Diss. ETH No. 16440

Analysis, Modeling, and Simulation of Diffusion Processes in Cell Biology

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZÜRICH

for the degree of
DOCTOR OF TECHNICAL SCIENCES

presented by
Ivo Fabian Sbalzarini

Dipl. Masch.-Ing. ETH
born on January 16th, 1977
citizen of Arbon TG, Switzerland

accepted on the recommendation of
Prof. Dr. P. Koumoutsakos, examiner
Prof. Dr. A. Helenius, co-examiner
2006

Abstract

Diffusion processes constitute a key mechanism for transport in biological cells. Nutrition, organization, growth, and signal transduction in cells are largely determined by diffusion mechanisms. The complex three-dimensional shapes of intracellular structures and the confinement of certain molecules to membranes however complicate the experimental analysis and computational simulation of diffusion in live cells. This thesis is concerned with the development and implementation of computational methods to analyze, model, and simulate diffusion processes in realistic cell environments.

In cell biology, mobile particles such as molecules, ions, vesicles, or viruses diffuse within the confines of the cell geometries. Two cases are considered: individually tracked particles, and their mean collective motion. The former case entails single particle tracking methods to directly follow the motion of individual particles. We present an accurate and computationally efficient image processing algorithm to determine trajectories of moving particles from digital videos. These trajectories are then analyzed with respect to their motion properties. We extend existing analysis methods to cases of anomalous diffusion and show that both the speed and the confinement of the motion can be quantified independently.

Automated trajectory analysis enables high throughput rates, minimizes human bias, and increases reproducibility. We therefore develop and apply methods for automatic trajectory classification, detection of motion patterns within trajectories, and adaptive data encoding to maximize classification performance. The developed computational tools are used in two studies of virus motion on the plasma membrane of live cells.

Analysis and simulation of the collective motion of abundant particles is based on continuum theory, which yields a model equation for the evolution of the resulting concentration field. Solving this governing equation is challenging for realistically complex cell geometries. We present particle methods to handle these complex geometries, and extend them to computations of diffusion on curved and moving surfaces.

The capability of numerically simulating diffusion both in spaces and on surfaces of complex shape allows to investigate the accuracy of fluorescence recovery experiments. We present for the first time the measurement of molecular diffusion constants in the endoplasmic reticulum of live cells by taking into account the complex geometry of the organelle.

All methods are implemented on the basis of a newly developed software library for hybrid particle-mesh simulations on parallel computers. The library is presented in this thesis and its parallel efficiency and scalability are demonstrated on a range of test cases.

Zusammenfassung

Diffusion ist einer der wichtigsten Transportmechanismen in biologischen Zellen. Ernährung, Organisation, Wachstum und Signalübermittlung in Zellen sind weitgehend von Diffusionsprozessen bestimmt. Die komplizierte dreidimensionale Form vieler innerzellulärer Strukturen, sowie die Bindung einzelner Moleküle an Membranen, erschweren jedoch die experimentelle Analyse und numerische Simulation von Diffusionsprozessen in lebenden Zellen. Die vorliegende Dissertation befasst sich mit der Entwicklung und dem Einsatz von rechnergestützten Verfahren zur Analyse, Modellierung und Simulation von Diffusionsprozessen in realen Zellen.

Diffusion in der Zellbiologie ist verbunden mit der Bewegung von Teilchen wie z.B. Molekülen, Ionen, Vesikeln oder Viren. Wir unterscheiden zwei Fälle: die individuelle Bewegung einzelner Teilchen, und die kollektive Bewegung einer grossen Anzahl von Teilchen. Im ersteren Fall benutzen wir Verfahren, die es uns erlauben den Bahnkurven einzelner Teilchen zu folgen und diese zu analysieren. Wir präsentieren ein effizientes und genaues Bildverarbeitungsverfahren, um die Bahnkurven der Teilchen aus digitalen Videoaufzeichnungen zu extrahieren. Diese Bahnkurven werden dann bezüglich ihrer Bewegungseigenschaften untersucht. Wir erweitern dabei existierende Analysemethoden auf Fälle anomaler Diffusion und zeigen, dass hiermit sowohl die Schnelligkeit als auch die Freiheit der Bewegung unabhängig quantifiziert werden können.

Eine durchgängig automatisierte Analyse ermöglicht hohen Datendurchsatz, reduziert menschgemachte Verzerrungen in den Daten und erhöht die Wiederholbarkeit. Wir entwickeln daher Methoden zur automatischen Klassifizierung von Bahnkurven, zur Identifikation von bestimmten Bewegungsmustern, sowie zur adaptiven Darstellung der Daten für maximale Klassifikationsgüte. Die entwickelten Verfahren werden in zwei Studien über die Bewegungen von Viren auf der Zellmembran verwendet.

Die Analyse und Simulation der kollektiven Bewegung einer grossen Anzahl von Teilchen basiert auf der Kontinuumstheorie, welche eine Modellgleichung für die Evolution des resultierenden Konzentrationsfeldes liefert. Das Lösen dieser Gleichung in realistisch komplexen Zellgeometrien ist numerisch herausfordernd.

Wir präsentieren Partikelmethoden für Diffusionssimulationen in komplexen Geometrien, und erweitern diese auf den Fall der Diffusion auf sich bewegenden gekrümmten Oberflächen.

Die Fähigkeit zur numerischen Simulation von Diffusionsprozessen in Räumen sowie auf Oberflächen von komplexer Gestalt ermöglicht es uns, die Genauigkeit von experimentellen Methoden der Fluoreszenzmikroskopie zu untersuchen. Wir präsentieren die erste Messung einer molekularen Diffusionskonstanten im endoplasmatischen Retikulum lebender Zellen unter expliziter Berücksichtigung der Geometrie der Organelle.

Sämtliche Simulationsprogramme wurden auf der Basis einer neu entwickelten Softwarebibliothek für Partikel-Gitter-Simulationen auf Parallelrechnern implementiert. Diese Bibliothek wird in der vorliegenden Dissertation beschrieben, und ihre Effizienz und Skalierbarkeit werden anhand mehrerer Testfälle demonstriert.

TO MY FAMILY AND FRIENDS

Acknowledgments

My foremost thanks go to Petros Koumoutsakos. Without him, this thesis would not have been written. Petros was a great adviser, an amazing source of ideas, and the driving support behind all of my work. He gave me the opportunity to work in a very dynamic and open environment, and to visit exciting places such as NASA Ames, Stanford University, and Caltech for extended periods of time. There I met Julie Theriot and Anthony Leonard, who are great scientists and teachers, and I am really privileged to having had the opportunity to work with them and learn from them.

I would also like to thank all the members of Petros' group for their continuous support, the discussions, lunches, and beers. I am particularly grateful to Michael Bergdorf for being a great discussion partner and source of scientific inspiration, as well as to Jens Walther, Michael Bergdorf, Simone Hieber, and Evangelos Kotsalis for being part of the PPM-team. Very special thanks go to Jens Walther; not only is he a constant support and a brilliant discussion partner, but also a true friend.

I am greatly indebted to Ari Helenius for awakening my love for biology and for being an excellent teacher and adviser. I particularly enjoyed the collaboration and discussions with Ari, and his group truly became my second home "upon the hill". Particular appreciation goes to Alicia Smith, Anna Mezzacasa, Arnold Hayer, and Helge Ewers. The close collaboration with them was a pleasurable, instructional, and rewarding experience, and I appreciate their work, without which major parts of this thesis would consist of blank pages.

I am grateful to Urs Greber, Yves Barral, Christoph Burckhardt, Cosima Lüdeke, and Stéphanie Buvelot Frei for giving me the opportunity to work with them, learn about their research, and for providing eclectic insights into biology. Credit also goes to all the students whose semester and diploma projects I supervised. Without their work, this thesis would have taken much longer to complete. Jo Helmuth deserves special mention for his ideas and work on trajectory segmentation.

My utmost thanks go to my family for the great personal support, and for enabling me to pursue the educational route that has led to this thesis. I am especially beholden to Christina for being a dependable support in even the most difficult of situations, and for her enduring patience, kind compassion, and undivided love.

Contents

List of Acronyms

2D	two-dimensional
3D	three-dimensional
Ad-2	Adenovirus-2
AGM	Adaptive Global Map
AMR	Adaptive Mesh Refinement
ANSI	American National Standards Institute
API	Application Programming Interface
CCD	Charge-Coupled Device
CMA	Covariance Matrix Adaptation
DiI	octadecyl indocarbocyanines
DNA	Deoxyribo-Nucleic Acid
DPD	Dissipative Particle Dynamics
EPS	Encapsulated Post-Script
ER	Endoplasmic Reticulum
FCS	Fluorescence Correlation Spectroscopy
FFT	Fast Fourier Transform
FMM	Fast Multipole Method, Fast Marching Method
FRAP	Fluorescence Recovery After Photobleaching
GFP	Green Fluorescent Protein
GMM	Gaussian Mixture Model, Group Marching Method
GTS	GNU Triangulated Surface library
GUI	Graphical User Interface
HMM	Hidden Markov Model
$i.i.d.$	*i*ndependent and *i*dentically *d*istributed
I/O	Input/Output
IVP	Initial Value Problem
KNN	k–Nearest Neighbor
LatA	Latrunculin A
LDL	Low Density Lipoprotein
MCD	Methyl-β-CycloDextrin
MD	Molecular Dynamics
MG	Multi-Grid
MPEG	Moving Picture Experts Group
MPI	Message Passing Interface
MSD	Mean Square Displacement
MSS	Moment Scaling Spectrum
ODE	Ordinary Differential Equation
OS	Operating System
PDE	Partial Differential Equation
PDF	Probability Density Function
$PI_{(4,5)}P_2$	Phosphatidyl-Inositol di-Phosphate
PIC	Particle-In-Cell
PM	Particle-Mesh
PP	Particle-Particle
PPM	Parallel Particle Mesh library
P^3M	Particle-Particle Particle-Mesh
PSE	Particle Strength Exchange
Py	Polyomavirus
Qdot	Quantum dot
ROB	Recursive Orthogonal Bisection
ROI	Region Of Interest
rSPH	remeshed Smooth Particle Hydrodynamics
RW	Random Walk
SAR	Stop-At-Rise
SNR	Signal-to-Noise Ratio
SPH	Smooth Particle Hydrodynamics
SPT	Single Particle Tracking
SSA	Stochastic Simulation Algorithm
STS	Super Time Stepping
SV40	Simian Virus 40
SVM	Support Vector Machine
TCP/IP	Transmission Control Protocol / Internet Protocol
TIFF	Tagged Image File Format
TIRF	Total Internal Reflection Fluorescence
VLP	Virus-Like Particles
VM	Vortex Methods
WENO	Weighted Essentially Non-Oscillatory
wt	wild type

List of Symbols

The following list explains the equation symbols used in this thesis. Naming conflicts are unavoidable, but the proper meaning of symbol is always unambiguously determined by the context. Vector and tensor quantities are printed in bold face.

Latin Characters

a	step displacement, edge length
b	background intensity, edge length
$\boldsymbol{b}$	small-scale solution
c	characteristic, compression ratio, reaction propensity
c_p	particle cost
d	dimension of the space (topology dimension)
d_0	capacity dimension
d_H	Hausdorff dimension
d_q	Renyi dimension of order q
d_s	spectral dimension
d_v	Vapnik-Chervonenkis dimension
d_w	dimension of the walk
d'	discrimination capability measure
e	base of natural logarithm, parallel efficiency, small scale
$\boldsymbol{e}$	canonical basis vector of $\mathbb{R}^d$
f	a function
f_i	feature i, false alarm rate
f_m	mobile fraction
g	a function, Riemann metric
h	inter-particle spacing, grid spacing
$\mathfrak{h}$	stochastic reaction parameter
h_i	hit rate
i, j	indices
k	kinetic reaction constant, outer band radius, normalized count
l	loss function, Lagrange polynomial
ℓ	trajectory index, characteristic length
m	number of samples or observations, mass
m_0, m_2	intensity moments of order 0 and 2
m_B	number of mesh points
m_i	miss rate
n	frame number, number of objects, refractive index
$\boldsymbol{n}$	(outer) surface normal
n_w	window width
p	point, particle, probability, pressure
q	reliability, particle
r	percentile threshold, radius, rate constant, order
r_c	cut-off radius
r_i	rejection rate
r_m	interaction radius
s	signal, standard deviation, speed
s_i	separation phases
t	time
$\boldsymbol{t}$	true class probabilities
$t_{1/2}$	recovery half-time
t_s	time stretching factor
u	concentration field
v	peak intensity level of an image
$\boldsymbol{v}$	velocity
$\boldsymbol{v}_r$	eigenvector
w	convolution kernel size, weight
$\boldsymbol{x} = (x, y, z)$	position
$\boldsymbol{x}$	data vector
y	class label
z	reduced cost
$\boldsymbol{A}$	image matrix, transition matrix
A_m	interaction area
B	normalization constant, a box
$\mathcal{B}$	set of output probability distributions
C	SVM parameter
C	a grid cell, a constant
$\boldsymbol{C}$	matrix of all point detections or characteristics
C^n	set of n-times continuously differentiable functions

$\boldsymbol{D}$	diffusion tensor
$\mathcal{D}$	data set
E	Euclidean space, large scale
E	expectation value
$\mathcal{E}$	test set
F	fluorescence intensity
$\mathcal{F}$	function space
G	Green's function
$\boldsymbol{G}$	link association matrix
H	neuron input level
H_0	null hypothesis
H_A	alternative hypothesis
$\mathcal{H}$	Heaviside step distribution
I	pixel intensity value, entropy
J	Bessel function, determinant of the Jacobian
K	image convolution kernel
$\mathcal{K}$	kernel feature space
K_0	kernel pre-factor
L	maximum displacement between frames, length
$\mathcal{L}_D$	differential operator
M	number of points, monitor function, molecules per mass
$\boldsymbol{M}$	matrix function
$\mathcal{M}$	a Riemannian manifold
Ma	Mach number
N	number of particles or objects
$\mathcal{N}$	normal (Gaussian) distribution
$\mathbb{N}$	set of natural numbers
NA	numerical aperture
N_B	number of particles in ROI
N_{iter}	number of iterations
N_{proc}	number of processors
O	output
$\mathcal{O}(\cdot)$	on the order of
P	probability density function, probability measure
$\mathcal{P}$	set of source points
Q	state, interpolation kernel
$\mathcal{Q}$	set of target points
Q_ϵ	integral operator

R	link range, true risk, interpolation kernel, gas constant, lateral resolution
$\mathbb{R}$	set of real numbers
R_e	empirical risk
Re	Reynolds number
S	score, parallel speed-up
$S^2(r)$	sphere of radius r
T	final time, normalized temperature
$\boldsymbol{T}$	projection operator, tensor of inertia
$\mathcal{T}$	training set
$\mathbb{T}$	ordered time space
T_f	sampling fraction
T_s	score threshold
U	wave front
$\mathcal{U}$	uniform probability distribution
V	volume
$\mathcal{V}$	adaptation velocity
W	interpolation kernel
$\boldsymbol{W}$	matrix of weights
X	domain limit, number of molecules
$\mathcal{X}$	set of observations
Y	spherical harmonic
$\mathcal{Y}$	set of class labels
Z	neuron activity level

Greek Characters

α	angle, model parameter, a coefficient
β	moment scaling spectrum slope, a coefficient
γ	moment scaling spectrum, a coefficient, specific heat ratio
Γ	cluster, circulation
δ	Dirac delta distribution, thickness, axial resolution
δ_{ij}	Kronecker delta symbol
δt	time difference, simulation time step
δx	bin size
Δ	difference of information
Δn	frame shift

Δt	sampling time, discretization time
Δx	displacement per frame
ϵ	kernel size
ε	small deviation, smallness parameter
ζ	mollifier function
η	isotropic kernel function
Θ	threshold, general kernel function
κ	inner band radius
$\hat{\kappa}$	curvature
λ	Poisson parameter, number of offspring, wavelength
λ_r	eigenvalue
Λ	a hidden Markov model
$\boldsymbol{\Lambda}$	generalized diffusion tensor
μ	moment of displacement, number of parents
$\boldsymbol{\mu}$	vector of means
μ_2	mean square displacement
ν, ν_2	scalar diffusion constant, SVM parameter ν
$\boldsymbol{\nu}_r$	scaled eigenvector
ξ	space stretching factor
$\boldsymbol{\xi}$	intrinsic coordinate on a manifold
π	ratio between circumference and diameter of a circle
Π	initial state probabilities, penalty, dimensionless number
ρ	residual
σ	kernel, standard deviation, step size
$\boldsymbol{\Sigma}$	covariance matrix
τ	time constant, adaptation time step
$\boldsymbol{\tau}$	stress tensor
ϕ	elementary cost
φ, ϑ	spherical angles
$\boldsymbol{\varphi}$	vector test function
Φ	a functional
$\boldsymbol{\Phi}$	mapping function
$\boldsymbol{\Phi}$	Jacobian
χ	indicator function
ψ	level function, a scalar function
$\boldsymbol{\Psi}$	stream function, a vector function
$\boldsymbol{\omega}$	particle strength, vorticity
Ω	domain of solution

$\hat{\Omega}$	reference space

Subscripts and Superscripts

0	initially, at point of reference
1, 2, 3	cartesian coordinate directions or components
app	apparent
B	relating to the bleached region of interest
eff	effective
exp	in the experiment
ϵ	mollified to size ϵ
f	filtered
h	discretized with resolution h
i, j, k	cartesian coordinate directions or components
ℓ	on trajectory ℓ
$\mathcal{M}$	intrinsic to a manifold
max	largest value of the variable
min	smallest value of the variable
n	at time step n
p	evaluated at particle locations, moment order
q	entropy order
sim	in the simulation
t	at time t
w	smoothed with resolution w
x, y, z	the component in the specified direction
∞	asymptotically
$\hat{\cdot}$	estimated, in reference space
$\tilde{\cdot}$	reconstructed, estimated, fluctuations
$\bar{\cdot}$	arithmetic mean

Special Symbols

$\mathbb{1}$	the identity matrix
$\forall$	for all
∇	the Nabla operator
∇^2	the Laplace operator

$\partial(\cdot)$	partial derivative, boundary of a domain
∞	infinity
$\sim$	has the physical units
$\Re$	real part of a complex number
$\#\{\cdot\}$	number of elements in a set (count measure)
$\langle\cdot\rangle$	average
$\|\cdot\|_2$	L_2 norm
$\|\cdot\|_\infty$	L_∞ norm
$\|\cdot\|$	absolute value, d-dimensional volume, determinant
$[\cdot]$	concentration of
$\{\cdot\}$	a set
$[\cdot,\cdot]$	closed interval
$(\cdot,\cdot)$	open interval
D/Dt	Lagrangian derivative with respect to time
$P[\cdot]$	probability of an event
$\wedge$	logical *and*
$\cap$	set intersection
$\cup$	set union
$\subset$	sub-set
$\in$	set membership
$\emptyset$	empty set

Introduction

With the availability of increasingly more quantitative experimental methods in biology and the life sciences, both the complexity of the systems and the amount of data produced are increasing. Over the past years the focus has shifted from considering isolated subsystems or individual chemical reactions to investigating the cell-wide interplay of processes and the internal organization of cells. Methods such as fluorescence microscopy, confocal microscopy, or total internal reflection microscopy are becoming standard tools to investigate intracellular transport phenomena. The prevalent lack of predictive theories as well as the vast amount of data available from these apparatuses make data analysis and inference from data one of the core challenges in the creation of knowledge.

A large number of cellular processes depend on the diffusion of macromolecules and substances of small molecular weight, such as metabolites and ions. Mechanisms such as convective flow and active transport do play a role in, e.g., vesicular transport or microtubule-dependent trafficking. Diffusion however constitutes the dominant mechanism for transport in important cell functions such as nutrition, organization, and signal transduction. The presence of internal membranes usually restricts diffusion to specific organelles or domains within the cell [8].

Computational tools are needed to analyze the data, extract patterns and features, or store and retrieve them. In diffusion analysis, two standard methods are widely used: single particle tracking and fluorescence recovery after photobleaching. Single particle tracking starts by recording videos of the motion of individual labeled particles. This technique can be applied to sparse systems, where the individual particles can be distinguished, and is covered in Part I of this thesis. Part II discusses the case where the collective motion of abundant particles is considered using continuum models. We focus on the case where the complex-shaped internal structures of the cell complicate the analysis, and where the absence of suitable models has long hampered fluorescence recovery methods. This thesis reviews and extends computational methods to simulate continuum diffusion processes both in complex-bounded domains and on complex curved surfaces.

The thesis is structured as follows:

Chapter 1. Automated Trajectory Acquisition by Video Analysis

We consider the problem of reconstructing the trajectories of moving particles from a digital video recording of their motion. This so-called feature point tracking is a crucial step in diffusion analysis of sparse systems as it determines the data throughput as well as the accuracy and the statistical significance of the results. Most of the existing feature point tracking algorithms make use of a-priori knowledge about the type of motion, such as, e.g., the trajectory smoothness. Other existing programs are computationally very expensive or need a lot of memory, thus preventing the processing of large amounts of long videos. Many landmark studies in biology have therefore used manual or semiautomatic tracking methods. *Can we construct an automatic feature point tracking procedure that is at the same time accurate and computationally efficient?* After presenting the suggested solution in Section 1.2, we test our claims of accuracy and efficiency in various benchmark cases in Section 1.3. We find that the present algorithm is at least as accurate and robust against image noise than the best previously available solutions, while being significantly faster.

Chapter 2. Trajectory Analysis

In Chapter 2 we consider a number of data analysis methods for trajectories. This starts from global methods that perform certain averaging operations along the whole trajectory. After reviewing the classical mean square displacement analysis method, we illustrate its limitations in analyzing anomalous diffusion processes. *Is there a global statistical analysis method that can be used for all types of motion and allows accurate classification?* In Subsection 2.1.2 we propose the moment scaling spectrum, originally introduced by Ferrari *et al.* [101], as a new, key measure for motion analysis.

Due to their averaging nature, global analysis methods are not suited to detect changes in the motion pattern within a single trajectory. While a certain level of time resolution can be achieved by applying the analysis in a moving window frame (Section 2.2), such analyses suffer from an inherent trade-off between resolution and accuracy as shorter and shorter trajectory pieces are analyzed at smaller window widths.

As we might for example be interested in analyzing residence times of particles that are temporarily stationary, trajectories are frequently decomposed into predefined segments. Such trajectory decomposition or segmentation is usually done

by hand, thus biasing the analysis result by the experimenter's choices and decisions. In analogy to the automated feature point tracking offering many advantages over manual tracking, we ask the question: *Can this biasing limitation be overcome by automatically decomposing the trajectory into pre-defined segments?* This would not only eliminate the bias, but also enable larger data sets and better reproducibility of the results. In Section 2.3 we propose a method to automatically detect and isolate certain patterns in trajectories. The presented trajectory segmentation technique is based on a neural network that is trained on demonstration samples before it is used on the actual data set.

Using the so-identified trajectory components enables *event-based analyses* as described in Section 2.4. We present three different counting and normalization strategies for intra-trajectory events, and explain them using two example events.

Chapter 3. Trajectory Classification

In Chapter 3, bias-free high-throughput analysis is extended to whole trajectories. This is complicated by the fact that trajectories constitute dynamic data, viz. ordered time series of position vectors. Data encoding is needed to exploit the temporal information and to reduce the dimensionality of the data. Moreover, trajectories exhibit multiple invariances with regard to translation, rotation, and symmetry, as rigid-body rotations or translations of a trajectory leave the motion patterns unchanged.

We fist introduce the problem of automatic classification of complete trajectories and survey the most frequently used computational (machine learning) techniques for automatic classification. Using a non-trivial data set from biology, we ask: *Which of the methods are suited for trajectory classification and how does their performance compare to human classification?* As a basis for our comparisons we introduce a measure to quantify the intrinsic separation capability of a classifier, machine or human. Human classification is done both by domain experts such as the experimenters themselves, and laypersons.

We find that the classification success critically depends on how the trajectories are represented in the computer. Using all the points that constitute the trajectory would lead to a very high-dimensional classification problem, requiring tremendous amounts of training data in order to achieve acceptable performance. Data compression and dimensionality reduction are thus mandatory concepts. But *how can we systematically find a good/optimal data representation in a given number of dimensions?* In Section 3.2 we present a framework consisting of an adjustable

encoder and an optimizer forming a closed loop with the classifier. The optimizer continuously adapts the encoder such as to maximize the classification performance. We test the influence of different classification performance measures and optimizer parameters on the outcome. The proposed algorithm is benchmarked on both synthetic and experimental trajectories, and the quality of the result is compared to human-found data encodings.

Chapter 4. Applications and Results

In Chapter 4, the feature point tracking, trajectory analysis, and trajectory classification methods presented thus far are applied to various problems of biological interest. While the case studies in Section 4.1 demonstrate the capabilities of the tracking algorithm, Sections 4.2 and 4.3 relate to collaborative research projects with two groups in biology. Both projects are concerned with the analysis of the motion of virus particles on the outside of the plasma membrane of live cells prior to internalization. Understanding these first steps of viral infection yields important *information about the uptake pathways that lead into the cell, about the internal organization of the plasma membrane, and about the cellular machinery involved in virus uptake*. The first project considers Polyomavirus and was carried out in close collaboration with the group of Prof. Helenius. The second project makes use of the automatic segmentation and classification techniques described earlier to investigate the role of the secondary receptor in the uptake pathway of human Adenovirus. This project was done in close collaboration with Prof. Greber at the University of Zürich and it exemplifies the statistical potential of large unbiased data sets.

Chapter 5. Particle Methods to Simulate Diffusion in Complex Geometries and on Curved Surfaces

Computationally solving the diffusion equation in real cell geometries is complicated by several factors. Resolving the complex geometries requires adaptive schemes and, in parallel simulations, adaptive domain decompositions. The computational efficiency of grid-based methods is drastically reduced when discretizing complex geometries, because the resulting linear algebraic systems fail to have the favorable structure associated with simpler geometries, resulting in fuller systems whose solution often scales with the square or even the cube of the number of computational elements. Moreover, the generation of robust computational

meshes in complex geometries remains a non-trivial task, despite the availability of several automatic procedures.

In this thesis we exploit the properties of mesh-free particle methods to simulate diffusion in complex cell geometries. A large number of problems in physics and engineering is most naturally described by particles, e.g. atoms in molecular dynamics simulations, charged particles in plasma physics, gravitational particles in astrophysics, or fluid elements in smooth particle hydrodynamics and vortex methods. Particle methods are not limited to their evident use in discrete systems, but can also be employed to simulate continuum systems. Hereby, computational particles are used to discretize the continuous functions by means of weights assigned to the particles. In continuum particle methods, the particles remain intimately linked to the physics they represent, as the governing equation is solved by appropriately evolving the locations and properties of the particles [160].

In Chapter 5 we summarize the fundamental concepts of continuum particle methods and proceed by reviewing existing particle methods for the solution of the diffusion equation in space. Motivated by the biological importance of membrane-bound molecules we then ask: *How can we construct a particle method to solve the diffusion equation on complex curved surfaces?* Intrinsic diffusion on manifolds has recently received considerable attention in computer graphics. In Section 5.3 we adopt a technique developed for video inpainting, and formulate it in the particle framework. Convergence and accuracy of the method are assessed in test cases before it is extended to the solution of reaction-diffusion problems on moving and deforming surfaces in Section 5.4. Representing complex-shaped surfaces at fixed resolution requires a large number of particles, rendering the method expensive. By means of the adaptive global mapping technique by Bergdorf *et al.* [27], we thus use multi-resolution particles, viz., particles whose size is locally adapted to the geometry that is to be resolved (Section 5.4).

Chapter 6. Simulations of Diffusion in Organelles of Live Cells

The method of fluorescence recovery after photobleaching is frequently used in biological experiments to observe the diffusive transport of a molecule. In this method, the molecule of interest is fluorescently tagged and expressed in the cell. Using intense laser light, a well-defined portion of the cell is bleached by irreversibly destroying the fluorophore. Influx of non-bleached molecules into the bleached region is recorded in a digital video. From such videos one wishes to infer the molecular diffusion constant of the molecule, as faster diffusion would lead to

faster fluorescence recovery. This analysis is however complicated if the space in which the fluorescent molecules are confined does not completely fill the bleached volume, i.e. if the bleached region is larger than the geometric structures within the cell, thus bleaching across domain boundaries. Moreover, the microscope video only shows a two-dimensional slice or projection of the three-dimensional cellular structures. Present methods to derive molecular diffusion constants from fluorescence recovery videos neglect these geometrical effects. *What are the errors made by neglecting the spatial organization of the cell in fluorescence recovery experiments?* And *can we correct for them?*

In Chapter 6 we consider diffusive fluorescence recovery in the Endoplasmic Reticulum (ER). The ER is a very complex-shaped organelle of cells, generally depicted as a convoluted network of connected tubular and lamellar structures. These structures are much smaller than the size of the bleached region and we expect that the complexity of the boundary shape has an influence on the observed recovery dynamics. To investigate these influences, we reconstruct the geometry of real ER samples from microscope image data (Section 6.2). In Section 6.3 the complexity of the shapes is quantified using the concept of fractal dimensions. This leads to the conclusion that the observed diffusion behavior appears qualitatively altered.

In Section 6.4 we describe the standard experimental technique used in fluorescence recovery assays and review currently used analysis models to determine diffusion constants from such measurements. The fractal geometry analysis allows us to derive a model that explains the data much better than previous ones. Using the simulation techniques described in Chapter 5 we then quantify the geometry-induced uncertainty in these classical analysis methods and propose a novel, geometry-safe method in Section 6.5. The presented method enables us to determine corrected molecular diffusion constants from fluorescence recovery data in complex geometries, and to assess existing analysis techniques.

Chapter 7. PPM – A Software Framework for Parallel Particle-Mesh Simulations

Particle methods formally amount to an N-body problem with a computational cost that potentially increases with the square of the number of particles. Although several strategies exist to reduce the computational cost to scale linearly with the number of particles, their wide-spread use is prohibited by several complications in the parallel implementation of such methods. For grid-based methods a number of flexible software libraries exists. These libraries provide a standard set of common

functions and greatly simplify the development of parallel simulation codes. To our knowledge, no such library or framework existed for particle methods.

Why don't we develop one? Particle methods also share a common set of operations and data structures, and a large number of applications in many fields of science and technology would benefit from an easy-to-use framework to develop robust and efficient computer simulations using particle methods. In Chapter 7 we describe a novel parallel particle mesh library, PPM, that was developed in the group of Prof. Koumoutsakos in order to enable flexible and rapid code development for particle methods and hybrid particle-mesh methods. After presenting the fundamental concepts and functions of the library, we demonstrate its parallel efficiency and computational performance on a number of test problems. This was a collaborative effort of five members of our group and many simulation programs using the PPM library have been written by other people.

Part I

Sparse Systems: Single Particle Analysis

Overview

In this part of the thesis we consider motion analysis in sparse systems. Such systems are characterized by the property that the motion of each particle can be followed individually. An efficient and accurate computational method to track particles in video recordings is described in Chapter 1, with details about the algorithm as well as benchmark results presented in Sections 1.2 and 1.3, respectively. Chapter 2 describes methods for analyzing the recorded trajectories. Starting from global (whole-trajectory) analysis methods, the resolution is gradually refined using moving window and segmentation techniques. Chapter 3 considers the problem of automatic classification of trajectories using methods from machine learning. In Section 3.1 the performance of different classification algorithms is assessed on a biological data set, and compared to the performance of human classification. In order to apply machine learning techniques, the trajectory data need to be encoded. To find good data representations for classification, we introduce a self-optimizing encoding strategy in Section 3.2. In Chapter 4, all methods presented in this part are applied to problems of biological interest. The main application considers the motion of virus particles on the plasma membrane of live cells prior to internalization. The examples demonstrate how the use of particle tracking and trajectory analysis can lead to biologically significant conclusions.

Single particle tracking in cell biology

Motion analysis on the single-particle scale starts from the trajectories of the moving objects. A *trajectory* is hereby defined as the "trace of positions followed by an object moving through space"[1]. Before such trajectories can be analyzed, they often need to be reconstructed from video recordings of the real moving objects. This (automated) *feature point tracking* step crucially determines the accuracy and the scope of the whole process. Feature point tracking consists of detecting the images of the particles in the digital video sequence, and linking those detections over time in order to follow the individual traces. The reconstructed trajectories can then be used to extract information about the behavior of the particles, their interactions, and their environment (cf. Chapters 2 and 3).

Besides biology, SPT has numerous applications in many fields of science and technology such as fluid mechanics (e.g. particle imaging velocimetry and particle tracking velocimetry [316]), computer vision (e.g. road following [196], human limb tracking [165]), navigation (e.g. vehicle navigation [237]), and material science (e.g. colloids [68]).

With increasing spatial and temporal resolution of the microscopy equipment, and with the wide-spread availability of techniques such as multicolor video microscopy and *Total Internal Reflection Fluorescence* (TIRF, cf. Chapter 4) microscopy [287], *Single Particle Tracking* (SPT) is becoming indispensable in cell biology. The quantitative analysis of the resulting trajectories provides important information about working mechanisms and structures in living cells [76]. SPT has been used first for descriptive studies of plasma membrane protein and lipid diffusion [329, 12, 110, 136, 106], and subsequently to address more complex issues of molecular transport [88, 235, 319, 100, 70]. Using SPT it has become possible to analyze cell motility [267], determine diffusion coefficients of single molecules [116], or measure the step displacements of molecular motors such as kinesin [108]. Descriptions and overviews of the employed trajectory analysis methods are available in the review by Saxton and Jacobson [242].

Video microscopy of fluorescently labeled virus particles moving on cell membranes and into internal organelles led to the pioneering study of Pelkmans *et al.* [212, 213]. Using frames from videos, they visualized and analyzed many of the key steps in the early pathway of the caveolar entry of *SV40* into live cells [212]. This analysis was performed by tracking the individual particles by hand, a procedure that becomes impossible when one needs to analyze the multitude of trajectories available by today's fast video techniques.

[1] source: www.wikipedia.org

Chapter 1

Automated Trajectory Acquisition by Video Analysis

In biology, a number of specialized, often application-specific, algorithms and computer programs for single particle tracking is available [55, 53, 298]. Most of them make use of a-priori knowledge about the physics of the problem in order to construct effective and robust feature point tracking procedures. Real applications however often involve the tracking of objects whose type of motion may not be known explicitly in advance. In these cases the tracking task is hindered by the absence of a suitable mathematical model, by the possible stochastic character of the motion, or by trajectories containing several modes of motion (e.g. smooth and non-smooth parts). While some of the existing feature point tracking algorithms are very accurate, they are also computationally intense, which prohibits tracking of long video sequences as often encountered in biological applications.

After defining the terminology of SPT in Section 1.1, we present a feature point tracking algorithm that was particularly developed for applications in cell biology (Section 1.2). The presented algorithm is fast and efficient, while attaining accuracy and precision that are comparable to far more computationally intensive algorithms (cf. Section 1.3). It is robust against imaging noise and intermittent detection of particles, making it suitable for tracking of long videos of mobile objects such as viruses on the plasma membrane, fast directed motion such as trafficking along microtubules, and particles with strong intensity fluctuations such as quantum dots (cf. Section 4.1). The presented algorithm is capable of processing a video of several thousand frames within a few seconds on a standard desktop computer. The algorithm relies on a minimum set of assumptions and reduced prior knowledge of the physical process, making it applicable to tracking problems with no a-priori information about the type of motion. Finally, the algorithm uses only few user-defined parameters, thus providing ease of use. These parameters are: the particle radius w, the intensity percentile r, the cutoff score T_s for the non-particle discrimination (see Subsection 1.2.1), the maximum link length L, and the number of future frames R for the linker. All parameters have a direct physical meaning and can easily be determined by inspection of a few frames of the movie. Moreover, the graphical user interface described in Appendix A.3 provides additional guidance and support for determining parameter values.

1.1 Definitions and problem statement

Before describing the tracking algorithm and its application to biological video sequences, we formally state the problem of feature point tracking and define the terms that are used throughout this chapter.

Consider physical *particles* that are mobile in a two-dimensional plane. Their motion is *observed* using imaging equipment and a digital (CCD) camera which generates a sequence of digital images at discrete time points. We call this sequence a *movie* and an individual image from it a *frame*. In each frame the images of the particles are visible as *feature points* (or *points*). The goal is to approximately reconstruct the motion of the observed particles. For a particle p, such a reconstruction consists of an ordered series of point locations $\{\boldsymbol{x}_p^t = (x_p^t, y_p^t)\}_{t=1}^T$ over the recording time points $t = 1, \ldots, T$ of the individual frames, and is called a (discrete) *trajectory*. To generate the trajectories, the *feature point tracking* algorithm has to perform two distinct steps: first it has to *detect* the feature points in every frame and then it has to *link* these point detections into trajectories. If a point is detected where there is none, we call it a *false detection*. The term *"spurious detection"* on the other hand refers to a correctly detected point where there was no particle of the desired kind in the real scene. Finally, linking two points that are not images of the same physical particle is called a *false link*. If a trajectory does not extend throughout the whole movie it is called an *incomplete trajectory*. The following sections describe and test an algorithm for *feature point tracking*, which is a sub-problem of *single particle tracking* and does not include treatment and analysis of the physical system under observation or the employed imaging equipment.

1.2 Feature point tracking algorithm

The automated reconstruction of trajectories from digital videos is developed under the assumptions of small feature points (compared to the length scale of background variations), limited speed, and short occlusions. The presented algorithm is self-initializing and capable of handling occlusion, exit, and entry. It is in the same functional class as the *IPAN tracker* introduced by Chetverikov and Verestóy [55],

except that the present work makes no assumptions about the smoothness of the trajectories. We present the algorithm in its two-dimensional form. It however also applies to tracking in three dimensions, provided three-dimensional video data are available. The only adaptation that needs to be made is to use position vectors with three components instead of two.

1.2.1 Feature point detection

The algorithm is initialized by determining the global[1] minimum I_{min} and maximum I_{max} of all intensity values occurring in the movie. All pixel intensity values I are then normalized as $(I - I_{\text{min}})/(I_{\text{max}} - I_{\text{min}})$. The use of global extrema preserves intensity variations across frames, serving as an important source of information in the linking step. The feature point detection consists of four steps:

1. Image restoration
2. Estimation of the point locations
3. Refinement of the point locations
4. Non-particle discrimination

The implemented algorithm has as a starting point the work by Crocker and Grier [68] for the detection of colloidal spheres in micrographs. In the following, the normalized frame image at observation time t is represented as a matrix $\boldsymbol{A}^t(x, y)$ of floating point intensity values between 0 and 1. The integer coordinate $x = 1, \ldots, N_x$ is the pixel row index and $y = 1, \ldots, N_y$ the pixel column index.

The **image restoration** step corrects for imperfections in the frame images. There are two different effects accounted for: (1) long-wavelength modulations of the background intensity due to non-uniform sensitivity among the camera pixels or uneven illumination, and (2) discretization noise from the digital camera. The former is straightforward to correct for since we assume the feature points to be small compared to background variations and thus well separated in spatial frequency. The background is removed by a *boxcar average* over a square region with extent of $2w + 1$ pixel:

$$\boldsymbol{A}^t_w(x, y) = \frac{1}{(2w+1)^2} \sum_{i=-w}^{w} \sum_{j=-w}^{w} \boldsymbol{A}^t(x + i, y + j)\,, \tag{1.1}$$

[1] i.e. across all the frames of the movie rather than within each frame individually

where the user-defined parameter w is an integer larger than a single point's apparent radius but smaller than the smallest inter-point separation. The camera *discretization noise*[2] is modeled as homogeneously Gaussian with a correlation length of $\lambda_n = 1$ pixel. The *de-noising filter* thus consists of a convolution of the image $\boldsymbol{A}^t$ with a Gaussian surface of revolution of half width λ_n [68]:

$$\boldsymbol{A}^t_{\lambda_n}(x, y) = \frac{1}{B} \sum_{i=-w}^{w} \sum_{j=-w}^{w} \boldsymbol{A}^t(x + i, y + j) \exp\left(-\frac{i^2 + j^2}{4\lambda_n^2}\right), \tag{1.2}$$

with normalization

$$B = \left[\sum_{i=-w}^{w} \exp\left(-\left(i^2/(4\lambda_n^2)\right)\right)\right]^2. \tag{1.3}$$

Both Eq. (1.1) and Eq. (1.2) amount to convolving the image with kernels of support $2w + 1$. The steps are thus combined and the final image restoration consists of a convolution of the original frame image with the kernel

$$\boldsymbol{K}(i, j) = \frac{1}{K_0}\left[\frac{1}{B}\exp\left(-\frac{i^2 + j^2}{4\lambda_n^2}\right) - \frac{1}{(2w+1)^2}\right]. \tag{1.4}$$

The normalization constant

$$K_0 = \frac{1}{B}\left[\sum_{i=-w}^{w} \exp\left(-\frac{i^2}{2\lambda_n^2}\right)\right]^2 - \frac{B}{(2w+1)^2} \tag{1.5}$$

allows comparison among images filtered with different values of w. The filtered image after restoration is given by:

$$\boldsymbol{A}^t_f(x, y) = \sum_{i=-w}^{w} \sum_{j=-w}^{w} \boldsymbol{A}^t(x - i, y - j)\boldsymbol{K}(i, j). \tag{1.6}$$

To perform above convolution, the image is temporarily padded to size $(N_x + 2w) \times (N_y + 2w)$ by repeating the first and last row and column w-fold outward

[2] Digital (CCD) cameras generate *shot pixel noise* during the process of photoelectron counting. Due to its discrete counting nature, this is a Poisson process [233]. The parameter λ in the Poisson probability density function is equal to the expected number of detected photoelectrons N. For $N > 9$ the Poisson distribution is well approximated by a Gaussian distribution (with errors $< 1\,\%$), and the discrete photon counts can be interpreted as continuous intensities.

each. Negative pixel values generated by the convolution are reset to 0. They are an artifact of the approximation of the camera noise by a Gaussian distribution, which breaks down at small intensity levels.

Estimating the feature point locations is done by finding local intensity maxima in the filtered image $\boldsymbol{A}_f^t$. A pixel is taken as the approximate location of a point if no other pixel within a distance of w is brighter, and if its intensity is in the upper r^{th} percentile of the frame's intensity distribution. The *intensity percentiles* are determined on a per-frame basis in order to be robust against possible global drift in image intensity over time, e.g. due to unspecific bleaching of the observed particles. The *local maximum selection* is implemented as a *grayscale dilation* [146] followed by the selection of all pixels that have the same value before and after the dilation. If such a pixel is in the upper r^{th} percentile of intensity values, it is taken as the candidate location of a point. The percentile criterion is needed to prevent the algorithm from selecting background pixels, that have low intensity both before and after the dilation.

The local maximum selection of point centers suffers from two deficiencies: (1) it is unable to reject noise, which leads to errors in the location estimate, and (2) it may include spurious detections such as random bright points in the background of the image or images of particle aggregates. This necessitates both a refinement of the detected locations and a subsequent non-particle discrimination.

Refinement of the point locations reduces the standard deviation of the position measurement. Other information gathered in the process can furthermore be reused later to reject spurious detections. We assume that the found local maximum of a point p at $(\hat{x}_p, \hat{y}_p)$ is near the true geometric center (x_p, y_p) of the particle. An approximation of the offset is given by the distance to the brightness-weighted centroid in the filtered (to reduce noise-induced positioning errors) image $\boldsymbol{A}_f^t$ [68]:

$$\begin{bmatrix} \varepsilon_x(p) \\ \varepsilon_y(p) \end{bmatrix} = \frac{1}{m_0(p)} \sum_{i^2+j^2 \leqslant w^2} \begin{bmatrix} i \\ j \end{bmatrix} \boldsymbol{A}_f^t(\hat{x}_p + i, \hat{y}_p + j)\,. \tag{1.7}$$

The normalization factor $m_0(p)$ is the sum of all pixel values over the feature point p, i.e. its *intensity moment of order 0*:

$$m_0(p) = \sum_{i^2+j^2 \leqslant w^2} \boldsymbol{A}_f^t(\hat{x}_p + i, \hat{y}_p + j)\,. \tag{1.8}$$

The location estimate is refined as: $(\tilde{x}_p, \tilde{y}_p) = (\hat{x}_p + \varepsilon_x(p),\, \hat{y}_p + \varepsilon_y(p))$. If either $|\varepsilon_x(p)|$ or $|\varepsilon_y(p)|$ is larger than 0.5 pixel, the candidate location $(\hat{x}_p, \hat{y}_p)$ is accordingly moved by 1 pixel and the refinement re-calculated.

The **non-particle discrimination** rejects spurious detections such as unspecific signals, dust, or particle aggregates. The implemented classification algorithm after Crocker and Grier [68] is based on the intensity moments of orders 0 and 2. The 0^{th} order moment of each point p has already been calculated in the previous step. The *second order intensity moment* is computed as:

$$m_2(p) = \frac{1}{m_0(p)} \sum_{i^2+j^2 \leqslant w^2} \left(i^2 + j^2\right) \boldsymbol{A}_f^t(\hat{x}_p + i, \hat{y}_p + j)\,. \tag{1.9}$$

The underlying assumption is that the majority of the detected observations corresponds to correct particles such that they form a dense cluster in the (m_0, m_2)-plane. Larger and dimmer or brighter structures such as aggregates or accumulations have different intensity moments and fall outside of the main cluster. Such outliers are identified by having each point p "carry" a 2D Gaussian

$$P_p(m_0, m_2) = \frac{1}{2\pi\sigma_0\sigma_2 N_t} \exp\left(-\frac{(m_0 - m_0(p))^2}{2\sigma_0} - \frac{(m_2 - m_2(p))^2}{2\sigma_2}\right) \tag{1.10}$$

with standard deviations σ_0 and σ_2, and N_t the total number of detected points. The contributions of all other points $q \neq p$ are summed for each point p at its location, giving a *score*

$$S_p = \sum_{q \neq p} P_q\left(m_0(p), m_2(p)\right)\,. \tag{1.11}$$

Every point detection having a score S_p above a certain user-provided threshold T_s is considered an observation of a "true" particle, all others are discarded.

Notice that the standard deviations σ_0 and σ_2 define the length scale of the clustering and can be chosen such as to normalize the cluster widths. Let I_{max} be the maximum intensity in the movie; $I_{\text{max}} = 1$ if the images are normalized as described earlier. We then have the bounds $0 \leq m_0 < I_{\text{max}}\pi w^2$ and $0 \leq m_2 < I_{\text{max}}\pi w^4/2$, which can be used to estimate values for σ_0 and σ_2. In our experience, a value of about $0.1 I_{\text{max}}\pi w^2$ seems to be a fair choice. Fig. 1.1 illustrates the non-particle discrimination clustering applied to a confocal image of fluorescently labeled Polyomavirus particles in a *PTK2 cell* (Helenius group). The image shows a confocal slice through the cell and thus contains observations of virus particles both on the plasma membrane and in the interior of the cell. The clustering is used to discard virus particles packed together in endocytic organelles, allowing analysis of individual free particles.

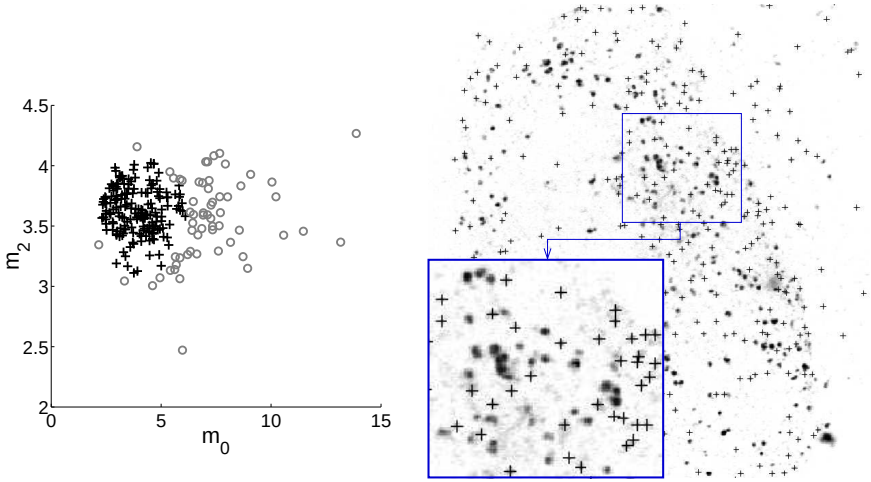


Figure 1.1: Left panel: Example of the non-particle discrimination clustering in the (m_0, m_2)-plane. Each symbol represents one detected feature point. The clustering with $\sigma_0 = \sigma_2 = 0.1$, and $T_s = 2.0$ (images normalized to $I_{max} = 1$) classifies the points marked by a plus symbol as "true" particles. Points outside the cluster are marked by circles and are classified as spurious detections. Right panel: Confocal image of fluorescently labeled Polyomavirus particles in a PTK2 cell (Helenius group; image intensity is inverted for printing purposes). The confocal slice contains both extracellular and intracellular regions. Some internalized virus particles are packed together in endocytic organelles that appear as larger fluorescent structures and are to be excluded from the trajectory linking. The result of the clustering shown in the left panel is illustrated with plus symbols marking "true" particles. Inset shows enlargement as indicated.

1.2.2 Trajectory linking

The feature point detection algorithm is applied to each frame image $\boldsymbol{A}^t$ and yields a set of T (total number of frames in the movie) matrices $\boldsymbol{C}^t \in \mathbb{R}^{N_t \times 2}$ with rows $[\tilde{x}_p, \tilde{y}_p]_{p=1}^{N_t}$, where N_t is the total number of points detected in frame t.

The linking algorithm identifies points corresponding to the same physical particle in subsequent frames and links the positions $\{\boldsymbol{C}^t\}_{t=1}^T$ into trajectories. This involves finding a set of *associations* between the *point location matrices* $\{\boldsymbol{C}^t\}_{t=1}^T$ such that a cost functional is minimized. The present implementation is based on a particle matching algorithm [71, 72, 73] using a graph theory technique [135] to determine optimal associations between two sets. This algorithm is extended so that each linking step may consider several frames to account for particle occlusion.

Let $\mathcal{P}$ the set of points p_i, $i = 1, \ldots, N_t$, in frame t and R a user-defined integer parameter specifying how many future frames are to be considered. For all sets $\mathcal{Q}_r$, $r = 1, \ldots, R$, of points q_j, $j = 1, \ldots, N_{t+r}$, in frame $t + r$ an *association matrix* $\boldsymbol{G}_r^t$ is defined:

$$\boldsymbol{G}_r^t(i,j) = g_{ij} = \begin{cases} 1 & \text{if } p_i \text{ in frame } t \text{ and } q_j \text{ in frame } t+r \text{ are produced by the same physical particle,} \\ 0 & \text{otherwise.} \end{cases} \tag{1.12}$$

We assume that there is always exactly one physical particle producing a single point detection. Note that this is a limiting assumption since particles could in principle coalesce or come so close that they are indistinguishable by the used imaging equipment, giving rise to one single point observation.

In order to allow the number of points to vary between frames, i.e. $N_t \neq N_{t+r}$, every association matrix is augmented with both a row g_{0j} and a column g_{i0} for *dummy particles* at times t and $t + r$, respectively. Linking a point to the dummy means that the corresponding particle disappeared from the observed part of the scene between frames t and $t + r$, and linking the dummy to a point means that the corresponding particle newly appeared. This leads to the following *topology constraint* on the matrices $\boldsymbol{G}_r^t$:

Constraint 1 *Every row $i > 0$ of $\boldsymbol{G}_r^t$ and every column $j > 0$ of $\boldsymbol{G}_r^t$ must contain exactly one entry of value 1, all others are zero. Only row 0 and column 0 are allowed to contain more than one entry of value 1.*

To find an optimal set of links $\{g_{ij}\}$, we need to define the *cost functional* to be minimized. In order to be able to use the efficient solution algorithm based on the transportation problem [135, 72], this functional needs to be linear in the association variables g_{ij} and may thus be written as the *linear combination*

$$\Phi = \sum_{i=0}^{N_t} \sum_{j=0}^{N_{t+r}} \phi_{ij} g_{ij} \,, \tag{1.13}$$

where ϕ_{ij} represents the *elementary cost* of associating point p_i in frame t with point q_j in frame $t+r$. The definition of ϕ typically involves the point positions, point characteristics, or, if available, temporal and spatial knowledge about the physics of the process. For the above functional to be linear, ϕ itself must not depend on the association variables g_{ij}. In our case, we use the quadratic distance between p_i, $i > 0$, and q_j, $j > 0$, as well as the quadratic differences in the intensity moments of order 0 and 2, thus:

$$\phi_{ij} = \left(\tilde{x}_{p_i} - \tilde{x}_{q_j}\right)^2 + \left(\tilde{y}_{p_i} - \tilde{y}_{q_j}\right)^2 + \left(m_0(p_i) - m_0(q_j)\right)^2 + \left(m_2(p_i) - m_2(q_j)\right)^2 \tag{1.14}$$

for $i, j > 0$. The cost of linking a point to one of the dummy particles $i = 0$ or $j = 0$ is set equal to: $\phi_{0j} = (rL)^2$, $j > 0$, and $\phi_{i0} = (rL)^2$, $i > 0$. This effectively places a limit to the allowed cost for point associations, since no association of cost larger than $(rL)^2$ can occur between regular points as the dummy association would be more favorable. The special case of linking a dummy to a dummy, i.e. $i = j = 0$, is of no concern and is arbitrarily set to $\phi_{00} = 0$. The parameter L is specified by the user and represents the maximum distance a point is allowed to travel between two subsequent frames, given its intensity moments remain constant. To accelerate the linking process, all costs $\{\phi_{ij} : \phi_{ij} > (rL)^2\}$ are set to ∞ and the corresponding g_{ij} is never considered in the following.

Initialization. The linking process starts by creating an arbitrary set of associations $\{g_{ij}\}$ which satisfies the topology constraint. Any valid association matrix $\boldsymbol{G}_r^t$ is acceptable since the linear nature of Φ ensures that the minimum of the cost function is unique [72]. Clever choice of the initial associations can however significantly reduce the number of iterations needed in the subsequent optimization process. The initial set of links is thus determined as follows: For each pair of frames (t, r), $r = 1, \ldots, R$, the association matrix $\boldsymbol{G}_r^t$ is initialized by assigning each point in frame t its nearest neighbor, using ϕ as the distance measure, in frame $t + r$ that is not already assigned to some other point. This means that for every

given $i = I$, $j = J$ is chosen such that ϕ_{IJ} is the minimum of all ϕ_{Ij} for which no other g_{iJ} is already set to one. This g_{IJ} is then set to one. If no such minimum is found, the point is linked to the dummy, i.e. g_{I0} is set to one. After having done this for all the points p_i, every J for which all g_{iJ} are 0 is determined and the corresponding g_{0J} are set to 1. This initialization generates a matrix $\boldsymbol{G}_r^t$ that fulfills the topology constraint. For low point densities this initial solution is already very close to optimal since only few conflicts occur, viz. the association that would have had the lowest elementary cost was already blocked by another one. To cope with regions of high point density, the association matrix is iteratively optimized.

Optimization. For each iteration, we scan through all g_{ij}, including the dummy particles, that are equal to zero *and* have finite associated cost ϕ_{ij}. For these we determine the *reduced cost* of introducing that association into the matrix. The reduced cost of an association $g_{IJ} = 0$, $I, J > 0$, is calculated from the elementary costs ϕ for $i, j > 0$ as follows: Let $g_{IL} = 1$ and $g_{KJ} = 1$, since every row and column must contain a 1 according to the topology constraint. Now if g_{IJ} was to be set to one, then g_{IL} and g_{KJ} must turn zero, otherwise points p_I and q_J would be in two places at once. Further, as point detections $i = K$ and $j = L$ must be related to some physical particle, it is necessary to set $g_{KL} = 1$. The reduced cost of setting g_{IJ}, $I, J > 0$, to 1 thus is:

$$z_{IJ} = \phi_{IJ} - \phi_{IL} - \phi_{KJ} + \phi_{KL} \qquad I, J > 0 \,. \tag{1.15}$$

If the reduced cost z_{IJ} is negative, introducing the association g_{IJ} into the solution is favorable, as it decreases the value of the cost functional Φ. In the case of a newly appearing particle, the association under consideration is at g_{0J} for some $J > 0$, and only the 1 in the same column at g_{KJ}, $K > 0$, is turned into a 0 and the dummy entry g_{K0} is set to 1. The reduced cost for an appearing particle thus is:

$$z_{0J} = \phi_{0J} - \phi_{KJ} + \phi_{K0} \qquad J, K > 0, L = 0 \,. \tag{1.16}$$

For a disappearing particle we similarly have:

$$z_{I0} = \phi_{I0} - \phi_{IL} + \phi_{0L} \qquad I, L > 0, K = 0 \,, \tag{1.17}$$

setting g_{I0}, $I > 0$, to 1, turning g_{IL}, $L > 0$, from 1 to 0, and setting the dummy g_{0L} to 1 as well. The special case $I = J = 0$ is set to $z_{00} = 0$. After calculating the reduced costs $\forall\{(i, j) : g_{ij} = 0 \wedge \phi_{ij} < \infty\}$, the g_{IJ} which corresponds to the most negative reduced cost $z_{IJ} = \min_{i,j} z_{ij}$ is set to 1, the corresponding g_{IL}

(if $I \neq 0$) and g_{KJ} (if $J \neq 0$) to zero, and g_{KL} to one. All the reduced costs are then re-calculated and the iteration is repeated until $z_{ij} \geqslant 0, \forall (i,j)$, which means that the *optimal set of associations*, with cutoff L, between frames t and $t+r$ has been found.

After doing so for all $r = 1, \ldots, R$ and a fixed specific t, all points in $\boldsymbol{C}^t$ that have been linked to the dummy particle in $\boldsymbol{C}^{t+1}$ are closer analyzed to re-connect broken trajectories as caused e.g. by particle occlusion, a sensitive non-particle discrimination, or a particle being close to the intensity percentile threshold. For each such point p_i in frame t, all association matrices $\boldsymbol{G}_r^t$, $r = 2, \ldots, R$ are scanned for valid associations to non-dummy points. If there are such associations, the one that has the smallest reduced cost is accepted and the corresponding point detections are linked.

Repeating the whole procedure for every frame t leads to an optimal (in the sense of the chosen cost functional Φ) linking of the detected point locations into trajectories over time. The computational cost of this linking algorithm formally scales as $\mathcal{O}\big(R(N^2 - N)\big)$ and the algorithm needs $\mathcal{O}\big(RN^2\big)$ memory. Associations between well-separated particles are however initially marked by an infinite cost and are never considered during optimization. This greatly improves the computational efficiency. The number of possible associations with finite cost values ϕ_{ij} is $\geqslant \max(N_t, N_{t+r})$, but much less than $(N_t + 1)(N_{t+r} + 1)$, depending on the actual distribution of the particles. In practice, the computational time for the present algorithm increases only slightly more rapidly than $\mathcal{O}(RN)$ and R is usually small (< 5). The trajectory linking hence takes less time than the feature point detection in most practical applications. A typical optimization of the association matrix $\boldsymbol{G}_r^t$ needs on the order of 10 iterations until the optimum set of links is found.

1.2.3 Computer implementation

The described feature point tracking algorithm is implemented in *ANSI C* as a multi-tier application using the *client-server* paradigm. The communication between the server and the clients is controlled by a simple packet-based protocol as specified in Appendix A.2.5. This makes it possible to reach the server from any remote computer that provides access to the network.

Server

The *server* application is completely written in standard ANSI C without compiler-specific extensions. This ensures portability of the code as it compiles and runs on every operating system that provides an ANSI C compiler.

The server consists of two parts: communication and point tracking. The point tracking part provides an *Application Programming Interface* (API) that is used by the communication part. This API implements the algorithm described in this section plus a set of functions for setting the tracking parameters, submitting a list of images, and retrieving the results. The technical documentation of the API is contained in Appendix A.2.3. The communication part connects the tracker API with multiple, potentially concurrent, clients. This is realized by *multi-threading* under the Microsoft Windows operating system, and *multi-processing* on all other platforms. The server accepts new connections from clients until a pre-defined maximum is reached. Further connection requests are denied by sending an appropriate protocol message to the client.

After establishing a connection, the server is listening to requests from the client. Initialization of the tracking procedure involves uploading all frame images of the movie. To avoid memory limitations, the server stores the frames on its hard disk. After at least two images of a valid file format are uploaded, the client can initiate the tracking process. The server then detects all particles in every frame and links the positions over time as described above. The tracking can be interrupted any time by a *reset signal* (cf. Appendix A.2.5).

The complete users manual of the server application is contained in Appendix A.1.1.

Client

Both a graphical and a *text-mode client* are provided. The latter is a lightweight implementation for efficient batch use, the former provides easy interactive use. Both clients are portable to a number of operating systems with the text mode client entirely written in ANSI C, and the graphical client in Java. Both clients have been tested on MacOS X, Windows, and Linux operating systems.

The client application provides an implementation of the communication protocol and a user interface. It sets the user-defined parameters of the tracking server, reads and submits an image sequence either as a series of TIFF images or directly from an MPEG-1 movie file, and receives the resulting trajectories from the server.

The text-mode client reads all information from input files and stores the results

in output files. The users manual in Appendix A.1.2 specifies the file formats.

The *graphical client* provides interactive use and assisted parameter choices. A preview functionality is available to easily test the effect of a parameter change. The graphical client also allows to filter the trajectories and to directly analyze their motion properties, diffusion constants [223], or moment scaling spectra [101] (cf. also Section 2.1). Furthermore, the graphical client can import and export trajectory data and parameter settings, and it also supports the printing and exporting of analysis plots. The complete users manual of the graphical client is contained in Appendix A.3.

Communication protocol

The packet-based communication protocol between the server and the clients is based on *TCP/IP* and uses a fixed-length packet header followed by a body of limited length (cf. Appendix A.2.5). The header specifies the message type and the length of the packet body. Receipt of every package is acknowledged to prevent loss of data. Since packet size is limited, it can be necessary to split an uploading file across multiple packets. They can be sent in any order, even interlaced with parameter packets. The server reconstructs the file after the last packet has been received.

Supported file types

TIFF images and MPEG-1 movie streams are currently supported. The support for TIFF images is provided by the open-source libtiff [286], MPEG-1 streams are handled by the open-source mpeglib [283].

1.3 Benchmarks

1.3.1 Computational cost

This section presents benchmark results testing the computational performance of the feature point tracking software. We also assess the memory usage of the program and describe a heuristic that is used to find the fastest algorithm for computing the convolution in the image restoration step of Subsection 1.2.1.

Timing results

The timing benchmarks are performed using movies consisting of 40 frames of size 512×512 pixel showing 24 particles moving along parallel straight lines. The kernel radius is set to $w = 6$ pixel, the maximum displacement to $L = 20$ pixel, and the threshold percentile to $r = 0.1\%$. The complete tracking process takes 25 seconds and uses 8 MB of main memory on a 1.8 GHz AMD Athlon computer with 256 MB of memory, running Microsoft Windows 2000 Professional. The software was compiled using Microsoft Visual Studio 6 with release compiler settings.

Using $w = 3$, a sequence of 100 frames of size 118×118 pixel showing 10 horizontally moving particles is tracked in less than 1 second on a 3.06 GHz Intel Pentium 4 computer running Linux. A movie of 3000 frames of size 214×214 pixel is processed in 14 seconds on the same machine.

A heuristic for time-optimal convolution

The convolution needed in the image restoration step can be computed either by direct evaluation of the sum or using *Fast Fourier Transforms* (FFT). For small kernel radii, the direct evaluation is faster, whereas the FFT method is preferable for large kernels. In order for the program to choose the appropriate method, we derive a simple heuristic from our timing results. Table 1.1 shows the computational time in milliseconds required by either method for different kernel sizes. We find the cross-over point at a kernel radius of 10 pixel.

The results for a constant kernel radius and varying image size are shown in Table 1.2. The cross-over point for $w = 12$ is around an image edge length of 400 pixel. For larger kernel radii, the cross-over point shifts toward smaller image sizes. For $w = 20$ it is at 256×256. For kernel sizes < 10 the direct method is always faster than the FFT (data not shown).

Using information about both the image size and the kernel radius, the time-optimal method of convolution is chosen by the program according to the following rules:

- For $w < 10$, the direct sum is used for all image sizes.
- For $10 \leqslant w < 20$, the choice depends on the size of the image. If the image edge length is smaller than the mean of the two nearest powers of two, the direct method is used, otherwise the FFT. The larger power of two is used as the effective image size since the FFT needs to pad the image.
- For $20 \leqslant w$, the FFT method is used for all image sizes.

Radius w [pixel]	Direct sum [ms]	FFT [ms]
1	4	15
2	4	16
3	5	16
4	8	17
5	11	20
6	17	24
7	21	26
8	25	29
9	30	31
10	35	35
11	40	38
12	47	42

Table 1.1: Computational time in milliseconds to calculate the convolution. For a fixed image of size 500×500 pixel, the kernel radius w is gradually increased. All results are averaged from 30 independent measurements.

Although these rules constitute a very simple model, they are effective enough to provide the proper choice of convolution method.

1.3.2 Accuracy and precision

The quality of the feature point detection is evaluated using synthetic frame sequences of moving point blobs. This method of evaluation is preferred over the common experimental practice of tracking a stationary/fixed particle and use the variance of the detected point positions as a measure of tracking quality. The true accuracy of the algorithm is given by its bias [53], which can not be estimated unless the precise and correct relative position of the particle with respect to the elements of the imaging system is known. The only way to achieve such conditions is the use of numerical simulations.

A good tracking algorithm has to meet two independent measures of quality: it should minimize *determinate errors* resulting from inaccuracies inherent to the algorithm and it should also minimize *indeterminate errors* from measurement fluctuations and imaging noise. While determinate errors systematically bias the position detections toward incorrect values, indeterminate errors fluctuate randomly. Following the terminology of Cheezum *et al.* [53], we refer to the measure of determinate errors as *accuracy* and the one of indeterminate errors as *precision*.

Image size [pixel]	Direct sum [ms]	FFT [ms]
256×256	13	20
300×300	17	23
350×350	23	27
400×400	30	31
450×450	38	36
500×500	47	42

Table 1.2: Computational time in milliseconds to calculate the convolution. For a constant kernel radius of $w = 12$ pixel, the image size is gradually increased.

Both accuracy and precision are estimated for a moving point source at different *Signal-to-Noise Ratios* (SNR) and pixel displacements per frame (Δx). Synthetic frames are created, showing particles moving along straight horizontal lines with a constant speed of Δx pixel/frame, cf. Fig. 1.3. Observation is simulated by centering a 2D *Gaussian blob*

$$I(x,y) = I_0 \cdot \exp\left(-\frac{(x-x_p)^2 + (y-y_p)^2}{4\sigma^2}\right) \tag{1.18}$$

of standard deviation $\sigma = 1$ pixel [284] at the current particle location (x_p, y_p) and sampling its value at the centers of all pixels $x = 1/2, 3/2, 5/2, \ldots$; $y = 1/2, 3/2, 5/2, \ldots$. Gaussian blobs are used as an approximation to (1) the sinusoidal intensity distribution of radially emitting spherical beads and (2) the square Bessel point spread function of a sub-resolution particle imaged using a microscope.

In order to model different SNR, a background (black) level of $b = 10$ is added to all pixels and the peak intensity v of the blobs is varied by setting $I_0 = v - b$ before adding the blobs to the images. For the noise model we assume that the images are acquired using a digital CCD camera. Such cameras produce *Poisson-distributed shot pixel noise* due to the discrete nature of photoelectron counting [233]. Pixel noise is thus simulated by replacing the intensity value I of each pixel with a random number from a Poisson distribution of expectation value $\lambda = I$. Fig. 1.2 illustrates the effect of such noise on a Gaussian blob. All random numbers are generated independently for every trial and frame, and the resulting frame images (cf. Fig. 1.3) are stored as unscaled 16-bit TIFF files.

The *SNR* is calculated as the difference in expected intensity levels between the particle peaks v and the background b, divided by the noise level σ_n on the

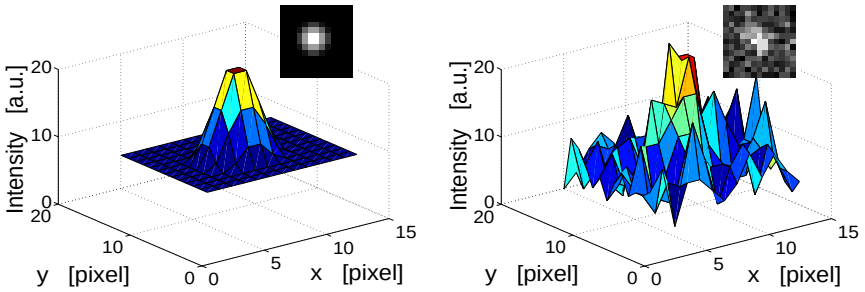


Figure 1.2: Example of a simulated particle observation before (left panel) and after (right panel) addition of Poisson noise. Insets show the particle images whose pixel intensity distributions are depicted in the surface plots below. The example shown uses a peak level of $v = 23.9$ and a background level of $b = 10$, resulting in a signal-to-noise ratio of 2.846.

particles. For the employed Poisson noise this is $\sigma_n = \sqrt{v}$ and thus:

$$\text{SNR} = \frac{v - b}{\sqrt{v}} \,. \tag{1.19}$$

This is the most conservative definition of SNR possible as using the noise level of the image background would lead to much larger values. These larger values are however inappropriate [53] since the stronger noise on the bright blobs is what affects the feature point detection and causes its inaccuracy. When measuring the SNR of real images, the noise ought to be estimated from the bright points instead of from the dark background. The peak pixel levels used in the present benchmark cases are given in Table 1.3 along with the corresponding resulting SNR values according to Eq. (1.19).

Accuracy and precision of the algorithm are quantified for different SNR and Δx using respectively the track *bias*

$$\text{bias} = \langle \hat{a} - a \rangle \tag{1.20}$$

and its *standard deviation*

$$\sigma = \langle (\hat{a} - \langle \hat{a} \rangle)^2 \rangle^{1/2} \,. \tag{1.21}$$

Hereby, $\langle \cdot \rangle$ denotes the ensemble average over independent trials, $\hat{a}$ are the reconstructed particle displacements from the tracking algorithm, and a the actual exact displacements.

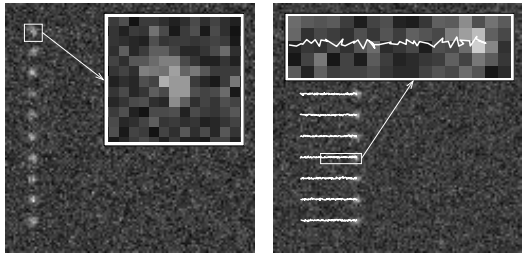


Figure 1.3: Example benchmark tracks. Each test case consists of an image sequence of 100 frames with 10 moving points, yielding 1000 independent displacement measurements $\hat{a}$ with known exact values $a = \Delta x$. The first (left panel) and last (right panel) frame of an example with $\Delta x = 0.27$ pixel, peak level $v = 23.9$, and background level $b = 10$ (SNR $= 2.846$) are shown with lines depicting trajectories as reconstructed by the present tracking algorithm. All 10 trajectories are of full length 100. Insets show enlargements as indicated.

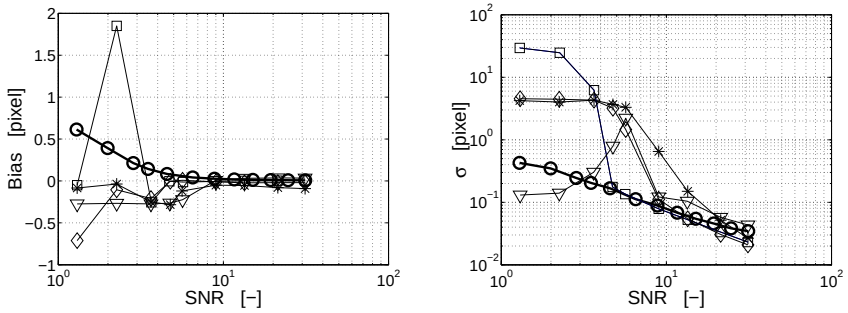
Fig. 1.3 shows both the first and the last frame at SNR $= 2.85$. The trajectories as reconstructed by the tracking algorithm are shown as solid lines in the right panel. The bold circles in Fig. 1.4 show the results for accuracy and precision versus SNR for a fixed displacement of $\Delta x = 0.27$ pixel. Fig. 1.5 shows bias and standard deviation versus the magnitude of the true particle displacement per frame between 0 and 1 pixel in steps of $1/11$ pixel at a fixed SNR of 31.3.

The *critical SNR* for the accuracy to become better than 0.1 pixel is around 4.2 for the present algorithm, indicating its good capability to handle noisy images. The precision σ is better than 1 pixel for all SNR larger than 1.3, cf. Table 1.4. For all SNR above 7.5, both the standard deviation and the bias are below 0.1 pixel. The present algorithm shows about the same accuracy as the more complex and computationally intense Gaussian fit and cross-correlation methods, while having better precision. The smooth and monotonic decay of both bias and standard deviation with increasing SNR are additional favorable properties of the present method, and the bias is virtually constant (and low) for all step displacements $\Delta x > 0.1$ pixel. The fact that the present algorithm avoids fitting a specific point spread function shape to the blobs in the frame images not only results in faster execution speed, but also renders it more general with respect to size and shape of the tracked objects.

In a second test, the simulated points are moving along straight lines of random angular orientation, thus exhibiting truly two-dimensional motion. Trajectories

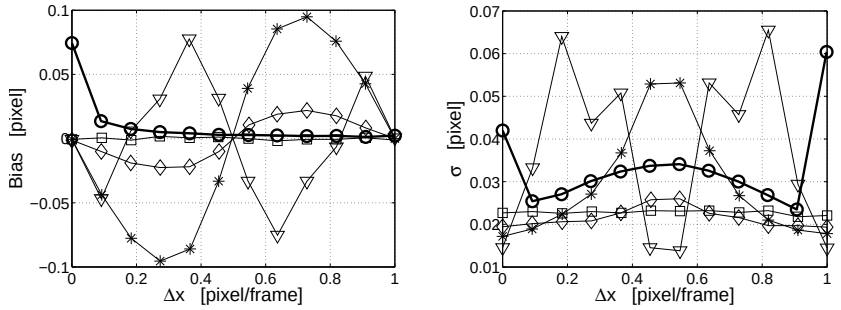
peak level v	SNR
15	1.291059
18.58	1.990510
23.9	2.846111
28.73	3.494379
38.1	4.556798
60.8	6.516668
97	8.832892
154.7	11.632132
246.6	15.067460
393.3	19.326731
627.1	24.642859
1000	31.306549

Table 1.3: Peak pixel levels v and resulting SNR used for the test cases in Fig. 1.4. The background level is fixed at $b = 10$.



(a) (b)

Figure 1.4: (a) Bias versus Signal-to-Noise Ratio (SNR) for a Gaussian blob moving at 0.27 pixel/frame. (b) Standard deviation versus SNR for the same cases. Each point is averaged from 1000 independent measurements. The present algorithm (bold circles) is compared to four existing algorithms as benchmarked by Cheezum et al. [53]: Gaussian fit (squares), Centroid (triangles), Sum of absolute differences (stars), Cross-correlation (diamonds).

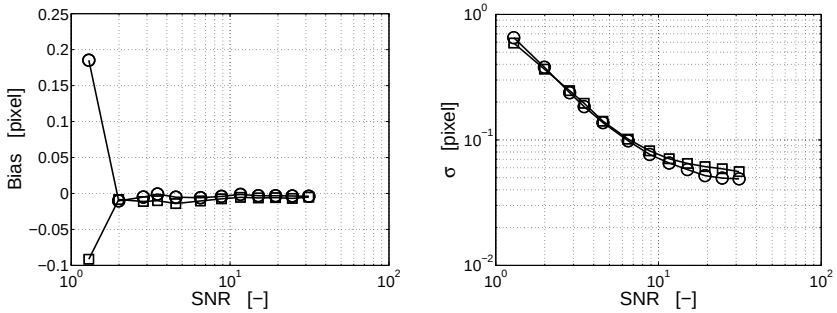


(a) (b)

Figure 1.5: (a) Bias versus actual distance moved per frame for a Gaussian blob at SNR $= 31.3$. (b) Standard deviation versus actual distance moved per frame for the same cases. Each point is averaged from 1000 independent measurements. The present algorithm (bold circles) is compared to four existing algorithms as benchmarked by Cheezum et al. [53]: Gaussian fit (squares), Centroid (triangles), Sum of absolute differences (stars), Cross-correlation (diamonds).

Algorithm	$\text{SNR}_{0.1\text{bias}}$	$\text{SNR}_{1.0\sigma}$
Present work	4.2	<1.3
Gaussian fit [53]	4.2	4.0
Centroid [53]	7.8	6.6
Sum of absolute differences [53]	6.9	8.1
Cross-correlation [53]	4.2	6.3

Table 1.4: Simulated SNR beyond which the bias remains below 0.1 pixel and σ below 1 pixel. Comparison of the present algorithm with the ones tested by Cheezum et al. [53].



(a) (b)

Figure 1.6: (a) Bias versus Signal-to-Noise Ratio (SNR) for 10 Gaussian blobs moving at random angular orientations with 0.27 pixel/frame. (b) Standard deviation versus SNR for the same cases. Each point is averaged from 3500 independent measurements. Circles indicate the x component, squares the y component of the respective measures.

can intersect and points can leave the image, in which case they reappear on the opposite side (*periodic boundary conditions*), and a new trajectory starts. This test mimics the situation of *finite dilution*. The same background and peak values are used as for the previous test (cf. Table 1.3), but bias and standard deviation are computed on the actual positions (x, y) – rather than the displacements a – as:

$$\text{bias}_x = \langle \tilde{x} - x \rangle \qquad \text{bias}_y = \langle \tilde{y} - y \rangle \tag{1.22}$$

and

$$\sigma_x = \langle ((\tilde{x} - x) - \langle (\tilde{x} - x) \rangle)^2 \rangle^{1/2} \qquad \sigma_y = \langle ((\tilde{y} - y) - \langle (\tilde{y} - y) \rangle)^2 \rangle^{1/2}\,, \tag{1.23}$$

where $\langle \cdot \rangle$ now denotes the ensemble average over all point detections in a movie. The results are shown in Fig. 1.6. While the standard deviation is comparable to the one in Fig. 1.4, the bias values are much lower than in the previous test. This is due to the fact that bias and standard deviation are correlated in the one-dimensional case, whereas they are independent here.

To test the trajectory linking in the case where two particles cross, we consider test movies showing 10 horizontally moving points and 10 vertically moving points, such that each pair of points exactly coincides in a certain frame. The

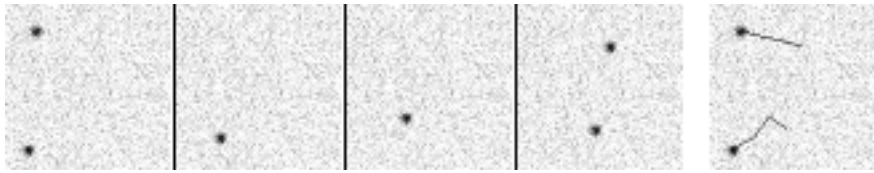


Figure 1.7: Sequence of two moving points with the upper one missing in two frames, e.g. due to occlusion or tight thresholding. The link range is $R = 3$, thus taking 3 subsequent frames into account for each linking step. The panel to the very right shows the correct recovery of the broken trajectory. (Image intensities are inverted for printing purposes.)

background intensity is again fixed at 10, and the peak intensity of the horizontally moving points is fixed at 23, corresponding to an SNR of 2.71. The peak intensity of the vertically moving points is gradually increased. Whenever two particles coincide, only one point observation is detected. Since the linking algorithm does not allow a point to be part of multiple links in any frame, one of the two trajectories must end. In the case where the two point sets are of equal brightness, the choice is random. In 50% of the cases, the vertical trajectory is continuous and the horizontal one pauses, and vice versa for the other 50%. If the point intensities (i.e. m_0) however differ, the trajectory of the brighter particle is consistently continued, whereas the dimmer one breaks. This is due to the particular choice of linking cost function, Eq. (1.14), where differences in m_0 are taken into account, and the fact that the brighter particle "masks" the dimmer one in the local maximum selection. A difference in SNR of 0.15 is sufficient for this to happen in 100% of the cases.

The case where a particle temporarily escapes detection is considered in Fig. 1.7. Extending the link range to $R > 1$ future frames (cf. Section 1.2.2) successfully prevents gaps in the resulting trajectories, as both points are available for linking.

Chapter 2

Trajectory Analysis

The automated reconstruction of trajectories from video sequences provides us with a wealth of information that can be exploited to quantify the particles' motions. Rather than a-priori distinguishing between "random" and "deterministic" types of motion, we use a single method of analysis for all trajectories. This is motivated by the fact that every motion can be viewed as a particular realization of a *stochastic process* of particle positions $\boldsymbol{x}(t)$. The motion process is hereby completely characterized by the probability of a particle originally at $\boldsymbol{x}_0$ to be at $\boldsymbol{x}$ after some time δt. The resulting probability density $P(\boldsymbol{x}|\boldsymbol{x}_0, \delta t)$ is called the *transition density* of the process. For unrestricted isotropic diffusion in d-dimensional space, the transition density is analytically known [48] to be the Gaussian

$$P(\boldsymbol{x}|\boldsymbol{x}_0, \delta t) = \frac{1}{(4\pi\nu\delta t)^{d/2}} \exp\left[-\frac{\|\boldsymbol{x}-\boldsymbol{x}_0\|_2^2}{4\nu\delta t}\right], \tag{2.1}$$

with ν the diffusion constant of the process. This result is obtained from the *central limit theorem* for a large number of independently moving *Brownian particles* [40, 92]. The same theory also applies to, e.g., an object moving deterministically along a straight line with constant velocity $\boldsymbol{v}$. Its transition density is given by the *Dirac distribution* $P(\boldsymbol{x}|\boldsymbol{x}_0, \delta t) = \delta(\boldsymbol{x}-\boldsymbol{x}_0-\boldsymbol{v}\delta t)$. This type of formulation thus allows us to use the same analysis methods for all types of motion and to consider trajectories of particles that change their behavior over time, a phenomenon frequently observed in biological applications.

In this chapter we present a hierarchy of trajectory analysis methods. Starting from global whole-trajectory methods, more locally resolved types of analyses are constructed. The global analysis is based on operators that reduce complete trajectories – or parts of a larger trajectory – to scalar numbers, employing certain averaging techniques along the trajectories (Section 2.1). After briefly reviewing the classical mean square displacement method [223, 242] in Subsection 2.1.1, we extend the global analysis by an additional operator that is based on the Moment

Scaling Spectrum (MSS) [101] of the trajectory (Subsection 2.1.2). We show in Chapter 4 that the combination of these two parameters allows to quantify both the "speed" and the "freedom" of the moving object independently. A simple analysis of the angle changes between subsequent displacement steps can also provide valuable information, as outlined in Subsection 2.1.3.

Temporal resolution can be achieved by considering an analysis window that moves along the trajectory. Within the moving window, independent global analyses are performed, resulting in a (smoothed) time series of quantification parameters, which allows to study their evolution along the trajectory as outlined in Section 2.2.

Moving window methods are limited by the inherent trade-off between temporal resolution, given by the width of the window, and statistical uncertainly. This can be overcome by decomposing the trajectory into pre-defined segments. Such *trajectory segmentation* techniques may be used to detect periods of immobility or super-random motion within a trajectory. In Section 2.3 we present an automatic trajectory segmentation procedure that is based on neural networks. The segments identified by such an algorithm can then be quantified separately to measure, e.g., the distribution of residence times in arrest zones or the mean speed during periods of directed transport.

Based on a segmentation of the trajectory, it becomes possible to define *events* as particular sequences of segments (Section 2.4). Event counts however need to be normalized by the expected number of events under purely random conditions. We present a Monte Carlo simulation technique and an analogy to chemistry as ways to provide such normalization.

2.1 Global trajectory analysis

The objective of *global trajectory analysis* is to reduce a complete trajectory to one or several scalar *quantification parameters*, describing certain characteristics of the motion. These quantification parameters can be defined as averages of functions of the trajectory data points $\boldsymbol{x}$, thus

$$\Phi(\Delta n) = \frac{1}{M-\Delta n} \sum_{i=0}^{M-\Delta n-1} f(\boldsymbol{x}(i), \boldsymbol{x}(i+\Delta n)), \tag{2.2}$$

where M is the total number of points in the trajectory and $\Delta n = 1, \ldots, M/T_f$ is the frame shift. Combining several such parameters allows to represent the trajectory as a point in *phase space*. Since the number of parameters is usually much

smaller than the number of points in the trajectory, this *dimensionality reduction* operation has to be designed to preserve the most important features of the motion.

After reviewing the classical mean square displacement method and its limitations, we propose an extension that is based on the work by Ferrari *et al.* [101]. This extension allows to treat a wide range motion types using a single theory.

2.1.1 Mean square displacement and diffusion constant

The second moment of the individual step displacements of a discrete trajectory is the most commonly used quantification parameter.

To define this parameter and its relation to the diffusion constant, let $\boldsymbol{x}_\ell(n) \in \mathbb{R}^d$ the position vector on trajectory ℓ at time $n\Delta t$ for $n = 0, 1, 2, \ldots, M_\ell - 1$, where M_ℓ is the total number of points in trajectory ℓ, i.e. its *length*. Δt is the real-time difference between two subsequent frames, viz. the *sampling time* of the discrete trajectory. The *Mean Square Displacement* (MSD) during a specific time interval $\delta t = \Delta n \Delta t$ is defined as

$$\mu_2(\delta t) = \langle \|\boldsymbol{x}(\delta t) - \boldsymbol{x}(0)\|_2^2 \rangle \,. \tag{2.3}$$

The average $\langle\cdot\rangle$ is taken over an ensemble of independent trajectories of the same motion process and can only be analytically computed for those special cases where the transition density is known. For isotropic *Brownian motion*, the transition density is given by Eq. (2.1) as long as $\delta t \ll L^2/(2d/\nu)$, where L is the diameter of the space available for diffusion. For this particular transition density, the MSD can be analytically computed as [223]

$$\mu_2(\delta t) = \iint P(\boldsymbol{x}_0)\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2 P(\boldsymbol{x}|\boldsymbol{x}_0, \delta t)\, d\boldsymbol{x}\, d\boldsymbol{x}_0 = 2\nu d \delta t \,. \tag{2.4}$$

Measuring the MSD of a trajectory of a diffusion process thus allows to determine its diffusion constant ν from the slope of μ_2 versus δt.

Diffusion with an overlaid deterministic drift of velocity $\boldsymbol{v}$ is treated similarly, leading to the expression $\mu_2 = 2\nu d\delta t + (\|\boldsymbol{v}\|_2 \delta t)^2$ [48]. The drift speed $\|\boldsymbol{v}\|_2$ can be determined from the curvature of μ_2 versus δt, or the slope in a logarithmic plot.

For most practical applications, the transition density $P(\boldsymbol{x}|\boldsymbol{x}_0, \delta t)$ is however unknown. The ensemble average is thus replaced by a time average, assuming that the process is *stationary* and *ergodic* [164]. This allows to compute the MSD for

any given trajectory as

$$\mu_2(\Delta n) = \frac{1}{M_\ell - \Delta n} \sum_{n=0}^{M_\ell - \Delta n - 1} ||\boldsymbol{x}_\ell(n + \Delta n) - \boldsymbol{x}_\ell(n)||_2^2 \,. \tag{2.5}$$

If the trajectory has a finite length $M < \infty$, the above average is interpreted as the mean of a set of random variables. As such, it has a *statistical uncertainty*, quantified by its variance [223]

$$\text{var}(\mu_2(\Delta n)) = \frac{(2\nu d \Delta n \Delta t)^2}{C} \,, \tag{2.6}$$

where

$$C = \frac{3\Delta n\,(M - \Delta n + 1)}{2\Delta n^2 + 1} \tag{2.7}$$

corrects for the fact that the samples in the time average Eq. (2.5) are correlated. The resulting variance for the measured diffusion constant is [223]

$$\text{var}(\nu) = \frac{2\Delta n}{3\,(M - \Delta n)} \,. \tag{2.8}$$

From Eq. (2.6) it is obvious that the error in the MSD increases as Δn approaches M. We thus compute the MSD using Eq. (2.5) only for $\Delta n = 1, \ldots, M/T_f$ with a factor $T_f > 1$. Usually $T_f = 5$, which bounds the standard deviation of the MSD at 40%.

2.1.2 Anomalous diffusion and moment scaling spectrum

The MSD as introduced in the previous section allows the determination of diffusion constants and velocities for normal Brownian diffusion processes and diffusion with superimposed uniform drift. *Normal diffusion* processes are characterized by an MSD that grows linearly with time shift, thus $\mu_2 \propto \delta t$. If this is not the case, the process is called *anomalous diffusion* [240]. Anomalous diffusion can for example originate from finite diffusion spaces, i.e. confinement, or long-tailed transition densities $P(\boldsymbol{x}|\boldsymbol{x}_0)$. For the former case, it is clear that the MSD can not continue to grow linearly as soon as the particles start to hit the boundary. Instead, we have the finite limit [223]

$$\lim_{\delta t \to \infty} \mu_2(\delta t) = 2\left(\langle \|\boldsymbol{x}\|_2^2 \rangle - \langle \|\boldsymbol{x}\|_2 \rangle^2\right) \,, \tag{2.9}$$

which is proportional to the size of the space accessible to diffusion. The case of long-tailed distributions $P(\boldsymbol{x}|\boldsymbol{x}_0)$ was first studied by Lévy [166]. It corresponds to a motion process where the particles occasionally "break out" and "fly" over a long distance. These *Lévy flights* are, e.g., observed in turbulent fluid flows [311], or in biological systems involving active transport by motor proteins. Mathematically, Lévy flights are characterized by a transition density $P(\boldsymbol{x}|\boldsymbol{x}_0)$ with an infinite second moment. In this case, the central limit theorem no longer applies and the diffusion constant is not defined [311]. These cases are called *superdiffusion* and their MSD grows faster than linear with time: $\mu_2 \propto \delta t^{\gamma(2)}$ with $1 < \gamma(2) < 2$. Cases with $0 < \gamma(2) < 1$ are termed *subdiffusion* and can for example originate from confinement of the particles. The special case of *Normal diffusion* is included with $\gamma(2) = 1$.

Ferrari *et al.* have shown that anomalous cases with $\gamma(2) = 1$ exist, so that the inversion of above characterization of normal diffusion does not hold. They considered a generalized version of the *telegraph model* [197, 101], describing the correlated random walks of an ensemble of particles that randomly switch between moving with velocity $+\boldsymbol{v}$, $\mathbf{0}$, or $-\boldsymbol{v}$. The switching probabilities between these three states depend on the elapsed time since the last switch. The particles thus carry an "age", making the process non-Markovian. We reproduce the results of Ferrari *et al.* [101] by simulating the telegraph model in the computer. The MSS as determined from these simulations is plotted in Fig. 2.1(a), and shows that $\gamma(2) = 1$. Standard MSD analysis would thus classify this process as normal diffusion. The transition density shown in Fig. 2.1(b) however reveals the non-Gaussian character of the process, certainly not corresponding to normal diffusion. This finding means that the MSD is not a sufficient criterion to distinguish Gaussian from non-Gaussian processes.

A sufficient criterion can be found by extending the trajectory analysis to the whole spectrum of *displacement moments* [101]:

$$\mu_p(\Delta n) = \frac{1}{M_\ell - \Delta n} \sum_{n=0}^{M_\ell - \Delta n - 1} \| \boldsymbol{x}_\ell(n + \Delta n) - \boldsymbol{x}_\ell(n) \|_2^p \,. \tag{2.10}$$

The MSD is included as the special case of $p = 2$. Each moment obeys a *scaling power law* [101]

$$\mu_p \propto \delta t^{\gamma(p)} \,, \tag{2.11}$$

for which the *scaling coefficient* $\gamma(p)$ can be determined from $\log \mu_p(\delta t)$ versus

$\log \delta t$ using linear regression. The function

$$\gamma \in C^1 : \mathbb{R}_0^+ \mapsto \mathbb{R}_0^+, \, p \to \gamma(p) \tag{2.12}$$

is called the *Moment Scaling Spectrum* (MSS) of the trajectory. Since $\gamma(0) \equiv 0$, the MSS always starts at the origin. Considering the complete spectrum of moments allows unambiguous classification of the process in Fig. 2.1 as anomalous diffusion.

Ferrari *et al.* have introduced a classification of dispersion processes according to their MSS. Processes with linear MSS are called *strongly self-similar*, those with non-linear MSS are called *weakly self-similar*. Mathematically, the distinction is based on the observation that the transition densities asymptotically collapse to a *self-similar form*

$$P(\boldsymbol{x}|\boldsymbol{x}_0, \delta t) \propto \delta t^{-\beta} P\left(\frac{\boldsymbol{x}}{\delta t^\beta}\right) \qquad \text{for } \delta t \to \infty \,. \tag{2.13}$$

A process is strongly self-similar if and only if $\beta = d\gamma(p)/dp = \text{const} \, \forall p$, meaning that the MSS is linear with slope β. For normal diffusion we have P a Gaussian and $\beta = 1/2$. For ballistic motion we have $\beta = 1$. Subdiffusive processes are characterized by $0 < \beta < 1/2$, and superdiffusive processes by $1/2 < \beta < 1$. Any process with non-constant $\beta(p)$ is weakly self-similar.

Despite the fact that the diffusion constant is mathematically not defined if P is a Lévy density, we artificially extend its definition by analogy to Eq. (2.4):

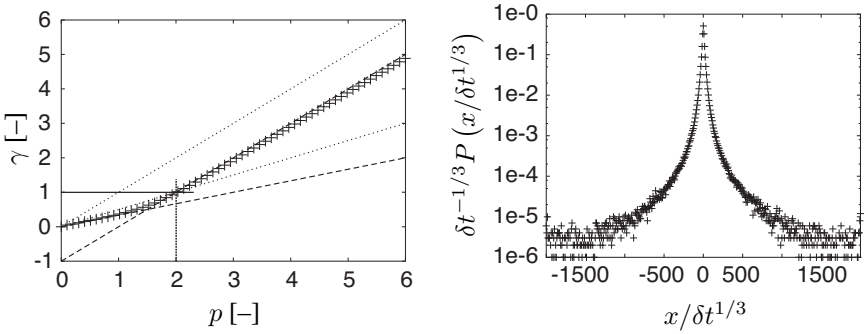
$$\mu_p = p \nu_p d \delta t^{\gamma(p)} \,. \tag{2.14}$$

The diffusion constant of a discrete trajectory is thus obtained from the y axis intercept y_0 of the linear regression of $\log \mu_p(\delta t)$ versus $\log \delta t$ as

$$\nu_p = (dp)^{-1} \cdot \exp(y_0) \,. \tag{2.15}$$

In the case of normal diffusion, ν_2 corresponds to the regular diffusion constant and we often omit the index 2 and simply write ν.

Determining β for an experimentally recorded trajectory provides a systematic characterization of particle motion, enabling more rigorous quantification and classification of biological dispersion processes [96]. To determine the slope β of a measured MSS, a linear least-squares fit is used. We do not constrain the regression to $\gamma(0) = 0$, since this would increase the residual in all weakly self-similar cases, while it is of no effect for strongly self-similar processes. This choice seems



(a) (b)

Figure 2.1: (a) Moment scaling spectrum determined from a Monte Carlo simulation of the generalized telegraph model [101]. The MSS values from the simulation are shown as crosses (+). The dotted lines mark the slopes 1/2 and 1, the dashed lines the asymptotes. The second order moment (MSD) is proportional to time, but the process is weakly self-similar as indicated by the non-constant MSS slope $d\gamma/dp$. (b) The self-similar transition density ($p = 1$) of the simulated process for large δt. The similarity exponent $\beta = 1/3$ corresponds to the slope of the lower asymptote in the MSS. Points marked by crosses (+) are determined from the Monte Carlo simulation.

appropriate as it avoids manifesting the non-universal prior of linearity. An MSS analysis typically combines β and ν_2 in a two-dimensional *phase space* to represent the motion. While ν_2 provides a measure of motion speed, β quantifies the motion type. Representing a trajectory in the (ν_2, β)-plane thus allows to observe both quantities simultaneously, as illustrated in Fig. 2.2.

Besides reduced ambiguity in anomalous diffusion cases, the MSS analysis has the additional advantage of better accuracy. Since higher-order moments are taken into account, the MSS curve is less noisy than the corresponding MSD curve (cf. Fig. 2.2). The linear regression typically used to determine the curve's slope is thus more robust. In addition, the *statistical truncation uncertainty* from finite trajectory lengths is reduced compared to the MSD. Analogous to Eq. (2.6), the variance of the moment of order p is given by

$$\mathrm{var}(\mu_p(\Delta n)) = \frac{(p\nu_p d\Delta n\Delta t)^2}{C}\,, \tag{2.16}$$

with the normalization constant C defined in Eq. (2.7). For the generalized diffusion constants we have

$$\mathrm{var}(\nu_p) = \frac{p\Delta n}{3\,(M-\Delta n)}\,. \tag{2.17}$$

Assuming that ν_p and $\gamma_p = \gamma(p)$ are uncorrelated, we find

$$\mathrm{var}(\log\mu_p) = \mathrm{var}(\log\nu_p) + \log^2\delta t\,\mathrm{var}(\gamma_p)$$

and thus

$$\mathrm{var}(\gamma_p) = [\mathrm{var}(\log\mu_p) - \mathrm{var}(\log\nu_p)]\log^{-2}(\Delta n\Delta t)\,. \tag{2.18}$$

Since all terms in the above expression are non-negative, the variance of γ_p is smaller than the variance of μ_p, making β the more accurate measure than $\gamma_2 = \gamma(2)$. For general motion processes with unknown transition density, it is impossible to express $\mathrm{var}(\log\cdot)$ in terms of the known $\mathrm{var}(\cdot)$. Fig. 2.3 shows the experimentally determined γ_2 and β for trajectories of Polyomavirus particles on the plasma membrane of live cells (cf. Chapter 4). The error bars indicate the standard deviation as determined numerically from the variance of the moments (Eq. (2.16)). The error bars for ν_2 are within the theoretical worst case of 40% for the used $T_f = 5$, as predicted by Eq. (2.8). These results illustrate that the MSS provides a more robust and accurate indicator than the MSD. For the Polyomavirus trajectories in Fig. 2.3, it would be impossible to distinguish a freely mobile virus particle

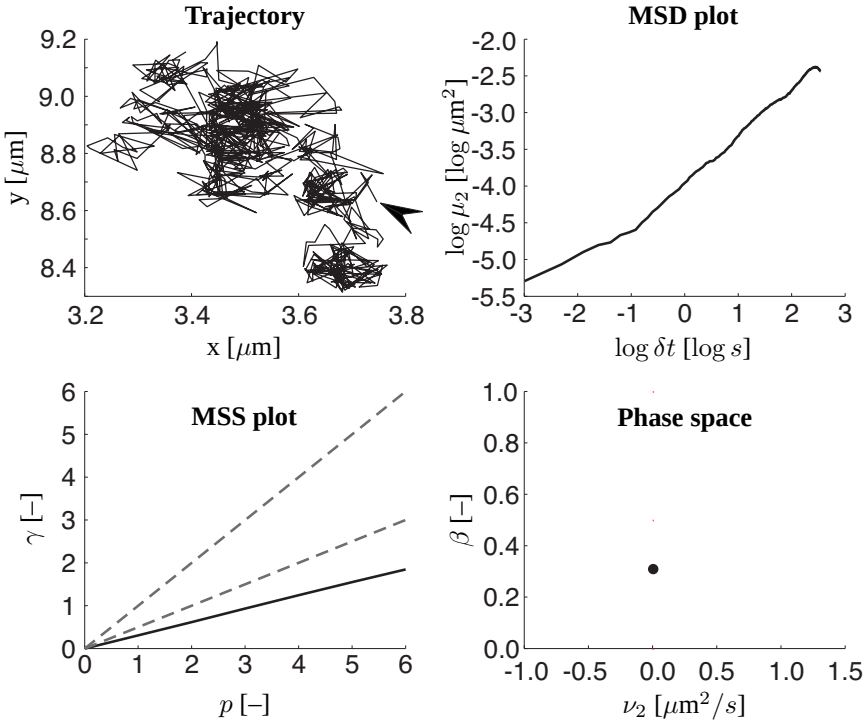
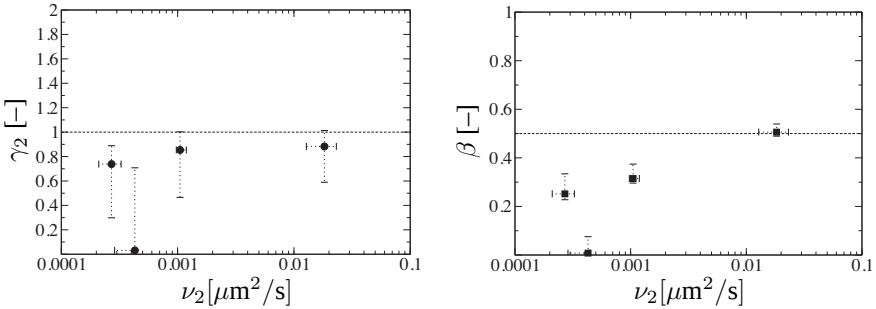


Figure 2.2: Illustration of the moment scaling spectrum analysis. The panel in the upper-left corner shows an experimentally recorded trajectory of an Adenovirus-2 particle on the plasma membrane of an M21 cell (Greber group, arrow head marks trajectory beginning). The mean square displacement according to Eq. (2.5) is shown in the upper-right panel. The moment scaling spectrum as defined in Eq. (2.12) is shown in the lower-left panel with the dashed lines indicating the slopes $1/2$ and 1. The resulting representation of the trajectory in the (ν_2, β) phase plane is shown in the lower-right panel.



(a) (b)

Figure 2.3: Comparison of the statistical uncertainties of MSD analysis and MSS analysis. (a) MSD analysis of Polyomavirus particle motion on the plasma membrane of live 3T6 cells (Helenius group). Error bars indicate the standard deviation according to Eq. (2.16) and numerical calculation of the downstream parameters. The large error bars of the MSD slope make it impossible to accurately classify the motion types. (b) MSS analysis of the same data set. The accuracy of the MSS slope is much better, enabling unambiguous classification.

from a completely stationary one using standard MSD analysis, since the standard deviation bars of the two classes significantly overlap. The MSS analysis, however, enables classification with a very high probability of success. Moreover, β is determined from a linear plot whereas γ_2 is the slope in a logarithmic plot. This means that the MSS analysis linearly weights the different parts of a trajectory, enabling unambiguous classification also in otherwise indistinguishable cases such as the ones depicted in Fig. 4.13.

In the present work we use β to classify different modes of motion or to quantify the "freedom" of a motion. If a particle is confined in a certain region, the process appears subdiffusive, which is robustly detected in β. Combining β and ν_2 provides a classification plane that allows to quantify both the type and the speed of the motion simultaneously. Classical MSD analysis would not allow us to distinguish between a particle that is confined in a moving region and a stationary particle [96]. As shown in Fig. 4.13, the MSS slope adds the crucial second dimension needed to discriminate these cases.

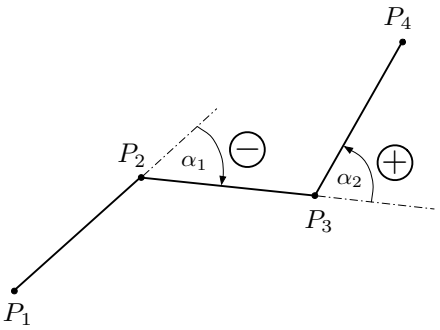


Figure 2.4: Definition of the signed angle α between subsequent steps of a trajectory; α_1 is negative, α_2 positive.

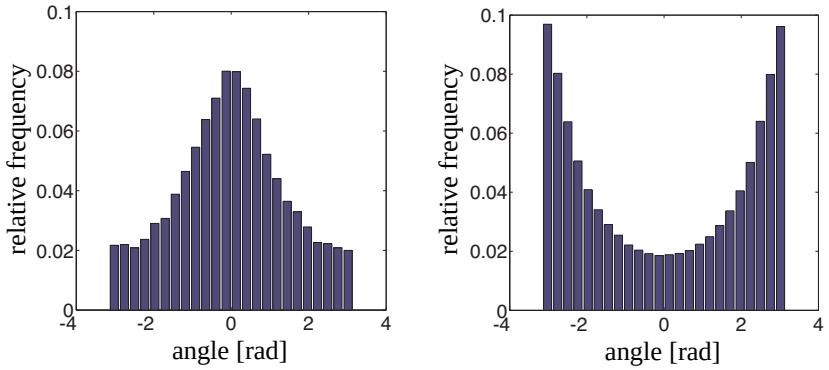
2.1.3 Direction angle histograms

An direct way of quantifying the degree of confinement of a trajectory is to consider the *distribution of angular direction changes* between subsequent displacement steps. The angles are defined by the direction vectors of two adjacent steps as illustrated in Fig. 2.4.

In a Brownian random walk [40], the direction of steps is uniformly distributed as there is no influence of previous steps on the present one (*Markov process*). The histogram of angles α of a random walk thus shows a flat distribution. Confined motion is characterized by a dip in the histogram center. Small angles – leading to larger end-to-end displacements – are less frequent than large angles. This corresponds to the tendency of the particle to "turn around" when it hits the boundary of the region to which it is confined. Superdiffusive motion is associated with a center peak in the angle distribution with forward steps being more probable than turns. This is illustrated in Fig. 2.5 using directed segments and phases of immobility of Adenovirus-2 trajectories (cf. Section 4.3). Since the histograms are built from individual displacements, the number of samples is large, allowing better statistics.

2.2 Moving window analysis

A straightforward way of extending any global analysis method to allow for temporal changes in the quantification parameters consists of applying it in a *moving window*. Suppose the global analysis computes the functional



(a) (b)

Figure 2.5: Distribution of angle changes for different types of motion. (a) shows the histogram for segments of directed motion, (b) for phases of confinement.

$$\Phi(\Delta n) = \frac{1}{M-\Delta n} \sum_{i=0}^{M-\Delta n-1} f(\boldsymbol{x}(i), \boldsymbol{x}(i+\Delta n)) \,. \tag{2.19}$$

Applying it in a *moving window* of length n_w and starting at point k then involves the functional

$$\Phi_k(\Delta n) = \frac{1}{n_w-\Delta n} \sum_{i=k}^{k+n_w-\Delta n-1} f(\boldsymbol{x}(i), \boldsymbol{x}(i+\Delta n)) \;,\; k = 0, \ldots, M-n_w \,. \tag{2.20}$$

Using $f = \|\boldsymbol{x}(i+\Delta n) - \boldsymbol{x}(i)\|_2^p$ generates the spectrum of displacement moments $\mu_p(\Delta n)$ in the moving window frame.

The moving window analysis defines an averaging operator along the trajectory, and only variations of length scales $\mathcal{O}(n_w)$ or larger can be resolved in $\Phi_k(\Delta n)$. Increasing the time resolution comes at the expense of larger statistical uncertainly. This is evident from the error estimates in Eqs. (2.6) and (2.16), since the number of sampling points per window decreases with increasing temporal resolution.

Using the MSS analysis of Subsection 2.1.2 in a moving window allows to detect changes in the motion type within a trajectory. This is important when analyzing

trajectories of unsteady processes as they often occur in biology. Fig. 2.6 shows an example trajectory of an Adenovirus-2 particle on the plasma membrane of a live M21 cell after binding to the receptor (Greber group, Section 4.3). The process is clearly not stationary as the virus seems to pause several times. These *transient confinement zones* can be seen as drops in the moving-window MSS (Fig. 2.6(c)).

2.3 Trajectory segmentation

In biological applications, a trajectory is often composed of a sequence of different types of motion. The examples in Fig. 2.7 illustrate this using trajectories of Polyomavirus particles on the plasma membrane of 3T6 cells (Helenius group, [96]). Many trajectories exhibit complex patterns of transient confinement zones, random motions, and segments of directed transport.

While the moving window MSS analysis provides a means of detecting longer periods of a certain type, it is not appropriate to detect short sequences. Detectable sequences have to be at least of the size of the window as they would be lost in the averaging otherwise. The statistical uncertainty (cf. Eq. (2.16)) however imposes a lower limit to meaningful window sizes.

In trajectories of biological motion, short periods of entrapment or active transport contain important information about the existence of certain molecular mechanisms. In addition, we are often interested in answering questions like *what is the mean residence time in entrapment?*, *what is the diffusion constant of the free motion?*, or *what deterministic velocity is contained in the biased stretches?*.

Both, detecting short sequences and addressing questions of the above type, requires a pattern-based decomposition of the trajectory prior to analysis. This *decomposition*, or *segmentation*, cuts the trajectory into pieces of pre-defined type. The segments of the individual types can then be analyzed independently, without the blurring effect of the moving window average.

The *dynamic classification* problem defined by the above task can be approached using *learning algorithms* [54]. In this section, we use an *artificial neural network* as described in Subsection 2.3.1. This network is trained on trajectories segmented by hand, and is then used on other trajectories of unknown segmentation (Subsection 2.3.2). Particular attention is payed to the *generalizability* of the results from the training data to the real data (Subsection 2.3.4), and the quality of classification is assessed using cross-validation on a disjoint set of *test data* (cf. Subsection 2.3.5).

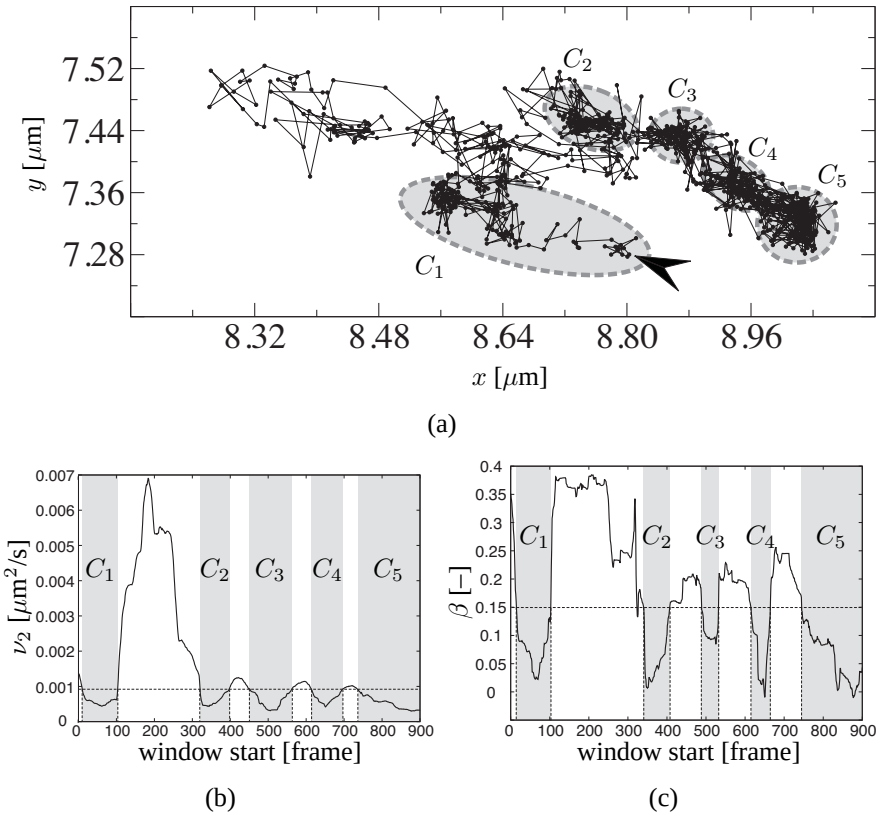
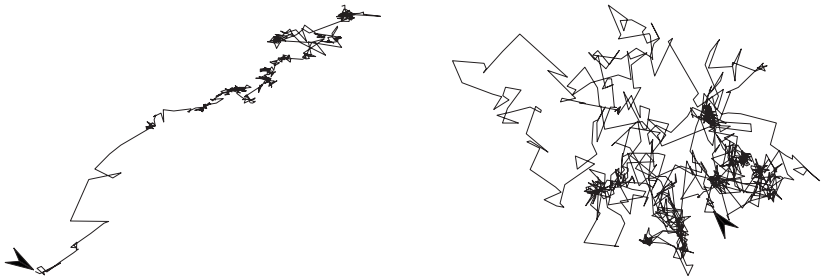


Figure 2.6: Moving window analysis of a trajectory of Adenovirus-2 on the plasma membrane of a live M21 cell (experiments: Greber group). (a) The trajectory recorded at 20 Hz time resolution. The start of the trajectory is indicated by the arrowhead. (b) Diffusion constant in a moving window of width $n_w = 100$ frames. (c) MSS slope in the same windows. Five transient confinement zones – C_1 to C_5 – can be identified as highlighted in the plots.



(a) (b)

Figure 2.7: Two example trajectories of Polyomavirus particles (Helenius group) on the plasma membrane of live 3T6 mouse fibroblast cells. The viruses display complex motion patterns with several transient confinement zones, directed segments, and random walks. Arrow heads mark trajectory beginnings.

2.3.1 Neural networks for classification

In classification problems, artificial neural networks can be used to represent a function that maps the data from a high-dimensional *input space* to a scalar *classification value*, indicating the *class* that the object belongs to [190, 42, 30].

The elements of an artificial neural network are modeled after the image of biological neurons. They mimic cells that communicate with their neighbors using electric signals. Similar to biological neurons, *artificial neurons* have a certain *level of activity*, and defined connections with a set of other neurons. The sum of all signals received over these connections determines the level of activity of the neuron. Once this level exceeds a certain *threshold*, the neuron sends an output signal to all its connected neighbors. Depending on the connectivity structure, a network of artificial neurons can represent certain families of *classification functions* between the input (data) space and the classification value. The particular function does need to be explicitly known. Rather, the network is trained to "learn" it from a set of *training data* with known classification.

The standard artificial neuron as depicted in Fig. 2.8 consists of a *transfer function* to which other neurons are connected. The *input level* H of the neuron is

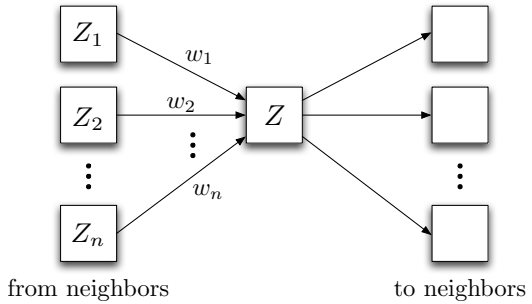


Figure 2.8: Transfer function model for artificial neurons.

computed from the activities Z_i of its upstream neighbors by the weighted sum

$$H = \sum_{i=1}^{n} w_i Z_i \,. \tag{2.21}$$

The set of *weights* $\{w_i\}$ determines the particular classification function that is represented by the network. These weights are determined during the *training phase*. The level of activity Z of each neuron is computed as a function of its input level H as

$$Z = f(H, \Theta) \,. \tag{2.22}$$

This involves a particular *activation function* f which determines the transfer characteristic of the neuron. Typical choices of activation functions are the *step function* $2\mathcal{H} - 1$ or any *sigmoid* smoothly increasing from -1 to 1. If the value of the activation function exceeds a fixed threshold Θ, the neurons sends its level of activity Z to all downstream neighbors.

A neural network can consist of any number of neurons with any interconnections. Typically, the neurons are organized in *layers*. The simplest structure consists of two layers: input neurons and output neurons. In layered networks, the neurons are only connected to neighbors in other layers, but not within a layer.

Special case: the multi-layer perceptron

Multi-Layer Perceptrons (MLP) are a special type of layered neural networks, characterized by a one-way signal flow [30]. Information is only propagated in the

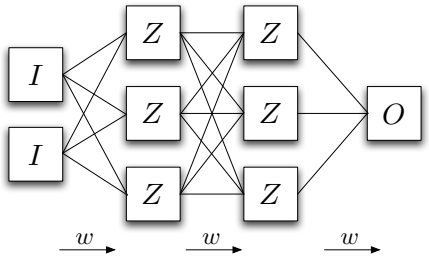


Figure 2.9: Example structure of a multi-layer perceptron. Information is only transmitted from left to right. The activities of the input neurons "I" are defined by the input data, whereas the activities of the neurons in the intermediate layers "Z" and in the output layer "O" result from the activities of their respective upstream neighbors according to Eqs. (2.21) and 2.22. Each connection i is characterized by an independent weight w_i.

forward direction, from the input layer to the output layer. Neither feed-back loops nor connections within the same layer are present in the network (cf. Fig. 2.9).

2.3.2 Learning decision boundaries

The *decision boundary* separates data of one class from data of other classes. Geometrically, the boundary is a manifold of *co-dimension* one in the input data space. A good decision boundary separates data from different classes without errors. The goal is to attain this property not only for the training data, but for all possible future data that are not known a-priori. This trade-off between generality and training performance is discussed further in Subsection 2.3.4.

In a neural network classifier, the decision boundary is implicitly defined by the connection weights w_i as learned on the training data. For linear activation functions f, a two-layer network implements a *linear classifier* and is only able to distinguish data from two classes if they are separable by a linear function, i.e. a hyper-plane in the input space. More complex decision boundaries can be realized using additional layers between the input and the output layers, or by non-linear activation functions f. One intermediate layer allows convex decision boundaries, two layers are sufficient to represent arbitrary decision boundaries.

A common way to *learn* the weights is to define a *loss function*, e.g. the sum of incorrectly classified points, that is to be minimized over the training data set. The weights w_i are then found by a standard *minimization algorithm*. In the case

of a linear classifier, there is a single unique minimum which is easily found by a gradient descent. In non-linear classifiers, several local minima may exist and a global optimizer is needed.

In the case of MLPs, an efficient training algorithm exists for arbitrary network structures. This so-called *backpropagation algorithm* [232] exploits the feed-forward structure of the network to find the optimal weights by adjusting them in the backward direction.

2.3.3 Selection of training data

The training set must be sufficiently diverse to contain all possible forms of motion from a certain class. We thus use real trajectories from the specific application to train the classifier, rather than using artificially generated samples. Samples of all allowed lengths have to be present in the training data, and particular care is taken that short segments can unambiguously be attributed to a class. This ensures that the segmentation is not misled by statistical fluctuations in random walks.

A second requirement concerns the size of the training set. For each input neuron > 10 samples are needed [30]. The effect of the size of the training set on the learning success is illustrated in Fig. 2.14.

2.3.4 Model selection

The complexity of the decision boundary is limited by the number of layers and the number of connections (weights). According to *Occam's razor*, the decision boundary should be as simple as possible to achieve good classification performance on data that are not contained in the training set. The lower bound for the simplicity is given by the desired performance on the training data, i.e. the tolerance used in the minimization algorithm when learning the connection weights. Completely adapting the decision boundary to the training data often leads to loss of generalization. This phenomenon is called *overfitting*. Its opposite, *underfitting*, refers to the situation where the decision boundary is too simple and both training performance and generalization degrade. This trade-off is depicted in the illustration in Fig. 2.10.

2.3.5 Cross-validation

A good method to avoid overfitting is to stop learning once the generalization capability of the classifier has reached a certain level. The generalization capability

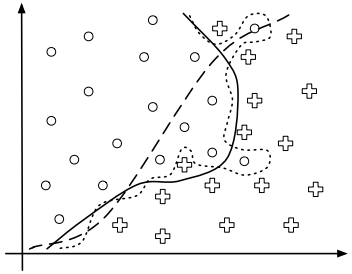


Figure 2.10: Planar distribution of sample data of two classes (crosses and circles). The decision boundary given by the dotted line corresponds to the situation of overfitting, the dashed line illustrates underfitting. A good decision boundary is depicted by the solid line.

can be estimated by disjointly dividing the data with known classification into a *training set* and *test set*. The network is trained using the training set and then evaluated on the test set. The mean classification error over k different partitionings of the data is used as an estimate for the *generalization capability*. Increasing mean errors in this k-fold *cross-validation* indicate overfitting.

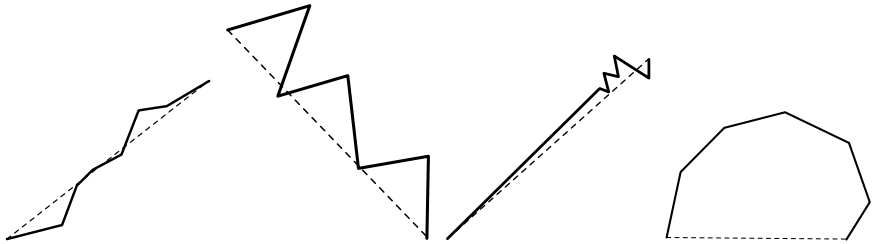
2.3.6 Segmentation of directed motion

We wish to detect segments of fast *directed motion* or "flights" (cf. Subsection 2.1.2) in a trajectory. Hereby, we are particularly interested in short segments (shorter than $10\Delta t$) that are not detectable in a moving window MSS analysis.

Criteria

Since there is no a-priori definition of what "directed" means, we have to invoke an *operational definition* by means of a set of criteria derived from expert knowledge. The goal is to reproducibly quantify the criteria used by a human domain expert to segment a trajectory. To construct an effective classifier, the criteria have to be independent of direction and scale and invariant with regard to rigid-body rotations and translations.

The term "directed" implies that the segment is composed of steps that are roughly headed in the same direction, Fig. 2.11(a). We thus choose the sum of the angles between the individual steps as the first criterion. This criterion is not sufficient as it would result in misclassification of the cases shown in Figs. 2.11(c)



(a) (b) (c) (d)

Figure 2.11: Sample segments for classification criteria (see text). The dashed line indicates the net path.

and (d). Additional criteria, relating the net displacement to the lengths of the individual steps, are needed to capture them. For a trajectory segment consisting of n points the following three criteria are found to yield good classification performance for directed motion:

- **linearity:**

$$\frac{1}{n-2}\sum_{i=1}^{n-2}\cos(\alpha_i)\,, \tag{2.23}$$

  where α_i denotes the angle between step i and $i+1$ as defined in Fig. 2.4.

- **relative net displacement:**

$$\frac{\|\boldsymbol{x}(n)-\boldsymbol{x}(1)\|_2}{\langle\|\boldsymbol{x}(i)-\boldsymbol{x}(i-(n-1))\|_2\rangle_{i=1\ldots n}}\,, \tag{2.24}$$

  where $\boldsymbol{x}(i)$ denotes the i^{th} position in the segment.

- **efficiency:**

$$\frac{\|\boldsymbol{x}(n)-\boldsymbol{x}(1)\|_2^2}{(n-1)\sum_{i=1}^{n-1}\|\boldsymbol{x}(i+1)-\boldsymbol{x}(i)\|_2^2}\,. \tag{2.25}$$

Using above criteria, the n-dimensional trajectory data are mapped onto a three-dimensional input data space for the classifier.

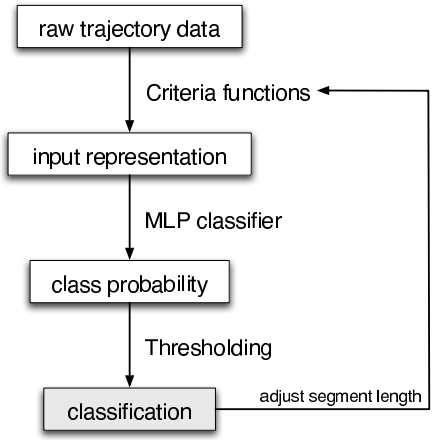


Figure 2.12: Flow chart of the trajectory segmentation algorithm.

Algorithm

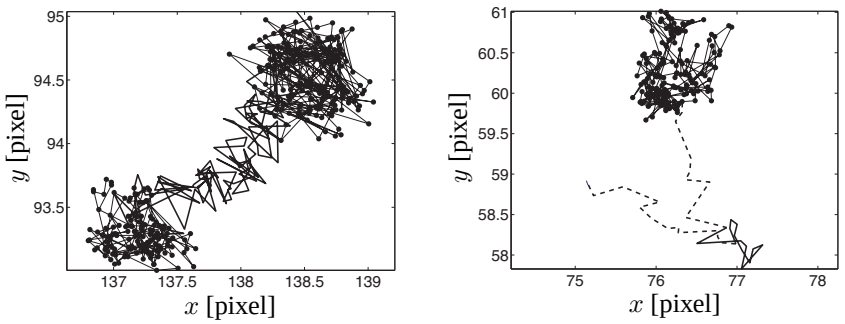
The MLP for the detection of directed motion has four input neurons, four neurons in an intermediate layer, and one output neuron. All neurons use the same activation function (cf. Subsection 2.3.1), namely the sigmoid

$$f(H) = \frac{1}{1 + \exp(-H)} \,. \tag{2.26}$$

The structure of the complete algorithm is shown in Fig. 2.12. The first step consists of computing the criteria given by Eqs. (2.23)–(2.25). This is done for all possible segments, yielding $M - (n-1)$ sets of criteria.

The computed criteria constitute the first three input values of an MLP that is trained to recognize the signature of directed motion. The fourth input is the length n of the segment. The scalar output of the MLP represents the estimated probability that the segment shows directed motion. Outputs above 0.9 are classified as directed.

The whole procedure is repeated for different segment lengths $n = 5, \ldots, 10$ to detect directed stretches of different durations.



(a) (b)

Figure 2.13: Example trajectories segmented by the present algorithm (data: Jo Helmuth). Dashed lines correspond to identified directed segments, solid lines mark diffusive segments, and thin lines with dots correspond to confinement zones.

Performance

Fig. 2.13 shows typical segmentation results using example trajectories of Adenovirus-2 particles on the plasma membrane of live M21 cells (cf. Section 4.3). Visually, the classification is in good agreement with expert knowledge in all of the cases.

The convergence of the fraction of misclassification (*risk*) during typical learning phases is shown in Fig. 2.14 for different sizes of the training set. 23 samples are not sufficient to learn the weights and the classifier suffers from overfitting as described in Subsection 2.3.4. Increasing the training set to 37 samples allows perfect classification of the training data. Further increases of the set size result in faster convergence. For all following applications, we use a training set of 100 samples.

Convergence of the cross-validated classification error during a typical training phase is shown in Fig. 2.15(a). While the training set is perfectly classified, the error on the test set remains around 18%.

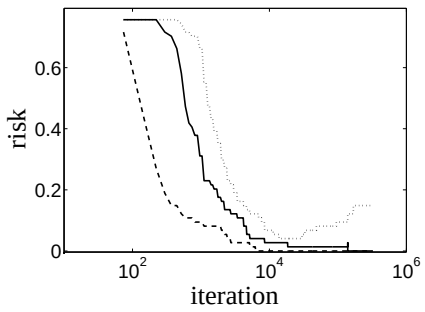


Figure 2.14: Dependence of the learning process on the size of the training set for directed motion (data: Jo Helmuth). While 23 samples are not sufficient (dotted curve), 37 samples allow perfect classification (solid curve). Increasing the size to 74 training samples results in faster convergence (dashed curve).

2.3.7 Segmentation of arrest zones

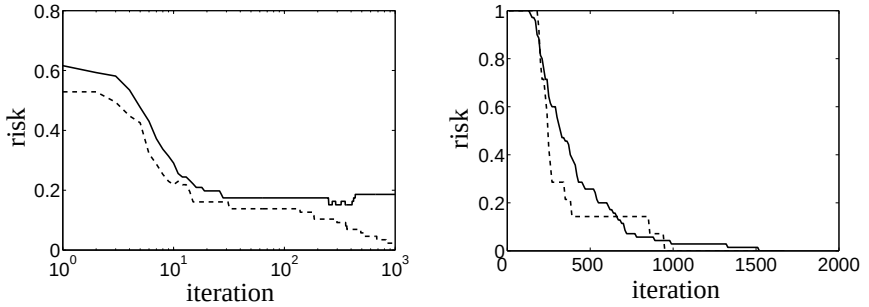
Criteria

In *arrest zones*, the particle is tightly confined. Arrest zones are thus the opposite of directed segments, such that the same three criteria as given by Eqs. (2.23)–(2.25) are used.

Algorithm

The MLP for immobility detection has three input neurons, corresponding to the three criteria. It has three neurons in the intermediate layer, and one output neuron. The activation function for all neurons is given by Eq. (2.26). The structure of the segmentation algorithm remains unchanged as shown in Fig. 2.12.

Random walks can only be discriminated from arrest zones if their diffusion constant is large enough to cause particle motion larger than the tracking uncertainly. Very slow random walks within the imaging noise are indistinguishable from arrest zones. The segment lengths used in the algorithm are thus chosen much larger than for the directed motion, i.e. $n = 40, \ldots, 100$.



(a) (b)

Figure 2.15: Performance of the MLP for classification of directed motion (a) and phases of immobility (b) (data: Jo Helmuth). The fraction of misclassification (risk) is shown for both the training set (dashed) and the test set (solid) in a cross-validation.

Performance

A typical result of the segmentation is shown in Fig. 2.13. Convergence of the cross-validated classification error during a typical training phase is shown in Fig. 2.15(b). While the training set is perfectly classified, the error on the test set remains around 2%.

2.4 Event-based trajectory analysis

Having a classification of trajectory segments as outlined in the previous section enables searching the trajectory for specific *events*, defined as particular sequences of segments. While detection of such events is straightforward, the interpretation of the resulting counts requires normalization with the expected number of events under purely random conditions.

In this section we present three different ways of normalization: an analogy to chemistry, a Monte Carlo simulation, and an analytic way. They are exemplified using two different events: a “sit-down” event, consisting of a segment of directed motion followed by an arrest zone, and a “pass-by” event of a directed segment of one trajectory passing by an arrest zone of another one. The conceptual difference is that the sit-down event is defined in a single trajectory whereas the pass-by event involves a pair of trajectories.

2.4.1 "Sit-down" event

The intra-trajectory *sit-down event* is defined as a segment of directed motion that is followed by a phase of immobility within the next five frames. The absolute count of such events depends on the number of segments of either class, and does therefore require normalization.

Analogy to chemistry

The situation is analogous to a *second order chemical reaction* of two reactants. By [d.m.] and [p.o.i] we denote the "concentrations" of segments of directed motion and phases of immobility, respectively. The "rate constant" r of successful encounters in a single trajectory is proportional to the product of the concentrations, viz.

$$r = k \cdot [\text{d.m.}] \cdot [\text{p.o.i}] \,. \tag{2.27}$$

The *concentrations* are naturally defined as the number of classified segments per unit trajectory length, thus:

$$[\text{d.m.}] = \frac{\#\{\text{d.m.}\}}{\#\{\text{trajectories}\} \cdot \langle M_\ell \rangle_\ell} \tag{2.28}$$

$$[\text{p.o.i.}] = \frac{\#\{\text{p.o.i.}\}}{\#\{\text{trajectories}\} \cdot \langle M_\ell \rangle_\ell} \,, \tag{2.29}$$

where $\langle M_\ell \rangle_\ell$ is the mean trajectory length. The number of events is given by the rate r and the number of attempts as:

$$\#\text{events} = r \cdot \langle M_\ell \rangle_\ell \cdot \#\{\text{trajectories}\} = k \cdot \frac{\#\{\text{d.m.}\} \cdot \#\{\text{p.o.i.}\}}{\#\{\text{trajectories }\} \cdot \langle M_\ell \rangle_\ell} = k \cdot p. \tag{2.30}$$

This allows to determine k from the observed number of events. The value of k is used as the *normalized event count*.

Monte Carlo simulations

The number of expected events under random conditions can be determined by a Monte Carlo simulation. Hereby, the segments of a trajectory are repeatedly randomized and the average number of observed events in the *randomized trajectories* is used to normalize the count in the original trajectory.

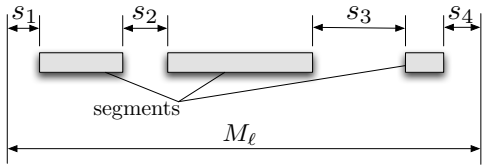


Figure 2.16: Situation for the randomization algorithm. Three segments are placed, giving rise to at most four random stretches s_1 to s_4.

The length of the randomized trajectories is fixed and equal to the length of the original trajectory. The directed and immobile segments do generally not cover the whole trajectory, as they may be separated by random phases s_i as shown in Fig. 2.16. In the Monte Carlo simulations, the separation phases s_i are randomly determined from uniformly distributed random numbers $\mathcal{U}_i$ between 0 and 1 as

$$s_i = \frac{\mathcal{U}_i \cdot \sum s_i}{\sum \mathcal{U}_i} \,. \tag{2.31}$$

Mean and variance of the number of expected events are determined from many realizations of randomized trajectories. The number of randomized trajectories used to compute the mean is determined by a prescribed target variance. The *normalized count* k is computed from the count in the original trajectory and the mean event count from all randomized realizations as

$$k = \frac{\#\{\text{counted}\}}{\langle \#\{\text{events in randomized}\} \rangle} \,. \tag{2.32}$$

Compared to the chemical analogy, the Monte Carlo simulation has the advantage of straightforward generalization to systems with larger numbers of "reactants".

2.4.2 "Pass-by" event

The inter-trajectory *pass-by event* is defined as a directed segment passing by another particle that is immobile during the time of passage. *Passage* is defined by the distance between the two particles being below a certain interaction threshold.

The segments from the trajectories are arranged in a three-dimensional space, where two dimensions correspond to physical space and the third one to time. A pass-by event is characterized by the center of one object being within the *interaction sphere* of another object.

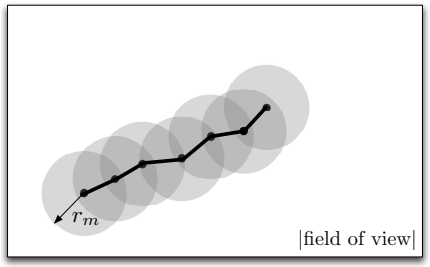


Figure 2.17: A segment of directed motion with the interaction area defined by circles around its nodes. Any phase of immobility within a circle corresponds to a pass-by event in space if the two segments are from different trajectories.

The total number of counted pass-by events in a set of trajectories depends on the number of segments and the lengths of the trajectories, and therefore needs to be normalized.

Estimator for the number of expected events

The expected number of pass-by events under uniformly random conditions is given by the *expectation value*

$$\mathsf{E} = \#\{\text{d.m.}\} \cdot \#\{\text{p.o.i.}\} \cdot p_{\text{close}} \,. \tag{2.33}$$

The numbers of directed segments and phases of immobility are counted over the whole set of trajectories, and p_{close} is the probability of a directed segment being close to an immobile phase in space and time.

Assuming statistical independence, the probability p_{close} is computed from the probability of encounters in space (p_s) and in time (p_t) as $p_{\text{close}} = p_s p_t$. The value of p_s can be estimated from the average length of directed segments and the interaction radius r_m as illustrated in Fig. 2.17. The average *interaction area* around directed segments is hereby approximated as

$$\langle A_m \rangle = \langle M_{\text{directed},\ell} \rangle_\ell \cdot 2r_m + \pi r_m^2 \,, \tag{2.34}$$

where the average is taken over all directed segments in the set of trajectories. Assuming a uniform distribution of directed segments in space, the probability of

an encounter in space hence becomes

$$p_s = \frac{\langle A_m \rangle}{|\text{field of view}|} = \frac{\langle M_{\text{directed},\ell} \rangle_\ell \cdot 2r_m + \pi r_m^2}{|\text{field of view}|} \,. \tag{2.35}$$

The temporal distribution of segments can not be assumed to be uniform over the time of observation. Let t_d and t_a be the starting times of a directed segment or an arrest zone, respectively. The probability p_t can be computed from the probability distributions $P(t_d)$ and $P(t_a)$ by integration over the time interval T of a spatial encounter, thus

$$p_t = \int_T \int_T P(t_d) P(t_a) \, dt_d \, dt_a \,. \tag{2.36}$$

The probability densities $P(t_d)$ and $P(t_a)$ are estimated from the relative frequencies in the data and represented as 4^{th} order interpolation polynomials for integration.

The normalized count

$$k = \frac{\#\{\text{counted}\}}{\mathsf{E}} \tag{2.37}$$

is computed from the actual count and the expected number E of pass-by events in uniformly random trajectories according to Eq. (2.33).

Chapter 3

Trajectory Classification

Automatic classification of trajectories is an important part of high-throughput and bias-free motion analysis. The term *bias-free* refers to the absence of selection of "representative" trajectories by a human experimenter. The goal of bias-free data analysis is to provide a reproducible algorithm and to achieve statistical significance by large sample sizes.

The classification of trajectories using machine learning techniques is complicated by several factors: Trajectories constitute dynamic data, viz. ordered time series of position vectors. Data encoding is needed to exploit the temporal information and to reduce the dimensionality of the data. Moreover, trajectories exhibit multiple invariances with regard to translation, rotation, and symmetry, as rigid-body rotations or translations of a trajectory leave the motion patterns unchanged. In order to effectively extract these patterns, encoding and classification methods have to be robust against invariances.

The application of machine learning techniques for automatic classification of trajectories mainly serves three scientific goals: First, one wishes to identify the biological or physical processes underlying the observed phenomenon by relating changes in the trajectory to perturbations in the experimental conditions (*causality detection*). Second, the information contents of a given trajectory with respect to a certain property of interest may be estimated (*capacity estimation*) and third, automatic identification and classification of vast amounts of experimental data can facilitate the process of interpretation (*data mining*).

Model-free classification of trajectories traditionally uses neural networks, e.g. to find invariant patterns within trajectories [157], or to classify encoded trajectories in feature space [204]. More recent approaches make increasing use of *self-organizing maps* [158] for trajectory classification. Applications include the detection of suspicious activity in trajectories of pedestrians as recorded by surveillance cameras [208] or the classification of user activity from the traces in a virtual world [238]. *Fuzzy logic* and maximum likelihood *Hidden Markov Models* (HMM) have also been successfully applied to the classification of pedestrian trajectories [183]. Other applications include biologically inspired motion classification schemes using vector quantization [147], speech recognition using mixture models of trajectories [114], and vehicle trajectory classification using HMM with binned data encoding [104]. Automatic trajectory classification also has emerging applications in credit card fraud detection [182], economics [143], and financial markets [144, 257].

In this thesis, we consider applications of machine learning techniques to the supervised classification of biological trajectories, in order to assess the suitability of various methods for biological applications [247], and to provide an automatic procedure for optimal data encoding. We first introduce the problem of trajectory classification using the particular example of keratocyte cell motility. After formalizing the problem, we assess the performance of a number of standard classification algorithms [54] on this application. The algorithms as described in Appendix B are compared among each other as well as to human classification using a separation measure from signal detection theory [120], that is introduced in Subsection 3.1.4.

The representation of the trajectories in data space plays a central role in all motion classification problems. Finding a set of highly discriminative characteristics is not only required for efficient classification, but also contains important information about the physical process that created the trajectory. In Section 3.2, we consider the problem of automatically finding an optimal set of characteristics for any given classification problem. This is done by using a self-optimizing data encoder which is adjusted to maximize the classification quality. The concept is demonstrated using the keratocyte motion data and compared to the handcrafted encoding of Section 3.1.

3.1 Automatic classification of keratocyte trajectories

We employ various machine learning techniques to the task of automatically classifying trajectories of moving *keratocyte* cells. The different algorithms as described in Appendix B are compared among each other as well as to expert and non-expert test persons using concepts from *signal detection theory* [120]. We find the algorithms to perform well when compared to humans, suggesting a robust tool for trajectory classification in biological applications.

We start by describing the sample data used in this section, and proceed to formally stating the problem of classification. The different machine learning tech-

niques are then assessed starting from clustering methods in the d-dimensional real space $\mathbb{R}^d$, proceeding to risk-optimal separation in $\mathbb{R}^d$, and dynamic signal source models in $\mathbb{R}^d \times \mathbb{T}$, where $\mathbb{T}$ denotes the discrete ordered time space.

3.1.1 Keratocyte trajectory data

Experimentally recorded trajectories of live cells moving on glass coverslips are used to test and assess the different classification algorithms. The cells are epidermal *keratocytes* (Theriot group, Stanford Medical School), taken from the scales of the fish *Gillichthys mirabilis*, commonly called "longjawed mudsucker". Isolated cells are cultured on glass coverslips and observed using an inverted phase-contrast microscope connected to a video camera. The two-dimensional trajectories of the cells are extracted from the videos with a sampling time of Δt =15 s. The observations $\boldsymbol{x}$ are the position/time points along the trajectories. The raw data space thus is $\mathcal{X} = \mathbb{R}^2 \times \mathbb{T}$.

Two different experiments are performed: For the *temperature data set*, fish are acclimated at 16°C, i.e. they are kept in water of this temperature for at least 3 weeks[1] prior to cell isolation. The movement of the isolated cells is then recorded at 10°C, 20°C, and 30°C using a temperature-controlled microscope stage. $N = 138$ trajectories (46 at each of the three observation temperatures) from 60 different cells are collected (shown in Fig. 3.1).

For the *acclimation data set*, all cells are observed at 20°C, but they are taken from three different fish populations acclimated at 10°C, 16°C, and 25°C, respectively. From this acclimation data, $N = 174$ trajectories (58 for each acclimation temperature) of 60 different cells are recorded (shown in Fig. 3.2). Both data sets contain samples from $n = 3$ classes.

Our hypothesis states that the reaction rates of the biochemical processes that contribute to cell motility depend on temperature. Temperature is therefore suspected to influence the motion. Using automatic classification, we want to address the questions:

- Can the trajectories recorded at different temperatures be distinguished? (temperature data)
- Are there persistent adaptations to temperature that are "remembered" by the cells? (acclimation data).

[1] After this time, the adaptive changes in liver lipid content are complete.

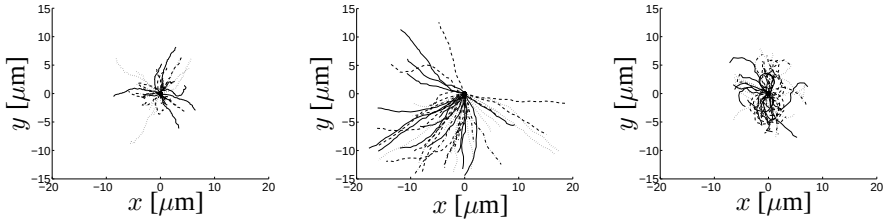


Figure 3.1: Temperature data set. 46 trajectories of moving keratocytes are used per class. The classes are defined by the three temperatures at which the observations are taken. All trajectories are shifted such that they start at the origin of the coordinate system.

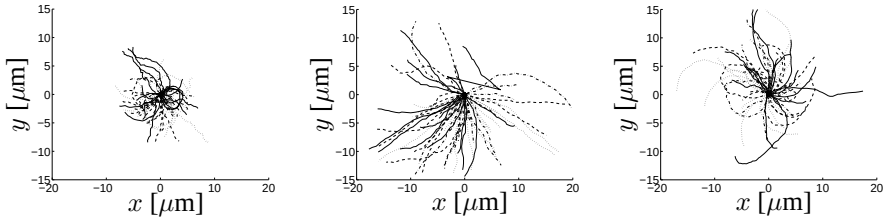


Figure 3.2: Acclimation data set. 58 trajectories of moving keratocytes are used per class. The classes are defined by the three temperatures at which the fish were acclimated for 3 weeks prior to the experiment. All trajectories are shifted such that they start at the origin of the coordinate system.

3.1.2 The classification problem

Given some sample trajectories from n classes, we wish to assign any new, previously unseen, trajectory to the proper class. The problem can be formalized as follows: We are given m empirical data points

$$(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m) \in \mathcal{X} \times \mathcal{Y}\,, \tag{3.1}$$

that are independent and identically distributed ($i.i.d.$) realizations of an unknown probability distribution $P(\boldsymbol{x}, y)$. $\mathcal{X}$ is the non-empty set from which the *observations* (sometimes called *patterns*) are taken and $\mathcal{Y} = \{1, \ldots, n\}$. The $y_i \in \mathcal{Y}$ are called *labels* and specify the class a particular pattern belongs to. Classification aims at *generalization* to unseen data points $\boldsymbol{x}$ with unknown labels y. We want to predict the $y \in \mathcal{Y}$, given some new observation $\boldsymbol{x} \in \mathcal{X}$. Formally, this amounts to estimating a function $f : \mathcal{X} \mapsto \mathcal{Y}\,, \boldsymbol{x} \to y = f(\boldsymbol{x})$, such that f optimally classifies unseen patterns $\boldsymbol{x} \in \mathcal{X}$. The criterion of optimality is to minimize the *expected risk*, which is the expectation value of the fraction of misclassified samples

$$R[f] = \int_{\mathcal{X}\times\mathcal{Y}} l\left(f(\boldsymbol{x}), y\right)\, dP(\boldsymbol{x}, y)\,, \tag{3.2}$$

where l denotes a suitably chosen *loss function*. A common choice is the *0/1-loss*, for which $l(f(\boldsymbol{x}), y)$ is 0 if $(\boldsymbol{x}, y)$ is a correct classification and 1 otherwise. The expected risk can not be minimized directly, since the underlying probability distribution $P(\boldsymbol{x}, y)$ is unknown. *Machine learning* algorithms thus approximate $R[f]$ based on the available information from the input-output *training data* of Eq. (3.1). The most common approximation is the *empirical risk*

$$R_e[f] = \frac{1}{m}\sum_{i=1}^{m} l\left(f(\boldsymbol{x}_i), y_i\right)\,. \tag{3.3}$$

Classification algorithms are distinguished by the different approximations to Eq. (3.2) they use, and the different methods employed to minimize these approximations.

3.1.3 Trajectory encoding

For most classification algorithms of Appendix B, the trajectories $\boldsymbol{x} \in \mathbb{R}^2 \times \mathbb{T}$ need to be transformed to vectors in $\mathbb{R}^d$. This *encoding* determines the data representation seen by the classifier and is of great importance to the classification process.

For *Hidden Markov Models* (HMM), we encode the trajectories by their momentary speed of motion, which is discretized into four equidistant bins for discrete HMMs (dHMM). One HMM is trained for each of the 3 classes. After evaluating the probability $P\left[\boldsymbol{x}|\Lambda_i\right]$ of a new observation $\boldsymbol{x}$ against the models Λ_i for all classes $i = 1, 2, 3$, $\boldsymbol{x}$ is assigned to the class that has the highest probability.

For all other algorithms, we find a good encoding by considering the following catalog of *characteristics*:

- average speed,
- standard deviation of speed,
- mean angle of direction change between 2 subsequent steps,
- standard deviation of those angles,
- net distance between first and last point of trajectory compared to the total path length,
- decay of the autocorrelation functions of speed and direction angle change, and
- minimum and maximum of speed and angle change.

Histograms of the distributions of these properties among the different classes of trajectories give evidence about the discrimination capability of each characteristic. For the present data, we find the mean and the minimum of the speed of a trajectory as good *encoding features*, allowing to represent the trajectories as vectors in $\mathbb{R}^2$. Fig. 3.3 shows the encoded data sets for both the temperature and the acclimation cases. It can be seen that the clusters mostly overlap, making the data *non-separable* in this encoding space.

3.1.4 Assessment of classification performance

Let $\mathcal{D} = \{(\boldsymbol{x}_j, y_j),\ j = 1, \ldots, N\}$ be the complete data set of all N recorded trajectories $\boldsymbol{x}_j$ with corresponding class labels y_j, a random $\mathcal{T} \subset \mathcal{D}$ with $\#\{\mathcal{T}\} = m$ the *training set*, and $\mathcal{E} \subset \mathcal{D}$ with $\#\{\mathcal{E}\} = N - m$ and $\mathcal{E} \cap \mathcal{T} = \emptyset$ the *test set*. An algorithm, trained on $\mathcal{T}$, classifies the trajectories $\boldsymbol{x}_j \in \mathcal{E}$ without knowing the correct y_j. The outcome of this classification is $\widetilde{y}_j$. The *hit rate* for class i is defined as $h_i = \#\{\boldsymbol{x}_j \in \mathcal{E} : \widetilde{y}_j = y_j = i\}/\#\{\boldsymbol{x}_j \in \mathcal{E} : y_j = i\} \in [0, 1]$. The *false alarm rate* (also called "false positives") for class i is given by $f_i = \#\{\boldsymbol{x}_j \in \mathcal{E} : \widetilde{y}_j = i \wedge y_j \neq i\}/\#\{\boldsymbol{x}_j \in \mathcal{E} : y_j \neq i\} \in [0, 1]$. The complementary quantities $m_i = 1 - h_i$ and $r_i = 1 - f_i$ are termed *miss rate* and *correct rejection rate*, respectively. In each classification experiment, both the hit rate and the false

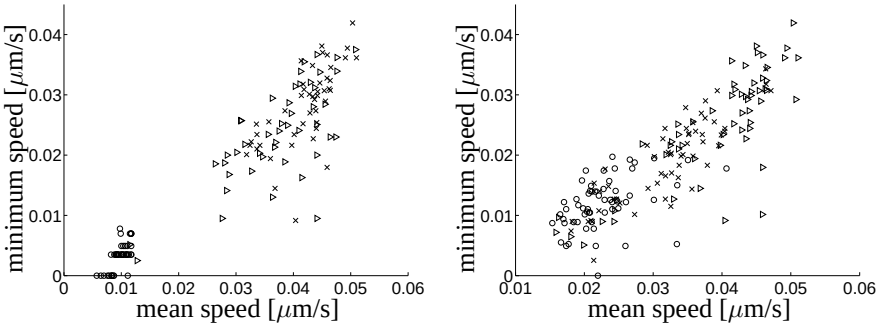


Figure 3.3: Encoded data sets. Both the temperature data set (left) and the acclimation data set (right) are encoded using the mean and the minimum of the speed along a trajectory. Data points from the 10°C temperature and 10°C acclimation classes are denoted by circles (○), those from the 20°C temperature and 16°C acclimation classes by triangles (△), and those from the 30°C temperature and 25°C acclimation classes by crosses (×).

alarm rate are recorded for each temperature class as they represent the minimal sufficient set of quality measures.

The different classification algorithms of Appendix B are trained on a subset $\mathcal{T}$ of $m = N/2$ data points from each class and then tested on the remainder of the data. For the KNN (Appendix B.1) we set $k = 5$, and for the SVM (Appendix B.3) a Gaussian kernel with standard deviation $\sigma = 0.05$ is used. The procedure is repeated 4 times for different partitionings of the data into training and test sets (*cross-validation*, cf. Subsection 2.3.5).

Measuring the classification quality

To quantify the *discrimination capability* of a classifier, we use the *theory of signal detection* [120], which was originally developed in psycho-physics and is widely used in experimental psychology. In this theory, the occurrences of observations that belong to class i and observations that do not belong to class i are assumed to be governed by two different Gaussian probability distributions as illustrated in Fig. 3.4. During training, the classifier learns a threshold C above which it assigns all observations to class i. If, after transformation to standard normal distributions, $C = 0$, the classifier is said to be *neutral*, for $C < 0$ it is called *progressive*, and

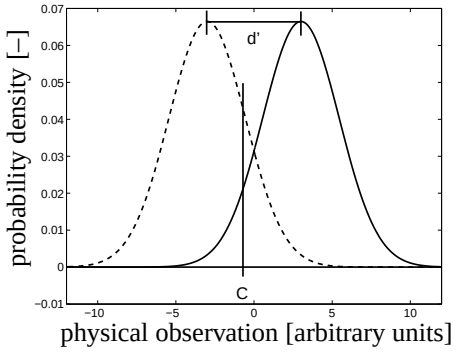


Figure 3.4: Schematic of the theory of signal detection. Observations that belong to a class i occur with a certain probability density (solid); observations that do not belong to that class occur with a different probability density (dashed). The classifier chooses a threshold C and assigns all observations above C to class i. The discrimination capability of the classifier is given by the normalized distance measure d' between the two density functions.

for $C > 0$ *conservative*.

The *discrimination capability* of the classifier is given by the separation distance d' between the two normalized (by their standard deviation) distributions. A value of $d' = 0$ corresponds to uniformly random guessing, where hits and false alarms grow at equal rates, and $d' \to \infty$ characterizes a perfect classifier.

In Fig. 3.4, the hit rate h_i corresponds to the area under the solid curve above C, and the false alarm rate f_i is the area under the dashed curve above C. For each class i, both C_i and d'_i can thus be computed from h_i and f_i as

$$d'_i = \sqrt{2}\left(\text{erf}^{-1}(2h_i - 1) - \text{erf}^{-1}(2f_i - 1)\right) \tag{3.4}$$

$$C_i = -\frac{1}{\sqrt{2}}\left(\text{erf}^{-1}(2h_i - 1) + \text{erf}^{-1}(2f_i - 1)\right) . \tag{3.5}$$

Classifiers are compared based on d', since algorithms that are capable of better separating the two probability distributions have a lower expected risk R.

Performance on the temperature data set

The temperature data set, introduced in Subsection 3.1.1, is classified using all the algorithms outlined in Appendix B. The results are evaluated according to the

class	h_i [%]	f_i [%]	d'	C
10°C	100.0	2.2	∞	–
20°C	54.4	24.5	0.8	0.29
30°C	46.7	22.9	0.7	0.41

Table 3.1: KNN on temperature data

h_i [%]	f_i [%]	d'	C
100.0	2.2	∞	–
54.3	15.7	1.1	0.46
68.5	20.7	1.3	0.15

Table 3.2: GMM on temperature data

class	h_i [%]	f_i [%]	d'	C
10°C	100.0	2.2	∞	–
20°C	51.1	27.2	0.6	0.29
30°C	41.3	24.4	0.5	0.46

Table 3.3: SVM on temperature data

h_i [%]	f_i [%]	d'	C
100.0	3.3	∞	–
77.2	28.3	1.3	-0.09
37.0	11.4	0.9	0.77

Table 3.4: dHMM on temperature data

class	h_i [%]	f_i [%]	d'	C
10°C	100.0	2.2	∞	–
20°C	76.1	30.5	1.2	-0.10
30°C	34.8	11.9	0.8	0.79

Table 3.5: cHMM on temperature data

previous paragraph. Tables 3.1 to 3.5 state the average percentage of hits and false alarms over all different partitioning of the data into training and test sets, as well as the normalized discrimination capabilities d' and thresholds C of the classifiers for each temperature class.

Fig. 3.5 displays the hit and false alarm rates of the classifiers for the three temperature classes. The averages over all data partitionings are depicted by solid bars, the error bars indicate the minima and maxima in the measurements. The d' values of the different classification methods are compared in Fig. 3.6.

Performance on the acclimation data set

All classification experiments are repeated using the acclimation data set as introduced in Subsection 3.1.1. The results are summarized in Tables 3.6 to 3.10.

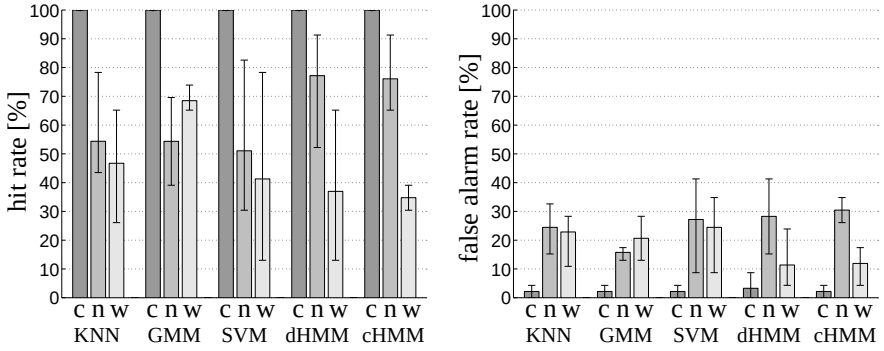


Figure 3.5: Hit rate and false alarm rate for all classifiers. The percentage of hits (left) and false alarms (right) on the temperature data is shown for each classifier in each of the three temperature classes: 10°C ("c"), 20°C ("n"), and 30°C ("w"). The error bars range from the smallest observed rate to the largest one (min–max bars).

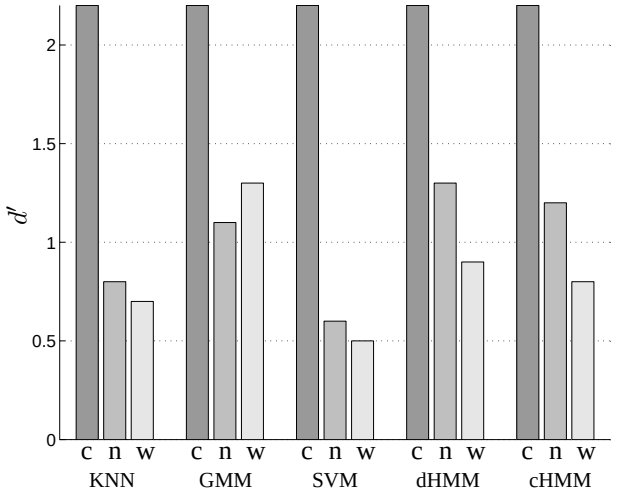


Figure 3.6: d' values of all classifiers. The value of the discrimination capability d' on the temperature data is shown for each classifier in each of the three temperature classes: 10°C ("c"), 20°C ("n"), and 30°C ("w"). The bars for "c" range to infinity, indicating the (almost) perfect separability of the data in this class (cf. Fig. 3.3).

class	h_i [%]	f_i [%]	d'	C
10°C	77.6	23.3	1.5	-0.02
16°C	59.5	15.5	1.3	0.39
25°C	41.4	22.0	0.6	0.50

Table 3.6: KNN on acclimation data

h_i [%]	f_i [%]	d'	C
88.0	21.1	2.0	-0.19
58.6	4.3	1.9	0.75
61.2	20.68	1.1	0.27

Table 3.7: GMM on acclimation data

class	h_i [%]	f_i [%]	d'	C
10°C	86.2	20.3	1.9	-0.13
16°C	62.9	9.9	1.6	0.48
25°C	54.3	18.1	1.0	0.40

Table 3.8: SVM on acclimation data

h_i [%]	f_i [%]	d'	C
84.5	20.3	1.8	-0.09
71.6	22.4	1.3	0.09
35.3	11.6	0.8	0.79

Table 3.9: dHMM on acclimation data

class	h_i [%]	f_i [%]	d'	C
10°C	75.0	19.4	1.5	0.09
16°C	56.0	6.9	1.6	0.67
25°C	61.9	27.2	0.9	0.15

Table 3.10: cHMM on acclimation data

h_i [%]	f_i [%]	d'	C
88.5	24.8	1.9	-0.26
47.3	16.5	0.9	0.52
33.8	23.8	0.3	0.57

Table 3.11: Humans on acclimation data

Fig. 3.7 shows the average hit and false alarm rates of the classifiers for the three temperature classes, along with their min–max bars. Based on the d' values, the classifiers are compared among each other in Fig. 3.8.

In addition to machine learning algorithms, the acclimation data set is also classified by humans. After training on a set of 30 trajectories and their labels, the *test persons* are presented one unknown trajectory at a time. Individual position measurement points are symbolized by circles in order to provide speed information. All trajectories are shifted to start at the origin of the coordinate system, and they are rotated by a random angle prior to presentation. Each person classifies 174 trajectories appearing in random order. The average result over 5 test persons is given in Table 3.11. The best-performing person who declared after the experiment to have looked at speed information only reaches $d' = 2.0$ for the 10°C class, $d' = 1.6$ for the 16°C class, and $d' = 0.7$ for the 25°C class. The globally best person reaches $d' = 2.1$, $d' = 1.9$, and $d' = 1.0$, respectively, by taking into account both speed and shape (curvature) information. The lowest result of the test group is $d' = 1.9$, $d' = 0.1$, $d' = -0.6$.

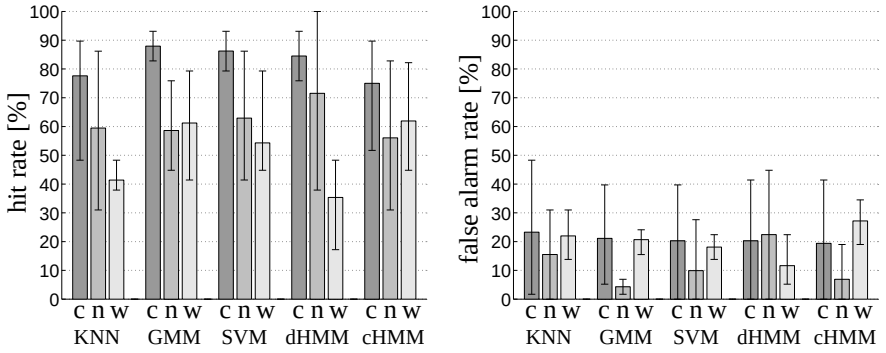


Figure 3.7: Hit rate and false alarm rate for all classifiers. The percentage of hits (left) and false alarms (right) on the acclimation data set is shown for each classifier in each of the three temperature classes: 10°C ("c"), 16°C ("n"), and 25°C ("w"). The error bars range from the smallest observed rate to the largest one (min–max bars).

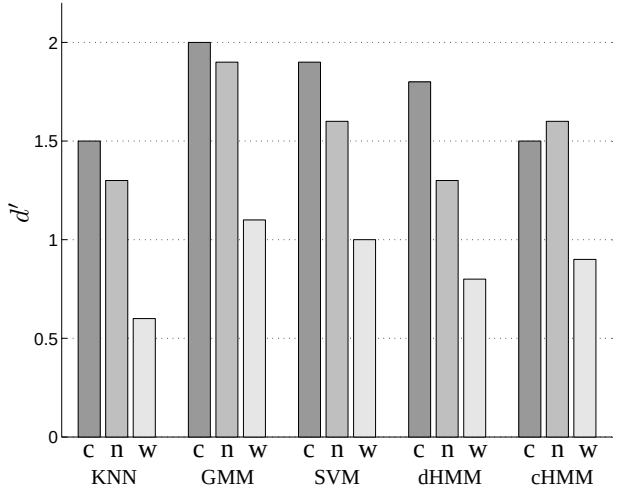


Figure 3.8: d' values of all classifiers. The value of the discrimination capability d' on the acclimation data set is shown for each classifier in each of the three temperature classes: 10°C ("c"), 16°C ("n"), and 25°C ("w").

Conclusions

From the results of this section we see that on non-separable clusters, GMM (Appendix B.2) and SVM (Appendix B.3) perform best. This provides evidence that the data are actually normally distributed. All classifiers are close to neutral with low normalized threshold values C. Compared to human classification, we find that the automatic classifiers are at least as good or better, while providing unbiased analysis and fast processing of large data sets. Humans are biased by prior expectations and suffer from fatigue and inaccuracy. The fact that the best human classification is as good as the best automatic classification indicates that the data encoding (Subsection 3.1.3) captures most of the relevant information.

3.2 Maximizing classification performance by encoding optimization

Classification performance strongly depends on the data encoding. Manually enlisting all interesting properties and finding a good encoding by exhaustive search as done in Subsection 3.1.3 is not a practical strategy for high-dimensional encoding spaces. Moreover, the manually found encoding may be sub-optimal or biased by prior expectations.

In this section, we consider the problem of automatically finding an optimal encoding. We present a solution that uses *meta-optimization* and demonstrate its performance on synthetic data with known optimal encoding as well as on the keratocyte data introduced in Subsection 3.1.1.

The method is based on defining a set of *signals* that are explicit functions of the trajectory position sequence $\{\boldsymbol{x}_i\}$. Using a set of *operators* that map $\mathbb{R}^M \mapsto \mathbb{R}$, these signals are reduced to scalar *characteristics*, such as the mean velocity, the minimum step length, etc. Encoding is performed in a d-dimensional *encoding space*, where each Cartesian axis is defined by a linear combination of characteristics. These linear combinations are called *features*, and the *encoder* is completely described by the set of weights used therein. Since usually $d \ll M$, the encoder performs a *dimensionality reduction* that is optimized with respect to the separability of the data in the encoding space.

The encoded trajectories are classified using a standard classifier, and the classification quality is measured. An optimizer adjusts the weights of the encoder such as to maximize the classification quality. This procedure, as depicted in Fig. 3.9, finds the set of most discriminating characteristics for any given classes of trajectories. Besides the actual classification, this set of *discriminating features* is itself

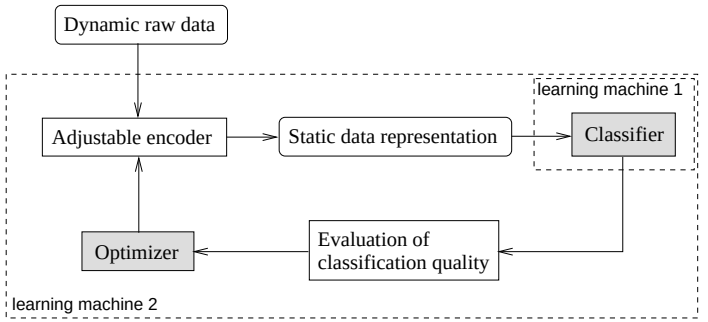


Figure 3.9: Outline of the principle: Dynamic data is encoded in order to use machine learning techniques such as classification. To find a good encoding, an optimizer is added to maximize the classification quality in an outer learning loop. The shaded blocks denote software modules that are easily replaceable.

interesting. Knowing in what respect two classes of trajectories differ the most possibly enables conclusions about the physical or biological process underlying the motion.

The *self-optimizing encoder* is implemented in a modular software framework, where both the classifier and the optimizer are easily replaceable. The modules as depicted in Fig. 3.9 perform the following tasks:

- Adjustable encoder: Compute the signals and characteristics of all trajectories and combine the characteristics to features using linear combinations with certain weights.
- Classifier: Separate the data into classes.
- Evaluation of classification quality: Quantify the classification quality using a set of test data and compute a measure of quality.
- Optimizer: Adjust the weights of the encoder to maximize the quality measure.

These modules are discussed in turn below.

3.2.1 A parametric encoder

The *encoder* performs the dimensionality reduction in a sequence of steps: starting from a signal $\in \mathbb{R}^M$, it applies a set of operators $\mathbb{R}^M \mapsto \mathbb{R}$. Each signal-operator

pair (i, j) defines a *characteristic* c_{ij}. From the set of all possible characteristics, d scalar features f_i are computed as linear combinations. The weights w_{ij} of the linear combinations form the set of parameters that describes the encoder.

Signals

Signals are explicit functions of the trajectory point sequence $\{\boldsymbol{x}_i\}$. A signal is a time series and usually of the same or similar length (dimension) as the trajectory itself. Commonly used signals include:

1. the path $\{\boldsymbol{x}_i\}_{i=1}^{M}$,
2. the speed along the trajectory $\Delta t^{-1}\{\|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i\|_2\}_{i=1}^{M-1}$,
3. the acceleration $\Delta t^{-1}\{\|\boldsymbol{x}_{i+2} - 2\boldsymbol{x}_{i+1} + \boldsymbol{x}_i\|_2\}_{i=1}^{M-2}$, and
4. the path length per net displacement $\{\sum_{j=1}^{i} \|\boldsymbol{x}_{j+1} - \boldsymbol{x}_j\|_2 / \|\boldsymbol{x}_i - \boldsymbol{x}_1\|_2\}_{i=1}^{M-1}$.

Operators

Operators are mapping functions $\mathbb{R}^M \mapsto \mathbb{R}$ that can be applied to signals $\{\boldsymbol{s}_i\}$. Commonly used reduction operations include:

a. the minimum $\min_i \|\boldsymbol{s}_i\|_2$,
b. the maximum $\max_i \|\boldsymbol{s}_i\|_2$,
c. the mean $\bar{s} = 1/M \cdot \sum_{i=1}^{M} \|\boldsymbol{s}_i\|_2$,
d. the standard deviation $1/(M-1) \cdot \sum_{i=1}^{M} (\|\boldsymbol{s}_i\|_2 - \bar{s})^2$, and
e. the median $s_I : \#\{\|\boldsymbol{s}_i\|_2 < s_I\} = \#\{\|\boldsymbol{s}_i\|_2 > s_I\}$.

Characteristics

A *characteristic* is defined as the result of applying a particular operator i to a particular signal j. All characteristics are normalized to the closed interval $[0, 1]$ and stored in a matrix $\boldsymbol{C} = c_{ij}$, whose columns correspond to the signals and the rows to the operators. If we have #op different operators and #sig signals, the matrix of characteristics thus is $\boldsymbol{C} \in \mathbb{R}_{[0,1]}^{\#\text{op} \times \#\text{sig}}$.

Features

A *feature* is a scalar value computed as a linear combination of characteristics. Each dimension of the *encoding space* is formed by one feature, such that d linearly independent features are required to span a d-dimensional encoding space. The features are computed by element-wise multiplication of the matrix $\boldsymbol{C}$ with a weight matrix $\boldsymbol{W} = w_{ij}$ of the same size, thus:

$$f_\ell(\boldsymbol{C}) = \sum_{i=1}^{\#\text{op}} \sum_{j=1}^{\#\text{sig}} w_{ij} c_{ij} \qquad \ell = 1, \ldots, d\,. \tag{3.6}$$

Non-linear features with continuous derivatives are approximated by their *Taylor series* expansion around 0:

$$f(\boldsymbol{C}) = f(\boldsymbol{0}) + f'(\boldsymbol{0})\boldsymbol{C} + \frac{1}{2!} f''(\boldsymbol{0})\boldsymbol{C}^2 + \ldots + \frac{1}{r!} f^{(r)}(\boldsymbol{0})\boldsymbol{C}^r + \text{h.o.t.}\,, \tag{3.7}$$

where all matrix operations are element-wise. The unknown derivatives are absorbed into the weights and $f(\boldsymbol{0}) \equiv 0$. Taking terms up to order r, the matrix $\boldsymbol{C}$ is thus augmented with all powers $k = 2, \ldots, r$ of the characteristics c_{ij}, and the weight matrix is enlarged accordingly:

$$f_\ell(\boldsymbol{C}) = \sum_{k=1}^{r} \sum_{i=1}^{\#\text{op}} \sum_{j=1}^{\#\text{sig}} w_{ijk} c_{ijk} \qquad \ell = 1, \ldots, d\,. \tag{3.8}$$

3.2.2 Measuring the classification quality

To estimate the quality of classification, we use k-fold *cross-validation* as outlined in Section 2.3.5. The following quantities are either known from the data or from the classifier output:

- class labels: y_i ,
- predicted class labels: $\widetilde{y}_i$,
- reliability for the class label prediction: q_i ,
- probability for a trajectory to belong to class j: p_j.

The following set of *fitness functions* is used to quantify the classification quality that is to be maximized:

1. **Quadratic Distance:** The quadratic Euclidean distance between the true class labels and the predicted ones, thus:

$$-\|\boldsymbol{y} - \widetilde{\boldsymbol{y}}\|_2^2 \,. \tag{3.9}$$

2. **Weighted Quadratic Distance:** The quadratic distance weighted with the reliability, i.e. the estimated probability that the classification is correct:

$$-\sum_i \left(y_i - \widetilde{y}_i\right)^2 q_i \,. \tag{3.10}$$

3. **Penalty:** A user-defined penalty Π is assigned to each miss and false alarm:

$$-\sum_{\text{class}} \Pi_{\text{miss}}^{\text{class}} \#\{\text{misses}\} + \Pi_{\text{fa}}^{\text{class}} \#\{\text{false alarms}\} \,. \tag{3.11}$$

4. **(Empirical) risk:** The relative frequency of misclassification:

$$-\frac{1}{\#\{i\}} \#\{i : \widetilde{y}_i \neq y_i\} \,. \tag{3.12}$$

5. **Cross entropy:** From the estimated probability vector that a sample belongs to a class, and the true class membership indicator $t_j = \delta_{jy}$, we compute the *difference of information* for each sample n as:

$$\Delta_n = \sum_i \left[t_{i,n} \log \frac{t_{i,n}}{p_{i,n}} + (1 - t_{i,n}) \log \frac{1 - t_{i,n}}{1 - p_{i,n}} \right] \,. \tag{3.13}$$

The fitness function is given by the negative cross entropy:

$$-\sum_n \Delta_n \,. \tag{3.14}$$

6. **Probability Distance:** The quadratic Euclidean distance between the estimated and the true vector of class membership probabilities, viz.

$$-\|\boldsymbol{t} - \boldsymbol{p}\|_2^2 \,. \tag{3.15}$$

7. **d':** The discrimination capability of the classifier as introduced in Subsection 3.1.4 is directly used as fitness function. To avoid numeric overflow, it is saturated at $d'_{\text{max}} = 1000$.

3.2.3 The classifier

In the *classifier* module, we use *Support Vector Machines* (SVM, cf. Appendix B.3). A multi-class SVM is constructed as a tree of binary SVMs, and the class probabilities are formed by summation of the node probabilities along the tree. In our software implementation we make use of the freely available *libSVM* [49].

We use two different variants of the SVM classifier: C-SVM, and ν-SVM. The parameters C and ν control the "softness of the decision boundary", i.e. the penalty assigned to a false classification in the training set. The influence of these parameters on the overall performance is tested in Subsection 3.2.5.

The ratio between the mean risk on the training sets and the mean risk on the test sets in the cross-validation is used to control the model complexity and to perform *model selection* based on the width of the SVM kernel function. For ratios > 3 the complexity is decreased, whereas it is increased for ratios < 1. This rule is based on the fact that the ratio between the true risk and the empirical risk, R/R_e, is approximately equal to $1 + 2c$, where c is the *compression ratio*. This theorem is known as the *Shibata criterion* [260]. The compression ratio hereby quantifies how much of the original data's complexity is retained by the model. In the case of SVM, complexity is quantified by the Vapnik-Chervonenkis dimension d_v as defined in Appendix B.3. A compression rate of $c = 1$ means that no learning occurs, $c > 1$ indicates overfitting (cf. Section 2.3.4).

3.2.4 The optimizer

The *optimization* of the encoder weights is done using the *Covariance Matrix Adaptation* (CMA) evolution strategy [125].

The CMA optimization algorithm has three parameters: the step size σ, the number of offspring λ per generation, and the number of parents μ in the population. The standard choice of $\lambda = 2\mu$ is used in all cases. For smooth fitness functions, we use $\mu = d$, otherwise $\mu = 1$. The influence of these parameters is considered in Subsection 3.2.5.

Since only the relative values of the encoder weights are important, they are normalized to $[-1, 1]$ to prevent the optimizer from drifting off. This normalization involves dividing all weights by the value of the absolute largest one and running the optimization process on the remaining $N-1$ independent variables.

Fitness function	Fitness	Risk	Iterations
Quadratic Distance	0.01356	0.0121212	2438
Weighted Quadratic Distance	0.01107	0.0121212	1058
Penalty, $\Pi = 1$	1.101e-4	0.0060606	5751
Risk	0.00606	0.0060606	2280
Cross entropy	45.5254	0.0121212	2014
Probability Distance	0.47998	0.0181818	258
d'	0.33194	0.0060606	8146

Table 3.12: Comparison of the final risk for the different fitness functions.

3.2.5 Parameter studies

We compare the effects of different parameter choices using artificially generated trajectories of biased random walks. The data set consists of three classes with step displacement lengths taken from the normal distribution $\mathcal{N}(\mathsf{E}, s^2)$. The three classes are defined by different pairs (E, s) as:

- class 1: $\mathsf{E} = 20$, $s = 5$
- class 2: $\mathsf{E} = 30$, $s = 6$
- class 3: $\mathsf{E} = 20$, $s = 10$.

The trajectories are encoded in two dimensions using the mean speed and the standard deviation of the speed as characteristics in a linear encoder. The total number of unknown weights thus is four. In order to gain a feeling for proper parameter choices in practical applications, we consider different fitness functions, different classifiers, and different parameter values for the CMA optimizer.

Comparison of different fitness functions

The feasibility of the different fitness functions introduced in Subsection 3.2.2 is tested using the CMA optimizer with $\mu = 4$, $\lambda = 8$, and a C-SVM with C=1000. Table 3.12 summarizes the results.

The fitness functions d', penalty, and risk work best, with the risk function requiring the minimum number of iterations for convergence.

Parameters	d'	Risk	Iterations
C-SVM, C=100	0.33214	0.0121212	100
C-SVM, C=1000	0.33214	0.0121212	215
C-SVM, C=10000	0.33194	0.0060606	10790
ν-SVM, $\nu = 0.4$	1000.00	0.0	17548
ν-SVM, $\nu = 0.6$	0.33214	0.0121212	2496

Table 3.13: Performance comparison for different classifier parameter values.

Comparison of different classifier parameter values

Using the same three-class data set, we estimate the influence of the classifier parameters on the overall performance. We test both the C-SVM and the ν-SVM for different values of C and ν, using the fitness function d'.

The parameter C of the C-SVM defines the penalty assigned to a misclassification in the training data. Its value ranges between 0 and ∞. The ν-SVM takes a different approach with the parameter ν limiting the training error per support vector.

Table 3.13 shows the different optima found. The ν-SVM with $\nu = 0.4$ results in data separability. The performance of the C-SVM improves with increasing C, at the expense of more iterations and increasing tendency of overfitting. At the same risk level, the C-SVM needs less iterations than the ν-SVM.

Comparison of different optimizer parameter values

Again using the same data set and a ν-SVM with $\nu = 0.6$, we assess the effects of the optimizer parameters. The step size σ is internally adjusted by the CMA and only its initial value needs to be specified.

We find the CMA with $\mu = 1$ to converge much faster (852 iterations instead of 2496) to the same fitness value (risk 0.0121212) than the CMA with $\mu = 2$. This indicates that the fitness function is mono-modal, such that the averaging between the two parents does not add to the performance. For higher-dimensional examples, larger values of $\mu \approx d$ are however expected to be beneficial. All optimizers use $\lambda = 2\mu$ offspring.

Concerning the initial step size, we find that $\sigma = 0.01$ performs slightly better than $\sigma = 0.1$. This however strongly depends on the particular problem, such that no general recommendation can be deduced.

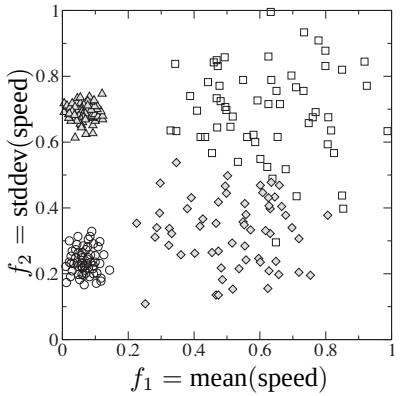


Figure 3.10: Data encoded by the mean speed and the standard deviation of the speed. The symbols from the different classes are not linearly separable under this a-priori encoding.

3.2.6 Benchmarks

Synthetic data

We first illustrate the functioning of the algorithm on synthetic test data. Artificial trajectories of biased random walks are generated. The individual step displacements have uniformly random directions and their lengths are sampled from the normal distribution $\mathcal{N}(\mathsf{E}, s^2)$. Four different (E, s)-pairs are used for the four classes:

- class 1: $\mathsf{E} = 20$, $s = 5$
- class 2: $\mathsf{E} = 20$, $s = 25$
- class 3: $\mathsf{E} = 60$, $s = 5$
- class 4: $\mathsf{E} = 60$, $s = 25$.

This defines an *a-priori encoding* based on E and s in a two-dimensional encoding space. Fig. 3.10 shows the encoded data set using this prior knowledge. Notice that the data are not linearly separable under this encoding.

The data are processed by the self-optimizing encoder described so far, using a two-dimensional encoding space with linear features. We use the signal "speed" and the operators "mean" and "standard deviation", creating a 2×2 weight matrix to be optimized. The initial encoding is defined by random guesses for the weights and is shown in Fig. 3.11(a). We use a C-SVM with C=1000 and a CMA optimizer

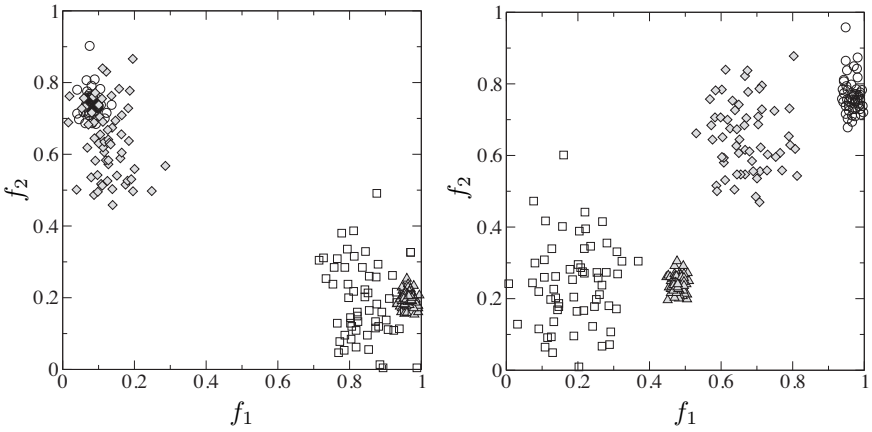


Figure 3.11: (a) Initial data encoding with random weights: $f_1 = 0.0267 \cdot$ mean(speed) $- 0.0078 \cdot$ stddev(speed) and $f_2 = -0.1276 \cdot$ mean(speed) $- 0.1167 \cdot$ stddev(speed). (b) Optimal data encoding found after 9 iterations of the CMA optimizer: $f_1 = -0.1358 \cdot$ mean(speed) $- 0.1096 \cdot$ stddev(speed), and $f_2 = -0.1833 \cdot$ mean(speed) $- 0.0521 \cdot$ stddev(speed). The classes become linearly separable.

with $\mu = 1$, $\lambda = 2$, and $\sigma = 0.1$. After 9 iterations of the CMA, the data encoding shown in Fig. 3.11(b) is found, where the two optimal features are:

$$f_1 = -0.1358 \cdot \text{mean(speed)} - 0.1096 \cdot \text{stddev(speed)} \tag{3.16}$$

$$f_2 = -0.1833 \cdot \text{mean(speed)} - 0.0521 \cdot \text{stddev(speed)}\,. \tag{3.17}$$

Under this encoding the data become linearly separable and perfect classification is possible. Using the same number of encoding space dimensions, and the same signal-operator space, the automatic encoder is thus able to perform much better than the direct use of prior knowledge.

Keratocyte trajectory data

We compare the self-optimizing encoder to the manually found data encoding of Subsection 3.1.3 for the acclimation data set (Subsection 3.1.1).

We use the signals "path", "speed", and "acceleration", and the operators "min", "max", "mean", and "standard deviation". Furthermore, we use a 4$^{\text{th}}$ order non-

	d'	Risk	Iterations
C-SVM, C=100	0.99898	0.306306	115521
ν-SVM, ν=0.4	0.99918	0.342342	79243

Table 3.14: Performance comparison on the keratocyte data set.

linear encoder, keeping terms up to $r = 4$ in Eq. (3.7), and an encoding space of dimension four. The total number of weights thus is 192, leading to a high-dimensional optimization problem. As fitness function we use the measure d', and the optimizer parameters are set to $\sigma = 0.1$, $\mu = 1$, and $\lambda = 2$.

Table 3.14 summarizes the results. After 115521 iterations, the C-SVM is able to classify the data with a residual risk of about 31%. This compares well to the minimal risk of 34% for the SVM with manual encoding (cf. Subsection 3.1.4). The presented automatic encoding method thus seems feasible to find good representations for biological trajectories, and it enables higher-dimensional encoding spaces where manual search becomes prohibitively expensive.

Chapter 4

Applications and Results

This chapter presents applications and case studies of the single particle tracking and trajectory analysis methods described so far. We start with three application cases from cell biology, demonstrating the feasibility of the tracking algorithm presented in Chapter 1 for purely random motion, fast directed motion, and the tracking of objects with strong intensity fluctuations. The third is used to demonstrate how the multi-frame linking algorithm (Subsection 1.2.2) recovers connected trajectories from intermittent point detections.

The complete spectrum of methods, from global analysis to segmentation to event-based analysis, is then demonstrated on trajectories of virus particles on the extracellular surface of the plasma membrane of live cells. Two different viruses – Polyomavirus and Adenovirus-2 – are fluorescently labeled and added to live cells. After binding to their respective receptors in the plasma membrane, the virus-receptor complexes exhibit intricate motion dynamics before being internalized by the cell.

The *Polyomavirus* study presented in Section 4.2 was done in collaboration with the group of Prof. A. Helenius and considers the motion of virus-receptor complexes under different chemical treatments of the cells. Analyzing the motion patterns under perturbations of the cytoskeleton or the lipid composition of the plasma membrane provides important information about the molecular mechanisms of virus uptake and the organization of the plasma membrane.

The *Adenovirus-2* study of Section 4.3 is a collaboration with the group of Prof. U. Greber, and it entails a particularly large data set with several tens of thousands of trajectories that were acquired, analyzed, and classified in a completely automatic way. This illustrates the suitability of the presented methods for high-throughput studies and enables the detection of weak interactions by statistical analysis. The large number of samples and the unbiased data processing permit significant statistics even for cases that would not be detectable otherwise. Adenovirus-2 relies on a duo of receptors to bind to the cell. It first associates with

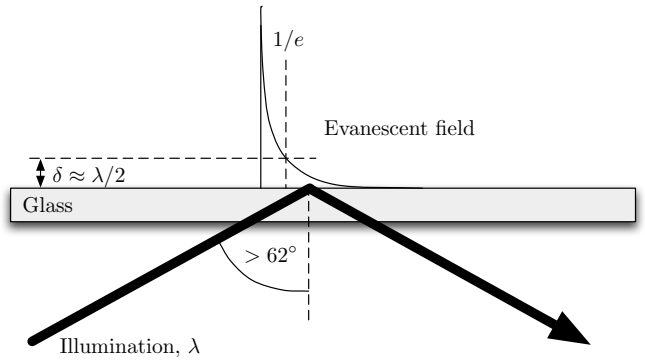


Figure 4.1: Total reflection at the glass-water interface and the evanescent field on the water side form the principles of TIRF microscopy. The illumination depth δ is defined by the $1/e$ decay of the evanescent field intensity and is about half of the wavelength λ of the incident light.

a *primary receptor* and then transfers to a *secondary receptor* before internalization. The present study compares the motion of viruses on wild type cells with the motion on cells that lack the secondary receptor. This leads to conclusions about the interplay of the two receptors and the virus transfer between them.

The videos underlying all of the studies presented in this chapter were recorded in the groups of Prof. Helenius and Prof. Greber using *Total Internal Reflection Fluorescence* (TIRF) microscopy. This microscopy technique makes use of the *evanescent field* in total beam reflection to achieve high *depth selectivity*.

Total internal reflection occurs at interfaces from optically dense to optically less dense media, if the incidence angle of the beam exceeds a critical threshold. In the microscope, the quartz glass of the coverslip forms such an interface with the water of the sample atop. For this quartz-water interface, the critical angle is about 62°. In TIRF microscopy as illustrated in Fig. 4.1, the angle between the interface and the laser beam used to excite the fluorescent markers is larger than this critical angle. On the low-density side of the interface, an evanescent field with exponentially decaying intensity develops. The $1/e$ *depth of penetration* (δ) of this field is about half the wavelength of the incident light. Hence, only a few hundred nanometers at the bottom of the sample are imaged in TIRF microscopy [288, 287].

The use of TIRF microscopy in the present application ensures that only the

virus particles within the evanescent field are imaged. These are the viruses on the plasma membrane[1]. Free viruses in solution and internalized viruses inside the cell are not illuminated, and are thus excluded from the observation. The depth selectivity of TIRF microscopy is an enabling feature of the present studies of virus motion on the plasma membrane of live cells.

4.1 Three case studies from cell biology

The feature point tracking algorithm described in Section 1.2 and the global MSS analysis method (Subsection 2.1.2) are demonstrated using three different case studies from cell biology. The three examples highlight different capabilities of the data analysis procedure:

1. *Tracking of freely diffusive motion:* endosomes containing fluorescently labeled Low-Density Lipoprotein (DiI-LDL) molecules,
2. *tracking of fast directed motion:* trafficking of internalized Adenovirus-2 (Ad-2) particles moving along microtubules, and
3. *tracking of objects with strong intensity fluctuations:* quantum dots (Qdot) on the plasma membrane.

These case studies help verify the robustness and applicability of the algorithms for a wide variety of problems in cell biology.

4.1.1 Moment scaling spectrum of endosome motion

In the first application, *endosomes* of *3T6 mouse fibroblast* cells are imaged. LDL is fluorescently labeled with DiI-red. Endosomes containing DiI-LDL are imaged using TIRF microscopy at 20 Hz with 80 nm/pixel resolution (Helenius group), and 2000 16-bit TIFF frames are recorded. Fig. 4.2 shows a few sample frames. The parameters used in tracking are listed in Table 4.1. The particle is successfully traced over 1446 frames before it fades out. The extracted trajectory is shown in Fig. 4.3(a).

The results of the MSS analysis (cf. Subsection 2.1.2) are shown in Fig. 4.3(b,c). The MSS shows an almost perfectly straight line of slope $1/2$. The particle thus undergoes free and normal diffusion. The diffusion constant is determined from the second moment μ_2 to be $\nu_2 = 1.8 \cdot 10^{-3}\,\mu\text{m}^2/\text{s}$. Fig. 4.7(a) shows the intensity

[1] The diameter of the Polyomavirus particle is about 45 nm, the one of Adenovirus-2 is about 90 nm.

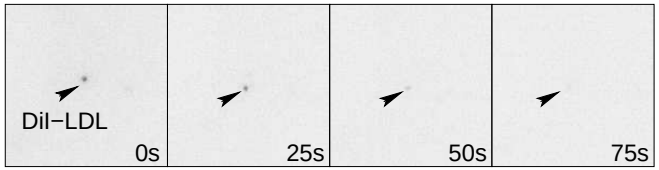


Figure 4.2: Time-lapse frame image sequence of a DiI-LDL-containing endosome (arrow head) in a 3T6 mouse fibroblast observed using TIRF microscopy at 20 frames/second. Each image shows a 12 μm $\times$ 12 μm region on 150$\times$150 pixels, corresponding to a resolution of 80 nm/pixel. The time difference in seconds to the first image of the sequence is given in the lower-right corner of each frame. Using $R = 1$ the particle is tracked over 1446 frames before fading out. (Image intensities are inverted for printing purposes.)

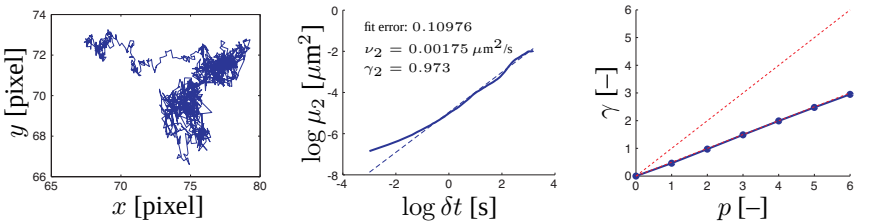
parameter	DiI-LDL	Ad-2	Noc	Qdot
particle radius w [pixel]	4.0	2.0	3.0	3.0
intensity percentile r [%]	0.1	2.0	1.0	0.05
cutoff score T_s [-]	0.0	1.0	4.0	0.0
max step length L [pixel]	5.0	5.0	1.0	1.0
link range R [frames]	1	4	2	1 or 10

Table 4.1: Summary of the tracking algorithm parameter settings used in the examples of this section.

of the endosome over time. The continuous fading could be due to photobleaching or the endosome moving into the cell and thus out of the evanescent field.

4.1.2 Tracking and analysis of Adenovirus-2 trafficking

The tracking of microtubule-dependent trafficking of intracellular *Adenovirus-2* (Ad-2) serves as a test for the algorithm in cases of fast directed motion. We analyze the original 16-bit frame images of Suomalainen *et al.* [276] (courtesy of Prof. U. Greber) that were tracked by hand for the original publication. Fluorescently labeled internalized Ad-2 particles in wild type *TC7 cells* are imaged using a wide-field fluorescence microscope. 104 frames are recorded with a resolution of 0.15 μm/pixel, and a 1.3 seconds time interval between frames. The complete protocol is contained in Ref. [276]. Fig. 4.4 shows a time lapse sequence of some frames with the unspecific photobleaching clearly visible. Tracking is done using the parameter values given in Table 4.1, and yielded 73 tracks of lengths between



(a) (b) (c)

Figure 4.3: Tracking of an endosome containing fluorescent DiI-LDL. (a) xy-path of the particle as tracked from the video recording. (b) Mean square displacement of the track, computed according to Eq. (2.5). The dashed line is the result of the linear least squares fit used to determine the slope and the diffusion constant (values given in the figure). (c) Moment scaling spectrum of the trajectory with dashed lines marking slopes $1/2$ and 1.

60 and 104 frames. Three example tracks are shown in Fig. 4.5(a), and the intensity of virus particle (a) over all 104 frames is shown in Fig. 4.7(b).

The control experiment of Suomalainen *et al.* [276] considers Ad-2 in *HeLa cells* that are treated with *nocodazole*, a microtubule depolymerizing drug. The tracker parameters are given in column "Noc" of Table 4.1. The total length of the control movie is 275 frames. 27 tracks of lengths between 80 and 252 frames are extracted by the tracking algorithm, and MSS analysis as outlined in Subsection 2.1.2 is performed for all of them.

Fig. 4.5 shows a scatter plot of all diffusion constants ν_2 and MSS slopes β for the two experiments. The existence of biased or directed motion in the wild type experiment is evident from the β values above 0.5. Still a significant fraction of trajectories with β around or below 0.5 exists, which means that those particles are not always transported actively. As can be seen from Fig. 4.5(a), intermediate pauses or changes in direction occur, causing the global β to drop. The nocodazole control never exhibits directed motion and particles are at most freely diffusive, which is evidence for the directed motion to depend on microtubules [276].

4.1.3 Tracking of Quantum Dots

We consider the tracking of quantum dots (Qdots) to demonstrate the function of the multi-frame linking algorithm as described in Subsection 1.2.2 for $R > 1$. *Quantum dots* (QuantumDot Corp., www.qdots.com) are extremely bright and

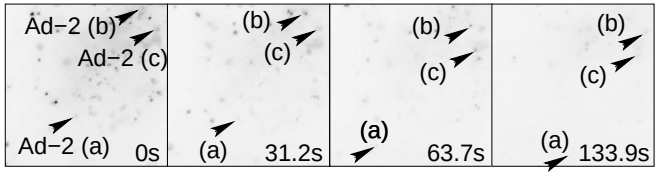
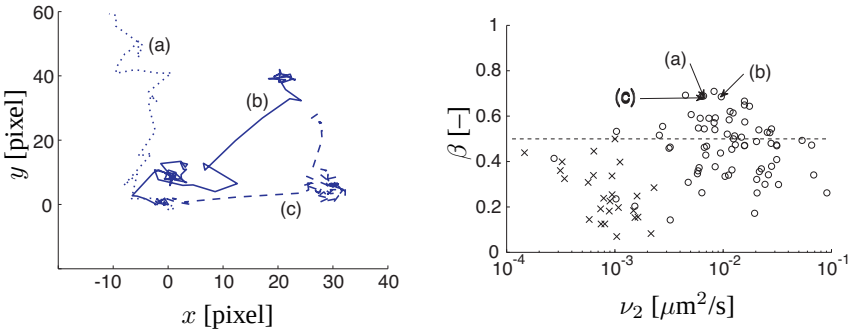


Figure 4.4: Time-lapse frame image sequence of Ad-2 moving along microtubules in TC7 cells observed using wide-field fluorescence microscopy and 1.3 seconds/frame. Each image shows a 37.65 μm $\times$ 37.65 μm region on 251$\times$251 pixels, corresponding to a resolution of 0.15 μm/pixel. The time difference in seconds to the first image of the sequence is given in the lower-right corner of each frame. All indicated particles (arrow heads) are tracked over the full 104 frames. (Image intensities are inverted for printing purposes.)



(a) (b)

Figure 4.5: Trafficking of Ad-2 particles along microtubules. (a) xy-paths of three sample particles as tracked over all 104 frames of the movie. All trajectories are shifted to start at point (0,0). Stretches of directed motion with intermediate random segments are visible. (b) Scatter plot of the global diffusion constants ν_2 and MSS slopes β for all tracks of the wild type experiment (circles: $\circ$) and the nocodazole control (crosses: $\times$).

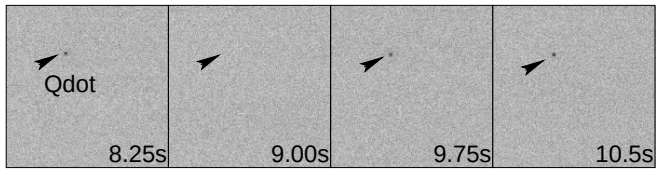


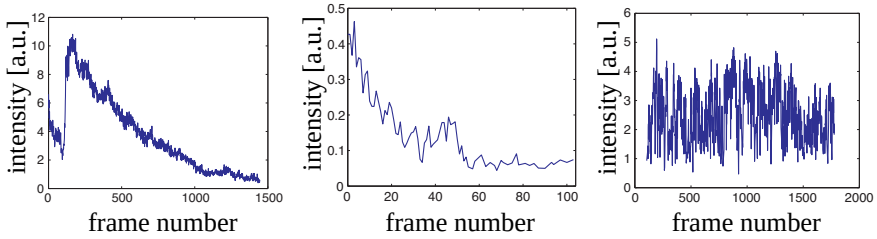
Figure 4.6: Time-lapse frame image sequence of a quantum dot (arrow head) on the plasma membrane of 3T6 mouse fibroblasts observed using TIRF microscopy at 20 frames/second. Each image shows a 12 μm $\times$ 12 μm region on 150$\times$150 pixels, corresponding to a resolution of 80 nm/pixel. The time difference in seconds to the beginning of the sequence is given in the lower-right corner of each frame. Using $R = 1$ the particle can be tracked over 21 frames; with $R = 10$ the longest trajectory spans 1068 frames. (Image intensities are inverted for printing purposes.)

photo-stable fluorescent *nano-particles*. They however exhibit strong fluctuations in their emission intensity (*blinking*), which complicates the linking of point detections into trajectories.

Biotinylized ConcanavalinA is bound to 3T6 cells for 30 seconds in PBS. Cells are dipped in imaging medium. 0.2 μM Streptavidin-coupled 25 nm Qdots are added. Using TIRF microscopy at 37°C (Helenius group), 2000 frames are recorded at 20 Hz video rate with 80 nm/pixel resolution. The images are stored as uncompressed 16-bit TIFF files.

Fig. 4.6 shows a few sample frames from the movie. The blinking is clearly visible as the Qdot has vanished in the second image. Good parameter settings for the tracking algorithm are determined using the graphical user interface documented in Appendix A.3. Their values are given in Table 4.1. Using two subsequent frames to perform trajectory linking (i.e. $R = 1$), the longest track that can be extracted is 21 frames in length. Setting $R = 10$ increases the track length to 1068 frames. This is a clear advantage since tracks as short as those in the $R = 1$ case would not allow to determine diffusion constants or other properties of the motion with significant statistics (cf. Subsection 2.1.2).

Fig. 4.7(c) shows the time evolution of the fluorescence intensity of the sample Qdot. The strong fluctuations ("blinking") are clearly visible, as well as its photostability and brightness. The Qdot in this example is almost stationary. The MSS shows a straight line of slope 0.083 (figure not shown), and the diffusion constant is below the detection limit.



(a) DiI-LDL (b) Ad-2 track (a) (c) Qdot

Figure 4.7: Particle intensities m_0 over time as returned by the tracking algorithm. The time evolution of the intensity is shown for each of the three test cases. The sum of all pixel values within the particle radius w is computed as intensity measure m_0, cf. Eq. (1.8). The strong intensity fluctuations ("blinking") of the Qdot can clearly be seen in (c). The continuous intensity decay of DiI-LDL and Ad-2 could be due to photobleaching or the particle moving out of focus.

4.1.4 Experimental assessment of the tracking accuracy

In order to assess the tracking quality, the SNR of the images are estimated using the noise in the bright image regions as outlined in Subsection 1.3.2. The program used to estimate the SNR is tested on the synthetic images of known SNR from Subsection 1.3.2. The SNR values of these test images are correctly measured to within $\pm 7\%$. The mean measured SNR of both the DiI-LDL and the Qdot samples is 3.1, averaged over all frames. The background intensity of the Qdot video is more than 3 times larger than the one of the DiI-LDL case. Using the results from Subsection 1.3.2, this SNR corresponds to both a tracking accuracy and precision of about 0.2 pixel (16 nm). The experimentally measured track standard deviation in the Qdot example is 0.4 pixel, which is consistent with the very small value of its MSS slope and illustrates the sensitivity of the latter measure.

Positioning errors result in observed *apparent subdiffusion* [187]. Using the model of Martin *et al.* [187], the measured diffusion constant for the DiI-LDL-containing endosome, and above estimate of the positioning error, the apparent slope in the double logarithmic MSD plot of DiI-LDL (Fig. 4.3(b)) is predicted to be $0.933 < \gamma_2 < 1$. This is in excellent agreement with our measured γ_2 of 0.973 and supports the conclusion that the motion of the endosome is normal diffusion, as properly indicated by the MSS slope.

4.2 Analysis of Polyomavirus motion on the plasma membrane

The present section reports the summarized results of applying the feature point tracking and trajectory analysis methods outlined so far to analyzing the motility of murine *Polyomavirus* (Py) on the extracellular surface of the plasma membrane of live cells [96]. Polyomavirus is a small (45 nm diameter), simple, non-enveloped DNA tumor virus [272] that uses either one of the *gangliosides GD1a* or *GT1b* as its *receptor* [291] and relies on clathrin-independent, cholesterol-dependent endocytosis to deliver its genome into the host cell for replication [112]. Instead of the infectious virus, we use *virus-like particles* (VLPs) that resemble the virus structurally, but do not contain the DNA genome [115].

Fluorescently labeled VLPs are added to live *3T6 Swiss albino mouse lung fibroblast cells*. After binding to the receptor, the motion of the VLPs is automatically tracked using the method described in Chapter 1. The goal is to use the labeled VLPs as molecular probes to investigate the organization of the plasma membrane and the interplay of integral membrane proteins (such as the virus receptors) with the cytoskeleton. The study makes use of different chemical perturbations of the cell:

- untreated *wild type* (wt) cells,
- cells treated with *latrunculin A* (LatA), an actin polymerization inhibitor, and
- cells treated with *methyl-β-cyclodextrin* (MCD), a cholesterol sequestration agent that reduces cellular cholesterol to levels below 40% of normal [96].

Control experiments also consider treatments with *jasplakinolide*, a drug that inhibits actin depolymerization and stabilizes actin filaments, and *genistein*, a tyrosine kinase inhibitor that blocks virus uptake. Further controls involve caveolin-deficient cells and quantum dots (cf. Subsection 4.1.3) coupled to *cholera toxin β*, which binds to the same receptor as Py. Since TIRF microscopy records the motion of particles on the bottom membrane of the cell, another set of control experiments considers the top surface of the cell to make sure that the dynamics, as quantified by the global MSS analysis, do not differ. Free unrestricted diffusion of the virus-receptor complex is measured in artificial *DPPE lipid bilayers*. All experimental work was done in the group of Prof. A. Helenius; protocols and controls are contained in the original publication [96].

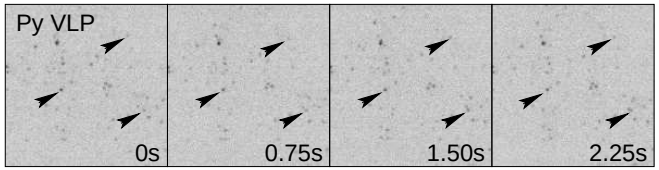


Figure 4.8: Time-lapse frame image sequence of Py VLP moving on the plasma membrane of a 3T6 cell, imaged using TIRF microscopy (Helenius group) at 20 frames/second. Three Py VLP are exemplarily highlighted by arrowheads. Each image shows a 12 μm $\times$ 12 μm region on 150$\times$150 pixels, corresponding to a resolution of 80 nm/pixel. The time difference in seconds to the first image of the sequence is given in the lower-right corner of each frame. The total movie extends over 2000 frames. (Image intensities are inverted for printing purposes.)

4.2.1 Virus particle tracking

Fluorescently labeled Py VLPs are bound to the plasma membrane of live 3T6 cells and imaged using TIRF video microscopy with a resolution of 80 nm/pixel and a frame rate of 20 Hz. All movies are 2000 frames long; a few example frames are shown in Fig. 4.8 for illustration. Feature point tracking is done using the algorithm described in Section 1.2 with parameters $w = 3$ pixel, $R = 1$ frame, $L = 10$ pixel, $T_s = 0$, and $r = 0.1\%$. Only trajectories longer than 100 frames are retained. We record a total of 220 trajectories on wt cells, 74 with LatA treatment, and 256 with added MCD. A few example trajectories of different motion behaviors are shown in Fig. 4.9.

The feature point tracking quality is assessed using the benchmark results of Subsection 1.3.2. The estimated mean SNR of the frame images is 1.35 ± 0.28, corresponding to a tracking accuracy and precision of 0.5 and 0.4 pixel, respectively. The SNR is properly estimated from the bright particle centers as outlined in Subsection 1.3.2. The estimation procedure was previously tested on benchmark images of known SNR and found to measure the values with an accuracy of $\pm 7\%$ (cf. Subsection 4.1.4). The smallest detectable diffusion constant is determined from trajectories of five stationary particles that are directly attached to the glass coverslip. The standard deviation of these tracks is 0.3 pixel, corresponding to a lower limit of $\nu_{2,\min} = 1.9 \cdot 10^{-7}\,\mu\text{m}^2/\text{s}$. The upper bound for the diffusion constant is given by the control experiments in artificial membranes as $\nu_{2,\max} = 3.2 \cdot 10^{-2} \pm 2.3 \cdot 10^{-2}\,\mu\text{m}^2/\text{s}$ ($\beta = 0.42 \pm 0.1$, $N = 39$), so that the dynamic range of the measurements spans about 5 orders of magnitude.

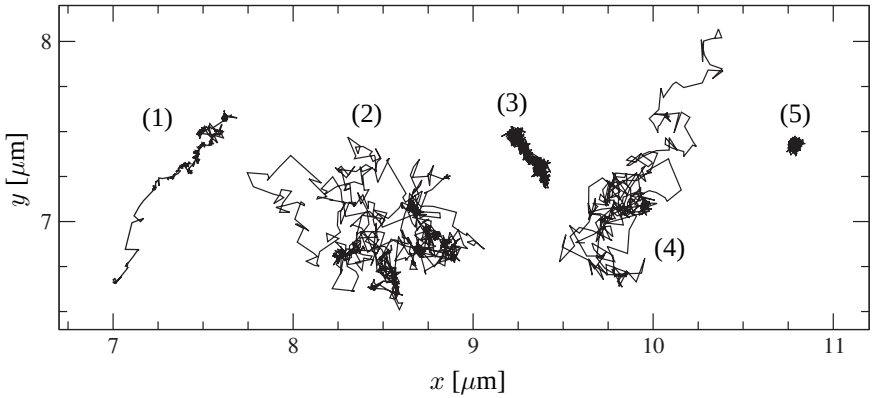


Figure 4.9: Five sample trajectories of Py VLP motion on the plasma membrane of 3T6 cells after binding to the receptor. Different modes of motion are observed: (1) biased motion, (2) mixtures of different segments, (3) confinement in a slowly moving area, (4) free and normal diffusion ($\beta = 0.49$), and (5) arrest zones. The trajectories are represented to scale, but arbitrarily shifted for presentation purposes.

4.2.2 Global analysis results

Examination of the recorded trajectories starts with a global MSS analysis (cf. Subsection 2.1.2) for all three drug treatments. Fig. 4.10 shows the (ν_2, β) scatter plot of all trajectories. Circles mark wt cases, diamonds correspond to MCD treatment, and crosses to LatA treatment. It is evident from this scatter plot that LatA causes a general increase in mobility and diffusion speed, whereas MCD completely eliminates freely diffusive motion.

This can be further quantified by the MSS slope (β) histograms as shown in Fig. 4.11 for all three cases. Treatment with LatA causes a clear shift to larger β, indicating a decrease in confinement. Treatment with MCD on the other hand causes almost all trajectories to be confined (small β) and completely eliminates the occurrence of free diffusion ($\beta = 1/2$). The histogram of step angle changes (Fig. 4.12, cf. Subsection 2.1.3) consistently confirms this observation.

The global MSS analysis in the (ν_2, β) *phase space* also enables unambiguous classification of the motion types of virus-receptor particles. In Fig. 4.13, the three main classes of motion – free diffusion, stationary particles, and particles that are confined in moving areas – are indicated by boxes. The labels of the classes correspond to the numbers in Fig. 4.9, where representative examples from each class

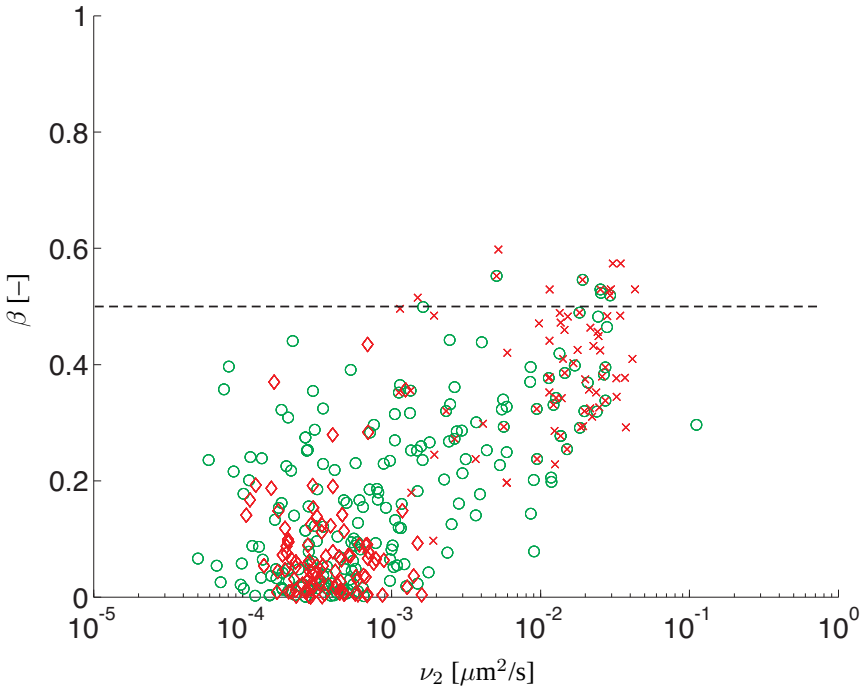


Figure 4.10: Scatter plot of global (ν_2, β) for all drug cases: wt (circles, $N = 220$), LatA (crosses, $N = 74$), MCD (diamonds, $N = 256$). Addition of MCD completely eliminates free diffusion and shifts the population to smaller ν_2. LatA shows the opposite effect.

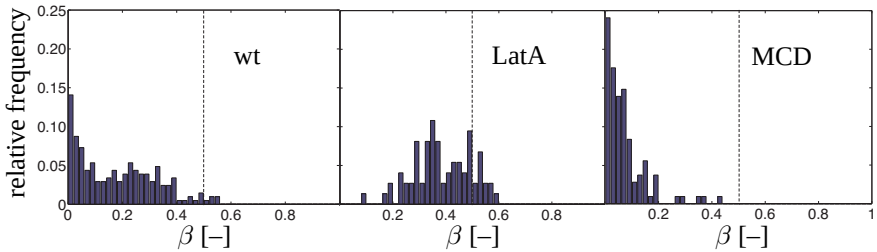


Figure 4.11: Histograms of the global MSS slope β over all trajectories of the three drug cases (see text for description). The total number of trajectories is: $N = 220$ for wt, $N = 74$ for LatA, and $N = 256$ for MCD.

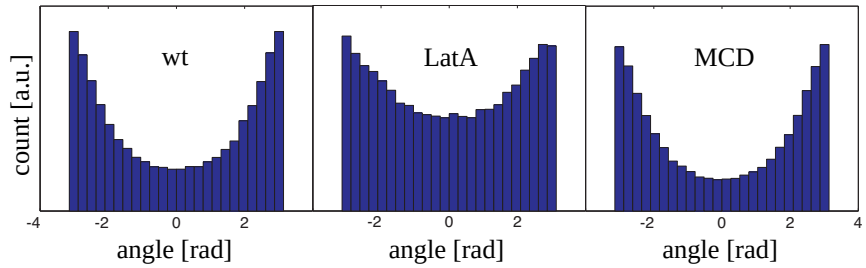


Figure 4.12: Histograms of angle changes between two successive displacements. The counts are taken over all displacements of all trajectories in each drug case (see text for description). A more pronounced the center dip corresponds to more confined motion. Brownian motion would generate a uniform distribution.

are shown. The error bars in Fig. 4.13 indicate the standard deviation according to Eqs. (2.16) and (2.17). It can be seen that the present MSS analysis enables the discrimination between stationary particles and particles that are confined in a slowly moving area. Using only MSD analysis, these two classes would be indistinguishable since they overlap in their diffusion constants [96].

VLPs that are confined to slowly moving zones, such as the one in Fig. 4.9(3), are frequently observed in wt cells. To show that the slow motion macroscopically corresponds to free diffusion, we analyze the corresponding trajectory segments separately. After smoothing with a running *boxcar average* filter of 10 frames width in time, these segments exhibit a macroscopic MSS slope of $\beta = 0.5 \pm 0.1$ and a *macroscopic diffusion constant* of $\nu_2 = 0.5 \ldots 1.5 \cdot 10^{-4}\ \mu\text{m}^2/\text{s}$, indicating free, but very slow, diffusion of the arrest zone itself.

4.2.3 Moving window analysis results

Virus binding and internalization is a dynamic process and the resulting trajectories are not expected to be realizations of a stationary random process. They rather change their mode of motion at least once during their duration. These changes contain important correlations with biochemical processes, wherefore we wish to gain an impression of the time evolution of the quantification parameters along the trajectories. We apply a moving-window MSS analysis with a window width of $n_w = 120$ frames, as introduced in Section 2.2. This yields smoothed traces in the (ν_2, β) *phase space*, representing the change over time. In the case depicted

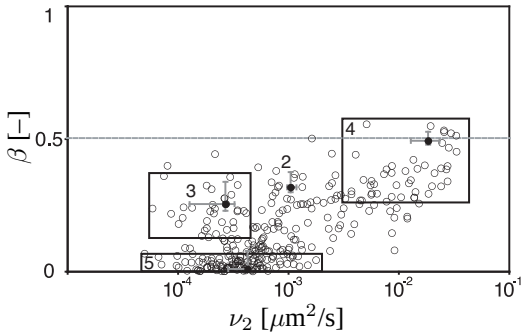


Figure 4.13: Scatter plot of global (ν_2, β) for Py trajectories on wt cells. The boxes indicate regions that correspond to particular motion types (source [96]). Representative examples for each type are shown by the corresponding numbers in Fig. 4.9. Error bars indicate the standard deviation of the statistical uncertainty according to Eq. (2.16).

in Fig. 4.14, the VLP-receptor complex is initially more or less freely mobile. The mobility then decreases as the particle becomes more confined. After a short period of immobility, the complex is confined in an arrest zone that is itself mobile (highlighted by the shaded ellipse in Fig. 4.14). This is the terminal arrest zone for this example and the VLP disappears from the movie after a few seconds of residence time.

4.2.4 Trajectory segmentation results

As suggested by the moving window analysis, the VLPs exhibit a characteristic motion pattern in which free mobility is followed by one or several (possibly mobile) arrest zones before the particle disappears from the image. In combination with the drug treatments of the cells, we use the trajectory segmentation technique outlined in Section 2.3 to investigate the nature of this motion pattern and possible molecular mechanisms for it. The segmentation analysis involves detection of short periods of directed motion as well as *arrest zones*.

To quantify the fraction of directed motion in function of the drug treatment, we compare the cumulative length of all directed segments in a trajectory to the total length of the trajectory. The resulting histograms, as shown in Fig. 4.15, clearly show that MCD effectively prohibits directed motion, a finding that is consistent with the MSS analysis in Fig. 4.11. Also consistently, LatA increases the directed fraction to an average of about 1/5 of the total length of the trajectories.

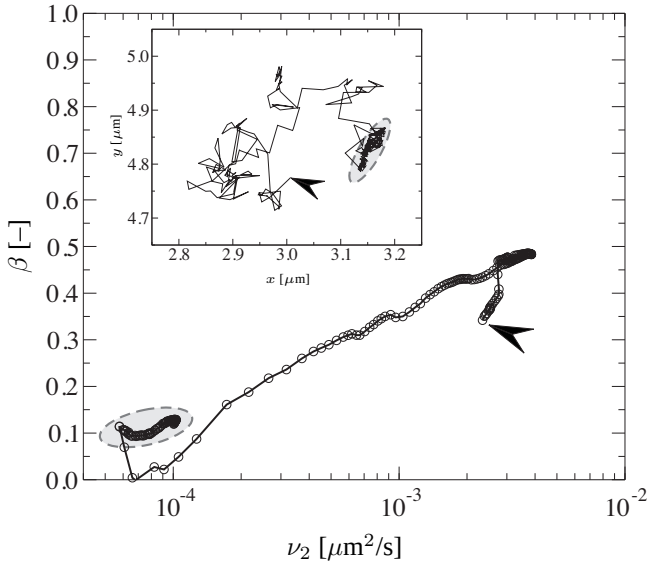


Figure 4.14: Moving window MSS analysis of an example Py trajectory on a wt cell. The inset image shows the trajectory in the xy-plane. The trace of this trajectory in the (ν_2, β) phase space is shown in the main plot. The arrow head marks the start of the trajectory and the terminal confinement zone is highlighted by the shaded ellipse. After being trapped in this mobile arrest zone for a while, the virus disappears from the movie.

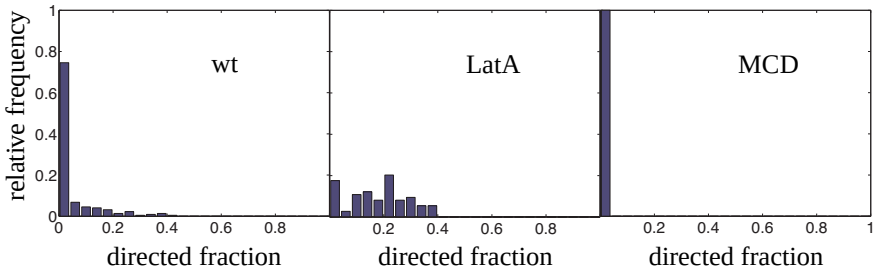


Figure 4.15: Histograms of the fraction of directed segments in each drug case (see text for description).

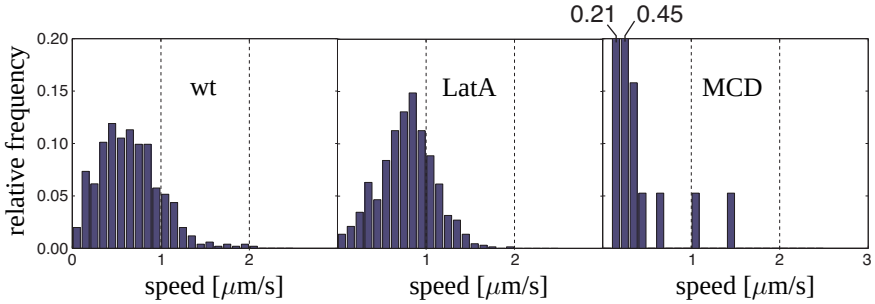


Figure 4.16: Histograms of speed in the directed segments of all drug cases (see text for description). The MCD case contained only 20 directed segments.

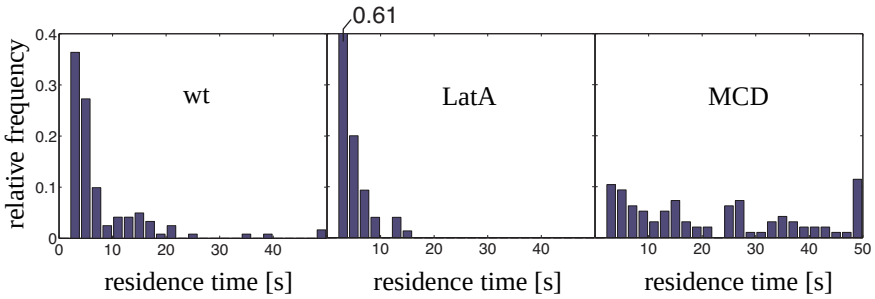


Figure 4.17: Histograms of residence time in transient arrest zones for those particles that leave the zone again.

Knowing the directed segments in each trajectory allows to determine the transport speeds in all of them. The measured distribution of speeds is shown in Fig. 4.16. Application of LatA shifts the distribution to faster speeds with a mean around $0.8\,\mu$m/s. The MCD data set contains only 20 directed segments in total, with all but four having a very low speed.

The segmentation algorithm is also used to identify *transient arrest zones* where the VLP temporarily pauses (within the accuracy of the tracking algorithm) before moving on. Fig. 4.17 shows the distribution of residence times in such arrest zones. Only the zones from which the particle leaves again are considered, as the residence time for the others can not be quantified (the end of the movie is arbitrary). While LatA virtually eliminates residence times $>$15 seconds, MCD causes a spread to longer times.

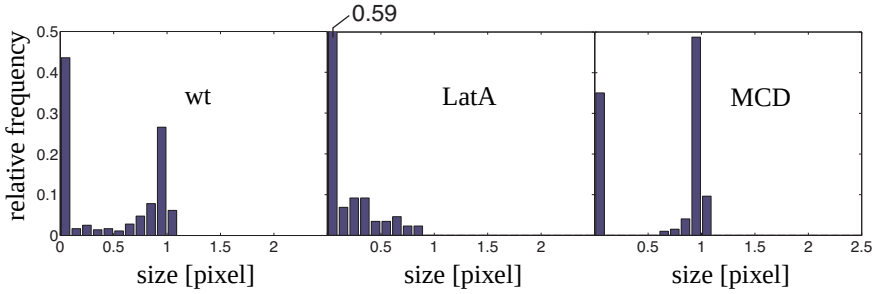


Figure 4.18: Histograms of the size of transient arrest zones for all drug cases.

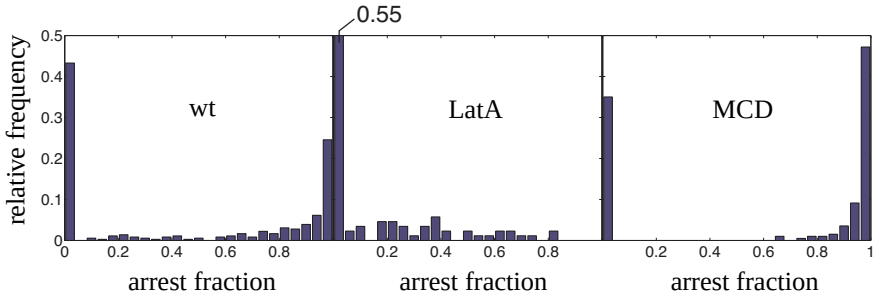


Figure 4.19: Histograms of the fraction of the total trajectory length spent in arrest zones for all drug cases.

The distribution of the sizes of the arrest zones is shown in Fig. 4.18 for all three classes. The size is measured as the standard deviation of the trajectory inside the arrest zone and typically is around 1 pixel in wt cells. Addition of MCD does not change this typical size, but increases its frequency. LatA eliminates most arrest zones of finite size.

Fig. 4.19 shows the distribution of the fraction of the whole trajectory length that is spent in arrest zones. In wt cells, around 45% of all trajectories do not contain arrest zones at all, while the fraction of those spending their whole life in arrest zones is around 25%. The LatA treatment causes 55% of the VLPs to never pause and the set of trajectories containing arrest zones only has vanished. MCD has the opposite effect with the fraction of permanently immobile particles rising to almost 50%.

The statistics of arrest zone sizes and VLP residence times are summarized in Table 4.2. Addition of LatA causes the zones to grow (looser confinement) and the

Case	zone size [nm]	residence time [s]
wt	36.8916 ± 21.6861	17.391760 ± 16.322162
LatA	60.3758 ± 28.3590	4.716092 ± 3.452447
MCD	29.1014 ± 9.48400	26.806853 ± 21.170700

Table 4.2: Average $\pm$ standard deviation of arrest zone sizes and residence times for all cases.

residence time is drastically reduced. MCD tightens the confinement and significantly increases the mean residence time.

4.2.5 Conclusions

Based on the presented global and local analyses, different modes of motion can consistently be identified for the VLPs [96]. Immediately after binding, the VLPs display 5 to 10 seconds of rapid, free diffusion at a rate comparable to the one observed in artificial lipid bilayers. The period of free diffusion ends with an abrupt decrease in mobility, and the particles become confined to areas with a diameter of 30 to 60 nm. Occasionally, VLPs break loose and enter another phase of free diffusion, before being trapped again in the same or a different arrest zone.

From the control experiments we see that the arrest zones do not overlap with clathrin-coated pits or caveolae and that the process of confinement does not seem to be directly linked to endocytosis [96]. Moreover, we find that inhibition of actin polymerization by LatA prevents confinement. It seems therefore likely that the confinement of particles reflects a basic property of the plasma membrane, and may be attributed to the general phenomenon of *compartmentalization* of the membrane by the cytoskeleton [258]. The "fence" or "corral" scenario involves partitioning of the cytosolic surface of the plasma membrane by tightly apposed dynamic *actin filaments*, forming a grid on the inside surface. The filaments prevent free diffusion of proteins and complexes with bulky cytoplasmic protrusions. While such proteins and complexes are free to diffuse within each partition of the grid, inter-compartment movement is restricted [235, 88, 258, 277].

In the case of VLPs, the receptors to which they bind are however constrained to the outer bilayer leaflet. It is not obvious how the trans-membrane coupling that leads to actin-mediated confinement occurs in this case. Various evidence [96] points to the explanation that each VLP clusters several ganglioside receptors and that these clusters induce the formation of *lipid rafts* [263] in the plasma membrane, such that lateral diffusion is subsequently restricted to those rafts.

We find that the initial phase of free diffusion is eliminated by depletion of cellular cholesterol by MCD. The fact that cholesterol depletion can lead to immobilization of plasma membrane components is not unprecedented in the literature. A reduction in lateral mobility of several GPI-anchored proteins, trans-membrane proteins, and lipids has been reported [304, 155, 162, 126]. One mechanism put forward to explain the effect is actin-independent, reversible agglutination of lipids into large, stable, ordered lipid domains [304, 155, 126]. On the other hand, it has been proposed that cholesterol depletion causes a drop in the $PI_{(4,5)}P_2$-level, which in turn affects the cortical actin cytoskeleton and thus lowers lateral mobility of membrane components [162]. Since LatA did not reverse the effect of cholesterol depletion, the lack of VLP mobility in cholesterol depleted cells is most likely due to the formation of large immobile membrane patches containing the receptors and the bound VLP. The latter explanation is less probable.

Taken together, our results lead to a model that begins with binding of the incoming VLP to the receptor. The lateral mobility of the complex formed is not constrained until some form of trans-bilayer coupling occurs that imposes strict, actin polymerization-dependent confinement of the VLP-receptor complex. The change is most likely caused by the addition of further components to the complex. These may form a direct bridge to the actin filaments, or a change in size or structure of the complex alone may suffice for it to become constrained. Regardless of whether the complex is trapped by actin or not, kinases and other signaling factors are recruited and proceed to turn on a cascade that eventually leads to the endocytic internalization of the VLPs.

Following and analyzing the motion of VLPs allowed us to investigate plasma membrane organization and the interplay between membrane proteins and the cytoskeleton [96].

4.3 High-Throughput analysis of Adenovirus motion

We wish to use the presented computational tools to provide a completely automated setup for data acquisition and analysis without any manual selection of samples. The large number of unbiased data points then allows to achieve high statistical significance even for small detectable changes.

This section considers human *Adenovirus-2* (Ad-2), a non-enveloped icosahedral DNA virus of about 90 nm diameter. Ad-2 infects epithelial cells of the respiratory and gastrointestinal tracts, using receptor-mediated clathrin-dependent en-

docytosis to enter the cells [141, 259]. Ad-2 relies on two different receptors: a *primary receptor* [28] and a *secondary receptor* which is an *integrin*). The virus first binds to the primary receptor from where it later transfers to the secondary receptor before internalization. The aim of the present study is to investigate the role of the secondary receptor and its influence on virus behavior on the plasma membrane. We therefore analyze the motion patterns of Ad-2 after binding to the primary receptor on human melanoma *M21 cells* under three different conditions: wild type (wt) M21 cells, genetically modified cells that lack the secondary receptor (M21L), and M21L cells with re-inserted secondary receptor (M21L4) as a control. Furthermore, we consider the following drug treatments of the cells:

- *nocodazole*, a microtubule-depolymerizing drug,
- *latrunculin A*, an actin-fiber polymerization inhibitor,
- *blebbistatin*, a myosin II inhibitor,
- *methyl-β-cyclodextrin*, a cholesterol sequestration agent,
- *jasplakinolide*, a drug inhibiting actin fiber depolymerization,
- *wiskostatin*, a drug that reversibly and selectively blocks actin polymerization, and
- *cytochalasin D*, a fungal toxin that disrupts actin filaments and inhibits actin polymerization.

Additional perturbations consider temperature (room temperature versus 32°C), and mutations of the virus: Ad-2-RAE, lacking integrin binding sites, and Ad-2-TS1, expressing a functionally defective protease. All experimental work was done in the Group of Prof. U. Greber at the University of Zürich.

The different movies were recorded at various time points within one hour after addition of the virus particles to the cells. To verify that there is no systematic bias due to transient effects, all analyses presented in this section are repeated, grouping the movies according to their time of recording. Doing this we find no correlation between the time after virus addition and any of the analysis results. This allows us to consider the whole data set at once, regardless of the time of recording.

4.3.1 Virus particle tracking

Movie frames are recorded using TIRF microscopy at a resolution of 80 nm/pixel and frame rates of either 20 Hz or 50 Hz (Greber group). A total of 342 movies is recorded at 20 Hz, and 3286 movies at 50 Hz. All movies are processed in a fully automated way using the feature point tracking algorithm outlined in Section 1.2

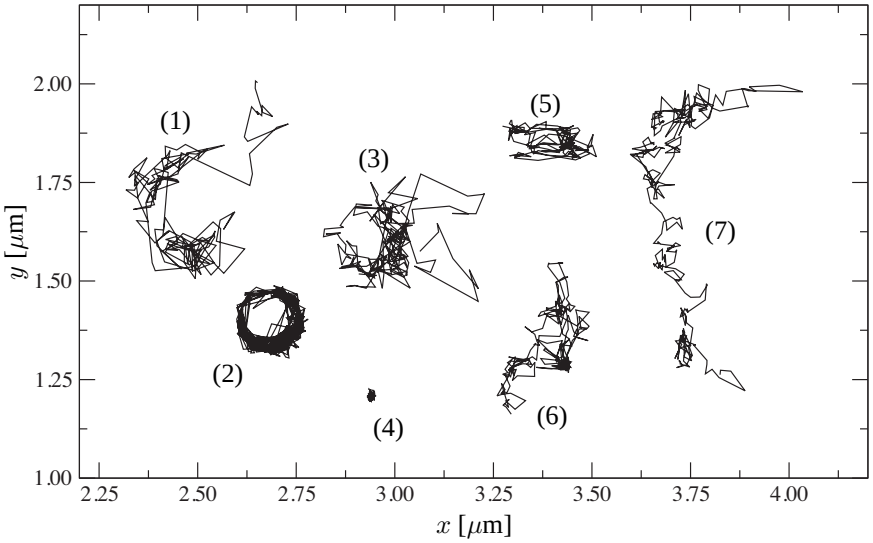


Figure 4.20: Sample trajectories of Ad-2 motion on the plasma membrane of M21 cells after binding to the primary receptor, observed using 50 Hz TIRF microscopy. Different modes of motion are observed: (1) free diffusion, (2) confinement in a slowly moving area, (3) subdiffusion, (4) arrest zones, (5)/(6) intermittent confinement, and (7) biased motion. The trajectories are represented to scale, but arbitrarily shifted for presentation purposes.

with parameters $w = 2$ pixel, $T_s = 0$, $r = 0.5\,\%$, $L = 5$ pixel, and $R = 1$ frame. Fig. 4.20 shows some examples of recorded trajectories on wt cells without drug treatment. In total, we record 9035 trajectories at 20 Hz and 46179 trajectories at 50 Hz. The 20 Hz movies consist of 1000 frames each, the 50 Hz movies contain 2535 frames each. The total number of frame images to process thus amounts to 8.7 million.

The smallest detectable diffusion constant, measured from trajectories of stationary particles on glass, is $\nu_{2,\min} = 10^{-6}\,\mu\text{m}^2/\text{s}$. The tracking uncertainty is estimated from the SNR and lies around 0.3 to 0.5 pixel in all cases.

4.3.2 Global analysis results

In the global MSS analysis according to Subsection 2.1.2, we compute the diffusion constants ν_2 and the MSS slopes β of all trajectories. Fig. 4.21 shows the

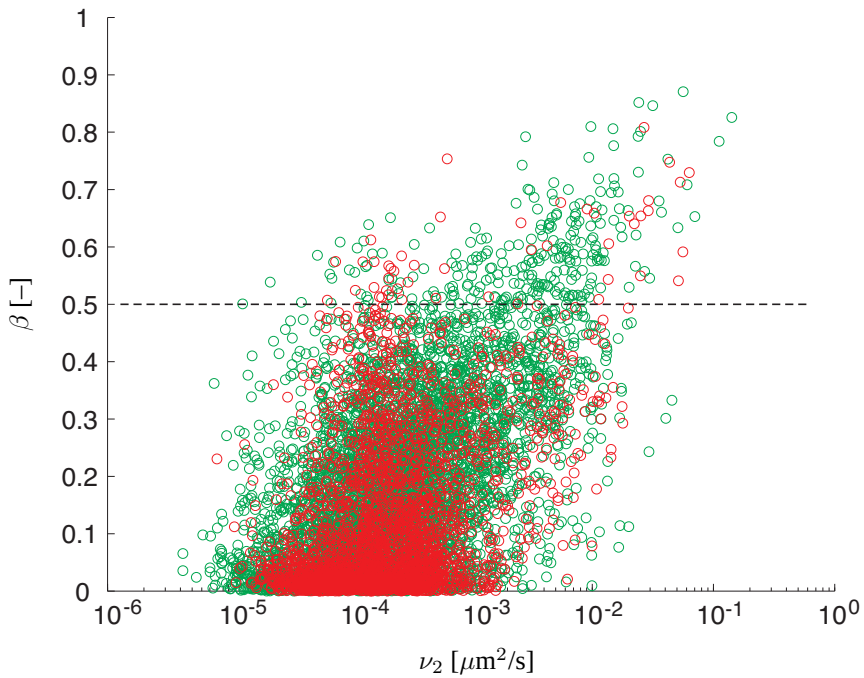


Figure 4.21: Scatter plot of global (ν_2, β) for the 20 Hz data. $N = 9035$ trajectories from the classes M21 (light gray), M21L (dark gray), and M21L4 (light gray) are shown (see text for details). The distributions of the three classes largely overlap.

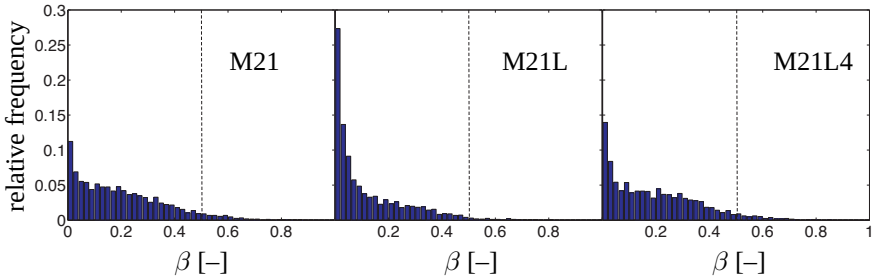


Figure 4.22: Histograms of the global MSS slope β of all 20 Hz trajectories from the three cases (see text for description). The total number of trajectories is: $N = 3313$ for M21, $N = 3208$ for M21L, and $N = 2514$ for M21L4.

(ν_2, β) scatter plot for the 20 Hz data set ($N = 9035$ trajectories). The distributions of the three classes mostly overlap, with wt M21 cells having a larger super-diffusive sub-population. This is evident from the histograms of the MSS slopes shown in Fig. 4.22, and indicates that the mobility of the virus-receptor complex is reduced in M21L cells lacking the secondary receptor. After re-addition of the secondary receptor, the wt distribution is recovered, which serves as a control for undesired side effects in the creation of the M21L cells.

4.3.3 Statistical data analysis

To investigate if the differences suggested by the global MSS analysis are *statistically significant*, we use *hypothesis testing* [271]. The proper test method or *test statistic* is selected based on information about the distribution of the data and the sample variances.

Data representation

The trajectories are encoded by their global diffusion constant ν_2 and MSS slope β. Statistical analysis is performed in this *phase space*, identified with $\mathbb{R}^2$. In statistical terminology, the observations are thus *paired*, of *interval type*, and sampled from *continuous random variables*. Furthermore, they are randomly selected since no human bias was introduced in tracking and analysis, and we assume the samples to be independent. The latter is a reasonable assumption since the trajectories come from different cells and were recorded on different days.

The standard *box plot* for the two variables is shown in Fig. 4.23. The diffusion constant ν_2 is plotted logarithmically for better visibility. We observe that the *Chambers notches* for the two classes do not overlap in any variable, providing strong evidence that the two medians differ. This is much more pronounced for β than it is for ν_2, indicating the better *discrimination capability* of the former measure.

Distribution of the data

Are the observations normally distributed? We perform a *Shapiro-Wilk test* for the *null hypothesis* (H_0) of a normal distribution. From the results as summarized in Table 4.3 it is evident that the null hypothesis of normality can be rejected for all variables with a confidence $> 99\%$. This is confirmed by the *quantile-quantile plots* for all variables and cases (figures not shown). Even with all stationary

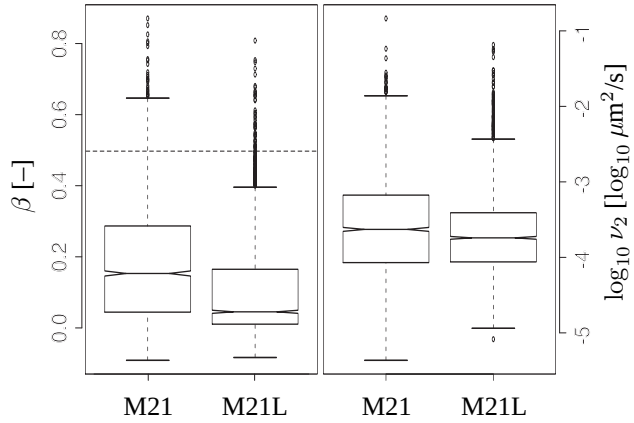


Figure 4.23: Standard box plot with Chambers notches for the MSS slope β (left) and the base-10 logarithm of the diffusion constant ν_2 (right) of M21 and M21L cells.

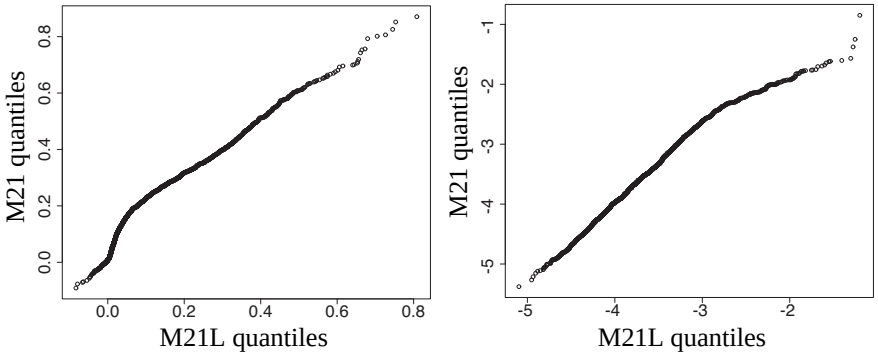
particles (characterized by $\beta < 0.1$) discarded, the data are far from a normal distribution.

Sample variance

A standard *Fligner-Killeen test* reveals that the variances of the two classes M21 and M21L are not the same, again with a confidence $> 99\%$. Among all tests for variance homogeneity, the Fligner-Killeen test is the one with the largest robustness against departures from normality [61].

variable	test statistic	p-value
β M21	0.9289	$2.2 \cdot 10^{-16}$
β M21L	0.8062	$2.2 \cdot 10^{-16}$
ν_2 M21	0.2082	$2.2 \cdot 10^{-16}$
ν_2 M21L	0.1709	$2.2 \cdot 10^{-16}$
$\log \nu_2$ M21	0.9884	$9.8 \cdot 10^{-15}$
$\log \nu_2$ M21L	0.9583	$2.2 \cdot 10^{-16}$

Table 4.3: Results of the Shapiro-Wilk normality test. The null hypothesis of normal distribution is rejected with a very high probability.



(a) (b)

Figure 4.24: Quantile-quantile plots for the MSS slopes β (a) and the diffusion constants $\log_{10} \nu_2$ (b) of M21 versus M21L cells.

Hypothesis testing

We test the null hypothesis H_0 that both classes are realizations of the same distribution against the two-sided *alternative hypothesis* H_A that they are sampled from different distributions. Since the data are not normally distributed and have different variances, standard t-tests and ANOVA tests must not be applied. Instead, we use the more general *Kolmogorov-Smirnov test*. The only assumption in this test – besides randomness and independence – is that the observations are sampled from continuous distributions, which is obviously the case. Performing the test we find that the p-values for all variables are below $2.2 \cdot 10^{-16}$, indicating that H_0 is highly unlikely. In other words, the two classes come from different distributions with a probability close to 1. The MSS analysis values of motion on M21 cells are thus significantly different from those of motion on M21L cells. Since the presence or absence of the secondary receptor is the only difference between the two cases, we can state that the receptor does significantly influence the virus trajectories.

A visual way of analyzing the differences in distribution between the two classes is to plot the *quantiles* of M21 versus the quantiles of M21L, which is done in Fig. 4.24. Due to the large number of samples, even slight departures from the straight line are significant.

Data correlations

We test whether significant *correlations* between (ν_2, β)-pairs exist in the two data sets. Using *Pearson's product moment correlation* (the "normal" r value), we find that the two variables are correlated. The null hypothesis H_0 that there is no correlation between ν_2 and β is rejected with a p-value of $2.2 \cdot 10^{-16}$. The 95% confidence interval for the correlation coefficient is found to be $[0.34, 0.40]$ for M21 and $[0.33, 0.39]$ for M21L, explaining about 14% of the variance in the data sets.

Pearson's correlation is however only appropriate if the data come from a bivariate normal distribution, which is not the case here (see above). We thus also compute *Kendall's rank correlation*, confirming that the two variables are correlated (p-value $2.2 \cdot 10^{-10}$ for uncorrelated H_0, rejected). The correlation coefficients are 0.36 for M21 and 0.20 for M21L. The difference in the correlation coefficients is significant. This result indicates that confined motion tends to be slower with a weak correlation in M21L, and a moderate one in M21.

Results

The tests show that the two classes M21 and M21L differ significantly. This is substantiated by three independent pieces of evidence: the Chambers notches indicate that the medians of the two classes are different, Fligner-Killeen tests indicate that the variances are different, and the Kolmogorov-Smirnov tests finally indicate that the functional shape of the two distributions differs, with none of them being a normal distribution. All tests are highly significant with p-values well below 10^{-6}. This striking clarity can be attributed to the large number of available data points.

Wild type M21 cells have a larger median[2] than secondary-receptor-devoid cells. This applies to both variables ν_2 and β, but is more pronounced for the latter.

We furthermore find that ν_2 and β are positively correlated, with a significantly stronger correlation in M21 cells.

4.3.4 Trajectory segmentation results

As confirmed by the statistical analysis, the secondary receptor has an influence on the virus motion. We use the trajectory segmentation technique presented in Section 2.3 to quantify the corresponding changes in the trajectory patterns.

[2] The mean would be meaningless due to the long tails of the distributions

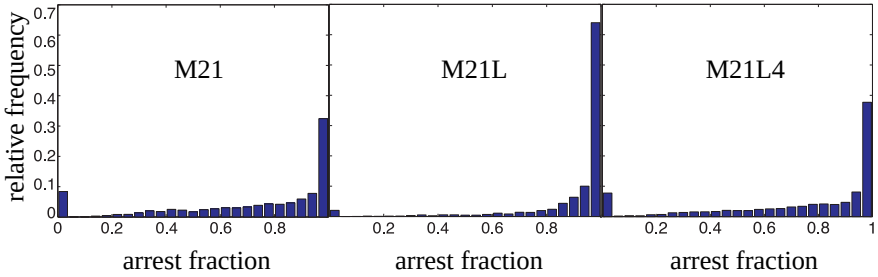


Figure 4.25: Histograms of the fraction of the total trajectory length that is spent in arrest zones for each drug case (see text for description).

In the 20 Hz data set, about 4000 arrest zones are identified on M21 cells, also about 4000 on M21L cells, and about 3000 on M21L4 cells. In order for an arrest zone to be distinguishable from a slow random walk, the latter must have an absolute MSD $> 0.1\text{pixel}^2$, corresponding to about $64\,\text{nm}^2$.

The histograms for the fraction of time spent in *arrest zones* are shown in Fig. 4.25. We find that trajectories on M21L cells are more often immobile than trajectories on the other two cell types. The fraction of trajectories that are never immobilized is about 10% in M21/M21L4 cells and drops to 2% for M21L cells. Moreover, we see that the distributions for M21 and M21L4 are almost identical, confirming that the removal of the integrin receptor did not have significant irreversible effects.

The distribution of residence times in *transient arrest zones* that ended before the end of the trajectory is shown in Fig. 4.26. The M21L4 control class is again similar to the wt distribution. Removal of the secondary receptor seems to enhance confinement with long residence times becoming more frequent. The size of the arrest zones is around 0.3 pixel in all three cases. Since this equals the tracking uncertainty due to imaging noise, the particles can be completely stationary during arrest phases. The fraction of particles that escape an arrest zone drops from 57% to 40% if the secondary receptor is absent.

The absence of the secondary receptor also influences the directed trajectory segments. The distribution of the lengths of directed segments within the trajectories of a class is shown in Fig. 4.27. While secondary-receptor-deficient cells exhibit directed motion in only about 10% of the cases, the ratio increases to around 30% for the other two cell types. The speed during directed segments is on average $2\,\mu$m/s in all three cases.

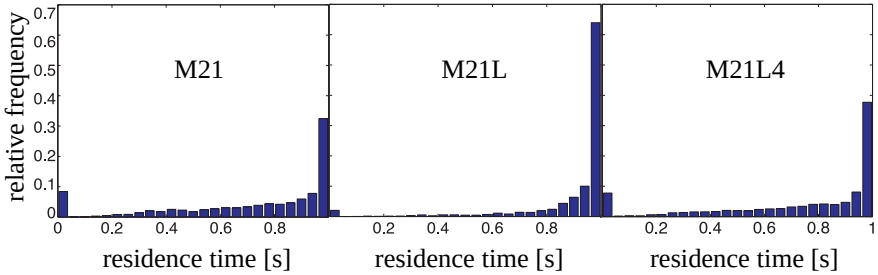


Figure 4.26: Histograms of the residence time in arrest zones for those particles that leave the zone again.

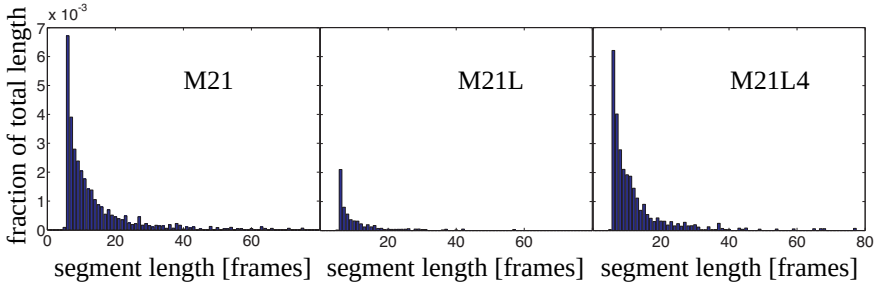


Figure 4.27: Histograms of directed segments of certain lengths. The cumulative length of all segments from each bin is compared to the total trajectory length in each class. The ratio is used as the bar height.

cell type	#events	$\langle M_\ell \rangle$	#traj	#d.m.	#p.o.i.	p	k
M21	72	406	2923	4004	4237	14.3	5.03
M21L	17	536	3110	1234	4275	3.17	5.36
M21L4	43	459	2280	3257	3383	10.5	4.10

Table 4.4: Statistics of the sit-down event (see text) and the normalized counts k as computed from the chemical analogy.

4.3.5 Event-based trajectory analysis

As outlined in Section 2.4, we can use the trajectory segmentation results to look for certain events. Arrest zones might be related to clathrin coated pits, directed segments to actin stress fibers. We are thus specifically looking for "sit-down" events, i.e. viruses moving in a directed fashion that suddenly become immobilized, and "pass-by" events, i.e. virus particles moving in a directed fashion toward or away from another, immobile virus. The counted numbers of occurrence of these events are normalized using the methods presented in Section 2.4.

Chemical analogy for the sit-down event

The absolute numbers of counted *sit-down events* in the 20 Hz data set are given in column "#events" of Table 4.4 for all three cell types. Using the absolute number of directed motion segments (#d.m.), the number of phases of immobility (#p.o.i.), the number of trajectories in each class (#traj), and the mean trajectory length $\langle M_\ell \rangle$, we can compute the normalization constants p and the *normalized counts* k according to Eq. (2.30).

Even though the absolute event counts significantly differ between the classes, the normalized frequencies are about the same. The occurrence of the sit-down event does therefore not seem to depend on the secondary receptor.

Monte Carlo simulation for sit-down event

Using a Monte Carlo simulation, the result of the chemical analogy is revisited. The number of events in the randomized trajectories (#randomized) is used to compute the *normalized count* k as shown in Table 4.5. The Monte Carlo results predict a decrease in sit-down events when the secondary receptor is removed. This contradiction to the chemical analogy indicates that directed segments and arrest zones are not statistically independent in Ad-2 trajectories. The occurrence

cell type	#events	#randomized	k
M21	72	163.8±15.2	0.402 to 0.485
M21L	17	64.9±7.6	0.235 to 0.297
M21L4	43	113.8±7.8	0.354 to 0.406

Table 4.5: Monte Carlo simulation results for the number of sit-down events in randomized trajectories (#randomized: mean ± standard deviation) with normalized counts k.

cell type	#events	E	k
M21	31	6.34	4.9
M21L	4	1.62	2.5
M21L4	21	4.6	4.6

Table 4.6: True and estimated (E) number of pass-by events in the three classes. The estimate is done for uniform conditions and represents the expected number of events in a random trajectory.

of arrest zones seems to be less likely after segments of directed motion than it would be anywhere else in the trajectory.

Estimator for the pass-by events

We use the estimator from Subsection 2.4.2 with an interaction radius of $r_m = 3$ pixel, corresponding to the size of a particle in the frame images. The effectively counted number of *pass-by events* (#events) and the expected value under uniform conditions (E) are given in Table 4.6. From this, the ratio k is computed as the *normalized count*.

The effective number of events in all cases is significantly larger than the expected value for uniformly random trajectories. Furthermore, there seems to be a difference between M21 and M21L cells. The control M21L4 is in good agreement with the wt cells. These results suggest that the pass-by event might have an underlying deterministic mechanism. The higher frequency in wt cells is consistent with the observation that the absence of the secondary receptor largely prevents directed motion (cf. Subsection 4.3.4).

4.3.6 Conclusions

We have demonstrated the feasibility of *unbiased automated high-throughput* particle tracking studies in cell biology. The large number of observations leads to statistically highly significant results for small deviations. For Ad-2 it can thus be stated with a probability close to 1 that the secondary integrin receptor has a significant influence on the motion of the virus bound to the primary receptor. In cells lacking the secondary receptor, the mobility of the particles is reduced. This is manifested by a drastic decrease in directed segments and prolonged residence in arrest zones. Neither the size of the arrest zones nor the speed of the directed motion do however seem to change. These findings could indicate that the presence of the secondary receptor facilitates (or enables) directed motion, which is supported by the observation that the upper quartile for M21 reaches above $\beta = 0.5$, whereas the one for M21L does not (Fig. 4.23). Furthermore, wt cells show a stronger correlation between the speed of motion (measure ν_2) and its freedom (measure β), as determined by Kendall's rank correlation for non-normally distributed data. This could signify that fast motion in wt cells is more likely to be directed, whereas receptor-devoid cells exhibit fast motion only by chance. The molecular nature of these effects will have to be addressed in future studies.

While no clear conclusion about the sit-down events can be made, there seems to be a deterministic tendency in the pass-by events. The normalized counts provide evidence that the secondary receptor enhances these events, most likely by increasing the probability of directed motion. This might suggest that bound viruses are transported or biased toward clathrin-coated pits, and that this transport or bias is mediated by integrins.

Part II

Dense Systems: Continuum Models

Overview

In the second part of this thesis we consider the collective motion of abundant diffusing particles for observation times that are much longer than the Brownian step time. In these cases, the continuum description holds and we focus on numerical methods for computationally solving the governing equation. We are using *particle methods* [138] to simulate diffusion in complex-shaped spaces as well as on curved surfaces. After presenting in Chapter 5 particle methods for diffusion in space and on surfaces, we apply these techniques to simulations of diffusion both in the lumen and in the membrane of the Endoplasmic Reticulum (ER). The ER is a cell organelle of highly complex shape, generally depicted as a convoluted and interconnected meshwork of tubular and lamellar structures [280]. We reconstruct realistic ER shapes from microscopy images (Section 6.2), quantify their geometric complexity (Section 6.3), and present a novel, simulation-based method to accurately determine molecular diffusion constants from Fluorescence Recovery After Photobleaching (FRAP) experiments in live cells (Section 6.5) [246, 245].

All computer programs used for the simulations in this part are based on a novel, efficient parallel software framework for the portable implementation of particle methods. This framework is presented and assessed in Chapter 7.

Governing equation of the continuum model

We consider a set of $N \to \infty$ particles undergoing *Brownian motion* [40, 92]. From continuum theory [202] we can define the particle volume density as the mean number of particles per unit volume. This density is called *concentration* and forms the key *field quantity* in diffusion problems.

The spatio-temporal evolution of the concentration field $u(\boldsymbol{x}, t)$ in a closed, bounded subset Ω of the d-dimensional *Euclidean space* E^d is modeled by the *Partial Differential Equation* (PDE)

$$\frac{\partial u}{\partial t} = \nabla \cdot (\boldsymbol{D}(\boldsymbol{x}, t) \nabla u(\boldsymbol{x}, t)) \qquad \text{for } \boldsymbol{x} \in \{\Omega \setminus \partial\Omega\}\,,\ 0 < t \leqslant T\,,$$

where $\boldsymbol{D}(\boldsymbol{x}, t)$ denotes the *diffusion tensor* and ∇ is the *Nabla operator* in E^d. This equation is termed the *diffusion equation*, since it constitutes a second order approximation to the mean field dynamics of a Brownian process [243]. Its *Green's function* naturally corresponds to the *transition density* of the underlying Brownian process, thus connecting the single-particle interpretation to the continuum model.

Terminology classifies diffusion problems based on the structure of the diffusion tensor:

- If $\boldsymbol{D}$ is proportional to the identity matrix, $\boldsymbol{D} = \nu_2 \mathbb{1}$, diffusion is called *isotropic*, otherwise *anisotropic*. Isotropic diffusion is characterized by a flux whose magnitude does not depend on its direction, and it can be described using a scalar *diffusion constant* ν_2. Microscopically, the diffusion constant is defined from the mean step displacement length a and the mean time τ between steps of the underlying Brownian motion as $\nu_2 \propto a^2/(2\tau)$. This ratio remains finite in the limit $a \to 0,\ \tau \to 0$.
- The case of a time-independent $\boldsymbol{D}$ is denoted as *stationary diffusion*, a time-dependent $\boldsymbol{D}$ gives rise to *unsteady diffusion*.
- A $\boldsymbol{D}$ that does not depend on space defines *homogeneous diffusion*. If $\boldsymbol{D}$ varies in space, diffusion is called *inhomogeneous*.

At $t = 0$ the concentration field is specified by an *initial condition*

$$u(\boldsymbol{x}, t = 0) = u_0(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega,\ t = 0\,.$$

The system is completed by problem-specific boundary conditions prescribing the behavior of u along $\partial\Omega$, the boundary of Ω. The most frequently used boundary conditions are the *Neumann* and *Dirichlet* conditions. The Neumann boundary condition fixes the normal derivative at the boundary ($\boldsymbol{n}$ is the outer unit surface normal on the boundary):

$$\frac{\partial u}{\partial \boldsymbol{n}} = \nabla u(\boldsymbol{x}, t) \cdot \boldsymbol{n} = f_N(\boldsymbol{x}, t) \qquad \text{for } \boldsymbol{x} \in \partial\Omega,\ 0 < t \leqslant T\,,$$

whereas the Dirichlet condition prescribes the concentration value

$$u(\boldsymbol{x}, t) = f_D(\boldsymbol{x}, t) \qquad \text{for } \boldsymbol{x} \in \partial\Omega,\ 0 < t \leqslant T\,.$$

If the boundary function f is 0 everywhere on $\partial\Omega$, the boundary conditions are called *homogeneous*.

Chapter 5

Particle Methods to Simulate Diffusion in Complex Geometries and on Curved Surfaces

In this chapter we briefly review particle methods for the solution of the diffusion equation in space, and extend them to simulations of diffusion on curved surfaces. After outlining the fundamental concept of continuum particle methods in Section 5.1, we recall the stochastic method of random walk (Subsection 5.2.1) and the deterministic method of *Particle Strength Exchange* (PSE) (Subsection 5.2.2) for simulating diffusion in space. Exploiting recent advances in computer graphics, we then introduce a method for simulating diffusion on curved surfaces (Section 5.3) and assess its accuracy and convergence. In Section 5.4, the new method is extended to reaction-diffusion processes on moving and deforming surfaces, and to employing *multi-resolution* concepts as introduced by Bergdorf *et al.* [27].

5.1 Fundamentals of continuum particle methods

Continuum particle methods are based on the approximation of smooth functions by integrals that are being discretized onto computational elements called *particles*. A particle p occupies a certain position $\boldsymbol{x}_p$ and carries an extensive physical quantity $\boldsymbol{\omega}_p$, referred to as its *strength*. The *particle attributes* – strength and location – evolve so as to satisfy the underlying governing equation in a *Lagrangian frame* of reference [160]. The simulation of the physical system thus amounts to tracking the dynamics of all N computational particles that carry the physical properties of the system that is being simulated. The dynamics of the particles are governed by sets of *Ordinary Differential Equations* (ODE) that determine the trajectories of the particles p and the evolution of their properties $\boldsymbol{\omega}$,

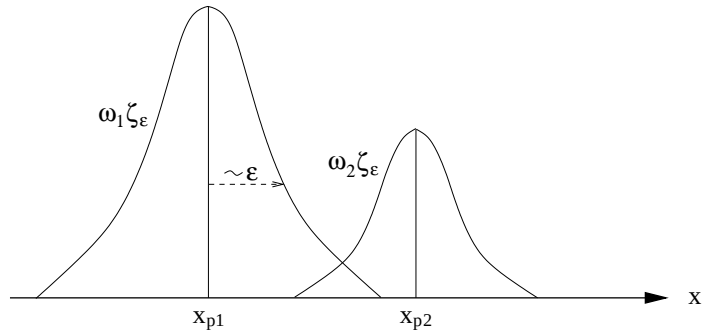


Figure 5.1: Two particles of strengths ω_1 and ω_2, carrying mollification kernels ζ_ϵ.

thus:

$$\frac{d\boldsymbol{x}_p}{dt} = \boldsymbol{v}(\boldsymbol{x}_p, t) = \sum_{q=1}^{N} \boldsymbol{K}(\boldsymbol{x}_p, \boldsymbol{x}_q; \boldsymbol{\omega}_p, \boldsymbol{\omega}_q), \qquad p = 1, \ldots, N$$
$$\frac{d\boldsymbol{\omega}_p}{dt} = \sum_{q=1}^{N} \boldsymbol{F}(\boldsymbol{x}_p, \boldsymbol{x}_q; \boldsymbol{\omega}_p, \boldsymbol{\omega}_q) \qquad p = 1, \ldots, N\,, \tag{5.1}$$

where $\boldsymbol{v}_p$ is the velocity of particle p. The dynamics of the simulated system are completely defined by the functions $\boldsymbol{K}$ and $\boldsymbol{F}$ that represent the physics of the problem. In *pure particle methods*, $\boldsymbol{K}$ and $\boldsymbol{F}$ emerge from integral approximations of differential operators; in *hybrid particle-mesh methods*, they entail solutions of field equations that are discretized on a superimposed mesh.

If the functions $\boldsymbol{K}$ and $\boldsymbol{F}$ are local, the *algorithmic complexity* of the sums in Eq. (5.1) is $\mathcal{O}(N)$. For long-range interactions, fast algorithms such as multipole expansions [122] are available to reduce the complexity to $\mathcal{O}(N)$ also in these cases. The issue of efficient parallel implementation of particle methods is addressed in Chapter 7.

5.1.1 Function approximation by particles

The approximation of a continuous function $u(\boldsymbol{x}) : \mathbb{R}^d \mapsto \mathbb{R}$ by particles can be developed in three steps:

- **Step 1: Integral Representation.** Using the Dirac δ-function identity, the function u can be expressed in integral form as

$$u(\boldsymbol{x}) = \int u(\boldsymbol{y})\,\delta(\boldsymbol{y}-\boldsymbol{x})\,d\boldsymbol{y} \qquad \text{for } \boldsymbol{x},\ \boldsymbol{y} \in \Omega\,. \tag{5.2}$$

  In *point particle methods*, this integral is directly discretized on the set of particles using a quadrature rule with the particle locations as quadrature points. Such a discretization does however not enable the recovery of the function values at locations other than those occupied by the particles.

- **Step 2: Integral Mollification.** *Smooth particle methods* circumvent this difficulty by regularizing the δ-function by a *mollification kernel* $\zeta_\epsilon = \epsilon^{-d}\zeta(\boldsymbol{x}/\epsilon)$, with $\lim_{\epsilon\to 0}\zeta_\epsilon = \delta$, that conserves the first $r-1$ moments of the δ-function identity (see Ref. [63] for details). The kernel ζ_ϵ can be thought of as a cloud or blob of mass, centered at the particle location, as illustrated in Fig. 5.1. The *core size* ϵ defines the characteristic width of the kernel and thus the spatial resolution of the method. The regularized function approximation is defined as

$$u_\epsilon(\boldsymbol{x}) = \int u(\boldsymbol{y})\zeta_\epsilon(\boldsymbol{y}-\boldsymbol{x})\,d\boldsymbol{y} \tag{5.3}$$

  and can be used to recover the function values at arbitrary locations $\boldsymbol{x}$. The *approximation error* is of order ϵ^r, hence

$$u_\epsilon(\boldsymbol{x}) = u(\boldsymbol{x}) + \mathcal{O}(\epsilon^r)\,, \tag{5.4}$$

  where r depends on the vanishing moments of the mollification kernel [63, 160]. For positive symmetric kernels, such as a Gaussian, $r = 2$ [63].

- **Step 3: Mollified Integral Discretization.** The regularized integral is discretized over N particles using the quadrature rule

$$u^h_\epsilon(\boldsymbol{x}) = \sum_{p=1}^{N} \omega^h_p \zeta_\epsilon(\boldsymbol{x}^h_p - \boldsymbol{x})\,, \tag{5.5}$$

  where $\boldsymbol{x}^h_p$ and ω^h_p are the numerical solutions of the particle positions and strengths, determined by discretizing the ODEs in Eq. (5.1) in time. The *strength* ω_p of particle p is an *extensive property* that depends on the particular quadrature rule. In this thesis we use the rectangular rule, thus setting

  $\omega_p = u(\boldsymbol{x}_p)V_p$ where V_p is the volume of particle p. Using this discretization we obtain the function approximation

$$u^h_\epsilon(\boldsymbol{x}) = u_\epsilon(\boldsymbol{x}) + \mathcal{O}\left(\frac{h}{\epsilon}\right)^s = u(\boldsymbol{x}) + \mathcal{O}(\epsilon^r) + \mathcal{O}\left(\frac{h}{\epsilon}\right)^s\,, \tag{5.6}$$

  where s depends on the number of continuous derivatives of the mollification kernel ζ_ϵ [63, 160], and h is the inter-particle distance. For a Gaussian $s \to \infty$.

From the *approximation error* in Eq. (5.6), we see that it is imperative that the distance h between any two particles be always less than their mollified support ϵ, thus maintaining

$$\frac{h}{\epsilon} < 1 \tag{5.7}$$

at all times. If this "*particle overlap*" condition is violated, the approximation error becomes arbitrarily large, and the method ceases to be well posed.

5.1.2 Operator approximation

To evaluate differential operators on particles, two strategies are distinguished: pure particle methods and hybrid particle-mesh methods.

Pure particle methods

In *pure particle methods*, differential operators on functions that are represented on particles are approximated by *integral operators*. The functions $\boldsymbol{K}$ and $\boldsymbol{F}$ in Eq. (5.1) thus represent the discretized versions of these integral operators. For diffusion, we are interested in the operators ∇^2 and $\nabla\cdot(\boldsymbol{D}\nabla)$. A conservative approximation by integral operators that allow consistent evaluation on scattered particle locations is reviewed in Section 5.2.2. Beyond diffusion, a general deterministic framework is available to approximate any differential operator by a corresponding integral [93].

In this thesis we use pure particle methods to simulate diffusion in space, as described in Section 5.2.

Hybrid particle-mesh methods

In hybrid *Particle-Mesh* (PM) methods, as pioneered by Harlow [127], some (but not all) of the differential operators are evaluated on a superimposed regular *Cartesian mesh* [138]. The functions $\boldsymbol{K}$ and $\boldsymbol{F}$ in Eq. (5.1) may thus contain contributions corresponding to the solutions of *field equations* on the mesh. Hybrid methods require:

- the *interpolation* of the $\boldsymbol{\omega}_p$ carried by the particles from the irregular particle locations $\boldsymbol{x}_p$ onto the M regular mesh points ($\boldsymbol{\omega}_m$) by:

$$\boldsymbol{\omega}_m^h = \sum_{p=1}^{N} Q(\boldsymbol{x}_m - \boldsymbol{x}_p^h)\boldsymbol{\omega}_p^h \qquad m = 1, \ldots, M\,, \tag{5.8}$$

- and the *interpolation* of the field solution $\boldsymbol{F}_m$ (and similarly $\boldsymbol{K}_m$ if present) from the mesh to the (not necessarily same) particle locations ($\boldsymbol{F}_p$):

$$\boldsymbol{F}_p^h = \sum_{m=1}^{M} R(\boldsymbol{x}_p^h - \boldsymbol{x}_m)\boldsymbol{F}_m^h \qquad p = 1, \ldots, N\,. \tag{5.9}$$

The accuracy of the method depends on the smoothness of $\boldsymbol{K}$ and $\boldsymbol{F}$, on the *interpolation functions* Q and R, and on the mesh-based discretization scheme employed for the solution of the field equations. To achieve high accuracy, the interpolation functions Q and R must be smooth to minimize *local errors*, and conserve the moments of the interpolated quantity to minimize *far-field errors* [160]. In addition, it is necessary that Q is at least of the same order of accuracy as R, to avoid spurious contributions to $\boldsymbol{F}_p^h$ [138]. This can be easily achieved by selecting the same type of interpolation, W, for both operations: $Q = R = W$. Accurate interpolation functions that conserve the moments of the interpolated quantity up to a certain order can be constructed in a systematic way [194].

In this thesis we use hybrid particle-mesh methods to simulate diffusion on curved and moving surfaces, as described in Section 5.3.

5.2 Particle Methods for diffusion in space

The simulation of spatial diffusion processes by particle methods can be formulated in the above framework, where the particles carry mass as their strength ω

and collectively represent the concentration field u. In the following, we first review the stochastic method of random walk and the deterministic *Particle Strength Exchange* (PSE) method. Using a one-dimensional test problem we then compare the accuracy and the convergence behavior of the two methods.

5.2.1 The method of Random Walk (RW)

The method of *Random Walk* (RW) [57] is based on the probabilistic interpretation of the Green's function solution of the diffusion equation:

$$u(\boldsymbol{x}, t) = \int_{-\infty}^{\infty} G(\boldsymbol{x}, \boldsymbol{y}, t) u_0(\boldsymbol{y})\, d\boldsymbol{y}\,. \tag{5.10}$$

In the case of d-dimensional isotropic free-space diffusion, i.e. $\boldsymbol{D} = \nu_2 \mathbb{1}$, Green's function is explicitly known to be:

$$G(\boldsymbol{x}, \boldsymbol{y}, t) = \frac{1}{(4\pi\nu_2 t)^{d/2}} \exp\left[-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|_2^2}{4\nu_2 t}\right]. \tag{5.11}$$

Probabilistically, G corresponds to the *transition density* as given in Eq. (2.1). This directly connects the continuum model to the single particle description of diffusion processes as described in Chapter 2. The RW method in d dimensions thus starts by either uniformly or randomly placing N particles p at initial locations $\boldsymbol{x}_p^0,\ p = 1, \ldots, N$. Each particle is assigned a strength of $\omega_p = V_p u_0(\boldsymbol{x}_p^0)$, where V_p is the particle's volume. The particles then undergo a random walk by changing their positions at each positive integer time step n according to:

$$\boldsymbol{x}_p^{n+1} = \boldsymbol{x}_p^n + \boldsymbol{\mathcal{N}}_p^n(0, 2\nu_2\delta t)\,, \tag{5.12}$$

where $\boldsymbol{\mathcal{N}}_p^n(0, 2\nu_2\delta t)$ is a vector of $i.i.d.$ Gaussian random numbers with each component having mean zero and variance $2\nu_2\delta t$; ν_2 is the molecular diffusion constant and δt is the simulation time step. Homogeneous Neumann boundary conditions can be satisfied by reflecting the particles at the boundary.

The method is consistent since the expected distribution of particle strength in space converges to the integral solution in Eq. (5.10) as we let $N \to \infty$. RW is a stochastic method. This limits its convergence capabilities since the variance of the mean of N $i.i.d.$ random variables is given by $1/\sqrt{N}$ times the individual variance of a single random variable. Moreover, the solution deteriorates with increasing

diffusion constant ν_2, since the variance of the random variables becomes larger. In the case of small ν_2, the motion of the particles can be masked by the sampling noise.

The RW method however readily extends to anisotropic diffusion processes and diffusion on curved manifolds, where the individual step displacements are simply projected onto the manifold [58].

5.2.2 The PSE method

The method of *Particle Strength Exchange* (PSE) [79, 80] is a *deterministic pure particle method* to simulate diffusion in space. As we show in Subsection 5.2.3, the PSE method is orders of magnitude more accurate than RW. PSE is based on approximating the diffusion operator by an integral operator that allows consistent evaluation on the particle locations. The PSE scheme has been devised by Degond and Mas-Gallic for both isotropic [79] and anisotropic [80] diffusion.

Isotropic diffusion by PSE

The *isotropic PSE* method [79] obtains an integral approximation to the *Laplace operator* by considering the solution at a location $\boldsymbol{y}$ and expanding it into a *Taylor series* around $\boldsymbol{x}$:

$$u(\boldsymbol{y}) = u(\boldsymbol{x}) + \sum_{i=1}^{r+1} \left[\frac{1}{i!} \Big((\boldsymbol{y}-\boldsymbol{x}) \cdot \nabla_{\boldsymbol{x'}} \Big)^i u(\boldsymbol{x'}) \right]_{\boldsymbol{x'}=\boldsymbol{x}} + \mathcal{O}\left(\|\boldsymbol{y}-\boldsymbol{x}\|_2^{r+2} \|u\|_\infty \right) . \tag{5.13}$$

Subtracting $u(\boldsymbol{x})$ on both sides, multiplying the whole equation by a *regularized kernel function* $\eta_\epsilon(\boldsymbol{x}) = \epsilon^{-d}\eta(\boldsymbol{x}/\epsilon)$ of size $\epsilon > 0$, and integrating over $\boldsymbol{y}$ yields:

$$\begin{aligned} &\int_{\mathbb{R}^d} (u(\boldsymbol{y}) - u(\boldsymbol{x}))\, \eta_\epsilon(\boldsymbol{y}-\boldsymbol{x})\, d\boldsymbol{y} = \\ &\sum_{i=1}^{r+1} \frac{1}{i!} \int_{\mathbb{R}^d} \left[\Big((\boldsymbol{y}-\boldsymbol{x}) \cdot \nabla_{\boldsymbol{x'}} \Big)^i u(\boldsymbol{x'}) \right]_{\boldsymbol{x'}=\boldsymbol{x}} \eta_\epsilon(\boldsymbol{y}-\boldsymbol{x})\, d\boldsymbol{y} \\ &\qquad + \|u\|_\infty \mathcal{O}\left(\int_{\mathbb{R}^d} \|\boldsymbol{y}-\boldsymbol{x}\|_2^{r+2} \eta_\epsilon(\boldsymbol{y}-\boldsymbol{x})\, d\boldsymbol{y} \right) . \end{aligned} \tag{5.14}$$

For the approximation to be consistent, we have to ask the following requirement for the kernel function η [79]:

$$\int_{\mathbb{R}^d} \prod_{i=1}^{d} x_i^{\alpha_i} \eta(\boldsymbol{x})\, d\boldsymbol{x} = \begin{cases} 0, & \forall\, \boldsymbol{\alpha} \in \mathbb{N}^d,\ \boldsymbol{\alpha} \neq 2\boldsymbol{e}_i,\ 1 \leqslant \sum_{i=1}^{d} \alpha_i \leqslant r+1 \\ 2, & \text{if } \boldsymbol{\alpha} = 2\boldsymbol{e}_i,\ i \in \{1,\dots,d\} \end{cases} , \tag{5.15}$$

where d is the dimension of the space, r is the order of the approximation, and $\boldsymbol{x} = (x_1, \dots, x_d) \in \mathbb{R}^d$. $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_d) \in \mathbb{N}^d$ is a d-dimensional index and $(\boldsymbol{e}_1, \dots, \boldsymbol{e}_d)$ is the *canonical basis* of $\mathbb{R}^d$. In the three-dimensional case, above requirement can be expressed as:

$$\int x_i x_j \eta(\boldsymbol{x})\, d\boldsymbol{x} = 2\delta_{ij} \qquad \text{for } i,j = 1,2,3 \tag{5.16}$$

$$\int x_1^{i_1} x_2^{i_2} x_3^{i_3} \eta(\boldsymbol{x})\, d\boldsymbol{x} = 0 \quad \text{if } i_1+i_2+i_3 = 1 \text{ or } 3 \leqslant i_1+i_2+i_3 \leqslant r+1 \tag{5.17}$$

$$\int \|\boldsymbol{x}\|_2^{r+2} |\eta(\boldsymbol{x})|\, d\boldsymbol{x} < \infty \tag{5.18}$$

for $i_1, i_2, i_3 \in \mathbb{N}_0^+$. The first condition is to normalize the kernel function. The second one requires all moments up to order $r+1$ to vanish, and the third one is required for a bounded truncation error. Using requirements (5.16) and (5.17), the only remaining terms in Eq. (5.14) are

$$\epsilon^{-2} \int_{\mathbb{R}^d} (u(\boldsymbol{y}) - u(\boldsymbol{x}))\, \eta_\epsilon(\boldsymbol{y}-\boldsymbol{x})\, d\boldsymbol{y} = \nabla^2 u(\boldsymbol{x}) + \mathcal{O}\left(\epsilon^r\right) , \tag{5.19}$$

and the *integral operator* that approximates the Laplacian is found as

$$\nabla_\epsilon^2 u(\boldsymbol{x}) = \epsilon^{-2} \int_{\mathbb{R}^d} (u(\boldsymbol{y}) - u(\boldsymbol{x}))\, \eta_\epsilon(\boldsymbol{y}-\boldsymbol{x})\, d\boldsymbol{y} . \tag{5.20}$$

While this operator is not the only possibility of discretizing the Laplacian onto particles, it has the big advantage of conserving mass exactly [63]. The approximation error is $\mathcal{O}(\epsilon^r)$ with r being the largest integer for which conditions (5.15) are fulfilled (see Ref. [63] for a rigorous error treatment). Eq. (5.20) is discretized using the particle locations as quadrature points, thus:

$$\nabla_{\epsilon,h}^2 u^h(\boldsymbol{x}_p^h) = \epsilon^{-2} \sum_{q=1}^{N} \left(V_q u_q^h - V_q u_p^h \right) \eta_\epsilon(\boldsymbol{x}_q^h - \boldsymbol{x}_p^h) , \tag{5.21}$$

where V_p is the volume of particle p. The final *PSE scheme for isotropic diffusion* then reads

$$\frac{d\omega_p}{dt} = V_p \nu_2 \epsilon^{-2} \sum_{q=1}^{N} \left(V_q u_q^h - V_q u_p^h\right) \eta_\epsilon(\boldsymbol{x}_q^h - \boldsymbol{x}_p^h) \qquad \forall\, p = 1, \ldots, N\,. \quad (5.22)$$

Since the *PSE kernel* η_ϵ is local, only the neighbors within a certain cut-off distance r_c significantly contribute to the sum of each particle. This reduces the algorithmic complexity to $\mathcal{O}(N)$ when neighbor lists such as *cell lists* (Subsection 7.2.4) or *Verlet lists* [301] are used. It can also be seen from Eq. (5.22) that in order to simulate diffusion the strengths of all the particles change, i.e. they exchange mass, while their locations remain the same, i.e. they do not move. This is dual to the method of RW and has the benefit that all the geometry and boundary condition processing only needs to be done once when initializing the particles. Therefore, we often simplify the notation by writing $\boldsymbol{x}_p$ instead of $\boldsymbol{x}_p^h$.

Anisotropic diffusion by PSE

In the case of a full diffusion tensor $\boldsymbol{D}(\boldsymbol{x}, t)$, an analogous derivation can be applied to find a deterministic particle representation of the *diffusion operator* $\nabla \cdot (\boldsymbol{D}\nabla)$. Degond and Mas-Gallic [80] have shown that the following regularized *integral operator* Q_ϵ is a consistent approximation of the anisotropic diffusion operator on a set of particles:

$$\nabla\cdot(\boldsymbol{D}\nabla u(\boldsymbol{x}, t)) \approx Q_\epsilon(t)\; u(\boldsymbol{x}, t) = \epsilon^{-2} \int_{\mathbb{R}^d} [u(\boldsymbol{y}) - u(\boldsymbol{x})] \sigma_\epsilon(\boldsymbol{x}, \boldsymbol{y}, t)\, d\boldsymbol{y}\,. \quad (5.23)$$

The regularized kernel $\sigma_\epsilon = \epsilon^{-d}\sigma(\boldsymbol{x}/\epsilon)$ again satisfies certain moment conditions [80]. The discretized particle approximation Q_ϵ^h is obtained by applying a quadrature rule to the integral operator $Q_\epsilon(t)$ using the particles as quadrature points:

$$Q_\epsilon^h(t)\; u_p^h(t) = \epsilon^{-2} \sum_{q=1}^{N} [V_q u_q(t) - V_q u_p(t)] \sigma_\epsilon(\boldsymbol{x}_p(t), \boldsymbol{x}_q(t), t)\,. \quad (5.24)$$

The regularized kernel is defined as:

$$\sigma_\epsilon(\boldsymbol{x}_p, \boldsymbol{x}_q, t) = \sum_{i,j=1}^{d} M_{ij}(\boldsymbol{x}_p, \boldsymbol{x}_q, t) \psi_{ij}(\boldsymbol{x}_q - \boldsymbol{x}_p)\,, \quad (5.25)$$

where $\boldsymbol{M} = (M_{ij}(\boldsymbol{x}, \boldsymbol{y}, t))$ is a matrix function of the diffusion tensor $\boldsymbol{D}$. For spherically symmetric kernels, a matrix smoothing function with elements

$$\psi_{ij} = \epsilon^{-2} \overline{\eta}_\epsilon\left(\boldsymbol{x}_p - \boldsymbol{x}_q\right) \cdot (\boldsymbol{x} - \boldsymbol{y})_i (\boldsymbol{x} - \boldsymbol{y})_j \quad (5.26)$$

is used [80], where $(\boldsymbol{x})_i$ denotes the i^{th} component of a vector $\boldsymbol{x}$. This reduces the system to a scalar kernel function $\overline{\eta}_\epsilon = \epsilon^{-d}\overline{\eta}(\boldsymbol{x}/\epsilon)$. Substituting into Eq. (5.25) yields the regularized anisotropic PSE kernel function

$$\sigma_\epsilon(\boldsymbol{x}_p, \boldsymbol{x}_q, t) = \epsilon^{-2}\; \overline{\eta}_\epsilon\left(\boldsymbol{x}_p - \boldsymbol{x}_q\right) \sum_{i,j=1}^{d} M_{ij}(\boldsymbol{x}_p, \boldsymbol{x}_q, t)(\boldsymbol{x}_p - \boldsymbol{x}_q)_i (\boldsymbol{x}_p - \boldsymbol{x}_q)_j\,. \quad (5.27)$$

Degond and Mas-Gallic [80] suggest $\boldsymbol{M}$ to be of the form

$$\boldsymbol{M}(\boldsymbol{x}_p, \boldsymbol{x}_q, t) = \frac{1}{2}\left(\boldsymbol{m}(\boldsymbol{x}_p, t) + \boldsymbol{m}(\boldsymbol{x}_q, t)\right)\,, \quad (5.28)$$

where

$$\boldsymbol{m}(\boldsymbol{x}, t) = \boldsymbol{D}(\boldsymbol{x}, t) - \frac{1}{d+2} \mathrm{Tr}\left(\boldsymbol{D}(\boldsymbol{x}, t)\right) \cdot \mathbb{1}\,. \quad (5.29)$$

The final *anisotropic PSE* scheme thus reads:

$$\frac{d\omega_p}{dt} = V_p \epsilon^{-4} \sum_{q=1}^{N} \Bigg[\left(V_q u_q^h - V_q u_p^h\right) \overline{\eta}_\epsilon(\boldsymbol{x}_p^h - \boldsymbol{x}_q^h) \cdot \sum_{i,j=1}^{d} M_{ij}(\boldsymbol{x}_p^h, \boldsymbol{x}_q^h, t)(\boldsymbol{x}_p^h - \boldsymbol{x}_q^h)_i (\boldsymbol{x}_p^h - \boldsymbol{x}_q^h)_j \Bigg]\,. \quad (5.30)$$

As in the isotropic case, the sum is only taken over the neighbors within a certain distance r_c, due to the local character of the interaction kernel. This is efficiently done using a *cell-list* (Subsection 7.2.4) or a *Verlet list* [301] algorithm. Since in a pure diffusion problem the particles do not move, we often write $\boldsymbol{x}_p$ instead of $\boldsymbol{x}_p^h$.

Valid spherically symmetric kernel functions $\overline{\eta}(r)$ for $d = 3$ can be found by introducing the spherical normalization constraint [80]

$$\frac{4\pi}{15} \int_0^\infty r^6\; \overline{\eta}(r)\, dr \stackrel{!}{=} 1\,. \quad (5.31)$$

This leads for example to the second-order accurate exponential kernel that is used for the simulations in this thesis:

$$\overline{\eta}_\epsilon\left(\boldsymbol{x}_p-\boldsymbol{x}_q\right)=\frac{4}{\epsilon^{-d}\pi\sqrt{\pi}}\,e^{-\frac{\|\boldsymbol{x}_p-\boldsymbol{x}_q\|_2^2}{\epsilon^2}}\,. \tag{5.32}$$

Boundary conditions

The PSE algorithm as described above only applies to infinite domains. For diffusion in constrained geometries, it needs to be modified to take into account the prescribed boundary conditions. For *homogeneous boundary conditions* in the case of flat (compared to the core size ϵ of the mollification kernel) boundaries, a straightforward method consists of placing *mirror particles* in an r_c-neighborhood outside of the simulation domain. In the resulting *method of images*, the integral operator becomes

$$\epsilon^{-2}\int_{\mathbb{R}^d}\left(u(\boldsymbol{y})-u(\boldsymbol{x})\right)\left(\Theta_\epsilon(\boldsymbol{y}-\boldsymbol{x})\pm\Theta_\epsilon(\boldsymbol{y}+\boldsymbol{x})\right)\,d\boldsymbol{y}+\mathcal{O}(\epsilon^r)\,, \tag{5.33}$$

with $\Theta_\epsilon=\nu_2\eta_\epsilon$ for the isotropic case and $\Theta_\epsilon=\sigma_\epsilon$ for the anisotropic case. The final scheme is thus represented as

$$\frac{d\omega_p}{dt}=V_p\epsilon^{-2}\sum_{q=1}^{N}(V_qu_q^h-V_qu_p^h)\left(\Theta_\epsilon(\boldsymbol{x}_q^h-\boldsymbol{x}_p^h)\pm\Theta_\epsilon(\boldsymbol{x}_q^h+\boldsymbol{x}_p^h)\right)\quad\forall\,p\,. \tag{5.34}$$

The positive sign between the two kernel functions applies for zero flux *Neumann boundary conditions*, whereas the negative sign is to be used in the case of zero value *Dirichlet boundary conditions*. The method of images is restricted to the case of homogeneous boundary conditions. For *inhomogeneous boundary conditions*, the particle strengths need to be adjusted in the vicinity of the boundary [161].

5.2.3 Comparison of PSE and RW

The convergence properties of the RW and PSE methods are illustrated on isotropic homogeneous diffusion on the one-dimensional (i.e. $d=1$) line $\Omega=[0,\infty)$, subject to the following initial and boundary conditions:

$$\begin{cases} u(x,t=0)=u_0(x)=xe^{-x^2} & x\in[0,\infty),\ t=0\\ u(x=0,t)=0 & x=0,\ 0<t\leqslant T\,. \end{cases} \tag{5.35}$$

Using the method of images, the exact solution of this problem is

$$u^{\text{ex}}(x,t)=\frac{x}{(1+4\nu_2t)^{3/2}}e^{-x^2/(1+4\nu_2t)}\,. \tag{5.36}$$

Both RW and PSE simulations of this test case are performed with a varying number of particles to study the spatial convergence behavior. In order to meet the boundary condition at $x=0$, the RW solution is calculated for $2N$ particles initially uniformly placed on the line $[-X,X]$, such that N particles have locations $x_p^0\geqslant 0$. The domain boundary X is chosen large enough such that $u(X,t)<\varepsilon$ (with ε being the *machine epsilon* of the computer) for the whole duration of the simulation. Each of the $2N$ particles is assigned a strength of $\omega_p=Xu_0(|x_p^0|)/N$. Then the particles undergo a one-dimensional random walk according to Subsection 5.2.1. To recover the solution at a later time step n, the domain of solution $[0,X]$ is subdivided into M disjoint intervals of size $\delta x=X/M$ and the particles are sampled in these intervals as follows: each interval $j=1,\ldots,M$ is assigned the sum of the strengths of all the particles having positions between $(j-3/2)\delta x$ and $(j-1/2)\delta x$, thus

$$u^{\text{RW}}((j-1)\delta x,n\delta t)=\frac{1}{\delta x}\sum_p\left\{\omega_p:(j-1)\delta x<x_p^n+\frac{1}{2}\delta x\leqslant j\delta x\right\}$$

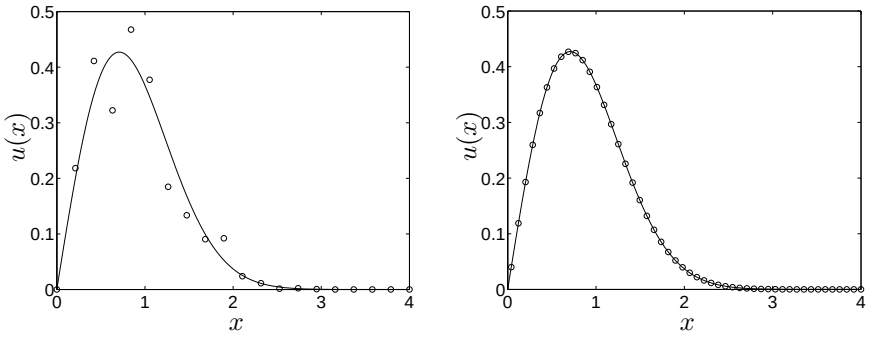
for $j=1,\ldots,M$.

For the PSE, the method as given by Eq. (5.22) is implemented. The boundary condition is treated in the same way as for the RW, i.e. the interval $[-X,X]$ is covered with $2N$ uniformly spaced particles at locations x_p, $p=1,\ldots,2N$. This is the *method of images* since it is equivalent to using mirror kernels as in Eq. (5.34). The inter-particle spacing is $h=X/(N-1)$. Initially each particle is assigned a strength of $\omega_p=Xu_0(|x_p|)/N$, as in the RW case. Eq. (5.22) is discretized in time using the explicit Euler method. The strengths of the particles are therefore updated at each time step $n=0,1,2\ldots$ as follows:

$$\omega_p^{n+1}=\omega_p^n+\frac{h\nu_2\delta t}{\epsilon^2}\sum_q\left(\omega_q^n-\omega_p^n\right)\eta_\epsilon(x_q-x_p)\qquad\forall\,p\in\{1,\ldots,2N\}\,.$$

For η_ϵ we use the 2nd order accurate Gaussian kernel

$$\eta_\epsilon(x)=\frac{1}{2\epsilon\sqrt{\pi}}e^{-x^2/4\epsilon^2}\,, \tag{5.37}$$



(a) RW (b) PSE

Figure 5.2: Comparison of RW (a) and PSE (b) solutions of the benchmark case. The solutions at time $T = 10$ are shown (circles) along with the exact analytic solution (solid line). For both methods $N = 50$ particles, a time step of $\delta t = 0.1$, $\nu_2 = 10^{-4}$, and $X = 4$ are used. The RW solution is sampled in $M = 20$ intervals of $\delta x = 0.2$. For the PSE a core size of $\epsilon = h$ is used.

which fulfills the requirements in Eq. (5.15) in one dimension at order $r = 2$. The concentration values at particle locations x_p and simulation time points $t_n = n\delta t$ are recovered as

$$u^{\text{PSE}}(x_p, t^n) = \omega_p^n \cdot N/X \, .$$

Fig. 5.2 shows the RW and PSE solutions at a final time of $T = 10$ for $N = 50$ particles and a diffusion constant of $\nu_2 = 10^{-4}$. The accuracy of the simulations for different numbers of particles is assessed by computing the final L_2 *error*

$$L_2 = \left[\frac{1}{N} \sum_{p=1}^{N} \left(u^{\text{ex}}(x_p, T) - u(x_p, T) \right)^2 \right]^{1/2} \tag{5.38}$$

for each N. The resulting convergence curves are shown in Fig. 5.3. For the RW we observe the characteristic slow convergence of $\mathcal{O}(1/\sqrt{N})$ [191]. For the PSE, a convergence of $\mathcal{O}(1/N^2)$ is observed, in agreement with the employed 2nd order kernel function. Below an error of 10^{-6} machine precision is reached. It can be seen that the error of the PSE simulations is several orders of magnitude lower

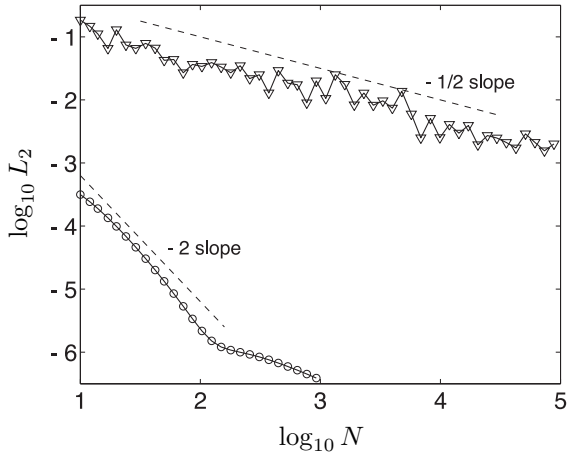


Figure 5.3: Convergence curves for RW and PSE. The L_2 error versus the number of particles for the RW (triangles) and the PSE (circles) solutions of the benchmark case at time $T = 10$ are shown. For both methods a time step of $\delta t = 0.1$, $\nu_2 = 10^{-4}$, and $X = 4$ are used. The RW solution is sampled in $M = 20$ intervals of $\delta x = 0.2$ and for the PSE a core size of $\epsilon = h$ is used. The machine epsilon is $\mathcal{O}(10^{-6})$.

than the one of the RW simulations for the same number of particles. Using only 100 particles, the PSE is already close to machine precision. It is evident from these results that large numbers of particles are necessary to achieve reasonable accuracy using RW in complex-shaped domains.

5.3 A level-set particle method for diffusion on curved surfaces

In computational science a number of techniques have been proposed to solve the *diffusion equation on curved surfaces*, requiring rectangular grids [1], surface triangulations [3], or using local representations and overset grid techniques [253]. These explicit techniques allow a piecewise linear representation of the surface and encounter severe difficulties in tracking large surface deformations. Monte Carlo techniques [58] for the simulation of diffusion processes suffer from slow convergence rates and they are not competitive with their deterministic counterparts for simulations in complex geometries [160, 246].

The simulation of diffusion on curved surfaces has also received considerable

attention in the area of computer graphics. We exploit recent advances from image and video inpainting [32] by representing surfaces implicitly using particle *level set* techniques [137]. The key concept amounts to considering the surface as a level set of a higher-dimensional scalar function. The resulting governing equations are solved in a Cartesian coordinate system spanning a region consisting of all points close to the surface. This technique has been recently employed for the simulation of isotropic diffusion on the plasma membrane of hl-60 cells [253].

Mathematically, we consider the diffusion of a scalar quantity u on a *Riemannian manifold* $\mathcal{M} \subset \mathbb{R}^d$ as governed by

$$\frac{\partial u(\boldsymbol{\xi}, t)}{\partial t} = \mathcal{L}_D u(\boldsymbol{\xi}, t) \qquad \boldsymbol{\xi} \in \mathcal{M}\,, \tag{5.39}$$

where the *intrinsic diffusion operator* on $\mathcal{M}$ is defined as

$$\mathcal{L}_D = \nabla_{\mathcal{M}} \cdot (\boldsymbol{D}(\boldsymbol{\xi}, t) \nabla_{\mathcal{M}}(\cdot))\ . \tag{5.40}$$

$\nabla_{\mathcal{M}}$ is the *intrinsic Nabla* operator on the surface $\mathcal{M}$ and $\boldsymbol{D}(\boldsymbol{\xi}, t)$ is the *diffusion tensor*. If the surface is closed and finite, no boundary conditions are required. We wish to discretize Eq. (5.39) onto particles.

In this chapter, we occasionally use the *summation convention* to keep the notation compact. In this convention, matrices are represented by their elements with the first index denoting the row and the second one the column, thus:

$$\boldsymbol{A} = (a_{ij})\,.$$

Moreover, all products are implicitly summed over all indices appearing more than once.

$$a_{ij} b_{jk} = \sum_j (a_{ij} b_{jk})$$

thus is the summation notation for the matrix product $\boldsymbol{AB}$.

5.3.1 Previous approaches

In order to discretize Eq. (5.39), i.e. to represent the differential operator $\mathcal{L}_D$, a parametrization of the manifold $\mathcal{M}$ is needed. In the simplest case a *global parametrization*

$$\boldsymbol{f} : \boldsymbol{\xi} = (\xi_i) \in \mathcal{M} \mapsto \boldsymbol{f}(\boldsymbol{\xi}) \subseteq \mathbb{R}^d \tag{5.41}$$

is available. This directly allows to compute the *Riemann metric*

$$g_{ij} = \frac{\partial f_k}{\partial \xi_i} \frac{\partial f_k}{\partial \xi_j} \tag{5.42}$$

and the *intrinsic Laplacian* on $\mathcal{M}$, given by

$$\nabla^2_{\mathcal{M}} = \frac{1}{\sqrt{|g|}} \frac{\partial}{\partial \xi_i} \left(\sqrt{|g|} g^{ij} \frac{\partial}{\partial \xi_j} \right)\ . \tag{5.43}$$

Hereby, $|g|$ denotes the determinant of g_{ij} and $g^{ij} = g_{ij}^{-1}$ its inverse. For arbitrary manifolds, global parametrizations however usually do not exist (e.g. for all objects homeomorphic to a sphere), or can not be explicitly determined.

Local parametrizations such as normal coordinates, local quadratic approximations, or splines can generally be found. They do however suffer from a number of shortcomings such as numerical instabilities in the case of normal coordinates, asymmetry in local quadratic approximations, or algorithmic complexity for the connectivity information required by splines.

5.3.2 Surface representation

The different ways of representing the surface $\mathcal{M}$ in the computer can be classified with respect to the connectivity information needed. *Triangulated surfaces* are an example of connectivity-based representations. Since establishing the connectivity information is computationally expensive, these meshes are preferably used in finite element methods for diffusion on surfaces [20]. Connectivity-less methods include scattered point clouds [124] and implicit representations [256].

We make use of the *implicit surface representation* technique, also called *level set* method. Hereby, the surface is given by the zero level of a smooth *level function* $\psi \in C^1 : \mathbb{R}^d \mapsto \mathbb{R}$, thus $\mathcal{M} = \{\boldsymbol{x} : \psi(\boldsymbol{x}) = 0\}$. For reasons of efficiency, we usually choose ψ to be the *signed distance function* for which

$$\|\nabla \psi\|_2 = 1\,. \tag{5.44}$$

Since $\psi(\boldsymbol{x} \in \mathcal{M}) = 0$, the function value of the signed distance function directly reflects the orthogonal distance to the surface.

5.3.3 Embedding in $\mathbb{R}^d$

As shown by Bertalmio *et al.* [32], the diffusion Eq. (5.39) on the surface $\mathcal{M}$ can be transformed into a PDE for generalized anisotropic diffusion in the surround-

ing $\mathbb{R}^d$. This is achieved by *embedding* the manifold in a small annular domain called "*band*", consisting of all points close to the original surface. The respective differential operators in $\mathbb{R}^d$ can then directly be discretized inside the band using particle methods [137, 160]. The embedding transformation works by constraining the fluxes to the tangential direction using the projection map

$$\boldsymbol{T} = \left(\mathbb{1} - \frac{\nabla\psi \otimes \nabla\psi}{\|\nabla\psi\|_2^2} \right) \|\nabla\psi\|_2 \,. \tag{5.45}$$

Here, ∇ is the regular Nabla operator in $\mathbb{R}^d$. The initial condition $u(\boldsymbol{\xi}, t=0)$ is only known on $\mathcal{M}$. It is thus extended to the band around $\mathcal{M}$ by solving to steady state the PDE

$$\frac{\partial u}{\partial t} + \text{sign}(\psi)(\nabla u \cdot \nabla\psi) = 0 \qquad \text{in } \mathbb{R}^d \,. \tag{5.46}$$

This enforces that the direction of the *diffusive flux* ∇u is orthogonal to the normal on $\mathcal{M}$, $\nabla\psi$, such that the extension is neutral with respect to the above mapping operator $\boldsymbol{T}$. The embedded governing equation for anisotropic *diffusion on the surface* $\mathcal{M}$ thus becomes

$$\frac{\partial u}{\partial t} = \frac{1}{\|\nabla\psi\|_2} \nabla \cdot \left(\boldsymbol{T}\widetilde{\boldsymbol{D}}\nabla u \right) \qquad \text{in the band,} \tag{5.47}$$

where ∇ is the regular Nabla operator in $\mathbb{R}^d$ and the tensor $\widetilde{\boldsymbol{D}}(\boldsymbol{x}, t)$ is obtained from the diffusion tensor $\boldsymbol{D}(\boldsymbol{\xi}, t)$ on the surface by extending it with an arbitrary radial component that is invariant under the projection map.

5.3.4 Level set reinitialization

The signed distance function to any surface can be constructed from an arbitrary smooth level function using *reinitialization*. This refers to the process of replacing the old ψ by a newly constructed one. For the new ψ to be the signed distance function, it has to be the solution of the *Eikonal equation*

$$\|\nabla\psi(\boldsymbol{x})\|_2^2 = 1 \qquad \text{in the band,} \tag{5.48}$$

or, equivalently, the steady-state solution of the PDE

$$\frac{\partial \psi}{\partial t} + \text{sign}(\psi)(\nabla\psi \cdot \nabla\psi - 1) = 0 \,. \tag{5.49}$$

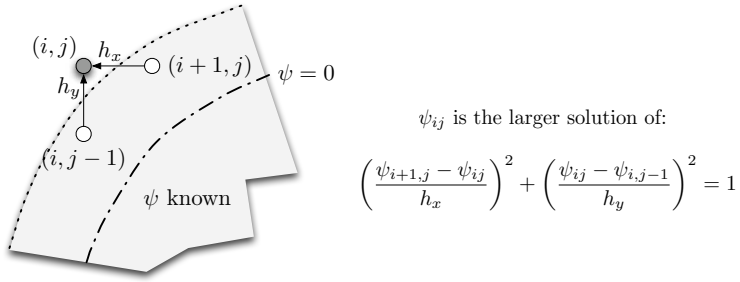


Figure 5.4: Principle of the fast marching method. The level function ψ is assumed to be known in a band (shaded in gray) around the surface $\psi = 0$. The algorithm successively enlarges the region where ψ is known. A new point is computed by solving the quadratic equation that emerges from the upwind discretization of $\|\nabla\psi\|_2^2 = 1$. To preserve causality, the points are updated in order of ascending distance to the surface.

Since the band is of finite width, this procedure requires an extrapolating method. The prevalent extrapolating narrow band algorithm for signed distance functions is the *Fast Marching Method* (FMM), introduced by Sethian in 1996 [255].

The FMM is a grid-based level set algorithm. It starts from all grid points immediately adjacent to the surface $\psi(\boldsymbol{x}) = 0$, where the level function is assumed to be known. The FMM thus needs to be initialized by computing the orthogonal distance to the surface for all points immediately adjacent to the surface. We use the second-order accurate surface locating algorithm by Chopp [56], employing tri-cubic interpolation near the surface, to determine these initial distance values. As a by-product, this algorithm also yields the location and the distance of the closest point on the implicit surface (*closest point transform*), which can be used to reconstruct function values on $\mathcal{M}$ or to compute the quadrature of a function along $\mathcal{M}$ [289].

From the first layer of points, the FMM successively expands the band in which the correct ψ is known as outlined in Fig. 5.4. The ψ values of new points are computed by solving the quadratic equation formed by the upwind finite-difference discretization of Eq. (5.48). To satisfy *causality*, the FMM updates the points in order of ascending distance ψ to the surface. This ensures that the upwind differences are only using values that are not going to change any more in future updates.

The original FMM uses first order *upwind differences* and requires the points

to be sorted according to their distance to the surface. The latter property prohibits a parallel implementation on distributed computer systems and renders the method inherently sequential. Sorting is avoided by the marching *Eikonal solver* of Kim [156], called the *Group Marching Method* (GMM). It advances several points per iteration and can therefore be implemented on parallel computers. Avoiding the global sorting also reduces the algorithmic complexity of the method from $\mathcal{O}(N \log N)$ to $\mathcal{O}(N)$, where N is the total number of points. To satisfy causality without sorting the points, each point is computed r times with r being the order of the upwind scheme. From all solutions, the one with the smallest absolute value is used, since the orthogonal distance to the surface is the shortest distance. The details of the algorithm are described in the original publication [156].

We note that upwind *WENO* [148] schemes are a viable alternative to the GMM. WENO schemes of up to fifth order have successfully been used for level set applications [214]. Because they constitute an iterative method that operates in the whole domain, rather than just in the narrow band around the surface, they are however usually more expensive than the GMM.

5.3.5 Orthogonal extension of the solution

In order to construct the initial condition in the band around $\mathcal{M}$ according to Eq. (5.46), we need to extrapolate the function u enforcing $\nabla u \cdot \nabla \psi = 0$. This extrapolation is referred to as *orthogonal extension* of u over ψ.

Extension corresponds to solving the Eikonal-like equation

$$\nabla u(\boldsymbol{x}) \cdot \nabla \psi(\boldsymbol{x}) = 0 \tag{5.50}$$

for $u(\boldsymbol{x})$. Orthogonal extension can thus be done with the same algorithms as reinitialization. In the GMM and FMM, the quadratic equation emerging from the upwind discretization of $\nabla\psi \cdot \nabla\psi$ is replaced by the linear equation from discretizing $\nabla u \cdot \nabla \psi$, and the right-hand side is changed from 1 to 0. Since ψ is known, upwind differences only need to be used in ∇u, and $\nabla\psi$ can be approximated using higher-order centered differences.

5.3.6 Convergence of the level-set algorithms

Both, the high-order initialization procedure of Chopp [56] and the GMM [156], modified to higher order, are implemented in the parallel PPM software library (cf. Chapter 7) for both two and three dimensions. All methods are available

for convergence orders 1, 2, and 3. Routines for orthogonal function extension, level function reinitialization, and closest point transform have been added. The accuracy of the present implementations is tested on the extension of the signed distance function to the unit sphere in $\mathbb{R}^3$. The analytical signed-distance function is $\widetilde{\psi} = \sqrt{x^2 + y^2 + z^2} - 1$, whereas the algorithm reconstructs ψ. Using the point-wise error $e = \psi - \widetilde{\psi}$ on the points in the narrow band, the following *relative error* measures are computed from all points adjacent to the surface:

$$L_2 = \frac{1}{\max_i |\widetilde{\psi_i}|} \left[\frac{1}{N} \sum_{i=1}^{N} e_i^2 \right]^{1/2} \tag{5.51}$$

$$L_\infty = \frac{1}{\max_i |\widetilde{\psi_i}|} \max_i |e_i| \,. \tag{5.52}$$

To test the orthogonal extension, the spherical harmonic $u = Y_1^0$ as given in Eq. (5.62) is extrapolated from the surface of the sphere into the band. The resulting convergence curves are shown in Fig. 5.5 for both the 2nd order and the 3rd order orthogonal extension GMM. The straight lines indicate the theoretical convergence slopes of order 2 and 3, respectively.

The convergence of the reinitialization algorithm is assessed and compared to the values published by Chopp [56]. Fig. 5.6(a) shows the convergence of the surface locating step to determine the initial values for the GMM using tri-cubic interpolation [56], and Figs. 5.6(b)–5.7(b) show the convergence of the GMM reinitialization in comparison to Chopp's higher order FMM [56].

5.3.7 Formulation of the numerical scheme

We formulate the numerical scheme for diffusion on surfaces within the framework of hybrid particle-mesh methods [127, 138]. Convection as used in Subsections 5.4.3 and 5.4.4 is treated by particle motion, whereas all level set operations and the diffusion operator are discretized on a Cartesian mesh with spacings (h_i) in all spatial directions i. We choose to evaluate the diffusion operator on regular locations because the convergence rate of the anisotropic PSE scheme deteriorates if the band is narrower than about 10ϵ (M. Bergdorf, personal communication, 2005). A band of minimal width is however a key property of the present method as the computational cost grows proportionally to the band width, and the capability of resolving fine surface structures decreases with increasing band width. Moreover, the level set algorithms described above require the data to be available

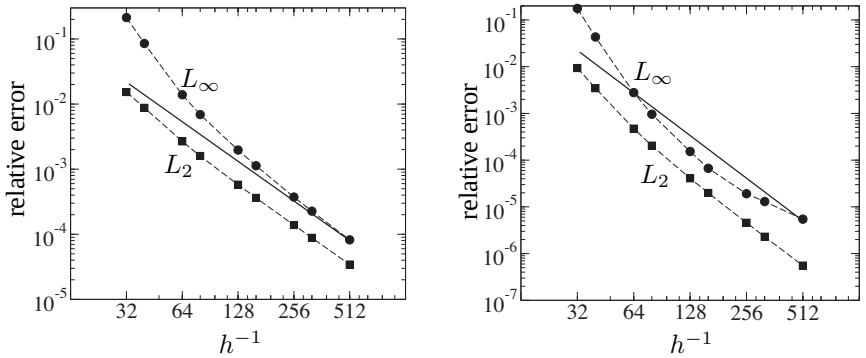


Figure 5.5: Convergence of the second order (a) and third order (b) orthogonal extension. The spherical harmonic $u = Y_1^0$ (Eq. (5.62)) is initialized in a band of half-width $3h$ and orthogonally extended outwards to a $9h$ band. This is done over the exact signed distance function $\widetilde{\psi}$, initialized in a $12h$ band. Second order centered differences are used to approximate $\nabla\psi$ in the extension. To compute the error, ∇u is approximated using 4^{th} order centered differences in the band $3h < |\widetilde{\psi}| < 6h$, and $\nabla\widetilde{\psi}$ is known analytically.

on regular locations. *Interpolation* as outlined in Subsection 5.1.2 is used to map the *property vectors* (u_p^h, ψ_p^h) between the regularly spaced mesh nodes and the irregular particle locations $\boldsymbol{x}_p^h$.

The method adapts to the shape of the surface as particles are only present in the narrow band $|\psi| < k$. Inside this band, $\nabla\psi$ is computed on the mesh using second-order *upwind differences* to avoid boundary errors, and the projected diffusion tensors $\boldsymbol{\Lambda} = \boldsymbol{T}\widetilde{\boldsymbol{D}}$ at all particle locations are determined. The right-hand side of Eq. (5.47) is computed in an *inner band* of half-width $\kappa < k$. The region where $\kappa \leqslant |\psi| < k$ serves as a *boundary layer*. After each time step, the solution u in the boundary layer is reconstructed using the second-order extension scheme as described in Subsection 5.3.5.

Discretization of the embedded Eq. (5.47) on the mesh is done for an arbitrary, space-dependent tensor $\boldsymbol{\Lambda}$ by using *Lagrange interpolation* polynomials for both u and $\boldsymbol{\Lambda}$ (element-wise). In order to obtain a second-order accurate operator discretization of minimal support, we choose a quadratic polynomial basis in each

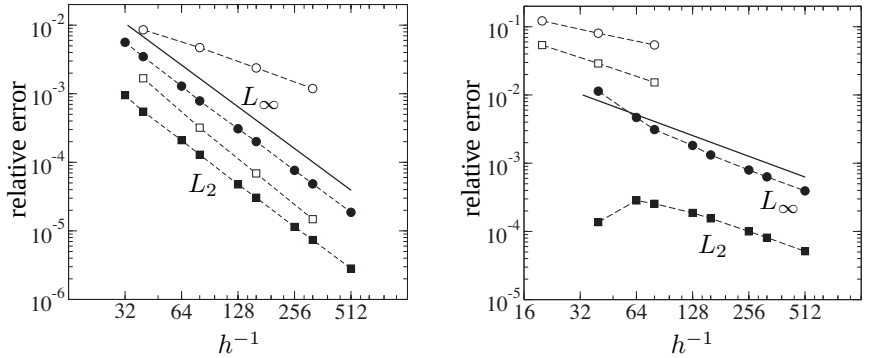


Figure 5.6: (a) Convergence of the surface locating algorithm [56], applied to the signed distance function around the unit sphere. The present implementation (filled symbols) is compared to the errors published by Chopp [56] (open symbols) and the 2^{nd} order scaling (solid line). (b) Convergence of the first order GMM. A signed distance function is extended from a band of half-width $3h$ to one of $9h$. All errors are computed within $3h < |\widetilde{\psi}| < 9h$. The present GMM (filled symbols) is compared to the errors published by Chopp [56] (open symbols) for a standard FMM, and the 1^{st} order scaling (solid line).

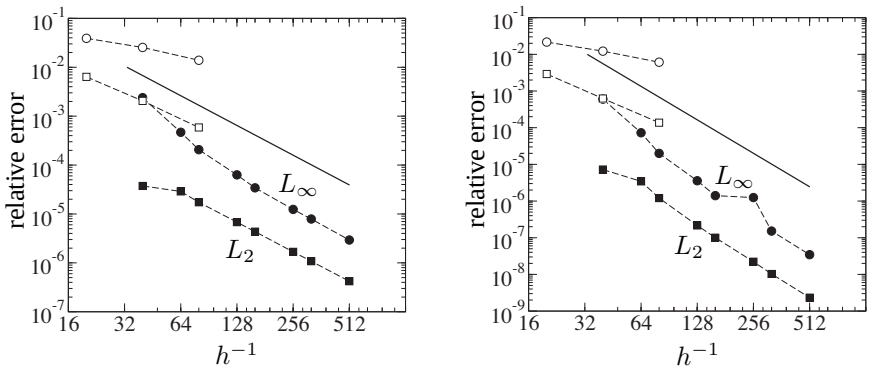
spatial direction i, as shown in Fig. 5.8:

$$l_{-1}(x_i) = \frac{1}{2}\frac{x_i\,(x_i - h_i)}{h_i^2} \tag{5.53}$$

$$l_0(x_i) = -\frac{(x_i + h_i)\,(x_i - h_i)}{h_i^2} \tag{5.54}$$

$$l_1(x_i) = \frac{1}{2}\frac{x_i\,(x_i + h_i)}{h_i^2}\,, \tag{5.55}$$

where the coordinate $\boldsymbol{x}$ is relative to each grid node (locally centered basis). The unknown function u and all elements of the matrix $\boldsymbol{\Lambda}$ are expressed as continuous functions in the above basis. In three dimensions, let $u^h(i,j,k)$ and $\boldsymbol{\Lambda}^h(i,j,k) = (\lambda_{mn}^h(i,j,k))$ be the discrete representations evaluated at mesh node (i,j,k). The



(a) (b)

Figure 5.7: Convergence of the second order (a) and third order (b) GMM. A signed distance function is extended from a band of half-width $3h$ to one of half-width $9h$. All errors are computed in the band $3h < |\widetilde{\psi}| < 9h$. The present GMM (filled symbols) is compared to the numbers published by Chopp [56] (open symbols) for a higher order FMM, and the theoretical scaling (solid line).

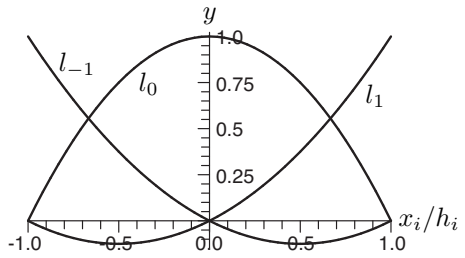


Figure 5.8: The three quadratic polynomial basis functions used in each spatial direction x_i. The polynomials are interpolating at the locations $-h_i$, 0, and h_i, with the coordinates locally centered at each mesh node. Higher-dimensional bases are formed by Cartesian products of the depicted polynomials.

interpolated continuous functions then are:

$$\widetilde{u}(\boldsymbol{x}) = \sum_{k=-1}^{1}\sum_{j=-1}^{1}\sum_{i=-1}^{1} u^h(i,j,k) l_i(x_1) l_j(x_2) l_k(x_3) \tag{5.56}$$

$$\widetilde{\lambda}_{mn}(\boldsymbol{x}) = \sum_{k=-1}^{1}\sum_{j=-1}^{1}\sum_{i=-1}^{1} \lambda^h_{mn}(i,j,k) l_i(x_1) l_j(x_2) l_k(x_3)\,. \tag{5.57}$$

These functions are used to symbolically compute the right-hand side of Eq. (5.47). Evaluating the resulting expression at the center node $\boldsymbol{x} = \mathbf{0}$ yields the final discretized form of the operator:

$$\nabla\cdot(\boldsymbol{\Lambda}(\boldsymbol{x})\nabla u(\boldsymbol{x})) \approx \left[\nabla\cdot\left(\widetilde{\boldsymbol{\Lambda}}(\boldsymbol{x})\nabla\widetilde{u}(\boldsymbol{x})\right)\right]_{\boldsymbol{x}=\mathbf{0}}\,. \tag{5.58}$$

By construction, this discretized operator has a compact support consisting of 27 particles in three dimensions, which corresponds to an interaction radius of only $1h$. The resulting stencil weights can be pre-computed, but may depend on space as $\boldsymbol{\Lambda}$ can be a function of $\boldsymbol{x}$. For $\boldsymbol{\Lambda} = \mathbb{1}$ in three dimensions, the standard 7-point stencil for the Laplacian is recovered.

Using more than three points in each direction, and thus a higher-order polynomial basis, allows to systematically construct higher-order operator discretizations.

5.3.8 Convergence of the diffusion method

The order of accuracy of the method described in Subsection 5.3.7 is determined on a test problem with known analytic solution. We consider isotropic diffusion on the surface of the unit sphere $S^2 \subset \mathbb{R}^3$, governed by

$$\frac{\partial u}{\partial t} = \nu\nabla^2_{\mathcal{M}} u \qquad \text{on } S^2\,. \tag{5.59}$$

The analytic solution in spherical coordinates (defined according to Bronstein *et al.* [38]) is obtained by *expansion to eigenfunctions* as

$$u(t,\vartheta,\varphi) = \sum_{\ell=0}^{\infty}\sum_{m=-\ell}^{\ell} c_{\ell m}(0) Y_\ell^m(\vartheta,\varphi) e^{-\nu\ell(\ell+1)t}\,, \tag{5.60}$$

with coefficients

$$c_{\ell m}(0) = \int_{S^2} (-1)^m Y_\ell^{-m}(\vartheta,\varphi)\, u(0,\vartheta,\varphi)\, dS\,. \tag{5.61}$$

To study convergence without the effects of series truncation, we use the special initial condition

$$u(0,\vartheta,\varphi) = Y_1^0(\vartheta,\varphi) = \sqrt{\frac{3}{4\pi}}\cos\vartheta\,, \tag{5.62}$$

for which the analytic solution simplifies to

$$u(t,\vartheta,\varphi) = Y_1^0(\vartheta,\varphi)e^{-2\nu t}\,, \tag{5.63}$$

due to the orthogonality of the spherical harmonics.

The boundary layer $\{\boldsymbol{x} : \kappa \leqslant |\psi(\boldsymbol{x})| < k\}$ is reinitialized after each time step using the second order orthogonal extension GMM, thus enforcing $\nabla\psi \cdot \nabla u = 0$ as described in Subsection 5.3.5.

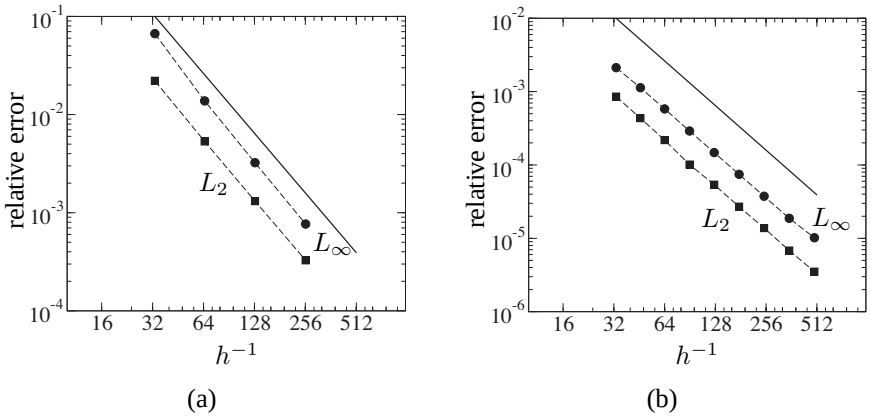
The convergence of the operator discretization given in Eq. (5.58) is shown in Fig. 5.9(a). The exact right-hand side is derived symbolically, using the known exact solution of the test problem and the known form of the diffusion tensor. Convergence for the complete diffusion solution after 10 time steps is shown in Fig. 5.9(b). To reconstruct the solution on the surface of the sphere from the mesh nodes in the band, we use linear interpolation along all grid lines that intersect the surface. It can be seen that the method is second order accurate for all grid sizes tested. The serial (1 processor) and the parallel (tested using 4 processors) versions of the implementation yield exactly the same results.

The presented method can be used to simulate diffusion on arbitrary surfaces. As an example, we consider the *Stanford bunny*[1]. The initial condition consists of a TIFF image, that is projected onto the bunny's surface (skin?). The values range between 0 and 1, with sharp transitions. Fig. 5.10 visualizes the solution at different times. The concentration on the surface is recovered from adjacent nodes using linear interpolation along grid lines that intersect the surface. An application to a problem of biological interest is given in Subsection 6.5.3.

5.3.9 Conservation of mass

While the PSE method in space is analytically conservative, the present surface diffusion method does not conserve mass exactly. Numerical errors in the orthogonal extension of the solution, as well as the interpolation to recover the solution on the surface, lead to mass drift. Using the global *re-scaling method* proposed by Xu *et al.* [322], conservation of mass can however be enforced. The surface

[1] source: http://graphics.stanford.edu/software/scanview/models/bunny.html



(a) (b)

Figure 5.9: Convergence of the discretized differential operator of Eq. (5.58) in the narrow band (a) and for diffusion on the unit sphere (b). The intrinsic diffusion constant is $\nu = 1$. The diffusion operator is evaluated on a support of $3 \times 3 \times 3$ particles in a narrow band of half-width $\kappa = 1h$ with an additional boundary layer of $1h$, thus $k = 2h$. Extension to the boundary layer is done using the 2^{nd} order GMM extension method. Time integration uses a 2^{nd} order TVD Runge-Kutta scheme with a time step of $\delta t = 10^{-5}$ until final time 10^{-4}. The solid lines indicate convergence of order 2.

(a) (b) (c)

Figure 5.10: Solution of the Stanford bunny test case at times $t = 0$ (a), $t = 2 \cdot 10^{-5}$ (b), and $t = 2 \cdot 10^{-4}$ (c). The intrinsic diffusion constant is $\nu = 0.1$, and the diffusion operator is supported on $3 \times 3 \times 3$ particles in a narrow band of half-width $\kappa = 3h$ and extended to a larger band of $k = 4.5h$ every 10 time steps, using the 2^{nd} order GMM extension method. An Euler scheme is used for 254 time steps of size $\delta t = 10^{-6}$. The solution is discretized using 2 million particles distributed in the band $|\psi| < k$.

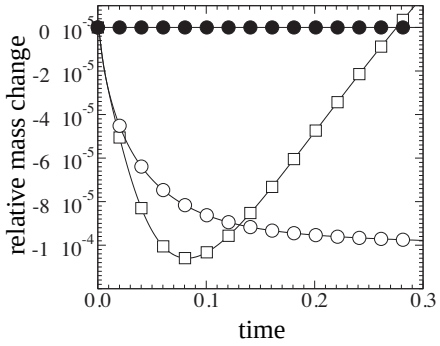


Figure 5.11: Evolution of the total mass on surface of the unit sphere with intrinsic diffusion constant $\nu = 1$. The diffusion operator is evaluated on a support of $3 \times 3 \times 3$ particles in a narrow-band of half-width $\kappa = 3h$. For the cases marked by circles, the solution is extended to a larger band of $k = 4.5h$ after each time step, using the 2^{nd} order GMM extension method. No extension is applied to the squares case. An Euler scheme is used with a time step of $\delta t = 10^{-4}$ until final time 0.3. The open symbols mark the case of no re-scaling, for the filled symbols the solution is re-scaled [322] at each time step.

integrals in this method are evaluated using linear interpolation along grid lines and the rectangular quadrature rule. Fig. 5.11 shows the total mass over time for the above sphere test case with initial condition

$$u(0, \vartheta, \varphi) = 1 + Y_1^0(\vartheta, \varphi)\,. \tag{5.64}$$

This initial condition is extended into the band using the second order GMM. If no orthogonal extension to the boundary layer is applied between time steps, the mass grows linearly. Even after the solution has reached its steady state, and no gradients are present any more, the mass continues to grow, which eventually leads to instabilities. Using extension, but no re-scaling, the mass approaches an asymptotic level as the solution reaches its steady state. Using both extension and re-scaling, the mass remains constant to machine precision for all times.

Since the mass drift depends on the local curvature of the surface, the global re-scaling method [322] is only exact for surfaces of constant curvature, such as a sphere. In other cases, conservation of mass is still enforced (by construction), but the solution is altered.

5.4 A multi-resolution particle method for reaction-diffusion on deforming surfaces

In the following we extend the numerical method described in the previous section to *reaction-diffusion* systems on *moving surfaces*. We consider reaction-diffusion systems governed by equations of the *Fisher-KPP* [102, 159] type:

$$\frac{\partial u_i}{\partial t} - \nabla_{\mathcal{M}} \cdot (\boldsymbol{D}_i \nabla_{\mathcal{M}} u_i) = f_i(\boldsymbol{u})\,, \tag{5.65}$$

where $\mathcal{M}$ is allowed to change over time. The concentration vector $\boldsymbol{u}$ contains one entry u_i per chemical species i and the diffusion tensors $\boldsymbol{D}_i$ are allowed to vary among species. All chemical reactions are described by the source terms $f_i \in C^1$.

After reviewing previous simulation methods and applications in biology, we start by describing the extension of the present scheme to reaction-diffusion systems, followed by the extension to moving surfaces. Finally, we present a multi-resolution implementation where the particle sizes are locally adapted to the surface.

5.4.1 Previous approaches and applications in biology

Coupled reaction-diffusion systems exhibit interesting stability properties that can give rise to the formation of stable concentration patterns called *Turing patterns* [293], or traveling *waves* [230].

Twenty years after the seminal work of Turing [293], Gierer and Meinhardt used reaction-diffusion systems to formulate their theory of *pattern formation* in biology [111]. They introduced the *Gierer-Meinhardt model*, which has become one of the most widely used pattern formation models, with significant applications also in computer graphics [294].

The first biological applications of reaction-diffusion models considered *morphogenesis* [293], following the idea that coupling of reaction-diffusion patterns to growth or motion could explain the geometries and shapes found in nature. Computer simulations that link pattern formation to growth and morphogenesis were studied by Marée [185]. After reviewing the morphogenesis modeling literature until 2000, Marée applied hybrid cellular automata-PDE simulations to explain stalk formation and cell differentiation in *slime mold*.

Computer simulations of reaction-diffusion patterns on *surfaces* first considered the unit sphere [51]. The numerical method was based on expanding the functions in terms of *spherical harmonics*, thus limiting its applicability to spherical

objects. The biological application that was considered in the original publication concerned growth of globular tumors [51].

Morphogenesis of more complex surfaces was simulated by Harrison *et al.* using a finite element method to solve the reaction-diffusion equation on triangulated surfaces [128]. The method allows to treat shapes as complex as branched unicellular algae. Moving-grid finite element techniques were later used to directly couple the motion of the boundary to the reaction-diffusion patterns on continuously deforming two-dimensional domains [181]. A different approach uses the solution of an interior Poisson problem to evolve the surface shape [180].

Besides morphogenesis, reaction-diffusion models also have important applications in cell motility [123] and cell modeling [221]. Miura *et al.* combined theory, experiments, and one-dimensional finite difference simulations to study pattern formation in cell cultures using reaction-diffusion models [193]. Emerging applications concern simulations of *cell signaling* pathways [33]. Since the first ODE model of the *chemotaxis* pathway in *Escherichia coli* was published by Bray *et al.* in 1993 [36], computer simulations have become increasingly more sophisticated in resolving spatial phenomena. A recent model by Lipkow *et al.* [172] explicitly includes diffusion of the key signal transduction molecule in the cytosolic space. The mobility of membrane-bound species was however not accounted for. Other reaction-diffusion signaling models include the *sporulation* control network model of Marwan [188], and the plant shoot meristem simulations of Jönsson *et al.* [149].

5.4.2 Reaction-diffusion in the present numerical method

We extend the simulation scheme outlined in Section 5.3 to reaction-diffusion problems on surfaces, as governed by Eq. (5.65). All components of the concentration vector $\boldsymbol{u}$ are hereby represented on the same set of computational particles, supporting property vectors $(\boldsymbol{u}_p^h, \psi_p^h)$.

Evaluating the *reaction terms* f_i amounts to a purely local exchange of strength among species at the same location. Reactions are thus evaluated independently for each particle within the narrow band. The rate of exchange between different u_i is directly given by the *reaction kinetics*, that are evaluated using either a deterministic method based on kinetic ODEs or a stochastic method such as the Gillespie SSA algorithm [113]. The latter is possible because individual particles constitute homogeneous reaction spaces since no spatial gradients are present within a particle. The deterministic solver makes use of the same time integrator as the diffusion and is thus restricted by the time step stability limit. The stochastic solver directly operates on (fictitious) molecule numbers and is used outside of the time integrator's right-hand side.

An example with moving reaction fronts

As an illustrative example we consider the reaction $a + b \to 2a$ with rate constant k. If we identify $[a] = u$ and normalize the total concentration to 1 everywhere, we have $f(u) = ku(1-u) \in C^1$. For the present reaction it is known that the solution of Eq. (5.65) has a traveling front $u(\boldsymbol{x}, t) = U(\boldsymbol{x} \cdot \boldsymbol{n} - st)$ with speed s along the local interface normal $\boldsymbol{n}$ [25, 26]. Such wave solutions exists for all speeds $s \geq s^* = 2\sqrt{f'(0)} > 0$ [25, 26]. For $s < s^*$, no fronts exist. If u is non-negative and continuous, the front thus propagates with constant speed

$$s^* = 2\sqrt{k}\,. \tag{5.66}$$

For the stochastic simulations we let X be the total number of molecules contained in a particle, and define an analog to *Avogadro's number*, viz. M, the number of molecules per unit mass. The solution u has the physical units of concentration, thus $u \sim X/(VM)$. The function f has units concentration/time, thus $k \sim 1/(ut) \sim (VM)/(Xt)$. Gillespie introduces the product $\mathfrak{h}c$ as the expected number of reactions per unit time, thus $\mathfrak{h}c \sim 1/t$. For the binary reaction above it is $\mathfrak{h} = X_a X_b$ and thus $c \sim 1/(Xt)$. For the relation between k and c we find $k = MVc$. This corresponds to Eq. 7b in Ref. [113] under the normalization $M = 1$. We interpret the *propensity* c as the probability that two molecules of species a and b react, provided they meet in space and time. This probability is independent of the volume and does not need to be adjusted if particles change size[2].

The reaction-diffusion system is completely described by the following set of dimensionless numbers:

$$\Pi_1 = \frac{V}{\ell^3}\,, \quad \Pi_2 = \frac{Mc\ell^3}{k}\,, \quad \Pi_3 = \frac{\nu_2}{\ell^2 k u}\,, \quad \Pi_4 = \frac{s\ell}{\nu_2}\,, \tag{5.67}$$

where V is the volume of the particle, ℓ the characteristic length scale of the problem, and s the front propagation speed. Π_1 is a dimensionless volume, Π_2 the ratio of stochastic and deterministic rate constants, Π_3 the ratio between diffusive and reactive mass exchange, and Π_4 the dimensionless front velocity.

[2] The probability of an encounter to occur however does depend on the volume. This is automatically accounted for in $\mathfrak{h}$ as the number of molecules X per particle changes if the particle is dilated or compressed.

We use the method presented in this chapter to simulate the example system on the surface of the unit sphere using the stochastic SSA algorithm [113] to evaluate the reaction terms. The initial condition is such that one half of the sphere contains only a, the other half only b. Reactions occur along the interface between the two species. Due to diffusion, the interface thickens up and, due to the reactions, it propagates into the region of b. At the end of the simulation, the sphere contains 100% a as all of b has been consumed by the reaction. For the case of $M = 10$ molecules per unit mass, Fig. 5.12(a) shows the total mass, integrated over the surface of the sphere, of a and b as they evolve in time. The front position is also shown, defined as the location where $[a] = [b] = 0.5$. As long as reactions occur, $[a]$ and $[b]$ are changing and the front travels at a more or less constant speed s, given by the slope of the dashed curve in Fig. 5.12(a). If the dimensionless front speed is plotted against the dimensionless reaction propensity, the curves for different diffusion constants ν_2 collapse as shown in Fig. 5.12(b). We also observe that the theoretical scaling predicted by Eq. (5.66) is well approximated, particularly if the reaction is at least a factor of 100 faster than the diffusion, corresponding to a *reaction-dominated system*.

5.4.3 Moving and deforming surfaces

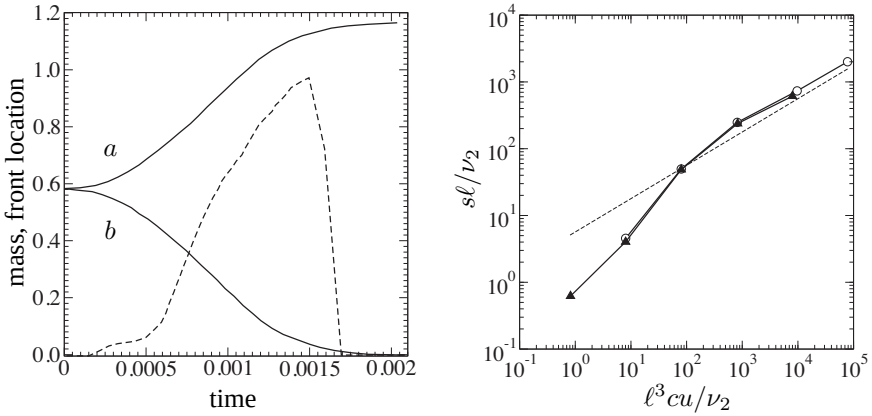
The particle character of the present method allows straightforward extension to moving and *deforming surfaces* $\mathcal{M}(t)$. To account for surface motion, the particle positions are changed according to the *velocity field* $\boldsymbol{v}(\boldsymbol{x}, t)$ of the deformation, thus:

$$\frac{d\boldsymbol{x}_p}{dt} = \boldsymbol{v}(\boldsymbol{x}_p, t)\,. \tag{5.68}$$

Since particles are only present in the narrow band of half width k, surface deformations can lead to compression or dilation of the band. The former would lead to surfaces breaking open, whereas the latter would eventually cause the method to become inconsistent as the overlap condition Eq. (5.7) ceases to be satisfied. The level set is thus reinitialized according to Subsection 5.3.4 after each convection step, and the particles are remeshed onto regular locations. All properties are interpolated using moment-conserving kernels as described in Subsection 5.1.2.

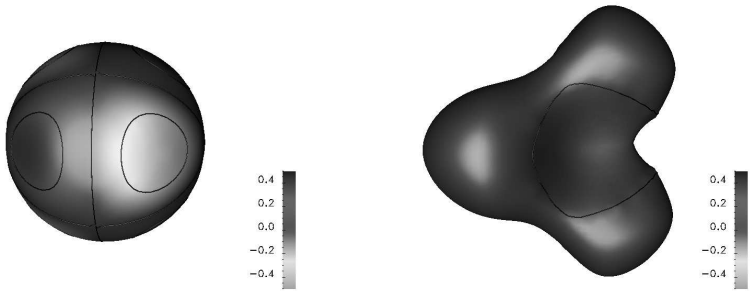
We test the method on the unit sphere by diffusing the initial condition

$$\Re(Y_3^2) = \frac{1}{4}\sqrt{\frac{105}{2\pi}}\sin^2\vartheta\cos\vartheta\cos(2\varphi)\,. \tag{5.69}$$



(a) (b)

Figure 5.12: (a) Evolution of the total mass of a and b (solid lines) for $M = 10$ molecules per unit concentration. See text for problem description. The location of the reaction front is shown by the dashed curve. The reaction front moves into the region of b until all of b is consumed. (b) Dependence of the front speed s on the reaction propensity c. In dimensionless numbers, the two curves for $\nu_2 = 0.1$ (open circles) and $\nu_2 = 1.0$ (filled triangles) collapse into one. The slope of the theoretical scaling according to Eq. (5.66) is indicated by the dashed line.



(a) (b)

Figure 5.13: Solution of the sphere test case at time $t = 0$ (a), and the final steady state at $t = 0.03$ (b). Diffusion on the surface with $\nu = 0.2$ is coupled to surface deformation according to the velocity field in Eq. (5.70) with $C = 0.25$. The diffusion and convection operators are evaluated in a narrow band of $\kappa = 2h$ and extended to $k = 3h$ every time step. A 9 stage STS Euler scheme is used for 301 time steps of size $\delta t = 10^{-4}$. Lines on the surface indicate the -0.25, 0.0, and 0.25 concentration iso-lines.

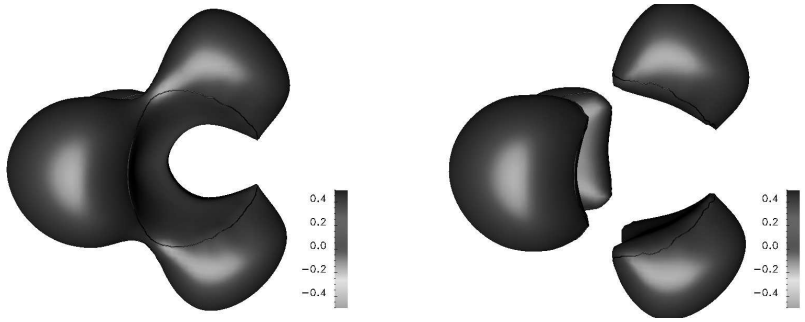
At the same time, the surface of the sphere deforms according to the velocity field

$$\boldsymbol{v}(\boldsymbol{x}, t) = Cu(\boldsymbol{x}, t)\boldsymbol{n}(\boldsymbol{x}, t)h/\delta t\,, \tag{5.70}$$

where $\boldsymbol{n}$ is the instantaneous outer unit normal on the surface. As diffusion homogenizes the concentration field, the velocity approaches zero and there is a stable steady-state shape. Fig. 5.13 shows an example simulation with $\nu = 0.2$ and $C = 0.25$. Starting from the same initial condition, but increasing C to 0.5 causes the shape to be torn apart before it reaches the steady state (Fig. 5.14). This demonstrates the ability of the employed particle level-set formulation to handle *topology changes* in the surface. This becomes important in biological applications where membrane fusion and fission are key processes.

5.4.4 Multi-resolution particles

The employed narrow-band level set method imposes a *resolution constraint* on the surface geometry: The bands from two opposite parts of the surface must never overlap, i.e. the smallest "feature" of the surface must be at least $2k$ in diameter.



(a) (b)

Figure 5.14: Solution of the same test case with $C = 0.5$ at times $t = 0.02$ (a) and $t = 0.03$ (b). The larger velocity causes topology changes in the surface before it reaches its steady-state shape. Lines on the surface indicate the -0.25, 0.0, and 0.25 concentration iso-lines.

If a uniform resolution is used, it becomes prohibitively expensive to resolve large complex-shaped geometries. Moreover, such high resolutions impose a stringent time step limit in order to maintain stability of the method.

Adaptive global map

We use the *Adaptive Global Map* (AGM) method, introduced by Bergdorf *et al.* in 2005 [27], to locally adjust the particle sizes with small particles close to the surface and larger ones in the outer part of the band. The AGM method is based on postulating a *reference space* $\hat{\Omega} \subseteq \mathbb{R}^d$ where the particles are uniformly distributed on a regular Cartesian grid of spacing $\hat{h}$, and all have the same volume $\hat{V}$. The *physical space* $\Omega \subseteq \mathbb{R}^d$ supports the irregularly spaced particles with adapted volumes. AGM entails a smooth *mapping function* $\boldsymbol{f} \in C^1 : \hat{\Omega} \mapsto \Omega$, which maps the reference space $\hat{\Omega}$ onto the physical space Ω. This mapping function defines the physical locations of the reference space particles by

$$\boldsymbol{x} = \boldsymbol{f}(\hat{\boldsymbol{x}})\,, \qquad \hat{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{x})\,. \tag{5.71}$$

The *Jacobian* of this map is

$$\boldsymbol{\Phi} : \phi_{ij} = \frac{\partial \hat{x}_i}{\partial x_j}\,, \qquad J = |\boldsymbol{\Phi}| = \det(\boldsymbol{\Phi})\,, \tag{5.72}$$

and defines the physical volumes of the particles as

$$V_p = \frac{1}{J(\boldsymbol{x}_p)} \prod_{i=1}^{d} \hat{h}_i \,. \tag{5.73}$$

In AGM, the mapping function $\boldsymbol{f}$ is represented on the same set of particles as the solution u, thus extending the property vector of the particles by one element to $(\boldsymbol{u}_p^h, \psi_p^h, \boldsymbol{f}_p^h)$. The function $\boldsymbol{f}$ is determined by the AGM method as the steady-state solution of the PDE

$$\frac{\partial \boldsymbol{f}}{\partial t} = \hat{\nabla} \cdot \left(M(\hat{\boldsymbol{x}}, t) \hat{\nabla} \boldsymbol{f}^\top \right) , \tag{5.74}$$

with the *monitor function* M guiding the resolution. This equation is solved to steady state using an implicit time integration scheme [27]. The monitor function M has to be non-negative and smooth. We choose the recommended form [27]

$$M(\boldsymbol{x}) = \sqrt{1 + \alpha \chi(\boldsymbol{x})} \,, \tag{5.75}$$

where α is a parameter and χ is a smoothly truncated *indicator function* that decays from the value 1 in the inner band to 0 outside. We choose:

$$\chi(\boldsymbol{x}) = \begin{cases} \exp\left[-\left(\psi(\boldsymbol{x})/\kappa \right)^4 \right] & , |\psi| < k \\ 0 & , \text{otherwise} \,. \end{cases} \tag{5.76}$$

The AGM determines the map $\boldsymbol{f}$ such that M becomes equi-distributed in reference space. The resolution in physical space is thus high (small particles) where M attains large values, while small values of M lead to coarse resolution with large particles. Above choice of the monitor function causes the AGM to concentrate the particles within the narrow band, leading to more efficient memory use and a relaxed narrow-band resolution constraint.

Diffusion operator

In the AGM method, all differential operators (diffusion, curvature, level set reinitialization, orthogonal extension, etc.) are evaluated in reference space. Particle-to-mesh interpolation (remeshing) is also done in reference space, where the grid is uniform. The level function ψ is a signed distance function in reference space,

but not in physical space. In order to simulate diffusion on the surface $\psi(\boldsymbol{x}) = 0$, we transform the governing Eq. (5.47) to reference space, where it becomes

$$\frac{\partial u}{\partial t} = \frac{J}{\|\nabla \psi\|_2} \hat{\nabla} \cdot \left(\boldsymbol{\Lambda} \hat{\nabla} u \right) \qquad \text{in } \hat{\Omega} \,, \tag{5.77}$$

and $\hat{\nabla}$ is the regular Nabla operator in reference space. The transformed diffusion tensor is given by

$$\boldsymbol{\Lambda} = J^{-1} \boldsymbol{\Phi} \boldsymbol{T} \widetilde{\boldsymbol{D}} \boldsymbol{\Phi}^\top \tag{5.78}$$

and can directly be used to evaluate the discretized operator of Eq. (5.58) in reference space.

Proof 1 *To simplify the notation, let $\boldsymbol{B} = \boldsymbol{T}\widetilde{\boldsymbol{D}}$. The right-hand side of Eq. (5.77) then is:*

$$\begin{aligned} & J \hat{\nabla} \cdot \left(\boldsymbol{\Lambda} \hat{\nabla} u \right) = \\ & J \hat{\nabla} \cdot \left(J^{-1} \boldsymbol{\Phi} \boldsymbol{B} \boldsymbol{\Phi}^\top \hat{\nabla} u \right) = \\ & J \frac{\partial}{\partial \hat{x}_i} \left(J^{-1} \phi_{ij} b_{jk} \phi_{lk} \frac{\partial u}{\partial \hat{x}_l} \right) = \\ & \qquad J \frac{\partial}{\partial \hat{x}_i} \left(J^{-1} \phi_{ij} \right) \cdot \boldsymbol{\varphi} + J J^{-1} \phi_{ij} \frac{\partial}{\partial \hat{x}_i} \left(b_{jk} \phi_{lk} \frac{\partial u}{\partial \hat{x}_l} \right) \end{aligned} \tag{5.79}$$

for a specific vector function $\boldsymbol{\varphi}$. By virtue of

$$\frac{\partial}{\partial \hat{x}_i} \left(J^{-1} \phi_{ij} \right) \cdot \boldsymbol{\varphi} = 0 \tag{5.80}$$

for any vector function $\boldsymbol{\varphi}$ (Eq. (20) in Ref. [64]), the first summand vanishes to 0. The remaining second summand is identical to

$$\boldsymbol{\Phi}^\top \hat{\nabla} \cdot \left(\boldsymbol{B} \boldsymbol{\Phi}^\top \hat{\nabla} u \right) , \tag{5.81}$$

which is the correct diffusion operator, transformed to reference space. ■

Surface curvature

To compute the local curvature $\hat{\kappa}(\boldsymbol{x})$ of the surface, we note that

$$\hat{\kappa} = \nabla \cdot \left(\frac{\nabla \psi}{\|\nabla \psi\|_2} \right) = \boldsymbol{\Phi}^\top \hat{\nabla} \cdot \left(\frac{1}{\|\boldsymbol{\Phi}^\top \hat{\nabla} \psi\|_2} \boldsymbol{\Phi}^\top \hat{\nabla} \psi \right) . \tag{5.82}$$

The curvature can thus be determined using the same operator as given in Eq. (5.58) with the *virtual diffusion tensor*

$$\boldsymbol{\Lambda} = \boldsymbol{\Lambda}_{\hat{\kappa}} = \frac{1}{J \|\boldsymbol{\Phi}^\top \hat{\nabla} \psi\|_2} \boldsymbol{\Phi} \boldsymbol{\Phi}^\top . \tag{5.83}$$

This tensor is not symmetric. Symmetry is however not required for the numerical scheme or the operator discretization of Eq. (5.58).

Convection operator

In order to move the particles in reference space, the physical velocity $\boldsymbol{v}$ needs to be transformed. The adaptation of the map causes an apparent motion, with the *adaptation velocity* for $\boldsymbol{v} = \boldsymbol{0}$ given by:

$$\boldsymbol{\mathcal{V}} = \frac{\partial \boldsymbol{x}_p^h(t)}{\partial t} \approx \frac{\boldsymbol{x}_p^n - \boldsymbol{x}_p^{n-1}}{\delta t} . \tag{5.84}$$

The transformed velocity in reference space is then given by [27]:

$$\hat{\boldsymbol{v}} = \boldsymbol{\Phi} \left(\boldsymbol{v} - \boldsymbol{\mathcal{V}} \right) . \tag{5.85}$$

Due to the adaptation velocity, the level function becomes distorted and has to be reinitialized according to Subsection 5.3.4 after every adaptation step, even if the physical velocity $\boldsymbol{v}$ is zero.

5.4.5 Algorithm

Using the methods and schemes outlined in this chapter, the complete algorithm for multi-resolution simulations of reaction-diffusion on moving surfaces becomes:

Algorithm 1 (Multi-resolution reaction-diffusion on moving surfaces)

Initialize fields $\boldsymbol{x}_p^0$, $\hat{\boldsymbol{x}}_p^0 = i\hat{h}$, ψ_0, *and* u_0.
`Loop` $n = 0, \ldots, T$ *with time step size* δt*:*
  Create particles from the fields ψ_n *and*
    u_n *using* $J_n = |\boldsymbol{\Phi}(\boldsymbol{x}_p^n)|$ *to scale values*
    to strengths. Particles have
    locations $\hat{\boldsymbol{x}}_p^n$ *and carry strengths:*
    $\boldsymbol{\omega}_p^n = (\boldsymbol{u}_p^n, \psi_p^n)/J_n$.
  Adapt the map using AGM: $\boldsymbol{x}_p^n \to \boldsymbol{x}_p^{n+1}$.
  Convert the physical velocity $\boldsymbol{v}$ *to reference*
    space velocity $\hat{\boldsymbol{v}}$ *using the old Jacobian:*
    $\hat{\boldsymbol{v}} = \boldsymbol{\Phi}(\boldsymbol{x}_p^n)(\boldsymbol{v} - \boldsymbol{\mathcal{V}})$ *with* $\boldsymbol{\mathcal{V}} = (\boldsymbol{x}_p^{n+1} - \boldsymbol{x}_p^n)/\delta t$.
  Move the particles using the reference space
    velocity $\hat{\boldsymbol{v}}$*:* $\hat{\boldsymbol{x}}_p^n \to \hat{\boldsymbol{x}}_p^{n+1}$.
  Interpolate particles to regular mesh using
    $J_{n+1} = |\boldsymbol{\Phi}(\boldsymbol{x}_p^{n+1})|$ *to scale strengths*
    to values. New fields $\to \hat{\boldsymbol{x}}_p$, ψ_{n+1}, u_n.
  Reinitialize ψ *in reference space, even if the physical*
    velocity $\boldsymbol{v} = 0$.
  Compute deterministic reaction and diffusion terms
    and update u*:* $u_n \to u_{n+1}$.
  Compute stochastic reaction terms.
`End`

All steps in the above algorithm are performed in reference space. The physical position of the particles is however always available from the AGM mapping function, and the particle volumes can be computed from the Jacobian according to Eq. (5.73). This allows to reconstruct the concentration field u in physical space at any time.

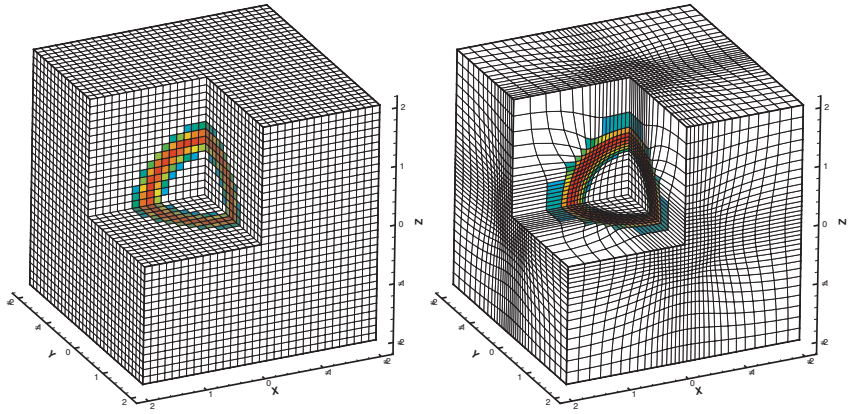
In order to initialize the physical locations $\boldsymbol{x}_p^0$ of the particles, the following initial AGM adaptation is performed:

Algorithm 2 (Initial adaptation)

initialize $\boldsymbol{x}_p = i\hat{h}\,, i = 0, \ldots, (N-1)$.
initialize the adaptation time step τ.
$\rho = 2 \cdot$*TOL.*
`While` $\rho >$*TOL:*
  compute monitor function at particle locations: M_p.

$\boldsymbol{x}_{n-1} \leftarrow \boldsymbol{x}_n$.
$\boldsymbol{x}_n \leftarrow AGM(\boldsymbol{x}_{n-1}, M_p, \tau)$.
compute the motion residual
$\rho = \frac{1}{N\hat{h}} \sum_{p=1}^{N} \|\boldsymbol{x}_p^n - \boldsymbol{x}_p^{n-1}\|_2$.
End

After this initial adaptation, the initial conditions for the level function ψ and all concentration strengths can be determined. Fig. 5.15 shows an example where the resolution is adapted to a narrow band of physical half-width $2\hat{h}$ around the unit sphere. Since ψ is the signed distance function in reference space, the physical width of the narrow band that is needed to evaluate the differential operators and level set algorithms is considerably smaller after AGM adaptation. The simulations for Fig. 5.13 and Fig. 5.14 were also done using the AGM scheme, with adaptation time step $\tau = 10.0$ and tolerance TOL$= 0.1\hat{h}$.



(a) (b)

Figure 5.15: AGM resolution adaptation to a narrow band around the unit sphere. (a) An initially uniform $32 \times 32 \times 32$ lattice with $\hat{h} = 0.125$ is adapted with an adaptation time step of $\tau = 1.0$ and a tolerance of TOL$= 0.1\hat{h}$. (b) The final distribution after 33 AGM iterations. The physical positions $\boldsymbol{x}_p$ of the particles as determined by Algorithm 2 are shown as mesh nodes. The physical width of the band needed to evaluate the differential operators is considerably smaller, thus allowing to resolve finer surface structures. Shading codes the contours of the monitor function, attaining large values close to the surface of the sphere and small values outside of the band.

Chapter 6

Simulations of Diffusion in Organelles of Live Cells

In this chapter we consider the application of the simulation techniques presented so far to diffusion processes in organelles of live cells. *Organelles* are the internal functional structures of cells, analogous to organs in whole organisms. The structures and shapes of organelles are hardly simple combinations of straight lines, spheres, and cubes and are thus not well described by the idealized constructs of Euclidean geometry. In biology this is true for many objects on a wide range of length scales. Known examples include the structures of taxonomic and phylogenetic trees [43], stability regions in population dynamic models [205], pneumonal and arterial trees [118], the shape of neurons [266], clusters of vesicles [163], the cytoskeleton [14], protein chain conformations [168], protein structures [167], nucleotide sequences [321], and electric currents through ion channels in cell membranes [169].

In the present work we consider the Endoplasmic Reticulum (ER), described in Section 6.1, as an example of biological interest. We reconstruct the shapes of real ER samples in the computer as described in Section 6.2. These reconstructed shapes are then used to quantify the complexity of the geometry using fractal analysis and theory as outlined in Section 6.3. We show that the geometric complexity of the organelle can lead to anomalous apparent diffusion on a larger, averaged length scale. This finding is important when quantitatively evaluating Fluorescence Recovery After Photobleaching (FRAP; Section 6.4) experiments. Owing to its experimental simplicity and versatility, FRAP has become one of the most widely used methods in modern cell biology. Its averaging nature however complicates the quantitative evaluation of FRAP data, as the involved influences from the shape of the organelle start to become important.

Using the particle methods outlined in the previous chapter, we simulate diffusion processes in the lumen (Subsection 6.5.2) and on the membrane (Subsection 6.5.3) of reconstructed ER shapes. Such simulations enable for the first time the measurement of geometry-corrected molecular diffusion constants from FRAP ex-

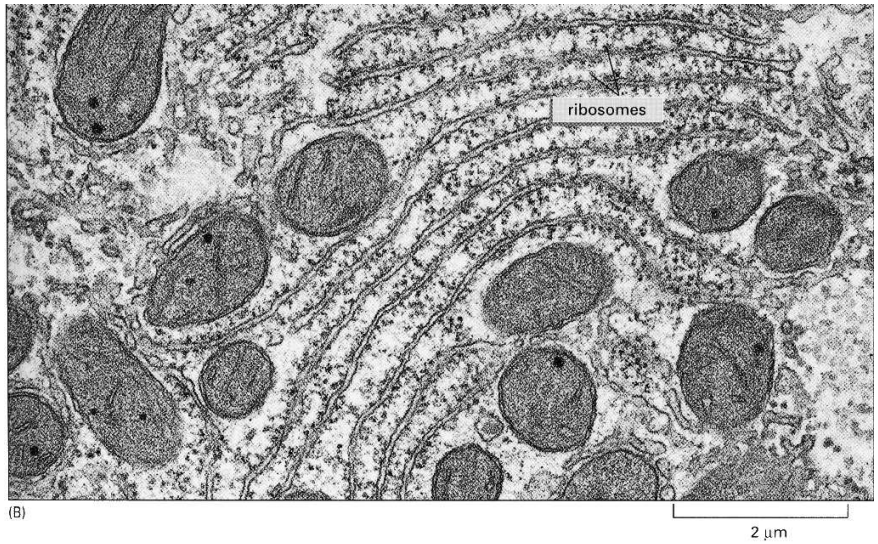


Figure 6.1: Electron micrograph of a region in a liver cell (source [8]). The ER is visible as cross-cut lamellar and tubular structures. The rough ER is covered with ribosomes, visible as small black dots on the outside of its membrane. The larger round structures in the image are mitochondria and peroxisomes.

periments in live cells. They also provide a means of validating the various existing models for diffusion in the ER [99, 207, 262, 300], and to rigorously quantify the geometric averaging artifacts in FRAP. Our simulations show that, unless properly accounted for, the geometric shape of the ER leads to a 2- to 4-fold underestimation of the molecular diffusion constants in FRAP analysis.

6.1 The Endoplasmic Reticulum (ER)

The *Endoplasmic Reticulum* (ER) is an organelle in eukaryotic cells. It is involved in protein synthesis, protein folding, and lipid synthesis. The electron micrograph in Fig. 6.1 shows a region of a cell where the ER is visible as a folded stack of membrane layers. The ER consists of a single contiguous membrane that is continuous with the *nuclear envelope* as shown in Fig. 6.2. The connected space

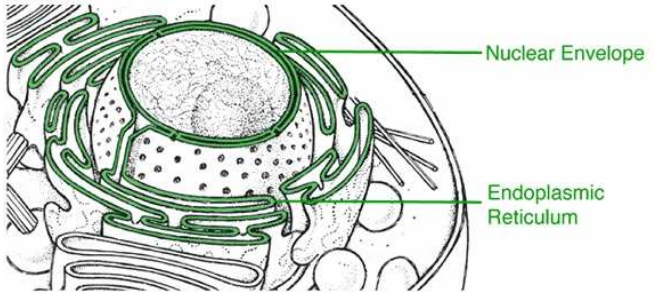


Figure 6.2: Schematic of the nuclear envelope and the contiguous ER membrane (source [77]).

enclosed by the ER membrane is called the *lumen*. The ER is generally depicted as a highly convoluted meshwork of tubular and lamellar structures in three dimensions [280] (Fig. 6.3), with individual tubules about 30 to 60 nm in diameter.

Morphologically, the ER can be partitioned into *rough ER* and *smooth ER*. The rough ER is mainly involved in protein synthesis, and the outside of its membrane is covered with ribosomes. The smooth ER tends to be concentrated in the *perinuclear region*, and it is mainly involved in lipid synthesis.

6.2 Computational reconstruction of real ER geometries

In order to be able to analyze the geometric properties of the ER and to simulate diffusion in its lumen and on the membrane, we reconstruct ER geometries from real live cells using *confocal fluorescence microscopy* and computational 3D reconstruction. ER geometries are recorded from live tissue culture cells expressing a soluble, resident, recombinant protein (ssGFP–KDEL; [281]). Using this marker and a stack of serial *confocal sections* (called a *z-stack*), we can experimentally define and computationally reconstruct the 3D shape of the ER. Cell lines, transfection, and confocal imaging were done in the group of Prof. A. Helenius.

In each cell, 50 0.1 μm optical z-sections are collected with a lateral resolution of $0.18\,\mu$m/pixel for the lumen, and $0.09\,\mu$m/pixel for the membrane cases. Imaging noise is removed using a *Gaussian filter* of half-width 200 nm, and the surface of the ER is reconstructed as a gray level iso-surface in 3D space using

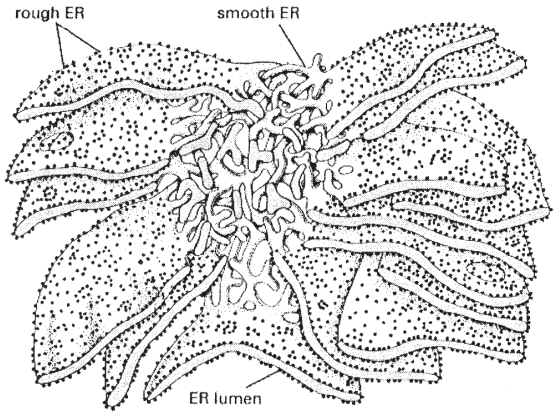


Figure 6.3: Schematic of the ER with distinction of rough ER (lamellar, with ribosomes) and smooth ER (tubular, without ribosomes). source [132].

Imaris 4.1.1 (BitPlane, Inc., Zürich, Switzerland), employing the same number of voxels as the section images have pixels. After removing detached parts of the surface, the ER membrane is discretized in Imaris and stored as a triangulation using planar triangles. The computer time for 3D reconstruction and triangulation of a complete ER using about 1 million triangles is about 2 to 5 minutes on a 3 GHz Intel Pentium 4 computer. The *intensity threshold* used for the iso-surface is set as high as possible to still result in a connected domain. This threshold is optimal according to the error analysis of the 3D reconstructions of artificial ER-like geometries presented in Subsection 6.2.1. After reconstruction, the surface is checked for consistency. It is required to enclose a connected space and to not contain any surface intersections or holes in the surface.

The triangulation of the surface is not an inherent feature of the present method and it represents only the format available from the image reconstruction software. Before being used in computer simulations, the surfaces are converted to level sets as described in Appendix C, and thus they do not remain piecewise linear.

For illustration purposes, one example of the numerous reconstructions is shown in Figs. 6.4 and 6.5. Fig. 6.4 shows the *confocal sections* of the fluorescently stained – but not fixed – ER of a live VERO cell (Helenius group). The result of the 3D reconstruction is shown in Fig. 6.5 for the complete ER, and an enlarged portion of a second sample.

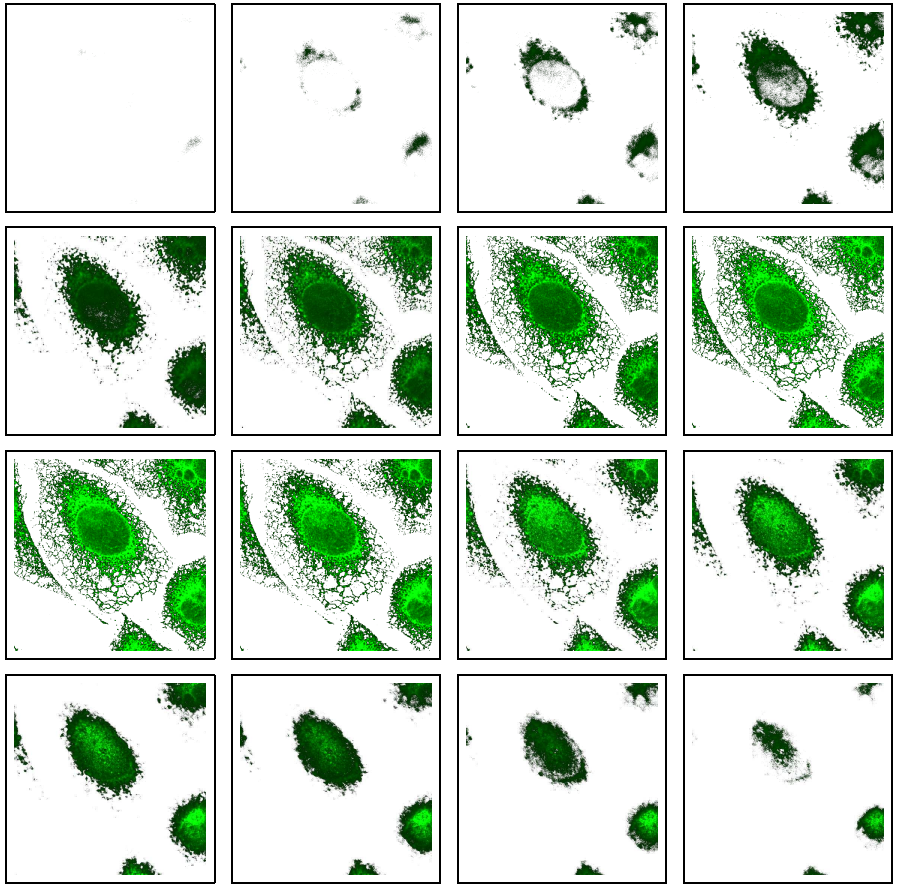
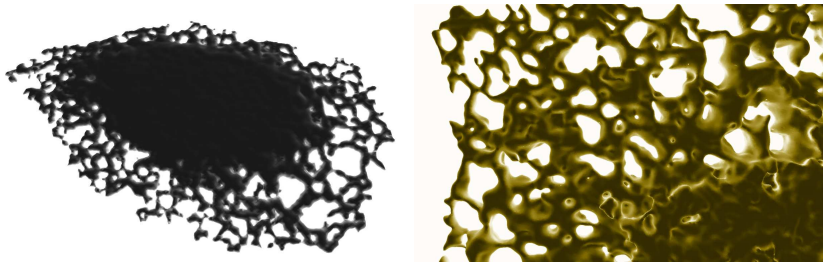


Figure 6.4: Sample z-stack of confocal sections used for 3D reconstruction. The ER of a VERO cell is fluorescently marked (Helenius group) and confocal images are taken at vertical distances of $\Delta z = 0.1\,\mu$m. Progressing from left to right and top to bottom, the focal plane moves from the bottom of the cell to its top.



(a) (b)

Figure 6.5: (a) Shaded view of the reconstructed ER surface from Fig. 6.4 in three dimensions. (b) Close-up of a reconstructed ER surface, illustrating the spatial resolution of the present geometry acquisition method.

6.2.1 Error analysis and influence of the microscope's optical anisotropy

In order to determine the optimal *threshold* (intensity iso-value) for the 3D reconstruction, we consider synthetic geometries for which the correct outcome is known. Artificial random networks of tubules are created in the computer on a lattice of M points. An example with $M = 20 \times 20 \times 3$ is shown in Fig. 6.6. These geometries are then *convolved* with a model of the anisotropic *point spread function* of the confocal microscope. For this purpose, the lateral resolution R of the microscope is expressed as

$$R = \frac{\lambda}{2NA}\,, \tag{6.1}$$

with λ the wavelength of the light emitted by the fluorophore (cf. Subsection 6.4.1) and NA the numerical aperture of the objective lens. The axial resolution δ is defined as the distance between the nearest and farthest planes simultaneously in focus and, according to [269],

$$\delta = \frac{3n\lambda}{2NA^2}\,, \tag{6.2}$$

where n is the refractive index of the medium. The ratio δ/R is called *optical anisotropy* of the microscope and it varies between about 1.6 and 5 for commercial confocal microscopes. The point spread function is modeled in each spatial

direction $r = x, y, z$ as [327]

$$P(r) = \left(2\frac{J_1(C_{x,y,z}r)}{r}\right)^2 \tag{6.3}$$

with

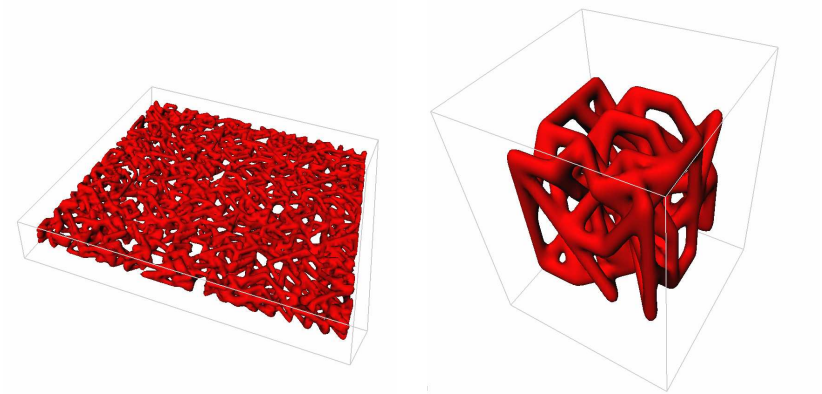
$$C_{x,y} = \frac{2\pi \cdot NA}{\lambda} \tag{6.4}$$

and J_1 the *Bessel function* of the first kind. For the axial direction, the point spread function is stretched according to the anisotropy as: $C_z = C_{x,y}R/\delta$. The values for the present work are: $\lambda = 510\,\text{nm}$, $NA = 1.4$, and $n = 1$.

Successive convolution of the artificial geometry with $P(r)$ in all three spatial directions yields a simulated z-stack of section images as shown in Fig. 6.7. To model *camera noise*, each pixel in these images is replaced by a Poisson-distributed random number with the expectation value equal to the original pixel value [53]. The section images are normalized such that all intensity values are between 0 and 255.

The geometries are then reconstructed from the simulated confocal sections using *Imaris* (BitPlane, Inc.), and the resulting reconstructed volumes are compared to the original ones. The deviation is quantified by the relative number of voxels that are incorrectly reconstructed, i.e. voxels that are missing in the reconstructed geometry, but are present in the original one, or vice versa. Fig. 6.8 shows the resulting total *reconstruction errors* for various thresholds and optical anisotropies for a test geometry with an expected number of 3 tubes connecting per branching point, $0.15\,\mu$m tubule radius, and an average distance of $1\,\mu$m between tubules. This corresponds to a *volume-filling fraction* of 0.3, which is close to the average volume-filling fraction of $1/3$ determined for real ER geometries. Similar studies are also done for larger tubules (radius $0.25\,\mu$m) and a lower connection density (2 tubes expected to connect per branching point). A total of 4 different geometries is analyzed for 5 different anisotropy values (1, 2, 3, 4, 6, 8) and a wide range of thresholds. The corresponding error plots are not shown here as they are analogous to the one in Fig. 6.8 and lead to the same conclusions.

We also determine the largest threshold for which the reconstructed network geometry remains connected. We find that the *optimal threshold* is close to or larger than this limit for all anisotropies larger than 1 and all geometries studied. For the experimental ER samples we thus always use the largest possible threshold which yields a connected reconstruction, since the most important objective toward



(a) Whole network (b) close-up

Figure 6.6: Example of an artificial tubular network geometry used to assess the quality of the 3D reconstruction. The random network is generated on a $20 \times 20 \times 3$ lattice with an average of 3 tubules being connected in each grid point. The radius of the tubules is $0.15\,\mu$m, the distance between two grid points is $1\,\mu$m. (a) The full network, imaged at a simulated anisotropy of 3 and reconstructed with a threshold of 50. (b) A $4 \times 4 \times 3$ subset imaged at anisotropy 1 and reconstructed with a threshold of 110.

realistic computer simulations is to preserve the topology of the organelle. The anisotropy of the confocal microscope used in the present work is about 2.14. The optimal threshold can thus be expected to be close to the largest feasible one.

Regarding the sensitivity of the reconstruction result with respect to the threshold setting, we find that varying the threshold by $\pm 10\%$ around the optimum changes the reconstruction error by $+4 \ldots 8\%$ for anisotropies of 2 and 3. Using the above-mentioned rule of thumb, such large threshold deviations should however never occur.

The reconstruction errors cause the tubules to appear thicker or thinner than they actually are. The error in the predicted diffusive flux is directly proportional to this size error. For an anisotropy of 2, the total relative reconstruction error is 28%, composed of 16% missing voxels and 12% excess voxels. A tubule thus appears on average 4% thinner than it actually is. This translates to a 4% error in the diffusion constant, which increases to 9% for an anisotropy of 3. Compared to the various experimental uncertainties, these errors constitute no significant reservation.

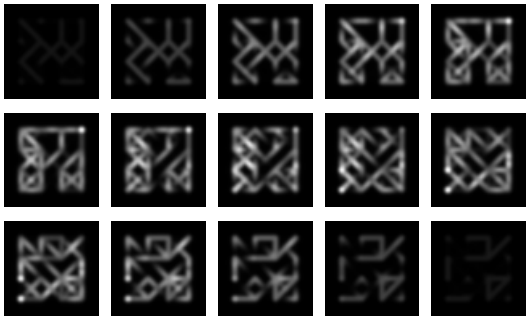


Figure 6.7: Artificial z-stack created from the sample geometry of Fig. 6.6(b). After convolving the geometry to model the effect of confocal imaging with an anisotropy of 4, 15 serial section images at an axial distance of $\Delta z = 0.2\,\mu$m are taken. The same geometry is also tested using up to 53 sections (images not shown). Progressing from left to right and top to bottom, the simulated focal plane moves from the bottom of the object to its top.

6.3 Fractal complexity analysis of the ER geometry

We analyze the geometric shape of the reconstructed ER from VERO cells and estimate some of its fractal properties. The *fractal dimension* [184] of a shape is often used to quantify the shape's *complexity*. In many situations, the fractal dimension is found to be a useful measure. Its definition however always involves some sort of limit to infinity and, since all physical and biological systems are finite, fractal dimensions are in principle not defined for them. Still, we can use the mathematical idealization as a model that is valid over a certain range of scales. Therefore, *fractal* in the present context means *pre-fractal*.

It is well known that the shape of a domain influences the processes taking place inside it or on its boundary. While this is already true for Euclidean shapes [140], it becomes evident for fractal geometries, where non-linear interactions in and between spatial and temporal scales determine the dynamics. Recent literature contains an increasing number of hints that confined diffusion in the ER appears anomalous [241, 314]. This is also known for diffusion in the cytoplasm [313, 19], where it can be attributed to the fractal nature of *molecular crowding* [14, 19]. *Anomalous diffusion* can however also be a consequence of diffusion on domains with non-integer fractal dimension, as shown in Appendix D. The present fractal analysis confirms that this is indeed the case for the ER. The geometric shape of

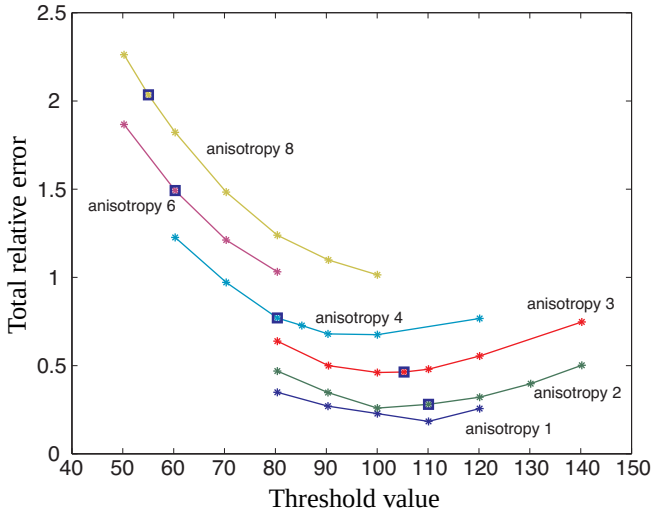


Figure 6.8: Relative reconstruction errors for the sample geometry of Fig. 6.6 and different optical anisotropies and reconstruction thresholds. The abscissa shows the pixel intensity iso-value value used for the 3D reconstruction. The intensity values in the images are normalized to the interval $[0, 255]$*. The ordinate shows the total relative reconstruction error, given by the number of missing voxels plus the number of excess voxels in the reconstructed volume, divided by the total number of voxels in the original volume. Lines are shown for various anisotropies. The squares mark the largest threshold for which a connected reconstruction results. For the idealized case of anisotropy 1, the reconstruction remains connected for all thresholds tested.*

the organelle can explain the apparent anomalous behavior, and it thus plays an important role in diffusion analysis.

6.3.1 Renyi entropies and generalized dimensions

Among the numerous definitions of fractal dimensions (see e.g. Table 1 in Ref. [69]), we use the generalized *Renyi dimensions* [228]

$$d_q = -\lim_{\delta\to\infty} \frac{I_q}{\log\delta} \qquad , q \in \mathbb{R}\,, \tag{6.5}$$

which are based on the *Renyi entropies* I_q, defined as follows: assume a disjoint partitioning of the embedding Euclidean space E^d into $M(\delta)$ cartesian cells $\{C_i^\delta\}_{i=1}^{M(\delta)}$, each of volume δ. Let p_i be the probability for the geometry under consideration to fill cell C_i. The Renyi entropies are then given by

$$I_q = \begin{cases} \frac{1}{1-q}\log\sum_{i=1}^{M(\delta)} p_i^q & , q \neq 1 \\ -\sum_{i=1}^{M(\delta)} p_i \log p_i & , q = 1\,. \end{cases} \tag{6.6}$$

The Renyi dimensions d_q are strictly positive and their values decrease with increasing q, converging to a limit d_∞. For $q = 0$, the dimension d_0 is identical to the *capacity* (or *box counting*) *dimension*. We consider the Renyi dimensions of orders $q = -1, 0, 1, 2$ to verify the fractal scaling behavior, i.e. to check that $d_{-1} > d_0 > d_1 > d_2$ holds over a sufficiently large range of length scales.

To estimate the Renyi dimensions, the probabilities p_i need to be approximated. This is done by uniformly scattering half a billion random points on the reconstructed surface of the ER, and counting the number of such points falling into every cartesian cell C_i. Dividing this count by the total number of scattered points approximates p_i for the cells. The grid that defines the cells C_i is subsequently coarsened by a factor of two in each direction, and the whole procedure is repeated until the number of cells in any direction falls below two. To minimize spurious effects from the random number generator and aliasing effects due to grid sampling, the whole procedure is repeated with five different random seeds and six different, slightly shifted, bounding boxes for the cell mesh. The measured entropies are averaged from all 30 repetitions, and the Renyi dimensions are determined as least squares regressions of the corresponding entropy versus the logarithm of the box size at each reduction step.

case	d_{-1}	d_0	d_1	d_2
1	2.5300	2.3565	2.2968	2.2783
2	2.7636	2.4870	2.4160	2.3863
3	2.5337	2.4005	2.3455	2.3159
4	2.5450	2.3903	2.3292	2.2927
5	2.6049	2.3494	2.2864	2.2647
6	2.6127	2.4002	2.3215	2.2837
7	2.2984	2.1490	2.1045	2.0911
8	2.7264	2.4361	2.3392	2.2687
9	2.9203	2.5695	2.4473	2.3672
all	2.61±0.18	2.39±0.11	2.32±0.10	2.28±0.08

Table 6.1: Measured Renyi dimensions (see text) for nine different ER geometries. The fractal scaling is confirmed over 1.7 orders of magnitude ranging from 0.01 cell diameters to 0.5 cell diameters.

We test whether the ER membrane can be viewed as a continuous *fractal surface* in space by applying above procedure to the reconstructed ER from nine different cells. The results are summarized in Table 6.1. For the tested ER shapes, the *capacity dimension* is $d_0 = 2.4 \pm 0.1$ and the ordering of the Renyi dimensions is satisfied in each individual sample. The fractal scaling persists over 1.7 orders of magnitude in length scales, ranging from 0.01 cell diameters to 0.5 cell diameters. At length scales relevant to whole organelle dynamics, the ER is thus expected to exhibit *fractal diffusion* characteristics.

6.3.2 Apparent diffusion on fractal domains

The laws of lateral diffusion on the membrane as well as of diffusion in the luminal space change when considering a fractal domain [22, 21, 97]. In particular, the expected MSD (cf. Subsection 2.1.1) of a normally diffusing particle during the time period δt changes from $\mathsf{E}\left(\|\boldsymbol{x}(t+\delta t) - \boldsymbol{x}(t)\|_2^2\right) \propto \delta t$ to [22]

$$\mathsf{E}\left(\|\boldsymbol{x}(t+\delta t) - \boldsymbol{x}(t)\|_2^2\right) \propto \delta t^{2/d_w}\,, \tag{6.7}$$

corresponding to apparent *anomalous diffusion* (cf. Subsection 2.1.2). The parameter d_w is called the *dimension of the walk*. This dimension is related to *Hausdorff's dimension* d_H and the *spectral dimension* d_s as [22]:

$$d_w = \frac{2d_H}{d_s}\,. \tag{6.8}$$

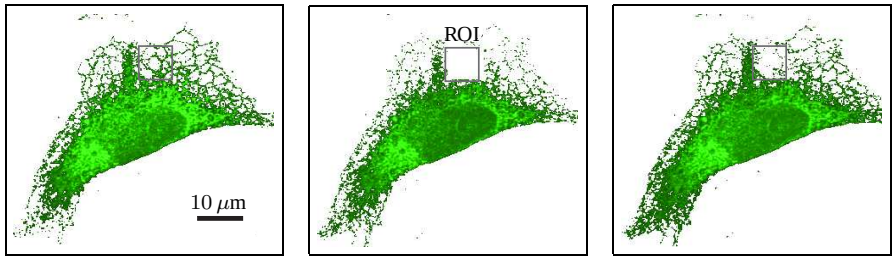
The spectral dimension d_s is connected to the *density of states*, i.e. the distribution of the eigenvalues of the diffusion operator on the domain of solution, and can not be measured for an arbitrary geometry such as the ER. According to Eq. (6.7), two shapes can only exhibit the same macroscopic diffusion behavior if their dimensions of the walk are identical [244]. This is a necessary condition.

Confined diffusion in the ER is expected to appear anomalous [241] at length scales between the diameter of individual tubules and the whole organelle, even if the underlying molecular diffusion is normal [207, 305, 313]. This is a direct effect of the *complexity* of the ER geometry (cf. proof in Appendix D), which we have shown to exhibit fractal scaling properties. The experimental results reported by Weiss *et al.* [314] confirm that the observed anomaly is purely caused by the geometry of the organelle and is independent of molecular structure and events. Any *model geometry* for diffusion in the ER would need to have the same d_w as the real ER shape. This is however impossible to achieve since the spectral dimension of any given ER sample can not be measured. Direct numerical simulations in the reconstructed shapes are thus needed to quantitatively understand the geometric influences.

6.4 Fluorescence Recovery After Photobleaching (FRAP)

The experimental technique of *Fluorescence Recovery After Photobleaching* (FRAP) is widely used to determine how substances move within live cells or on cellular membranes [317]. In FRAP, a *Region Of Interest* (ROI) in the cell that contains the fluorescently tagged molecules is bleached using strong laser light. The influx of non-bleached molecules from adjacent areas into the ROI is recorded and analyzed over time, as illustrated in Fig. 6.9. When applied quantitatively, FRAP allows to determine the molecular diffusion constants of fluorescent molecules, including soluble and membrane-bound proteins [174].

FRAP has been used since the 1970s to investigate lateral mobility on the cell surface [15]. Later it has been extended to the investigation of protein dynamics within the cell [274] and was also used to follow events during cell division and signaling, and to measure protein interactions and conformational changes [227]. The use of FRAP rapidly increased with the availability of methods to tag intracellular proteins with *Green Fluorescent Protein* (GFP) and its derivatives. This



pre-bleach bleach, $t = 0$ $t = 2\,\text{min}$.

Figure 6.9: Example of a FRAP micrograph sequence in a VERO cell (Helenius group). The Region Of Interest (ROI) is highlighted by the gray box.

allows visualization of the proteins and enables measurements of their dynamics in live cells. Diffusion constants of GFP and GFP-tagged proteins have been reported for the cytoplasm [278], the nucleus [216], the ER [75, 203], mitochondria [210], the Golgi complex [60, 254], and for different membranes in the cell [89, 90, 186].

6.4.1 Green fluorescent protein

Green Fluorescent Protein (GFP) is a spontaneously fluorescent protein isolated from coelenterates, such as the fluorescent pacific jellyfish *Aequoria victoria*. The role of the active center of GFP is to transduce (by energy transfer) the blue chemiluminescence of another protein (aequorin) into green light. GFP is well suited for photobleaching studies since it is a bright, stable, non-toxic fluorophore with low bleaching under imaging conditions. When illuminated at high intensity, GFP looses its fluorescence irreversibly, but without damaging intracellular structures [317]. This process is referred to as *bleaching*. Various mutants of GFP with stronger fluorescence and/or different absorbance and emission peaks are known and used [324].

To track proteins other than GFP itself, they are covalently modified to include a GFP domain. This is done by molecular cloning of GFP cDNA that is then expressed in cultured cell lines. GFP can function as a protein tag, as it is tolerated for both N- and C-terminal fusion by a wide variety of proteins. Many of those have been shown to retain their native function after addition of GFP. This enormous flexibility as a *noninvasive marker* in live cells enables numerous applications of GFP, FRAP being one of them.

6.4.2 Estimating the diffusion constant from FRAP data

In order to obtain molecular diffusion constants from *fluorescence recovery curves*, the dependence of the curve's shape on the molecular ν_2 needs to be modeled. Fitting such a *FRAP model* to an experimentally determined recovery curve yields the estimated diffusion constant.

We distinguish between the *molecular diffusion constant* and the *apparent diffusion constant*. The former is directly measured by single-molecule techniques, such as single molecule tracking [142, 175] (Chapter 1) or fluorescence correlation spectroscopy [314], and is to be used in the diffusion equation to model the process in the continuum. The latter is the constant determined by *coarse-grained* methods such as FRAP, averaging over a certain *observation volume*. These apparent values depend on the geometry of the observation volume (see Appendix D for a mathematical derivation and rigorous definitions). Deriving molecular diffusion constants from apparent ones is important when comparing experiments made in different organelles or cells, as well as for mathematical modeling and computational simulations of the observed diffusion process.

Current techniques, as summarized below, do however not take into full account that the organelles to which the fluorescent molecules are confined often have a complex three-dimensional shape, and that they may only occupy a fraction of the bleached and unbleached volumes. The importance of accounting for the specific geometry of the organelle increases with increasing *complexity* of the organelle's shape and with decreasing *volume-filling fraction* in the bleached and unbleached regions. This issue has been frequently discussed in the literature [94, 75], but no procedure exists to quantify the magnitude of the uncertainty introduced, let alone to calculate more accurate molecular diffusion constants from FRAP curves.

Although theoretical descriptions of particle diffusion in two-dimensional membranes have been derived for a variety of situations, including periodically nonplanar membranes [6], binding, particle crowding [249], and mobile as well as immobile obstacles [239, 252], no such theory exists for the three-dimensional lumen of complex compartments such as the ER, or for their curved membranes.

FRAP models are based on postulating certain dynamics for the process and in some cases also a *model geometry* for the organelle under consideration. In our case, we assume the dynamics to be described by the diffusion equation. Two classes of models can be distinguished: *closed-form models* and *simulation-based models*. The former are based on an analytic solution of the model problem. This solution is then fitted to the experimentally determined FRAP curves to determine the diffusion constant. Such models lack the capability of accounting for the

specific organelle shape and should be used with caution when comparing diffusion constants between different cells or compartments [227]. Simulation-based models provide more flexibility by numerically solving the model problem.

6.4.3 Closed-form equation models

Closed-form models are based on the analytic solution of a model problem, where the diffusion constant constitutes the model parameter that is to be determined by data fitting. Due to the limitations of analytic solvability, diffusion is either calculated in a flat plane rather than in three dimensions [75], or by using (semi-)empirical correlations determined from calibration experiments [99]. The following overview summarizes the most frequently used closed-form models from the literature. The list is however incomplete as several specialized models exist, e.g. for strip bleaching [94].

Exponential recovery model

Solving the homogeneous, isotropic diffusion equation in two dimensions, and assuming homogeneous flux into the ROI, the predicted fluorescence recovery is given by the *exponential recovery model*

$$F(t) = \int_{\mathrm{ROI}} u(\boldsymbol{x}, t)\, d\boldsymbol{x} = F_0 \left(1 - e^{-\alpha \nu_2 t}\right) , \tag{6.9}$$

where F_0 is the *pre-bleach intensity* and α is a constant that depends on the shape of the ROI and that can be determined analytically. The value of the parameter ν_2 is obtained from data fitting.

Empirical correlation model

The following *empirical correlation model* is found by considering anomalous diffusion in simple geometries and calibrating with experiments [75, 227]. The predicted fluorescence content in the ROI is:

$$F(t) = F_a + \frac{[F_a + f_m(F_0 - F_a)]\,(t/t_{1/2})^{\alpha}}{1 + (t/t_{1/2})^{\alpha}} , \tag{6.10}$$

where F_a is the intensity just after bleaching, F_0 is the intensity just before bleaching, and α is the model parameter. The *recovery half-time* $t_{1/2}$ is estimated from

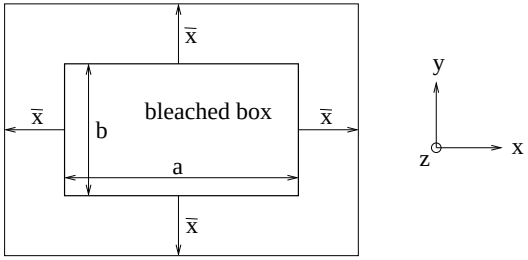


Figure 6.10: Simplified geometrical situation around the bleached ROI.

the measured curve by nonlinear fitting. The *mobile fraction* f_m is defined as [227]

$$f_m = \frac{F_\infty - F_a}{F_0 - F_a} , \tag{6.11}$$

where F_∞ is the asymptotic fluorescence intensity at large times. The mobile fraction contains information about membrane barriers, binding reactions, and microdomains in the membrane, as these phenomena can prevent or temporarily restrict the free diffusion of molecules.

A second order physical model

This semi-empirical model [244] is based on the situation in the vicinity of the ROI as depicted in Fig. 6.10. For simplicity, it considers the complementary problem of diffusion of bleached protein out of the ROI.

Let a, b, and c be the lengths of the edges of the ROI in all three spatial directions. Without loss of generality, we assume that the initial concentration of bleached protein inside the ROI is unity. The total mass of bleached protein in the box is thus equal to abc. After some time, the concentration front has diffused a mean distance of $\overline{x}$ in both the x and y direction. The new volume in which the bleached molecules are contained is thus given by $(a + 2\overline{x})(b + 2\overline{x})c$. Assuming a homogeneous distribution as well as conservation of mass, the new mean concentration in the ROI hence becomes

$$\frac{abc}{(a + 2\overline{x})(b + 2\overline{x})c} . \tag{6.12}$$

If we assume that for each bleached molecule that leaves the ROI, a fluorescent one enters in exchange, the fluorescence intensity in the ROI is given by the complement of the above concentration. Re-scaling the model to an asymptotic level of F_∞ instead of 1 yields

$$F(\overline{x}) = F_\infty \left(1 - \frac{ab}{(a+2\overline{x})(b+2\overline{x})}\right) . \tag{6.13}$$

To model the influence of the local ER geometry, we set $\overline{x}^2 = \alpha^2 t^{2\beta}$, using the fractal diffusion concepts outlined in Section 6.3. Substituting into Eq. (6.13) yields the *second order physical model*:

$$F(t) = F_\infty \left(1 - \frac{ab}{ab + 2\,(a+b)\,\alpha t^\beta + 4\alpha^2 t^{2\beta}}\right) , \tag{6.14}$$

with parameters α and β. This model includes both information about the ER geometry (in the sense of fractal dimensions) and the size of the ROI (in a and b). It is based on first physical principles and can thus be expected to have some extrapolation capabilities.

Since it is not possible to measure the spectral dimension of a real ER shape (cf. Section 6.3), the connection of the model parameters α and β to the physical diffusion constant ν_2 is however unknown. The model can thus only be used to determine recovery half-times, but not diffusion constants.

Comparison of closed-form model fits

The fitting quality of the closed-form models presented so far is assessed on a simulated FRAP curve (cf. Appendix F and Subsection 6.5.2 for the simulation details). Fitting of the models to the simulated FRAP curves is done using a *Nelder-Mead simplex* algorithm to minimize the L_2 fitting error. The resulting optimal model parameters, the fitting residual ρ_{min}, and the needed number of iterations N_{iter} are given in Table 6.2, the corresponding curves are shown in Fig. 6.11. While the empirical model is already significantly better than the exponential recovery model, the second order physical model explains the data another one to two orders of magnitude better.

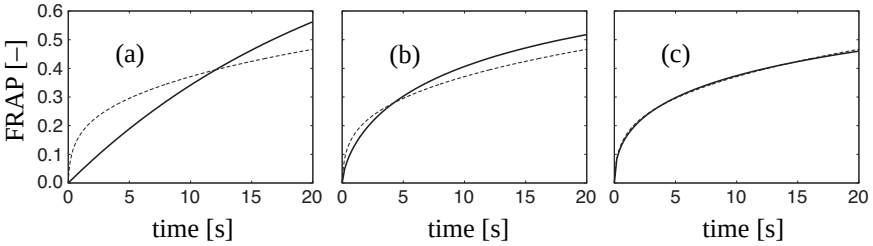


Figure 6.11: Best fits of the exponential model (a), the empirical model (b), and the second order physical model (c) to a simulated FRAP curve (dashed) in a real ER geometry (cf. Appendix F and Subsection 6.5.2). The dashed curve has $F_0 = 1.0$, $F_a = 0.0$, and $F_\infty = 0.99$ (cf. Appendix F.1). The residuals and the optimal parameter values are given in Table 6.2.

Model	optimal parameters	ρ_{min}	N_{iter}
Exponential	$(\alpha\nu_2)_{\text{opt}} = 0.04342$	6.434	24
Empirical	$\alpha_{\text{opt}} = 0.66775$, $t_{1/2,\text{opt}} = 16.290$	1.368	991209
Physical	$\alpha_{\text{opt}} = 3.6143$, $\beta_{\text{opt}} = 0.45771$	0.01244	18318

Table 6.2: Optimal model parameters, residual fitting error, and number of iterations needed to converge for the different closed-form FRAP models.

6.4.4 Computational FRAP models

Since closed-form equation models do not account for the specific shape of the organelle in individual cells, they have limited extrapolation capabilities. If the FRAP curve depends on the particular shape of the organelle, these models predict different diffusion constants, even if the actual molecular diffusion constants are identical. This is due to the fact that closed-form models attribute any variation in the FRAP curve to changes in the diffusion constant and can not account for geometry-induced variations.

Several approaches have been made to use *simulation-based FRAP models*. In such a model, simulated recovery curves with known diffusion constant in the simulation are fitted to experimental data in order to deduce molecular diffusion constants. Fitting is only done in time, while the FRAP values are left unchanged [246]. The *effective molecular diffusion constant* ν_{eff} is then computed from the *computational diffusion constant* ν_{sim} and the time-stretching factor t_s from the fit as

$$\nu_{\text{eff}} = \frac{\nu_{\text{sim}}}{\xi^2 t_s}\,, \tag{6.15}$$

where ξ is the ratio of length units between simulation and experiment.

In the following, we review the most important advances in computational modeling and introduce a novel simulation model that fully accounts for the three-dimensional shape of the organelle.

Monte Carlo simulations in artificial model geometries

Using the classical method of random walk, as outlined in Subsection 5.2.1, Ölveczky and Verkman [207] performed computer simulations to calculate solute diffusion in an orthogonal meshwork of interconnected cylinders. Random walk is an intuitive method for simulating diffusion and is suitable for handling complex geometries. Its slow convergence rate however hampers the accuracy of the results, as shown in Subsection 5.2.3. Ölveczky and Verkman found that the apparent diffusive transport in the cylinder meshwork is about half as fast as in free space. Moreover, they found the diffusion to effectively appear anomalous, even if the molecular diffusion is normal. In Appendix D we give a mathematical explanation for this phenomenon. This showed that geometry has a significant impact on apparent diffusion, and that diffusion constants are underestimated by models that neglect the confinement. The shapes of real ER may however not be accurately mimicked by random artificial cylinder meshes.

Two-dimensional variable density simulations

Siggia *et al.* [262] used finite differences [265] to computationally solve the diffusion equation in the imaging plane of the observation microscope. The geometry of the ER was treated by taking a smoothed post-bleach fluorescence intensity micrograph as the initial condition. In the course of the simulation, the geometry was however no longer explicitly taken into account, mainly due to the numerical limitations of the employed finite difference method[1]. Instead, the *connection density* of the ER was assumed to be represented by the local fluorescence intensity in a pre-bleach image. Based on this assumption, a statistically averaged transport model was introduced. Depending on the particular transport model, variations in the apparent diffusion constant of up to a factor of three were observed [262]. As already stated in the original publication [262], the validity of the model is questionable when 3D effects become important, when image regions of saturated pixel intensities exist, or when concentration variations are present in the pre-bleach image. The first situation for example occurs when two compartments that overlap in the projection are in fact disconnected in 3D.

Three-dimensional free space simulations

A more recent approach by Braga *et al.* [35] made use of finite difference simulations to derive a FRAP model in the nucleoplasm. The model thoroughly treats the initial condition of the recovery dynamics by explicitly considering the 3D intensity distribution of the bleaching laser beam as well as *premature recovery* during bleaching. Braga *et al.* report a molecular diffusion constant of $33.3\pm3.6\,\mu\text{m}^2/\text{s}$ for GFP in the *nucleoplasm* of *HeLa cells*. Their work did also neglect the geometric shape of the compartment under consideration. The model as well as the simulations were done in 3D free space. For short times this is certainly a valid

[1] Finite differences are based on numerical approximations of the derivatives of the governing equation on a computational mesh. These approximations result in re-formulating the governing PDE as sets of linear systems of equations that can be solved computationally. For simple geometries, the resulting algebraic systems can be structured (e.g. in tridiagonal matrices), so that efficient numerical solvers can be applied, resulting in computations that scale linearly with the number of the discretization points. The efficiency of grid based methods is however drastically reduced when discretizing complex geometries. The resulting discretized equations fail to have the "nice" structure associated with simpler geometries, resulting in fuller systems whose solution often scales with the square or even the cube of the computational elements. Moreover, the generation of the grid in complex geometries remains a challenging task, despite the availability of several methods to render such procedures automatic. In addition, the order of accuracy of the numerical approximation of the governing equation is reduced near complex boundaries.

assumption in the nucleoplasm. Compartments of more complex shape, such as the ER or mitochondria, can however not be expected to be treated accurately with this scheme.

Three-dimensional simulations in realistic geometries

A more accurate three-dimensional analysis of diffusion in complex-shaped organelles is needed in order to overcome the limitations of the above-mentioned methods, and to provide much-needed validation of the various closed-form models that are currently in use. We propose a novel simulation strategy [246, 245] that uses realistic ER geometries as reconstructed from micrographs (cf. Section 6.2). The reconstructed geometries are directly used as *computational domains*. We solve the diffusion equation both in the lumen and on the surface of such reconstructed ER shapes. The resulting simulated FRAP curves are used to quantify the geometry-induced uncertainty in closed-form FRAP models, and to directly determine geometry-corrected diffusion constants by means of the fitting procedure of Eq. (6.15). The outline of the method is shown in Fig. 6.12, its application to real FRAP experiments in the ER of live cells is presented in the following Section 6.5.

6.5 Results of FRAP simulations in reconstructed ER geometries

Using reconstructed ER geometries according to Section 6.2, the numerical methods outlined in Chapter 5, and the scalable parallel software implementation presented in Chapter 7, enables us to simulate FRAP experiments with a minimum number of assumptions. Within the imaging accuracy of the confocal microscope used for 3D reconstruction, the only assumption consists of postulating the governing equation, which is supposed to be the diffusion equation. Fully resolved simulations eliminate the need for modeling either the geometry or the process of confined diffusion, and effectively allow assessment and refinement of existing FRAP models.

In this section we present results concerning the geometric effects and the geometry-induced uncertainties in FRAP experiments both in the lumen and on the membrane of the ER. As outlined in Subsection 6.4.4, these simulations can also be used to determine geometry-corrected molecular diffusion constants of soluble and membrane-bound molecules from FRAP data.

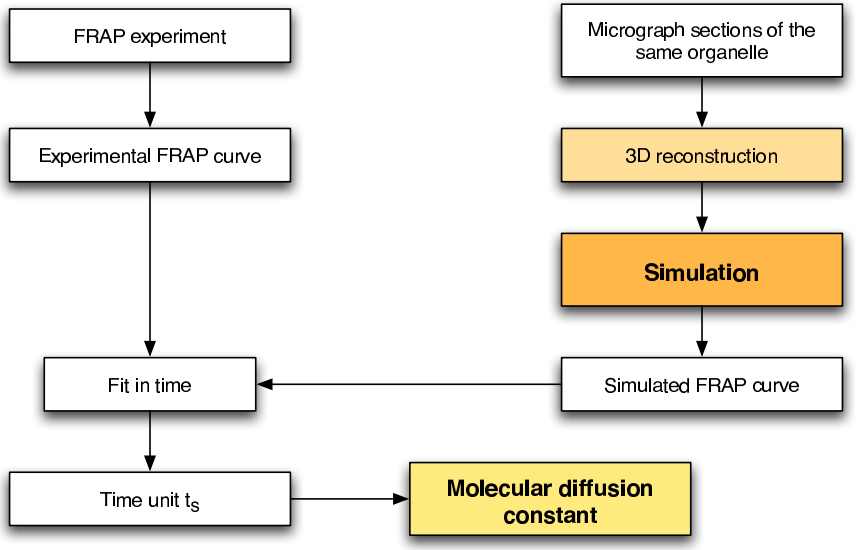


Figure 6.12: Overview of the method proposed in this thesis to determine molecular diffusion constants from FRAP data by means of spatially resolved computer simulations. See text and Eq. (6.15) for details.

6.5.1 A proposed method for determining molecular diffusion constants from FRAP data in complex-shaped organelles

Following the strategy outlined in Subsection 6.4.4, our simulations and experiments lead to a novel method of determining molecular diffusion constants from FRAP data. The procedure as summarized in Fig. 6.12 is as follows:

1. After transfection and incubation, the organelle of interest is imaged as a z-stack of serial confocal sections. After this recording of the geometry, the actual FRAP experiment is performed. It is important that the organelle under consideration does not significantly move or deform during this step.
2. The z-stack of images is used to determine the reconstructed surface of the organelle as an iso-surface of pixel intensity. Various commercial and free software packages are available to do this. The iso-value is chosen such that the topological features of the organelle are conserved. The ER should, e.g., remain connected.
3. The reconstructed volume is used as the computational domain for computer simulations of diffusion using scaled units of time and an arbitrary, scaled, computational diffusion constant. The initial condition is given by the FRAP setup.
4. The computed fluorescence recovery curve is fitted to the measured data points using a linear least squares regression in time.
5. The molecular diffusion constant in the experiment is calculated from the computational diffusion constant, the time scale factor (from the fit) and the length scale factor (from microscope/camera resolution) according to Eq. (6.15).

To make an example, assume that the simulation uses a computational ν_{sim} of 75 (in scaled simulation units). In order to convert from scaled *simulation units* to *physical units*, the time and length scales need to be determined. The length scale is known from the pixel resolution of the z-stack images and the voxel size used in the 3D reconstruction. Say that the images are acquired with a lateral resolution of $0.18\,\mu$m/pixel, and that the 3D reconstruction uses the same number of voxels as the z-stack images have pixels. If the size of an individual voxel is set to 66 (arb. units), one simulation length unit corresponds to $2.7\,$nm in physical units. The time scale factor is determined by fitting the simulated recovery curve to the experimental one in time. This provides all the information needed to compute the physical molecular diffusion constant according to Eq. (6.15).

6.5.2 Application to soluble proteins

We describe simulations using the method of PSE [79] (cf. Subsection 5.2.2) to estimate the influence of organelle shape on FRAP of a luminal solute, and to obtain more accurate measurements of molecular diffusion constants *in vivo*. The PSE method enables whole-organelle simulations, as its deterministic nature renders it orders of magnitude more accurate than random walk for the same number of particles, allowing fully resolved 3D simulations in realistic organelle geometries using about 10^6 particles. Resolving a full ER using random walk would require some 10^{10} particles (cf. Subsection 5.2.3), which is infeasible on present-day workstation computers. Moreover, the grid-free character of PSE avoids the geometric limitations of finite differences and the complications of robust grid generation. The details of the simulation procedure are described in Appendix F.1, the experimental protocols are given in Appendix E.1

The influence of confinement in complex shapes

The bleached region in a standard FRAP experiment usually has a diameter or edge length of around 0.1 cell diameters. This means that at length scales relevant to FRAP, the ER surface is complex enough to exhibit fractal characteristics, and the laws of apparent diffusion are expected to change as outlined in Section 6.3.

To study the effects of organelle geometry, we perform computer simulations of the diffusive fluorescence recovery in various reconstructed ER geometries. A well-characterized, fluorescent, recombinant protein (*ssGFP–KDEL*; [281]) is expressed in the ER of *VERO cells* (Helenius group). The protein is synthesized with a cleavable signal sequence sufficient for ER-luminal targeting. At the C-terminus, it has a *KDEL* sequence that serves as an ER localization sequence and prevents secretion [201]. Using fluorescence correlation spectroscopy [305], the molecular diffusion of the closely related *ssYFP–KDEL* has been shown to be non-anomalous in the ER lumen of HeLa cells, i.e. the molecule does not exhibit sub-diffusive properties on molecular length and time scales (M. Weiss, personal communication, 2002). We thus solve the normal isotropic diffusion equation. Using confocal fluorescence microscopy and a set of serial sections (z-stacks), the 3D shape of the ER filled with the fluorescent protein is defined in 19 different cells as described in Section 6.2. The ER reconstructions are used as computational domains for PSE simulations of diffusion of luminal solutes (cf. Appendix F.1). The speed of diffusive recovery is influenced by the geometry of the organelle near the bleached region. The specific shape of the organelle far from the ROI is insignific-

ant. All simulations use an assumed homogeneous Neumann boundary condition, implying that the protein cannot cross the membrane. The numerical L_2 error is as low as $6 \cdot 10^{-3}$ in all simulations. Only bleached regions in the cell periphery are considered, since the geometry of the dense *perinuclear ER* is not well resolved (cf. Fig. 6.5(a)).

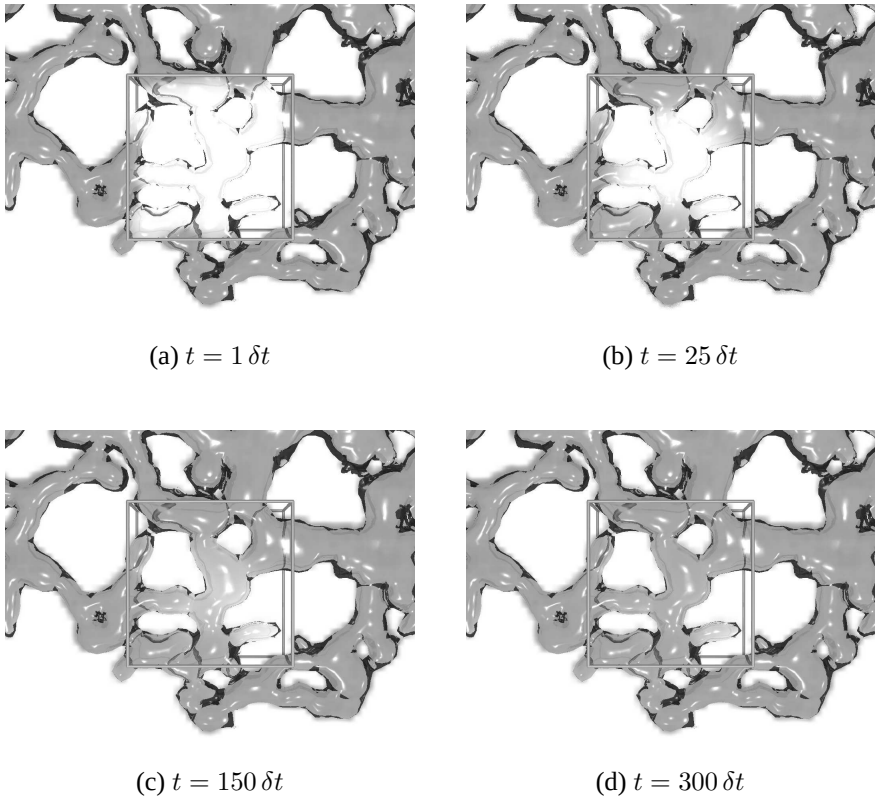
Computer-generated images of ER samples at different stages of simulated recovery are shown in Fig. 6.13. The local concentration of unbleached solute is shown as a density cloud inside the reconstructed ER structure. The bleached volume is depicted by its outline. The solute can be seen to diffuse into the bleached region from the edges, and the rate by which each element in the bleached region recovers depends on the distance from the edge and on the local geometry.

To study the effects of confinement in the ER lumen, we compare the PSE simulations of diffusion in the ER to simulations in a cubic 3D box. When the same molecular diffusion constant is used in both simulations, much faster recovery is observed in the box, as shown in Fig. 6.14. Depending on the actual ER geometry, the *apparent diffusion constant* observed in the ER is 1.8 to 4.2 times lower than the one observed in the box. Ignoring the effect of 3D confinement in complex geometries thus leads to significant underestimation of molecular diffusion constants.

Comparison to experimental data

To compare the results obtained from the simulations with experimental data, FRAP experiments according to Appendix E.1 are conducted in ssGFP–KDEL expressing cells, for which the ER shape is first established from a 3D confocal reconstruction. Using the PSE method, simulated FRAP curves are computed in the same geometries as those used in the actual experiments. This is done for 12 different FRAP experiments in 8 different cells. The simulated curves are then fitted to the experimentally measured FRAP curves using time stretching as described in Subsection 6.4.4. Stretching time by a factor of t_s and at the same time multiplying ν_2 by $1/t_s$ leaves the solution unchanged, as the diffusion constant can be incorporated into the governing equation as a scaling in time.

As shown in Fig. 6.15(a) for two of the cases, the simulated and experimentally determined FRAP curves are virtually indistinguishable after fitting. Similar overlap is observed in all instances. We conclude that the simulations are consistent, and accurate enough to be used to predict the effects of organelle geometry on FRAP as well as to derive geometry-corrected molecular diffusion constants from FRAP data. Fig. 6.15(b) shows a similar comparison between simulated and ex-



(a) $t = 1\,\delta t$ (b) $t = 25\,\delta t$

(c) $t = 150\,\delta t$ (d) $t = 300\,\delta t$

Figure 6.13: Snapshots of the concentration distribution from a sample PSE simulation in a reconstructed ER geometry. The results at times $t = 1\,\delta t$ (a), $t = 25\,\delta t$ (b), $t = 150\,\delta t$ (c), and $t = 300\,\delta t$ (d) are shown for a molecular diffusion constant of $\nu_2 = 3 \cdot 10^{-5}\, b^2/\delta t$. All units are scaled with the simulation time step $\delta t = 0.01$ and the lateral edge length $b = 50$ of the bleached region. The ER membrane is visualized as a transparent surface and the concentration of green fluorescent protein as a volume density cloud inside it. The bleached region is represented by its edges. Only the part of the ER around the bleached region is shown.

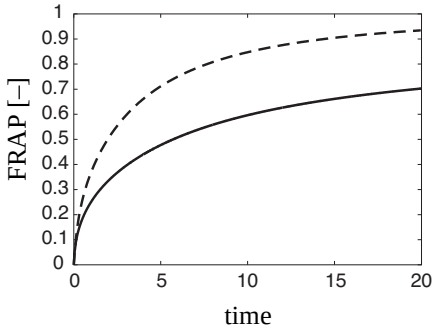


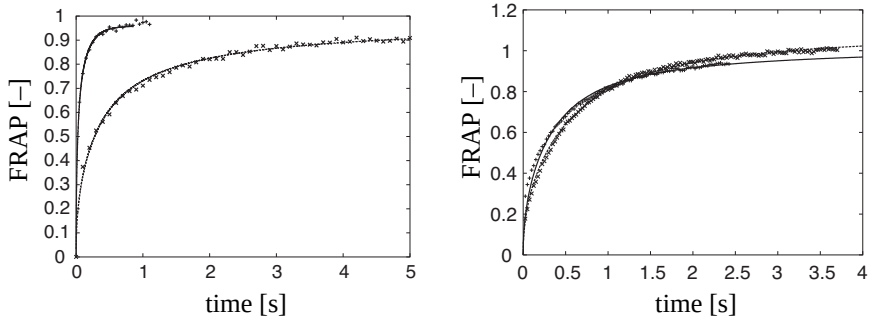
Figure 6.14: The influence of confinement: Diffusion in a cube versus diffusion in an ER. Both simulations are done using the same molecular diffusion constant. Depending on which specific ER sample is used, the recovery half-time for the ER case (solid) is 1.8 to 4.2 times the one of the cube (dashed). Both curves are normalized by their respective asymptotic level to allow geometric comparison.

perimentally measured FRAP curves for two different ROIs in the same ER. The two bleached regions are overlapping and the recovery curves are thus expected to be similar.

Fig. 6.16 visually compares the *fluorescence recovery dynamics* from an experiment and the corresponding simulation. Note that the experimental images show confocal sections, whereas the simulation visualization shows the top-view onto the closed three-dimensional object. The recovery percentages of the simulation and the experiment match within $\pm 1\,\%$.

The influence of the particular geometry

The observed variation in the *factor of underestimation* is due to the different shapes of the individual ER samples. This is illustrated in Fig. 6.17(a), where simulated recovery curves for different ER geometries are compared. All simulations are done using the same value for the molecular diffusion constant. Still, the recovery curves and recovery half-times scatter over a wide range. Not surprisingly, changes in the size or in the position of the ROI in the same ER lead to similar variations, as shown in Fig. 6.17(b). The specific *local geometry* of the organelle around the ROI is thus responsible for variations of about a factor of 2.5 in the observed apparent diffusion constant. Methods that deduce molecular



(a) (b)

Figure 6.15: (a) Simulated FRAP curves compared to experimental measurement data for different ER. The computer simulations are done using to the method of particle strength exchange as outlined in Subsection 5.2.2. The experiment is a standard FRAP experiment according to Appendix E.1, preceded by the recording of a stack of serial sections used to reconstruct the geometry. The simulated FRAP curves (lines) are stretched in time to fit the experimental data (symbols). As time and diffusion constant are inversely proportional, this allows to estimate the molecular diffusion constant, while fully taking the specific geometry into account. For the two examples shown, the molecular diffusion constants are determined to be 34.4 $\mu m^2/s$ (faster curve, $+$), and 34.2 $\mu m^2/s$ (slower curve, $\times$), respectively. All curves are normalized by their asymptotic value to allow comparison. (b) Simulated FRAP curves compared to experimental measurement data for different locations of the bleached region. Two FRAP experiments, followed by corresponding PSE simulations, are performed for two different, but overlapping, bleached regions in the same ER. The result after fitting the simulation results (lines) to the measurements (symbols) is shown. The two bleached regions are given in microscope coordinates as: $\times$ (191,190)-(229,228) and $+$ (218,196)-(256,234), and the molecular diffusion constants for this alternate transfection case (see text) are determined as 1.9 $\mu m^2/s$ ($\times$) and 2.0 $\mu m^2/s$ ($+$), respectively. All curves are normalized by their asymptotic value to allow comparison.

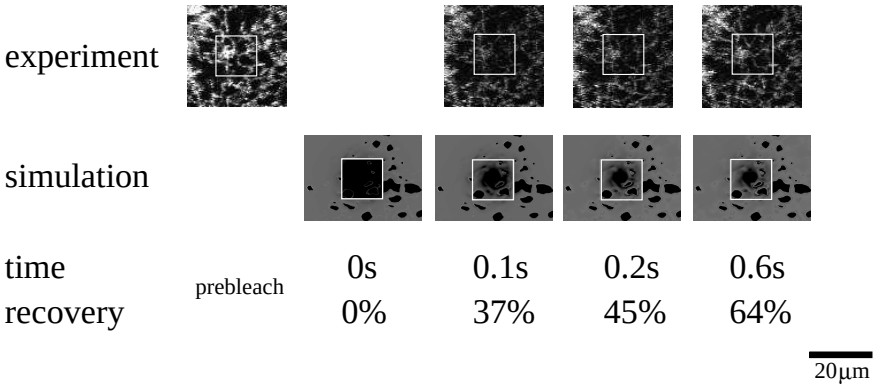
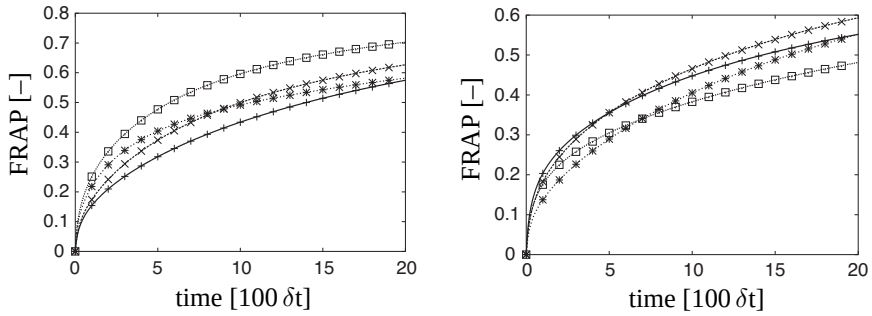


Figure 6.16: Visual comparison between FRAP experiment and computer simulation. Micrographs from a standard FRAP experiment (Appendix E.1) are compared to visualizations from the corresponding computer simulation. The case corresponds to the slower curve in Fig. 6.15(a). Experimental images were acquired every 100 ms with a spatial resolution of 0.18 μm/pixel. The simulation entailed 6.8 million particles and comprised the whole ER. The figure only depicts the portion of the ER in the vicinity of the region of interest. The molecular diffusion coefficient is determined from the fit shown in Fig. 6.15(a) to be 34.2 $\mu m^2/s$. The bleached region is indicated by its outline. No experimental image was acquired during bleaching. Note that the experimental images show a confocal section through the middle of the cell, whereas the visualizations from the simulation show the top-view onto the closed three-dimensional geometry. The recovery percentages of the simulation match those of the experiment to within $\pm 1\,\%$.



(a) (b)

Figure 6.17: (a) Comparison of simulated FRAP curves for four different ER samples. All simulations are done using the same computational diffusion constant and the same simulation parameter settings (see Appendix F.1 for details). All curves are normalized by their asymptotic value to allow comparison. The variations observed in the FRAP curves are solely caused by the different geometries of the ER samples. The recovery half-times vary within the interval $[5.7 \ldots 14.2] \cdot 100\delta t$. (b) Comparison of simulated FRAP curves for different bleached areas in the same ER sample. The bleached regions are given by the microscope coordinates of their lower-left and upper-right corners as follows: $\square$ (225,125)-(300,200) / $*$ (350,200)-(400,250) / $+$ (250,125)-(300,175) / $\times$ (80,300)-(130,350). The simulation parameters and computational ν_2 are kept constant and all curves are normalized by their asymptotic value to allow comparison.

ν_2 values using a "representative" model geometry, or statistically averaged shape models, are unable to account for this situation. Any closed-form model or averaged geometry model [262] thus suffers from this uncertainly, which can only be reduced below 250% by taking the *specific geometry* of the individual organelle into account.

Geometry-corrected diffusion constant of ssGFP–KDEL

The scaling of units as outlined in Subsection 6.5.1 allows to determine *geometry-corrected molecular diffusion constants*. With a length unit of $2.7\,\text{nm}$ and $t_s = 1.6 \cdot 10^{-5}\,\text{s}$ from curve fitting, we find the molecular *diffusion constant of ssGFP–KDEL* in the ER lumen of VERO cells to be $\nu_{\text{eff}} = 34 \pm 0.95 \mu\text{m}^2/\text{s}$, averaged from 8 computer-evaluated FRAP experiments. Depending on the particular ER geometry, the molecular diffusion constant obtained without correcting for the organelle shape is 1.8 to 4.2 times lower. Ignoring the effect of shape thus leads to significant underestimation of molecular diffusion constants.

The reported diffusion constant of pure GFP in water at room temperature is $87\,\mu\text{m}^2/\text{s}$ [75]. This indicates that the material filling the ER lumen is of more than 2.5-fold higher viscosity than water.

Assessment of the method

The diffusion constant of GFP in the ER lumen is reported in the literature as around $10\,\mu\text{m}^2/\text{s}$ [75]. Our value being about 3.5 times larger is consistent with the result that neglecting the geometry leads to underestimation of the molecular diffusion constant by a factor of 1.8 to 4.2. This is further supported by the work of M. Weiss *et al.* who used fluorescence correlation spectroscopy – a particle-level method that directly determines molecular diffusion constants – to measure the molecular ν_2 of the closely related ssYFP–KDEL in the ER lumen of *HeLa cells*. The value obtained by Weiss *et al.* is $30\,\mu\text{m}^2/\text{s}$ (M. Weiss, personal communication, 2002), which is in reasonable agreement with our results, given that two different cell types are considered. Braga *et al.* found a value of $33.3 \pm 3.6\,\mu\text{m}^2/\text{s}$ for GFP in the nucleoplasm of HeLa cells [35], where geometric complexity is of little concern.

A thorough comparison of our approach to the method of Siggia [262] is conducted on FRAP experiments of ssGFP-KDEL in the ER lumen of VERO cells. Great care is taken to record non-saturated images and meet all requirements of both methods.

To ensure that we correctly use the computer program of Siggia *et al.* [262], we test it on an artificially generated time lapse sequence of images showing homogeneous diffusive recovery of a rectangular bleached area in a flat 2D plane, simulated using finite differences and zero flux boundary conditions. The computational ν_2 is $25\,\mu\text{m}^2/\text{s}$, the pixel size is $0.18\,\mu\text{m}$, and Gaussian pixel noise with a standard deviation of 10% of the peak intensity value is added. A sequence of 200 512×512 pixel images with sampling time $\Delta t = 0.05\,\text{s}$ is simulated and evaluated using Siggia's program. The diffusion constant is correctly recovered as $25.08\,\mu\text{m}^2/\text{s}$.

Comparing two different bleached regions from one ER, we observe that the molecular diffusion constants determined by the present method are much less scattered than the ones obtained using Siggia's program. Even though we have no reason to expect the molecular ν_2 to be constant throughout the entire ER, variations of a factor of three and absolute values ranging from 23 to $79\,\mu\text{m}^2/\text{s}$, as predicted by Siggia's method, seem unlikely.

The sensitivity of our method is assessed using an alternative transfection procedure (cf. Appendix E). Again, two different spots of the same ER are bleached and analyzed. PSE evaluations of the corresponding FRAP experiments, shown in Fig. 6.15(b), yield molecular diffusion constants of $2.04\,\mu\text{m}^2/\text{s}$ and $1.86\,\mu\text{m}^2/\text{s}$ for the two areas.

6.5.3 Application to membrane-bound proteins

In the past, FRAP on biological surfaces has mostly used planar diffusion models. For FRAP and the related continuous fluorescence microphotolysis, calculations exist for planar membranes [215], for spherical membranes [41], and for *singly-connected* periodically curved membranes (cosine surfaces) [6]. Several real biological membranes are however much more complex. They can contain tubular networks, holes, large curvature variations, and they are usually not singly-connected. Moreover, diffusion in biological membranes can appear anisotropic even though it is molecularly isotropic in all observed instances [145]. The apparent anisotropy in membrane FRAP experiments is due to different membrane curvatures is different spatial directions [7]. Taking the exact surface geometry into account is thus mandatory for isotropic FRAP models. We present computer simulations of diffusion, using the numerical method presented in Section 5.3 on reconstructed ER membrane surfaces. This allows to investigate the influences of geometry on FRAP and to derive corrected molecular diffusion constants from FRAP data in the ER membrane.

Fig. 6.18 shows the visualized concentration field from a sample simulation at different times after bleaching.

Comparison to experimental data

The simulations are validated by comparing them to FRAP experiments in the same ER shapes. Fig. 6.19(a) shows three fits of simulated FRAP curves to experimental data. Clearly, simulation and experiment are in excellent agreement in all but one of the cases. Closer inspection of the membrane geometry in the differing case reveals the overhanging membrane section shown in Fig. 6.20. At early times, lateral recovery could thus be occluded in the experiment, whereas the simulation always integrates the concentration over the whole membrane surface. In addition, the biochemistry may be different in this more lamellar part of the ER membrane.

The influence of membrane geometry on FRAP experiments

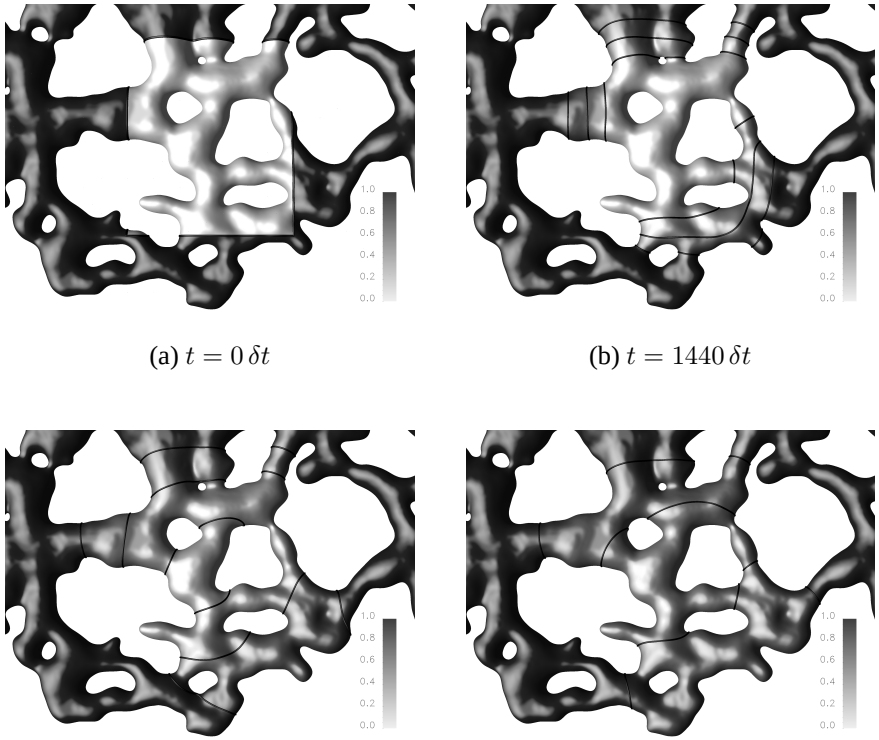
To estimate the geometry-induced uncertainty in membrane FRAP experiments, we perform simulations in the membranes of ER samples that are reconstructed from different cells. The same computational diffusion constant is used in all instances, and all simulation parameters are kept constant (cf. Appendix F.2). The geometrical differences in the membranes are therefore the only source of variation. As shown in Fig. 6.19(b), the recovery half-times vary by a factor of 1.76, which can be explained by membrane curvature effects [6]. The specific shape of the membrane thus affects the recovery half-times to vary by a factor of about 2 for different ER samples.

Geometry-corrected diffusion constant of tsO45-VSV-G

The *geometry-corrected molecular diffusion constant* of GFP-labeled *tsO45-VSV-G* [107] is determined in *VERO cells* as outlined in Appendix E.2. From 4 samples we find $\nu_{\text{eff}} = 0.16 \pm 0.07\,\mu\text{m}^2/s$ at the non-permissive temperature of 40°C. This is a factor of 2.7 lower than the previously published value of $0.45 \pm 0.03\,\mu\text{m}^2/s$ [203], which was obtained in *COS-7 cells* without shape correction.

Comparison of membrane and luminal FRAP

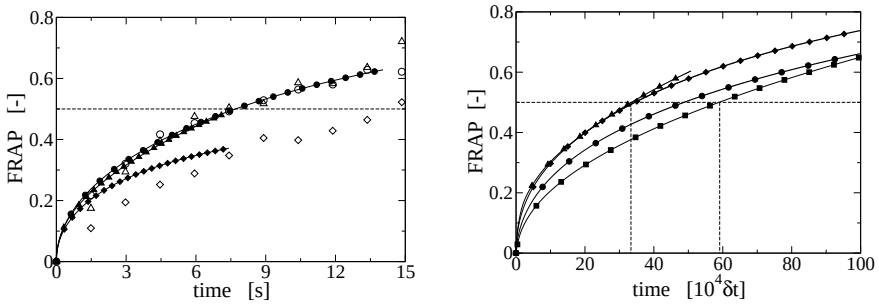
Fig. 6.21 shows the simulated FRAP curves in the lumen and on the membrane of the same ER sample for the same computational diffusion constant. The recovery



(a) $t = 0\,\delta t$ (b) $t = 1440\,\delta t$

(c) $t = 8640\,\delta t$ (d) $t = 17640\,\delta t$

Figure 6.18: Simulation of diffusion in the ER membrane. The concentration on the membrane of a reconstructed ER geometry is shown at times $t = 0\,\delta t$ (a), $t = 1440\,\delta t$ (b), $t = 8640\,\delta t$ (c), and $t = 17640\,\delta t$ (d). The computational diffusion constant is $\nu_{sim} = 1 \cdot 10^{-5}\,b^2/\delta t$, scaled with the simulation time step $\delta t = 0.025$ and the edge length of the bleached ROI $b = 50$. The diffusion operator is supported on $3 \times 3 \times 3$ particles in a narrow band of half-width $\kappa = 2h$ and extended to a larger band of $k = 3h$ every time step using the 2^{nd} order GMM extension method. An Euler scheme with a time step of $\delta t = 0.025$ is used for the first 20 time steps, then a 9-step STS with a time step of 0.45 is used until the final time $24000\,\delta t$. The level function and the concentration field are discretized using 1.7 million particles. The concentration on the surface is recovered from the adjacent particles using linear interpolation along inter-particle lines, and visualized as surface color density. The three black lines indicate the 25%, 50%, and 75% recovery iso-lines.



(a) (b)

Figure 6.19: (a) Comparison of simulations and experiments. The simulated FRAP curves (lines with filled symbols) are stretched in time to fit the experimental data (corresponding open symbols). As time and diffusion constant are inversely proportional, this allows to estimate the molecular diffusion constant, while fully taking the specific geometry into account. For the three examples shown, the molecular diffusion constants are determined to be 0.13 $\mu m^2/s$ (circles), 0.24 $\mu m^2/s$ (triangles), and 0.12 $\mu m^2/s$ (diamonds). All curves are normalized by their asymptotic value to allow comparison. (b) The influence of the ER membrane geometry. All simulations are done using the same computational diffusion constant and the same simulation parameter settings (Appendix F.2). All curves are normalized by their asymptotic value to allow comparison. The variation observed in the FRAP curves is therefore only caused by the different geometries of the ER samples. The recovery half-times scatter in the interval $[33 \ldots 58] \cdot 10^4 \delta t$.

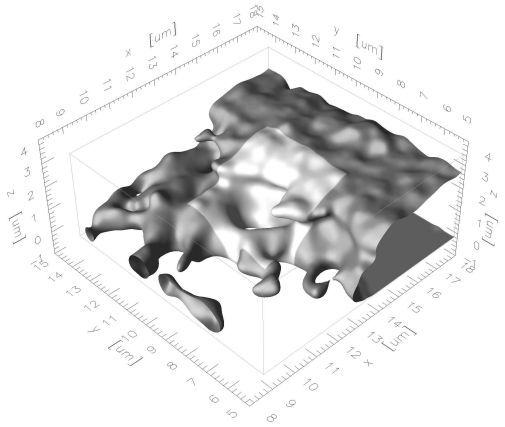


Figure 6.20: Visualization of a reconstructed ER membrane piece with overhanging regions. A $9 \times 9\,\mu m$ *neighborhood around the bleached ROI (bright spot) is shown. The case corresponds to the diamond symbols in Fig. 6.19(a), where the discrepancy between the simulation and the experiment could be caused by the depicted overhanging membrane piece.*

half-times are 111 for the luminal protein and 485 for the membrane-bound protein. This indicates that the diffusion behavior of molecules in the ER membrane differs significantly from the volumetric diffusion of soluble molecules in the lumen of the same ER. The apparent speed of recovery differs by a factor of about 4, even if the molecular diffusion constants of the two species are identical.

6.5.4 Conclusions

The results of this section demonstrate that for complex-shaped organelles neither the confinement caused by the 3D shape of the organelle, nor the specific geometry of the sample can be neglected when experimental fluorescence recovery data are used to derive molecular diffusion constants. Moreover, diffusion of membrane molecules is significantly different from diffusion of soluble components in the ER lumen. Models used to calculate diffusion in the ER or any other intracellular organelle have to take these influences into account if they should be free of systematic errors. In the case of the ER, the correction factors are in the 2 to 4-fold range. The actual magnitude depends on the complexity of the particular 3D shape as well as on the local density of small-scale structures. If one is in-

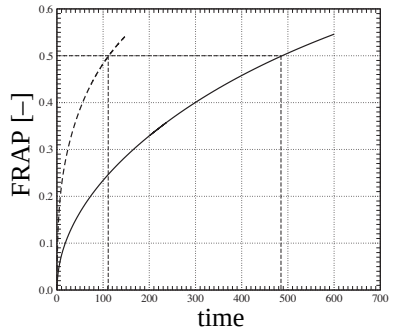


Figure 6.21: Comparison of FRAP curves in the lumen (dashed) and on the membrane (solid) of the same ER sample. Both curves correspond to the same computational diffusion constant and are normalized by their asymptotic value. The recovery half-time (in simulation time units) is 111 for the luminal protein and 485 for the membrane protein, indicating that recovery in the lumen is significantly faster than recovery on the membrane.

terested in determining *molecular weights* based on measured molecular diffusion constants, this uncertainty correspond to an 8 to 64-fold error in weight, since a soluble molecule's volume scales with the third power of its molecular diffusion constant [174]. For membrane-bound proteins the situation is even worse as their radius depends exponentially on the diffusion constant [234].

We demonstrated that FRAP models derived for planar membranes yield incorrect molecular diffusion constants when applied to curved membranes. The factor of about 2 can be explained by purely geometric effects. Moreover, diffusion appears anisotropic if the membrane has different curvatures in different directions [7]. Isotropic models are thus only valid when one accounts for the real membrane geometry in the vicinity of the ROI. Membrane FRAP models should not be applied to luminal proteins and vice versa, as the apparent diffusion constants differ by a factor of about 4.

Our results indicate that it is possible to perform accurate and fully resolved computer simulations in experimentally recorded organelle shapes from confocal sections. This enables the estimation of the geometry-induced uncertainties in the calculation of molecular diffusion constants from FRAP data. The employed numerical particle methods are crucial in doing so, since they avoid many of the problems that grid-based methods have in complex geometries, and their convergence is fast enough to limit the number of particles to feasible ranges. The computational cost of the algorithms is low enough for them to be used in quantitative analyses of FRAP experiments. Moreover, all simulation programs are parallelized using the techniques presented in Chapter 7. This allows to further reduce the computational time by using a cluster of computers that are connected by a network.

Chapter 7

PPM – A Software Framework for Parallel Particle-Mesh Simulations

The dynamics of particle methods are governed by the interactions of the N computational particles, resulting in an N-body problem with a computational cost that nominally scales as $\mathcal{O}(N^2)$ (cf. Section 5.1). For short-ranged particle interactions, as in simulations of diffusion according to Chapter 5, the computational cost scales linearly with the number of particles. In the case of long-range interaction potentials such as the Coulomb potential in electrostatics, the gravitational potential in astrophysics, or the Biot-Savart law in *Vortex Methods* (VM), *Fast Multipole Methods* (FMM) [122] reduce the computational cost to $\mathcal{O}(N)$. Alternatively, long-range interactions can be described by equivalent *field equations*, such as the Poisson equation, that are solved on meshes, resulting in hybrid *Particle-Mesh* (PM) algorithms as outlined in Subsection 5.1.2 [127, 138]. The computational cost of hybrid methods scales as $\mathcal{O}(M)$, where M denotes the number of mesh points used for resolving the field equations. The choice between FMM and PM techniques is dictated by the boundary conditions of the problem with FMM techniques allowing more flexibility on their specification, while PM schemes are well suited for periodic systems. An important factor, distinguishing FMM and PM techniques, is the parallelization efficiency of these methods, as the mesh regularity of the PM algorithm enables implementations that are typically one or two orders of magnitude faster than corresponding FMM [65, 309] implementations. Moreover, FMM-based particle methods have limited scalability on shared memory systems [66], while their implementation in distributed memory environments is difficult due to the inherent global nature of the underlying tree data structure. It is important to observe, however, that even when FMM are used for the evaluation of the particle interactions, the need for hybrid PM algorithms is imperative in adaptive particle methods – such as VM or *Smooth Particle Hydrodynamics* (SPH) – for the reinitialization of the distorted particle locations [160] in order to maintain the overlap condition in Eq. (5.7).

Despite their versatility and physical link, the parallel implementation of PM techniques is complicated by several factors:

- exploiting the symmetry of the particle interactions requires the sending back of ghost contributions to the proper real particle,
- the simultaneous presence of particles and meshes prohibits a single optimal way of parallelization,
- complex-shaped computational domains and strong particle inhomogeneities require spatially adaptive domain decompositions,
- particle motion may invalidate the existing domain decomposition, causing rising load imbalance and hindering the implementation of multi-stage integration schemes,
- inter-particle relations constrain decompositions and data assignment.

Moreover, different physical problems parallelize differently: *Molecular Dynamics* (MD) simulations require elapsed times below one second per time step to allow the hundreds of thousands or millions of time steps typically required in these simulations. This severely limits the available time for communication. Computational fluid dynamics simulations require less time steps with each time step taking tens of seconds or minutes, thus leaving less stringent requirements on the communication overhead. Gravitational systems often develop strong inhomogeneities (large particle density variations) and require adaptive domain decompositions and load (re-)balancing.

Many of the available domain decomposition, load balancing, solver, interpolation, and data communication methods are applicable to a wide range of particle or hybrid PM algorithms, regardless of the specific physics simulated [46, 160]. Meta-languages such as *Linda* [47] inspire us to provide a general-purpose platform for the parallel implementation of PM algorithms. The inherent loss of computational efficiency and portability of meta-languages [295] is however to be avoided. In this chapter we present the newly developed *Parallel Particle Mesh library* PPM [248]. It provides a generic, physics-independent infrastructure for simulating discrete and continuum particle, mesh, and hybrid particle-mesh systems, and it bridges the gap between infrastructure libraries and application-specific simulation codes.

The core of the PPM library provides several adaptive domain decomposition schemes, multiple processor assignment methods, load balance monitoring, dynamic load balancing, data mapping (sending and receiving), update of overlap regions, parallel file I/O, optimized inter-processor communication using Vizing's

approximate solution of the minimal edge coloring problem [303], neighbor lists (cell lists and Verlet lists [301]), routines for adaptive trees, as well as particle-to-mesh and mesh-to-particle interpolation. This core infrastructure is supplemented with commonly used numerical methods such as mesh-based solvers, evaluation of differential operators on particles [93], FMM, parallel FFT, and multi-stage ODE integrators. Moreover, the PPM library provides bindings for the external libraries fftw, MathKeisan FFT (NEC, Inc.), and METIS[1] [152].

The design goals of PPM include ease of use, flexibility, state-of-the-art parallel scaling, good vectorization, and platform portability.

Ease of use is provided by limiting the number of user-callable functions and using generic interfaces for overloading different variants of the same task. The PPM library has demonstrated its ease of use in the process of developing several client applications for PSE, SPH, VM, and Dissipative Particle Dynamics (DPD). The library is portable through the use of standard languages (Fortran 90 and C) and libraries (MPI) and it was successfully compiled and used on Intel/Linux, Apple/MacOS X, IBM Power/AIX, NEC SX/SUPER-UX, and AMD/Linux systems on 1 to 242 processors. Computational efficiency is achieved by dynamic load balancing, dynamic particle re-distribution, explicit *message passing*, and the use of simple data structures.

After reviewing presently available parallel codes and libraries for scientific simulations in Section 7.1, we present the concepts and modules of the PPM library in Section 7.2. The parallel scaling and efficiency of PPM are assessed in the benchmark cases presented in Section 7.3.

7.1 Review of software and libraries for parallel computer simulations

A number of parallel implementations and libraries have addressed the aforementioned parallelization issues in various fields of application. They globally fall into two categories: *infrastructure libraries* that do not implement particular numerical methods and focus on data communication, and application-specific *simulation libraries*.

Infrastructure libraries have been developed to alleviate the large programming overhead associated with the parallel implementation of simulation applications for various physical problems. Most of these libraries are non-numerical. Examples include the parallel scalable I/O library *PASSION* [282], supporting parallel file I/O including data pre-fetching and data sieving. Run-time support for Monte Carlo simulations is provided by the *PARTI* library [195]. Besides different domain decomposition schemes, this library also implements the *Stop-At-Rise* (SAR) heuristic [195] for deciding when to dynamically re-decompose the problem to achieve a good trade-off between the cost of domain decomposition and the arising load imbalance from particle motion. The *Parallel Utilities Library* (PUL) [52] provides domain decomposition methods, data communication, and parallel file I/O for mesh-based schemes that use either structured or unstructured meshes. The *Bisect* package [328] provides recursive orthogonal bisection domain decomposition, communication through overlap regions, and a parallel bin sorting algorithm for load balancing. Purely numerical infrastructure is provided by libraries such as *PETSc* [18], a parallel linear algebra and equation solving package. PETSc integrates and interfaces BLAS, ScaLAPACK, and many other mathematical libraries.

[1] A library for graph partitioning for load assignment.

Application-specific simulation libraries can be classified into purely mesh-based implementations with no support for particles, purely particle-oriented implementations, and hybrid *Particle-In-Cell* (PIC) libraries. Purely mesh-based libraries include Prometheus, Hypre, and various finite element libraries such as the one in Ref. [74]. *Prometheus* is a parallel multigrid library for finite element applications [4] that is based on PETSc and *ParMETIS* [152, 153]. *Hypre* is a library of efficient parallel pre-conditioners (mainly algebraic multigrid) and solvers such as conjugate gradient and GMRES [98]. Mesh-based multilevel and multi-resolution simulations are supported by libraries that implement the *Adaptive Mesh Refinement* (AMR) framework [29] such as, e.g., *AMROC* [17, 81].

In purely particle-based applications with long-range potentials, parallel FMM [121, 236, 310] and parallel Barnes-Hut methods [117] have been used. Parallel atomistic simulations [131, 226, 219, 130, 95] and MD codes [170, 231, 178, 275] such as *PARALLACS* [200], *GAUSSIAN* [105], and *CPMD* [67, 189, 13] are well established. The parallelization techniques range from event-driven MD [192], over domain decomposition techniques [222], multiple levels of parallelism [292], and multi-resolution techniques [151, 315], to MD using FMM [34]. These techniques have enabled large simulations comprising several millions of atoms [16, 44]. Parallelization for short-ranged many-body potentials [62] and specific adaptive load balancing schemes for MD [83] are available as well.

Hybrid PM methods include PIC and *Particle-Particle Particle-Mesh* (P^3M), capturing sub-grid scale phenomena by particles [138]. Parallel application codes using P^3M are found, e.g., in astronomy [37] and in large cosmological simulations [179]. *Hydra* [211] is a parallel adaptive P^3M code. Parallel PIC codes

[78] are widely used in plasma physics, where the motion of charges is tracked by particles while Maxwell's equations are solved on the mesh. Probably the first parallel PIC code, published in 1990 [171], was a two-dimensional plasma simulation with random particle-to-processor assignment, running on a massively parallel CONVEX computer. An object-oriented PIC implementation for accelerator dynamics [224], and an HPF/MPI code with limited scalability [176] have also been published. More recent PIC codes include *VORPAL*, a versatile plasma simulation code [206], and a magneto-hydrodynamics code using particle decomposition techniques instead of domain decomposition [84]. *PICARD* is a parallel PIC library, rather than a specialized application code [46]. It can be applied wherever the PIC formulation is used.

State-of-the-art simulation codes show good parallel efficiencies up to large numbers of processors. The electromagnetic PIC code *QUICKSILVER* [220] for example demonstrated a parallel efficiency of 60% solving a scaled-size irregular case on 1024 processors. 90% were achieved in the ideal uniform load case on 3200 processors [220]. Parallel molecular dynamics using CPMD [67] has reached 50% efficiency on 1280 processors of an IBM p690 at 15% of peak performance. Astrophysical Barnes-Hut tree codes for gravitational systems have achieved 70% to 90% efficiency on 128 [297] to 256 processors [86]. Using the *GADGET* cosmology code [270], a recent simulation at the Max Planck Institute for astrophysics has used more than 10 billion particles (3082^3 mesh) in an SPH simulation on 512 IBM p690 processors. Lattice Boltzmann simulations of flow in porous media demonstrated linear scaling up to 128 processors [209]. A finite element code based on the multigrid library Prometheus was used for bone biomechanics simulations on up to 4088 IBM Power3 processors at 7.5% of peak performance. The computational mesh consisted of 125 million elements, corresponding to half a billion unknowns, and a parallel efficiency of 41% on 128 processors was achieved [5]. Yokokawa *et al.* [326] presented a direct numerical simulation of incompressible turbulence using 4096^3 mesh points with a parallel efficiency of 48% on the 512 NEC SX-5 processors of the Earth Simulator. The ASCI Red project even achieved 74% efficiency solving the Poisson equation on 9632 processors using a 362880^2 mesh.

7.2 The PPM library

7.2.1 Concepts and fundamental assumptions

The use of the PPM library requires that the simulated systems are formulated in the framework of PM algorithms as outlined in Section 5.1. The field equations are solved using structured or uniform Cartesian meshes. As a result, the physical and computational domains are rectangular or cuboidal in two or three dimensions. Complex geometries are handled by immersed boundaries, through the use of source terms in the corresponding field equations, or through boundary element techniques. Adaptive multi-resolution capabilities are possible using mapping concepts as adapted to particle methods [27] (cf. Subsection 5.4.4).

The simultaneous presence of particles and meshes requires different concurrent domain decompositions. These decompositions divide the computational domain into a minimum number of cuboidal *sub-domains* with sufficient granularity to provide adequate load balancing. The concurrent presence of different decompositions allows to perform each step of the computational algorithm in its optimal environment with respect to load balance and the computation-to-communication ratio. For the actual computations, the individual sub-domains are treated as independent problems and extended with *ghost mesh layers* and *ghost particles* to allow for communication between them. *Ghosts* are copies of true mesh points or particles that either reside on a neighboring processor or account for periodic boundary conditions. Ghost particles and/or ghost mesh points are needed for all local operations such as finite support particle-particle interactions, finite difference stencils, and PM interpolations.

The PPM library supports *connections* and *relations* between particles, such as particle pairs, triplets, quadruplets, etc. These relations may describe a physical interaction, such as chemical bonds in molecular systems, or a spatial coherence, such as a triangulation of an immersed boundary or an unstructured mesh. Multiple sets of connections may coexist as the list of connections is allocated and stored by the client program. The entries of the connection lists point to the unique, global particle index which must be stored separately for each particle in this case.

To achieve maximum efficiency, the PPM library splits all the major routines into an initialization step, processing, and finalization. This ensures reusability of potentially expensive initialization, and facilitates memory management. On the whole-library level, the routines `ppm_init` and `ppm_finalize` provide this functionality. The routine `ppm_init` sets the problem dimensionality (two or three), the floating point precision for the internal communication buffers (single

precision or double precision), the numerical tolerance for floating point comparisons (differences smaller than this tolerance are considered zero), the level of detail in output and log messages, and the output devices for error and log messages. Calling `ppm_finalize` deallocates all internal buffers and closes all I/O devices.

Memory for internal lists and communication buffers is allocated by the PPM library. All other memory, such as simulation data (particles, fields) and index lists (cell lists, Verlet lists, etc.), is held by the client application. This ensures user-control over the data and allows multiple different sets of particles, connections, and fields to be used concurrently. The number of decompositions, sub-domains, particle sets, fields, and meshes is only limited by the cumulative memory capacity of all processors.

7.2.2 Topologies

A *topology* is defined by the decomposition of space into sub-domains with the corresponding boundary conditions, and the assignment of these sub-domains onto processors. Multiple topologies may coexist and library routines are provided to *map* particle and field data between them as described in Subsection 7.2.3. Fields are defined on *meshes*, which in turn are associated with topologies. Every topology can hold several meshes. The only constraint is that sub-domain boundaries must align with mesh lines/planes.

As the domain decomposition may take several seconds to complete, a given topology is assumed to persist through longer periods of the simulation. For problems with free-space boundary conditions the extent of the computational domain is adjusted in order to enclose all particles at any time. An extra margin may be added to the computational domain to avoid repeated update of the topology. For problems in confined systems, subject to, e.g., periodic boundary conditions, the extent of the computational domain is fixed and the decomposition is performed filling the entire space, disallowing void space(s). This assures that particles can not leave the computational domain, which would require an immediate, potentially expensive, re-decomposition.

In order to achieve good *load balance*, both the *load distribution* and the computational cost of the topology creation are monitored throughout the simulation. The *SAR heuristic* [195] is used in the PPM library to decide when problem re-decomposition is advised, i.e. when the cost of topology re-definition is amortized by the gain in load balance. Moreover, all topology definition routines can account for the true computational cost of each particle, for example defined by the actual

number of its interactions. A routine is provided to compute this number based on the lengths of Verlet lists.

Domain decompositions

The PPM library provides a number of different adaptive domain decomposition techniques for particles, meshes, and volumes, the latter defining geometric sub-domains with neither meshes nor particles present. These decompositions currently include: recursive orthogonal bisection, x-, y-, and z-pencils, xy-, xz-, and yz-slabs, cuboids, and a user-defined decomposition. *Recursive Orthogonal Bisection* (ROB) is based on the adaptive binary tree described in Subsection 7.2.9, where subdivisions are allowed in all spatial directions. *Pencil decompositions* prohibit subdivisions in one direction, resulting in an adaptive decomposition where each sub-domain extends over the whole computational domain in at least one spatial dimension. Such decompositions are useful when performing fast Fourier transforms. In *slabs*, two directions are fixed. *Cuboids* are created using adaptive quad- and oct-trees in two and three dimensions, respectively, and the *user-defined decomposition* allows the client program to explicitly specify the sub-domains. After checking the validity of such a decomposition, the PPM library directly proceeds with assignment of the sub-domains to the processors.

In addition, a special *null decomposition* is provided, that does not perform any domain decomposition. It creates only one "sub-domain" which is the computational domain itself. This trivial "decomposition" is used to evenly distribute the particles among processors, irrespective of their spatial location. The resulting special topology is called the *ring topology*, where the sub-domain is assigned to every processor. The ring topology supports full $\mathcal{O}(N^2)$ calculations, and also allows to distribute data of initially unknown processor affiliation (cf. Subsections 7.2.3 and 7.2.4).

Assignment of sub-domains onto processors

Load balancing in the PPM library comprises two main components: domain decomposition and *assignment* of the sub-domains onto the processors. While the former has to ensure sufficient granularity and partitioning of the computational cost, the latter has to provide even distribution of the computational load among processors, accounting for possible differences in processor speeds. The computational cost for each sub-domain – as determined by the number of particles, the number of mesh points, or the true computational cost – is known from the domain

decomposition. The individual processor speeds are measured internally by PPM, solving a small *Lennard-Jones* system [10] with an increasing number of particles until all processors report sufficient timing statistics.

Using this information, PPM provides several methods of assigning the sub-domains onto the processors. The PPM-*internal* method assigns contiguous blocks of sub-domains to processors until the accumulated cost of a processor is greater than or equal to the theoretical average cost under uniform load distribution. The average is weighted with the relative processor speeds. In addition, four different METIS-*based* [152] assignments and a user-defined assignment are available.

In conjunction with a user-defined domain decomposition, the *user-defined assignment* scheme allows the client program to enforce a specific processor affiliation for each sub-domain. For a METIS assignment, the sub-domain partitioning problem is first translated to the equivalent *graph partitioning* problem. Two different conversions are supported by METIS: in the *primal* scheme, each sub-domain is represented by a node in the graph and the neighborhood relations by the edges of the graph; the *dual* scheme represents sub-domains by graph edges. Graph partitioning is then performed such as to minimize either *edge cut* or *communication volume* [152, 153]. The relative processor speeds and the computational costs of the sub-domains are accounted for by means of weights that are assigned to the nodes and edges of the graph.

Boundary conditions

At the external boundaries of the computational domain, Neumann, Dirichlet, free space, symmetric, and periodic boundary conditions are supported. These conditions complement the particular mesh-based solver that is being employed. More involved boundary conditions and complex boundary shapes are represented inside the computational domain by defining connections among the particles, or using *immersed interfaces*.

Benchmark tests

To assess the behavior of the different topology schemes, we compare them on four test cases using 16 processors. The quality of decomposition is quantified by the standard deviation of the number of particles across processors, and by the average number of ghost particles needed per processor. The domain is decomposed using a non-adaptive binary tree, ROB, and an adaptive oct-tree. Assignment of the sub-domains onto the processors is done using the external library METIS [152] to

Particle distribution	Non-adaptive tree	ROB	Adaptive oct-tree
Standard deviation of particles per processor			
Uniform	422	268	265
Sphere	62501	1865	2626
Spiral	73350	2336	6011
Diagonal line	108255	148	161
Average number of ghost particles per processor			
Uniform	33847	33832	33750
Sphere	35268	36526	28187
Spiral	10584	31102	22297
Diagonal line	20050	28940	46232

Table 7.1: Comparison of different domain decomposition schemes on four test problems, each involving 1 million particles on 16 processors (see text). For all schemes, the equi-distribution of particles and the total communication overhead, measured by the number of ghost particles, are reported.

minimize the total length of the communication boundaries. One million particles are distributed in the unit cube in four ways: uniformly, on the surface of a sphere with radius 0.25, on a spiral, and on the diagonal line from the point $(0, 0, 0)$ to the point $(1, 1, 1)$. The computational time needed to construct the topologies on 16 2.2 GHz AMD Opteron 248 processors is about 30 milliseconds per sub-domain in all cases, and the results are summarized in Table 7.1.

7.2.3 Mapping

PPM topologies implicitly define a data-to-processor assignment. Mapping routines provide the functionality of sending particles and field blocks to the proper processor, i.e. to the one that "owns" the corresponding sub-domain(s) of the computational space. Three different mapping types are provided for both particles and field data:

1. a *global mapping*, involving an all-to-all communication,
2. a *local mapping* for neighborhood communication, and
3. *ghost mappings* to update the ghost layers.

In addition, a special *ring shift mapping* is provided for particle data on the ring topology, and a *connection mapping* for taking into account links between particles.

The global mapping is used to perform the initial data-to-processor assignment or to switch from one topology to another, whereas the local mapping is mainly used to account for particle motion during a simulation. Communication is scheduled by solving the *minimum edge coloring* problem using the efficient approximation algorithm by *Vizing* [303, 85, 87]. Ghost mappings are provided to receive ghost particles or ghost mesh points, and to send ghost contributions back to the corresponding real element, for example after a symmetric particle-particle interaction or a particle-to-mesh interpolation. The ring shift mapping sends data-sets around all processors, while each processor keeps a local copy of its original data. Finally, connection mappings are provided to distribute connections among processors according to an existing distribution of particles, and to update the connection lists when particles have moved across processor boundaries.

All mapping types are organized as *stacks*. A mapping operation consists of four steps: (1) defining the mapping, (2) pushing data onto the *send stack*, (3) performing the actual send and receive operations, and (4) popping the data from the *receive stack*. This architecture allows data stored in different arrays to be sent together to minimize network latency, and mapping definitions to be re-used by repeatedly calling the push/send/pop sequence for the same, persisting mapping definition. The individual mapping types only differ in their definition step, while push, send, and pop are identical.

All mapping subroutines of PPM are available in separate optimized versions for scalar and vector data. Supported data types for particles and fields are: single and double precision floating point, single and double precision complex numbers, integer numbers, and logical values. Different data types can be mixed within the same stack, in which case they are converted to the stack data type as defined by the `ppm_init` routine (cf. Subsection 7.2.1).

Mappings of field data can be masked. In this case, an optional *binary mask* selects which mesh points are to be mapped and which ones are not. The values of non-mapped points remain unaffected by the mapping operation.

Global mapping

The *global mapping* in the PPM library involves communication of all processors with all others. Individual communication steps are synchronous, and scheduled such that no conflicts occur. Thus, all processors i, $i = 1, \ldots, N_{\text{proc}}$, send to $i + r$ (mod N_{proc}) while they receive from $i - r$ (mod N_{proc}). This is repeated for all shifts $r = 1, \ldots, N_{\text{proc}} - 1$.

The definition of a global mapping involves the creation of an index list of particles or mesh blocks that fall outside the local processor, and a list of their new processor affiliation.

As an example, the global mapping of `Np` particles with positions `xp(:,:)` (`ndims` $\times$ `Np` floating point array) and vector properties `wp(:,:)` (`nprop` $\times$ `Np` floating point array) consists of the following sequence of calls to the `ppm_map_part` routine:

```
CALL ppm_map_part(xp,ndims,Np,Mp,tid,ppm_param_map_global,info)
CALL ppm_map_part(wp,nprop,Np,Mp,tid,ppm_param_map_push  ,info)
CALL ppm_map_part(wp,nprop,Np,Mp,tid,ppm_param_map_send  ,info)
CALL ppm_map_part(wp,nprop,Np,Mp,tid,ppm_param_map_pop   ,info)
CALL ppm_map_part(xp,ndims,Np,Mp,tid,ppm_param_map_pop   ,info).
```

`Np` is the number of non-ghost particles on the local processor before the mapping, `Mp` is the new number of particles after the mapping, and `tid` is the unique identification number of the target topology for the mapping. This target topology needs to be defined beforehand, using the PPM topology creation routine `ppm_mktopo`. The error status of each step is returned in `info`. For the scalar version of the mapping routines, the second argument is absent and the first one is a rank 1 array. For reasons of efficiency, the first call to `ppm_map_part` not only defines the mapping by creating all the lists, but also directly pushes the particle positions onto the send stack. The pop action thus needs to be issued once more than the push action.

Mapping an `ndim`-dimensional vector field `fld`, that is defined on the mesh `mid`, requires the following sequence of calls to `ppm_map_field`:

```
CALL ppm_map_field(fld,ndim,tid,mid,to_mid,gs,ppm_param_map_global,
     info)
CALL ppm_map_field(fld,ndim,tid,mid,to_mid,gs,ppm_param_map_push  ,
     info)
CALL ppm_map_field(fld,ndim,tid,mid,to_mid,gs,ppm_param_map_send  ,
     info)
CALL ppm_map_field(fld,ndim,tid,mid,to_mid,gs,ppm_param_map_pop   ,
     info).
```

The identifier `to_mid` specifies the target mesh, which needs to be defined on the target topology `tid`. The width of the ghost layer in units of the mesh spacing is given in `gs(:)` for each spatial direction. The scalar version of the routine lacks the second argument and the rank of the field array `fld` is reduced by one.

Local mapping

During a *local mapping*, each processor only communicates with those processors that have sub-domains that are adjacent to any of its own sub-domains. The optimal communication sequence would satisfy that no conflicts occur, and that the minimum number of communication steps is needed. The problem of finding this sequence can be formulated in a *graph representation*, where each processor is a node of the graph. An edge in the graph denotes a neighborhood relation, i.e. a necessary communication. The goal is to find a coloring of the edges such that no two edges of the same color meet in any node, and such that the minimum number of different colors (corresponding to communication steps) is needed. This *minimum edge coloring* problem is NP-complete [139]. An approximate solution can however efficiently be found using the algorithm of *Vizing* [303]. This solution guarantees that at most one color more is used than the minimum number [45]. PPM uses this algorithm to pre-compute and store the *communication schedule* for each defined topology [85, 87].

Periodic boundary conditions at the outer faces of the computational domain are automatically accounted for. Executing a local mapping is analogous to executing a global mapping, except that the parameter `ppm_param_map_partial` is used in the first call to the corresponding mapping routine.

Receiving ghosts

The PPM library provides the mapping routines `ppm_map_part_ghost` and `ppm_map_field_ghost` for handling ghost layers. *Receiving ghosts* and obtaining the copied values is done using the `ppm_param_map_ghost_get` parameter. For particles, ghosts can have both a position and values. Notice that in the case of stationary particles, the stack architecture of PPM's mappings allows to update the ghost values without re-defining the lists or re-sending the ghost positions.

Sending ghost contributions back

When performing symmetric particle-particle interactions or when using particle-to-mesh interpolation, the value at the location of a ghost may change. This gives rise to *ghost contributions* to the corresponding real particle or mesh point on the source processor. In order to add these contributions back onto the proper real element, PPM provides a *ghost sending* mechanism (`ppm_param_map_ghost_put`). The library automatically keeps track of which

real element(s) correspond to a ghost particle or a ghost mesh point, and of possible periodic images in the case of periodic boundary conditions.

Ring shift

When the PPM ring topology (cf. Subsection 7.2.2) is used to compute direct interactions or to distribute data of not globally known processor affiliation, the *ring shift mapping* (`ppm_map_part_ring_shift`) is needed. In this mapping, each processor keeps a local copy of its data while a second copy is "sent around the ring". This means that processor i receives the data from processor $i-1 \pmod{N_{\text{proc}}}$ while sending its previous data to $i+1 \pmod{N_{\text{proc}}}$. The ring shift mapping performs one such step upon each call. It thus has to be executed $N_{\text{proc}}-1$ times for the traveling set to visit all processors. After every ring shift, each processor can perform local operations using its original local data as well as the current traveling set. During a complete cycle, all possible pair combinations are thus considered.

Mapping of particle connections

To allow disjoint initialization or input of particles and the corresponding *connections*, the latter are typically initialized or read separately and are not sorted by processor. The PPM *connection mapping* is provided to properly distribute the connections among the processors. The ring topology is conveniently used for this mapping, in which each processor picks those connections from the data being transmitted on the ring that have entries that correspond to one of its particles. After the connection mapping, every processor has those connections that are associated with any of its particles. This allows non-symmetric calculations of the interactions. If symmetry is used, only one processor performs the calculation. The mapping is thus followed by a *pruning of the connections* to assure that only the one processor that has all the member particles of a connection keeps it. This is a sufficient condition since the ghost layers in PPM are non-overlapping, as illustrated in Fig. 7.1.

As the particles move to other processors during the course of a simulation, the connections are updated accordingly. In the case of symmetric interaction evaluations, the mapping is again followed by the pruning step as described above, in order to remove redundant connections.

7.2.4 Particle-Particle interactions

The evaluation of *Particle-Particle* (PP) interactions is a key component of PM algorithms. Sub-grid scale phenomena can require local particle-based corrections [306], differential operators can be evaluated on irregular locations [93], or the main dynamics of the system may be governed by particle interactions.

The PPM library implements PP computations using cell lists, Verlet lists [301], and the full $\mathcal{O}(N^2)$ direct method. Both *symmetric* and *non-symmetric* interactions are supported, the former to reduce the amount of duplicated work. In each method, the interaction potential or kernel can be specified either by a function pointer to a user function, by passing a look-up table of kernel values, or by choosing one of the predefined PPM-internal kernels: first order derivatives in two and three dimensions, Laplacian on particles in two and three dimensions, and quadratic spline kernels for derivatives in SPH.

In addition to the routines performing the actual computations, the PPM library also provides a routine to create look-up tables from either a function pointer or an internal kernel. Such tables can then be passed to any of the compute routines for the evaluation.

Alternatively, the client program can implement its own interaction routines. Template subroutines for the use of cell lists, Verlet lists, direct interactions, and connection interactions are provided.

Direct interactions

The *direct evaluation* of the full N-body problem makes use of the PPM ring topology, which is based on the null decomposition as introduced in Subsection 7.2.2. Particles are evenly distributed among processors, irrespective of their location in space. This results in optimal load balance [46], but high communication overhead. Using the ring topology to perform direct interactions, each processor keeps a copy of its assigned particles while a second copy is "sent around the ring" using the ring shift mapping as described in Subsection 7.2.3. After each ring shift, all processors compute the pair interactions between the local stationary set and the current traveling set. For asymmetric interactions, contributions are only added onto the local set, for symmetric interactions the traveling set is updated as well. This is repeated until the sets of all processors have completed their trip around the full, for asymmetric interactions, or half of the ring, for symmetric interactions. In the case of symmetric interaction evaluations, the accumulated contributions on the traveling set are finally sent back to their origin for summation.

Cell list interactions

Cell lists are provided for local, *short-range* interactions. Hereby, particles are sorted into equi-sized cuboidal cells, whose size reflects the interaction cut-off. A particle then only interacts with the other particles in the same cell and with all particles in neighboring cells, accessible through index lists. In PPM, cell lists are defined per sub-domain and *ghost cells* are used around each sub-domain as illustrated in Fig. 7.1. If two sub-domains are on the same processor, no communication is needed to populate the ghost cells. The evaluation of PP interactions based on cell list, as well as the set-up of the cell lists, is $\mathcal{O}(N)$. The lists need to be updated when particles have moved.

PPM provides routines to create cell lists, to sort particles into cells, viz. create the index lists, and to compute the actual PP interactions using cell lists. An additional routine is provided for determining which cell-cell interactions have to be considered. Using asymmetric PP interactions, each cell interacts with all of its neighbors as shown in Fig. 7.1(a). For symmetric interactions however, only half of the neighbor cells, and half of the particles in the center cell, are to be considered. In order to limit the ghost layer to half of the boundaries of the sub-domain and to achieve parallel scaling in memory, a novel interaction scheme involving *diagonal interactions* is introduced as illustrated in Fig. 7.1(c). This scheme reduces the amount of memory overhead and communication for symmetrically evaluated PP interactions by 33% in two dimensions and 40% in three dimensions compared to the classical approach depicted in Fig. 7.1(b), and it enables the symmetric evaluation of interactions between connected particles. Given the cells are numbered in ascending x, y, (z), starting from the center cell with number 0, the cell-cell interactions in PPM are: 0–0, 0–1, 0–3, 0–4, and 1–3 in two dimensions (cf. Fig. 7.1(c)), and 0–0, 0–1, 0–3, 0–4, 0–9, 0–10, 0–12, 0–13, 1–3, 1–9, 1–12, 3–9, 3–10, and 4–9 in three dimensions.

The difference in computational time between symmetric and non-symmetric PP interactions is assessed using a PSE diffusion solver according to Subsection 5.2.2. The elapsed time per time step is found to decrease by a factor of 1.72 when changing from asymmetric to symmetric interactions on two processors. Due to the additional overhead caused by sending back the ghost contributions, this factor is below 2.

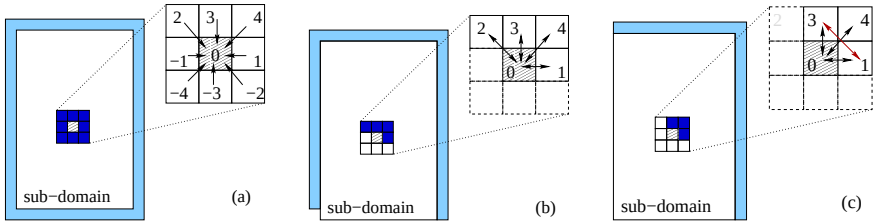


Figure 7.1: Cell-cell interactions and ghost-layer arrangement. (a) For non-symmetric particle-particle interactions, the ghost layer (light gray) extends all around the sub-domain. Interactions are one-sided. (b) In traditional symmetric cell list algorithms, ghost layers are required on all but one boundary of the domain. (c) In PPM*, diagonal interactions are introduced (1–3). Ghost layers are now symmetric and do not overlap with any other ghost layers of neighboring sub-domains. This results in less communication, better scaling in memory, and simpler algorithms, e.g. when considering connected particles. The two-dimensional case is depicted. See text for interactions in the three-dimensional case.*

Verlet list interactions

For spherically symmetric interactions, cell lists contain up to $27/(4\pi/3) = 81/(4\pi) \approx 6$ times more particles than actually needed. *Verlet lists* [301] are provided to reduce this overhead. For each particle they involve an explicit list of all other particles it has to interact with. The radius of the Verlet sphere is usually chosen to be the interaction cut-off plus a certain safety margin, called *skin*. The lists need to be rebuilt as soon as any particle has moved farther than this safety margin. In three dimensions, interactions using Verlet lists are at most $81/\left((4\pi)\cdot(1+\text{skin})^3\right)$ times faster than cell list interactions. In PPM, Verlet lists are set up using intermediate cell lists to reduce the algorithmic complexity to $\mathcal{O}(N)$. Routines are provided for generating Verlet lists as well as for computing the interactions based on them. Memory requirements however usually limit the application of Verlet lists to small or well-distributed problems.

Connection interactions

Besides free-space PP interactions, PPM also supports interactions based on inter-particle connections. No neighbor lists are required in this case since a connection consists of an explicit list of all its member particles. Since the connections are mapped according to the particle distribution as described in Subsection 7.2.3,

the *connection interactions* can be evaluated without communication. The global particle index specified by the connection is hereby translated to the local particle index by a direct look-up in a global list.

7.2.5 Particle-Mesh and Mesh-Particle interpolations

All hybrid PM methods involve interpolation of irregularly distributed *particle quantities* from particle locations onto a regular mesh, and interpolation of *field quantities* from the grid points onto particle locations.

These interpolations are utilized for two purposes, namely:

- the communication of the particle solver with the field solver, and
- the reinitialization of distorted particle locations.

While the first issue is a well-established notion in PM techniques, the *reinitialization* of particle locations and strengths when particle locations get distorted is a critical, albeit often overlooked, aspect of particle methods for the simulation of continuum systems [160]. Particle overlap is needed in order to ensure convergence of the method (cf. Section 5.1.1), and it is achieved by periodically interpolating the particles onto a regular mesh and replacing the current set of particles by new particles, created at the locations of the mesh points. This procedure is referred to as *remeshing*.

The PPM library provides routines that perform these operations. The *interpolation weights* $W(\boldsymbol{x}_m - \boldsymbol{x}_p)$ can be pre-computed and stored to facilitate adjustments of the interpolation, or to interpolate several sets of quantities. Adjustments are e.g. needed in the vicinity of solid boundaries or immersed interfaces. If the weights are not pre-computed, they are determined during the actual interpolation. The used d-dimensional interpolants are tensor products of one-dimensional interpolation kernels, such as B-Splines and extrapolations of B-Splines [194] with m points in their support. The amount of memory required per particle therefore is $\mathcal{O}(dm)$ instead of $\mathcal{O}(m^d)$ for the general case. Currently implemented interpolants include first and second order B-Splines, and the M_4' function [194].

The interpolation of mesh values onto particles readily vectorizes: the interpolation is performed by looping over the particles and receiving values from mesh points that lie within the support of the interpolation kernel. Therefore, the values of individual particles can be interpolated independently.

The interpolation of particle values onto the mesh, however, leads to *data dependencies* as the interpolation is still performed by looping over particles, but a mesh point may receive values from more than one particle. To circumvent this

Scheme	CPU time	vector operation ratio	avg. vector length
colored	2.69 s	99%	230.6 words
classical	30.1s	0.36%	4.1 words

Table 7.2: Comparison of the vector performance of classical particle-to-mesh interpolation and the present coloring scheme on a NEC SX-5 vector computer. 2 million particles are interpolated onto $128 \times 128 \times 128$ *regularly spaced mesh points.*

problem, the PPM library implements the following technique [307]: when new particles are created in the course of remeshing, we assign colors to the particles such that no two particles within the support of the interpolation kernel have the same color. Particle-to-mesh interpolation then visits the particles ordered by color to achieve *data independence*. This *coloring scheme* enables *vectorization* of particle-to-mesh interpolations, as confirmed by a test on a NEC SX-5 vector computer. The results are summarized in Table 7.2. Without the present coloring scheme, interpolation in hybrid PM methods would be prohibitively expensive on vector architectures.

7.2.6 Mesh-based solvers

In PPM, meshes can be used to solve the field equations associated with *long-range particle interactions* [138], or to discretize the differential operators in the governing equations of the simulated physical system. These operators are often local and their computational cost scales linearly with the number of mesh points.

A large class of pair interaction potentials in particle methods can be described by the Poisson equation as it appears in MD of charged particles via electrostatics (Coulomb potential), fluid mechanics in stream-function/vorticity formulation (Biot-Savart potential), and astrophysics (gravitational potential). The *Poisson equation* is expressed as:

$$\nabla^2 f(\boldsymbol{x}) = g(\boldsymbol{x})\,. \tag{7.1}$$

The PPM library provides fast *Poisson solvers* based on FFTs and geometric Multi-Grid (MG).

Fast Fourier Transforms

PPM provides an *FFT-based solver* for the Poisson Eq. (7.1) with periodic boundary conditions. A multi-dimensional FFT is parallelized using a sequence of one- or two-dimensional FFTs, that are performed on pencil and slab topologies as introduced in Subsection 7.2.2. The data array to be transformed is optimally stored if the transformation operates along the leading dimension of the array, thus ensuring unit memory stride. Therefore, the data are transposed if necessary before performing the individual low-dimensional FFTs. A complete three-dimensional FFT thus consists of mapping the data onto a temporary xy-slab topology, performing a two-dimensional FFT, mapping onto a temporary z-pencil topology, and performing a one-dimensional FFT. The run time of the mapping and the transposition steps strongly depends on the machine's network speed as it involves global communication of a large amount of data.

The actual serial low-dimensional FFTs are performed using the external libraries fftw or MathKeisan (on NEC SX vector architectures).

Multigrid methods

The geometric MG method is implemented in PPM as a fast iterative solver for the Poisson Eq. (7.1). The advantage of parallel *MG solvers* consists in restricting communication to the ghost layers, whereas the corresponding FFTs require several global mappings. This advantage is particularly significant on distributed memory machines, where the bandwidth of the network connection may become the performance-limiting factor in FFT solvers.

The PPM MG supports both the V and W cycle [290]. The Laplacian is discretized using five and seven point stencils in two and three dimensions, respectively. As residual smoother we employ the red-black *successive over-relaxation* scheme, which includes the *Gauss-Seidel* smoother as a special case. The optimal value for the over-relaxation parameter is approximately 1.15 [325]. Furthermore, the full-weighting scheme [290] is used for the *restriction* of the residual, and bilinear (in two dimensions) or tri-linear (in three dimensions) interpolation for the *prolongation* of the function corrections [290].

The PPM MG solves the two and three dimensional Poisson equation for scalar and vector fields. Boundary conditions can be periodic, Neumann, or Dirichlet. In the vector case, each component may satisfy different boundary conditions. The solver is structured into initialization, computation, and finalization to ensure efficient memory-management.

7.2.7 ODE Solvers

Simulations using particle methods entail the solution of systems of ODEs as outlined in Section 5.1. The characteristics of the *Initial Value Problems* (IVP) represented by these ODEs explicitly reflect the physics of the system that is being simulated.

The PPM library provides a set of explicit integration schemes to solve these IVPs. The *ODE solver* of PPM is designed as a "black-box" solver. The user selects the method to be used and provides as a function pointer a routine that computes the right-hand sides of the ODEs. Allocation of storage (for the stages of multi-step schemes) and the actual computation of the stages is performed by the library. Second order ODEs are solved by transforming them into a system of first order problems, and parallelism is achieved by mapping the integrator stages along with the other particle quantities. At the last stage of the integrator, the previous stages are available on the processor that currently hosts the particles, and the final particle update is completed without further communication. *Low-storage schemes* have the additional advantage of requiring little communication.

The set of available integrators currently includes forward Euler with and without super time stepping [9], 2-stage and 4-stage standard Runge-Kutta schemes, Williamson's low-storage third order Runge-Kutta scheme [318], and 2-stage and 3-stage TVD Runge-Kutta schemes [261].

7.2.8 Parallel I/O

File I/O in distributed parallel environments exist in two different modes: *distributed* and *centralized*. By distributed we denote the situation where each processor writes its part of the data to its local file system. Centralized I/O on the other hand produces a single file on one of the nodes, where the data contributions from all processors are consolidated. The latter is convenient for small or aggregated data, and for writing files that are later read on a different number of processors, e.g. to continue an interrupted simulation.

The PPM library provides a parallel I/O module which supports both binary and ASCII read and write operations in both modes, distributed and centralized. The *I/O mode* is transparent to the client application. Write operations in the centralized mode can concatenate or reduce (sum, replace) the data from individual processors; read operations can transparently split the data in equal chunks among processors, or send an identical copy to each one. The basic assumption behind the *split mode* is that no processor is able to hold all the data in memory.

Read and write operations are performed by the same routine. The actual operation is determined by an input flag that can be set to either `ppm_param_io_read` or `ppm_param_io_write`. This interface facilitates writing and reading simulation data files in consistent formats and order.

To improve performance of the centralized mode, network communication and file I/O are overlapped in time using *non-blocking message passing*. All standard data types are supported: real, double, single precision complex, double precision complex, integer, logical, and character strings.

7.2.9 Adaptive trees

A general *tree construction* is provided for both internal and client use. It supports non-adaptive and adaptive *binary trees*, *quad-trees*, and *oct-trees*. At any stage of the tree, the space is subdivided into M boxes $\{B_k\}$. The indices i and j are used to denote coordinate directions. Adaptivity and subdivision behavior are guided by two *cost functions* ϕ_1, ϕ_2. Both cost functions are linear combinations of the three cost contributions: particle costs c_p (user-specified or unity per particle), mesh points (number of mesh points in the box $m_B = \prod m_{B,i}$), and geometry (volume of the box $|B| = \prod |B|_i$), with user-provided coefficients α, β, γ:

$$\phi_{\{1,2\}}(B_k) = \alpha_{\{1,2\}} \sum_{p \in B_k} c_p + \beta_{\{1,2\}} m_{B_k} + \gamma_{\{1,2\}} |B_k| . \tag{7.2}$$

The first cost function ϕ_1 guides the *adaptivity* of the tree since the next subdivision is applied to the box B_K of largest ϕ_1. The second cost function ϕ_2 defines the direction(s) of subdivision and the position(s) of the subdivision plane(s). Suppose B_K is to be subdivided next. In order to create the minimum ϕ_2-cut when subdividing the box, the *tensor of inertia* $\boldsymbol{T} = (T_{ij})$ is computed from the particle locations $\boldsymbol{x}_p = (x_{p,i})$ and costs c_p as:

$$T_{ii} = \sum_{p \in B_K} \left(\sum_{j \neq i} x_{p,j}^2 \right) c_p \,, \qquad T_{ij} = \sum_{p \in B_K} x_{p,i} x_{p,j} c_p \,. \tag{7.3}$$

The *eigenvectors* $\boldsymbol{v}_r$ of $\boldsymbol{T}$ are scaled with the corresponding *eigenvalue* λ_r: $\boldsymbol{\nu}_r = \lambda_r (\boldsymbol{v}_r / \|\boldsymbol{v}_r\|_2)$ and projected onto the unit coordinate vectors $\boldsymbol{e}_i$. The number of mesh points in this direction ($m_{B,i}$) and the length of the box in this direction ($|B|_i$) are normalized and added to form a *score value* S for each coordinate

direction:

$$S(\boldsymbol{e}_i) = \alpha_2 \sum_r \boldsymbol{e}_i \cdot \boldsymbol{\nu}_r + \beta_2 \left(\sum_j m_{B_K,j} \right)^{-1} m_{B_K,i} + \gamma_2 \left(\sum_j |B_K|_j \right)^{-1} |B_K|_i \,. \tag{7.4}$$

The spatial subdivision directions are chosen in order of ascending score. The client program can however specifically disallow subdivisions in certain coordinate directions to enforce pencil-type or slab-type boxes. The actual position of a cut perpendicular to direction I is determined as the corresponding component of the *center of mass* of ϕ_2 within the box B_K:

$$\phi_2(B_K)^{-1} \left[\alpha_2 \sum_{p \in B_K} x_{p,I} c_p + \mu_I(B_K) \left(\beta_2 m_{B_K} + \gamma_2 |B_K| \right) \right] , \tag{7.5}$$

subject to the constraint that a client-specified minimum box size is not under-run. In the above equation, $\boldsymbol{\mu}(B_k) = (\mu_i(B_k))$ denotes the geometric center of box B_k. To terminate the tree, multiple concurrent stopping criteria can be prescribed.

7.3 Parallel efficiency and benchmark results

The parallel efficiency of the PPM library is measured based on the following five tests:

1. solving the Poisson equation using FFTs,
2. solving the same equation using geometric MG,
3. simulating diffusion using PSE in the endoplasmic reticulum of live cells,
4. simulating a perturbed double shear layer using remeshed Smooth Particle Hydrodynamics (rSPH) for compressible flows, and
5. simulating the same problem using vortex methods for incompressible flows.

Performance is tested for both a *fixed-size* and a *scaled-size problem* for all cases except the diffusion simulation. In the fixed-size problems, the number of mesh points and particles is kept constant, i.e. the work load per processor decreases with increasing number of processors. In the scaled problems, mesh point and particle numbers grow proportionally to the number of processors, resulting in a constant work load per processor.

Timings and parallel efficiency figures are collected on the *IBM p690* computer of the Swiss National Supercomputing Centre (CSCS). The machine consists of 8 Regatta nodes with 32 1.3 GHz Power4 processors per node. Within each node, the machine at CSCS is configured in 4 groups with 8 processors sharing 12 GB of memory. Each processor has a peak performance of 5.2 GFlop/s, and the nodes are connected by a 3-way Colony switch system. Of the total of 256 processors, only up to 242 can be used for computation since the remaining ones are dedicated to file I/O and login. All software is compiled with version 10.1 of the IBM XL Fortran compiler for AIX using the flags `-O5 -qarch=pwr4 -qtune=pwr4 -qunroll=yes -qcache=auto -qhot -qipa=inline -qstrict`. In each test, we measure the elapsed wall-clock time t_{ij} for each time step j on each processor $i = 1, \ldots, N_{\text{proc}}$ using the Fortran `SYSTEM_CLOCK` intrinsic. To account for synchronous communication steps we report the maximum of these times over all processors. This maximum is averaged over 5 to 10 samples to compute the *speedup* S and the *efficiency* e:

$$S(N_{\text{proc}}) = \frac{t(1)}{\text{mean}_j \max_i t_{ij}(N_{\text{proc}})} \cdot \frac{N\left(N_{\text{proc}}\right)}{N\left(1\right)}, \tag{7.6}$$

$$e(N_{\text{proc}}) = \frac{S(N_{\text{proc}})}{N_{\text{proc}}}. \tag{7.7}$$

Hereby, $t(1)$ is the time on a single processor (linearly extrapolated if not measured), $t_{ij}(N_{\text{proc}})$ is the time on N_{proc} processors, $N(1)$ is the problem size on a single processor, and $N(N_{\text{proc}})$ is the problem size on N_{proc} processors. To account for the $\mathcal{O}(N \log N)$ scaling of the FFTs, the second factor of the speedup is accordingly adjusted in the benchmarks of the FFT-based Poisson solver.

Vectorization and parallel efficiency on vector machines is tested using the *NEC SX-5* computer at CSCS. This is a shared memory machine with 16 NEC SX-5 vector processors. Each processor has a peak performance of 8 GFlop/s and contains 64 vector registers of a length of 256 words (2048 bytes) each. The total memory of the machine is 64 GB. Out of the 16 processors, up to 15 are used in the present benchmarks, leaving one processor to the operating system. The software is compiled using the NEC SXF90 compiler, version 2.0, revision 305, with the flags `-R5 -C hopt -f4 -float0`.

Additional benchmarks are performed on a 16 processor distributed memory cluster of 2.2 GHz AMD Opteron 248 nodes running under Linux. The nodes are connected by a switched gigabit ethernet network. On this machine, the Intel Fortran 90 compiler, version 8.1, is used with the flags `-O3 -xW`, and the free MPI implementation mpich 1.2.6.

7.3.1 Parallel FFT-based Poisson solver

We test the performance of the FFT-based Poisson solver by solving the scalar Poisson Eq. (7.1) with the right hand side

$$g(x, y, z) = \sin(2\pi x)\sin(2\pi y)\sin(2\pi z), \quad x, y, z \in [0, 1]\,, \tag{7.8}$$

subject to periodic boundary conditions. All Fourier transforms are performed using the parallel FFT routines of the PPM library as described in Section 7.2.6.

The parallel speedup and efficiency for the scaled problem as shown in Fig. 7.2 exhibit two characteristic regions. The first one ranges from 1 to 8 processors, the second one from 8 and beyond. From 1 to 8 processors the efficiency drops significantly, due to conflicts and congestion in the *shared memory* architecture within each compute node. As the size of the problem and the number of processors exceed the shared memory, the loss of efficiency is markedly reduced. This is verified in a separate benchmark (Fig. 7.2: $\times$), in which only one processor per node is used. In this case, the congestion is removed and the efficiency significantly improves to 68% on 16 processors. Solving the Poisson equation to machine precision on a $128 \times 128 \times 128$ mesh takes 0.6 seconds on a single processor. The corresponding scaled system on 64 processors ($512 \times 512 \times 512$) requires 2.4 seconds.

Speedup and efficiency for the fixed-size problem are shown in Fig. 7.3. The FFT Poisson solver shows an efficiency above 30% (average 50%) on 128 processors using a $512 \times 512 \times 512$ mesh. Again, the scaling improves beyond 8 processors, similar to the scaled case. The solution time for a $512 \times 512 \times 512$ mesh is 2.4 seconds on 64 processors, and 1.6 seconds on 128 processors.

7.3.2 Parallel multigrid Poisson solver

We test the parallel performance of the PPM MG Poisson solver and compare it to the FFT-based solver using the same scalar Poisson Eq. (7.1) with the same right hand side Eq. (7.8), subject to periodic boundary conditions. The initial value of

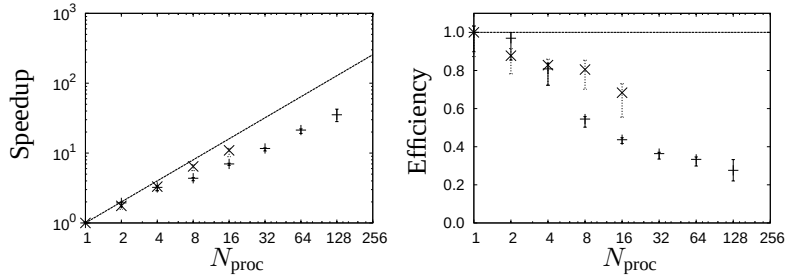


Figure 7.2: Parallel speedup and efficiency of the FFT-based Poisson solver for the scaled-size problem starting with $128 \times 128 \times 128$ mesh points on one processor ($+$). Using only one processor per node, the bottleneck of the shared memory is removed ($\times$). Each point is averaged from 5 samples, error bars indicate the min-max span. All timings are performed on the IBM p690.

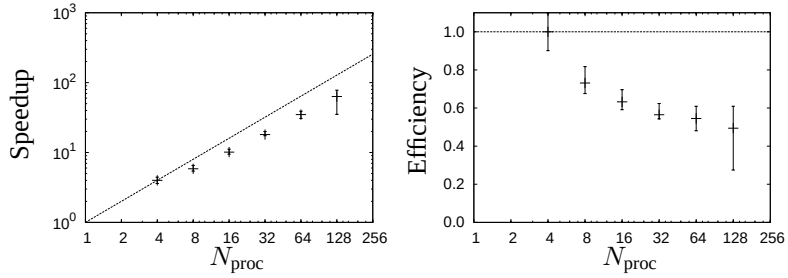


Figure 7.3: Parallel speedup and efficiency of the FFT-based Poisson solver for the fixed-size problem with $512 \times 512 \times 512$ mesh points on 4 to 128 processors. Each point is averaged from 5 samples, error bars indicate the min-max span. All timings are performed on the IBM p690.

f is zero everywhere and we use the V(2,1) cycle with one smoothing step at the finest level.

We conduct three tests. The first one involves the fixed case with $256 \times 256 \times 256$ mesh points, while the two others are scaled cases, one starting from a $128 \times 128 \times 128$ mesh, the other one starting from $256 \times 256 \times 256$. Efficiency and speedup for the scaled cases are shown in Fig. 7.4 and for the fixed case in Fig. 7.5. Again we observe a strong decrease in the parallel efficiency up to 8 processors due to the congestion of the shared memory. This is removed when using only one processor per node in a pure *distributed memory* setup, cf. Figs. 7.4 and 7.5, and the efficiency improves to 90% on 16 processors for the scaled case. The effective efficiency based on the timing obtained on 8 processors is 92% for the large scaled case on 64 processors, and the efficiency for a $1024 \times 1024 \times 1024$ system is 66% for the large scaled case on 64 processors. For the latter system, the elapsed wall-clock time is 10.5 seconds per V-cycle, and thus 42 seconds for the four V-cycles needed to reduce the L_2 error to 10^{-4}. A system with half a billion unknowns is solved in the small scaled case on 242 processors at 48% efficiency in 1.7 seconds per V-cycle. This compares well to the 41% efficiency achieved by the Prometheus multigrid library [5] on 128 IBM Power3 processors for the same problem size. Moreover, the present solver sustains 15% of the machine's peak performance, whereas Prometheus sustained 7.5% [5].

The vectorization of the PPM MG solver is tested on the NEC SX-5 using up to 8 processors. The PPM MG sustains a performance of 2.4 GFlop/s per processor (30% of peak performance) with a vector operation ratio of 95% and a parallel efficiency of 96%. On this machine, a single V cycle on a $512 \times 512 \times 512$ system takes 1.21 seconds on 8 processors.

7.3.3 Particle strength exchange in complex geometries

We use a client application for the simulation of three-dimensional diffusion using the PSE method as outlined in Subsection 5.2.2. This test demonstrates the capability of the library in handling complex-shaped domains, and it helps to assess load balance and inter-processor communication for an irregular domain decomposition. Since the particles in this simulation are stationary, only ghost values need to be communicated and the Verlet lists are only created once.

We simulate isotropic homogeneous diffusion in the lumen of the ER of a live cell using the methods introduced in Chapter 6. All ODEs are integrated in time using the explicit Euler scheme, and we use the second-order accurate isotropic PSE kernel given in Eq. (F.4).

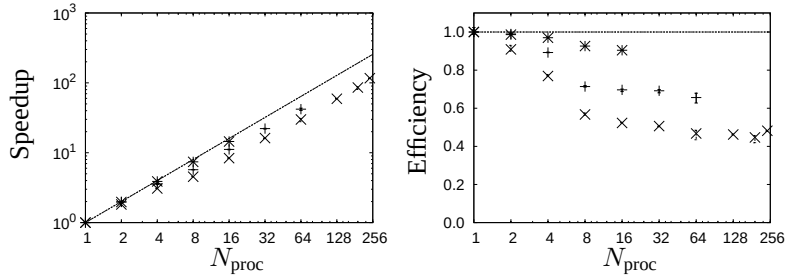


Figure 7.4: Parallel speedup and efficiency of the MG Poisson solver for the scaled-size problems. The initial mesh resolutions on one processor are $256 \times 256 \times 256$ $(+)$ and $128 \times 128 \times 128$ $(\times)$, respectively. Using only one processor per node in the large case, the bottleneck of the shared memory is removed $(*)$. Each point is averaged from 5 samples, error bars indicate the min-max span. All timings are performed on the IBM p690.

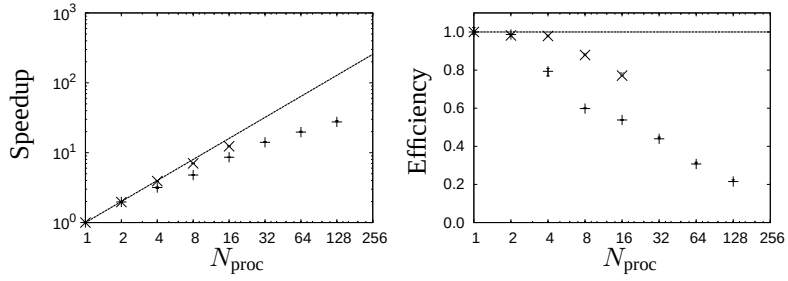


Figure 7.5: Parallel speedup and efficiency of the MG Poisson solver for the fixed-size problem with $256 \times 256 \times 256$ mesh points on 2 to 128 processors $(+)$. Using only one processor per node, the bottleneck of the shared memory is removed $(\times)$. Each point is averaged from 5 samples, error bars indicate the min-max span. All timings are performed on the IBM p690.

Parallel speedup, timing, and efficiency

The problem size is fixed at 3.4 million particles, uniformly distributed inside the ER geometry. The simulation uses double-precision floating point arithmetics and Verlet lists with a cut-off of 2ϵ. Each particle thus interacts with 32 neighbors. Domain decomposition is done using adaptive ROB (cf. Subsection 7.2.2) with the z direction fixed as the organelle is very thin in the z direction. Fig. 7.6 shows the resulting decomposition into 9311 sub-domains. Since the particles do not move in a PSE simulation, empty sub-domains are discarded. The elongated domains at the periphery are a consequence of the ROB domain decomposition.

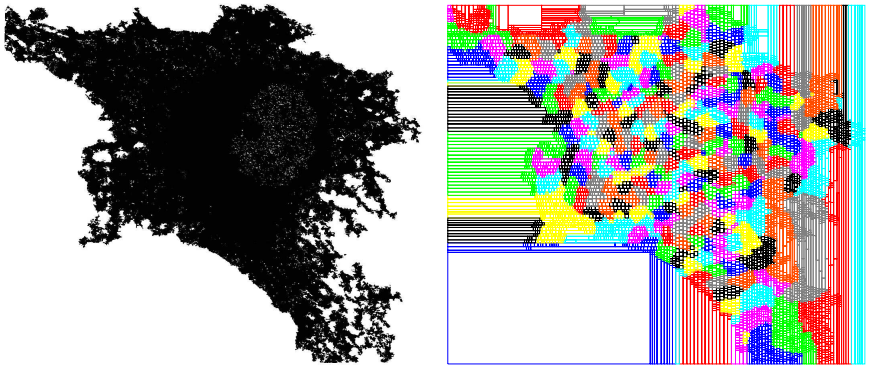
The parallel performance is tested on 4 to 242 processors. Fig. 7.7 summarizes the results. The simulation sustains 20% of the peak performance on the IBM p690, thus reaching a total of 250 GFlop/s on 242 processors at 84% efficiency. The observed decrease in parallel efficiency to 84% on 242 processors can be explained by arising load imbalance when distributing a constant number of sub-domains onto an increasing number of processors. The *load balance* is quantified by

$$\frac{\text{mean}_j \min_i t_{ij}\,(N_{\text{proc}})}{\text{mean}_j \max_i t_{ij}\,(N_{\text{proc}})}. \tag{7.9}$$

If we use the actual number of interactions of each particle as that particle's computational cost for the topology creation, we observe values in the range of 90% to 95%. Using an assumed unit cost per particle, the load balance is on the order of 10% to 60%, depending on the actual number of processors used. Using cell lists, one time step on 4 processors takes 67 seconds instead of the 13.5 seconds for the simulation using Verlet lists. This difference is expected since each particle interacts with 216 neighbors in the cell list case.

On the NEC SX-5 vector computer, more than 99% of the loops in non-initialization routines vectorize and the average vector length is > 254 words. The parallel efficiency is 88% on 8 processors and 86% on 15 processors. One time step in the latter case takes 1.15 seconds. 2.64 GFlop/s are sustained on each of the 15 processors thus reaching 33% of the machine's peak vector performance. Again using the actual number of interactions, given by the length of the Verlet list, as the computational cost for each particle, the load balance exceeds 80% on up to 15 processors.

The largest simulation performed using this PPM client uses 1 billion particles. The computation is based on cell lists and a cut-off of 1ϵ, i.e. each particle interacts with 26 neighbors. The simulation is performed on 64 processors of the IBM



(a) (b)

Figure 7.6: (a) Top view of the computational domain used for the present PSE test case. (b) The resulting PPM domain decomposition on 242 processors using recursive orthogonal bisection in the x and y directions (z direction fixed). Rectangles show the 9311 sub-domains, shading codes processor affiliation. The elongated peripheral domains are a consequence of the recursive orthogonal bisection decomposition.

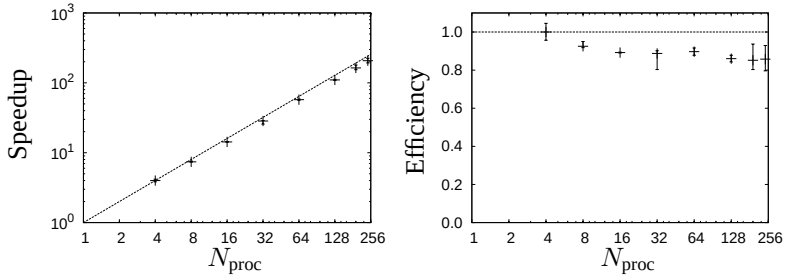


Figure 7.7: Parallel speedup and efficiency of the PPM PSE client for the fixed-size problem with 3.4 million particles on 4 to 242 processors. Each point is averaged from 10 samples, error bars indicate the min-max span. All timings are performed on the IBM p690.

p690, uses 1.4 GB of memory per processor, takes 54 seconds per time step, and sustains 20% of the peak performance. Extrapolating from the previous runs, 50 to 60 seconds per time step are expected for this large a simulation. The measured 54 seconds fall within this range, confirming linear scaling to large numbers of particles.

7.3.4 Three-dimensional remeshed smooth particle hydrodynamics

We test a PPM client application that implements a novel, computationally efficient formulation of the *remeshed SPH* [50, 248]. This rSPH client was implemented by Simone Hieber, and in the present benchmark it is applied to the simulation of a three-dimensional compressible double shear layer [109]. In order to measure the parallel performance, we consider a computational domain fully populated with particles, so that the reported performance measures are independent of the particular flow problem.

The rSPH client solves the three-dimensional *Navier-Stokes equations* for viscous compressible isothermal flow in non-dimensional Lagrangian form, expressed as:

$$\frac{Du}{Dt} = -u\nabla \cdot \boldsymbol{v}\,, \tag{7.10}$$

and

$$u\frac{D\boldsymbol{v}}{Dt} = -\frac{1}{\mathrm{Ma}^2\gamma}\nabla p + \frac{1}{\mathrm{Re}}\nabla \cdot \boldsymbol{\tau}\,, \tag{7.11}$$

where the pressure p is given by

$$p = Tu\,, \tag{7.12}$$

and the components of the stress tensor $\boldsymbol{\tau}$ are:

$$\tau_{ij} = \frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\partial v_k}{\partial x_k}\,. \tag{7.13}$$

Hereby, δ_{ij} is the *Kronecker delta* symbol, Re is the *Reynolds number*, and T the *temperature*, normalized by the characteristic temperature T_0 of the flow. It is set to $T = 1$ for all simulations (iso-thermal fluid). The *Mach number* is defined as $\mathrm{Ma} = \frac{v_0}{\sqrt{\gamma R T_0}}$, where v_0 denotes the magnitude of the characteristic velocity, γ the

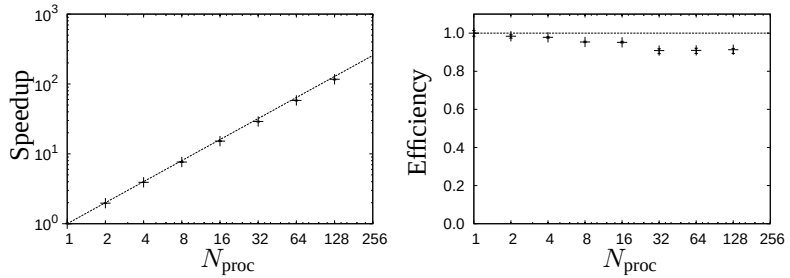


Figure 7.8: Parallel speedup and efficiency of the PPM *rSPH client for the scaled-size problem starting with 2 million particles on one processor. Each point is averaged from 5 samples, error bars indicate the min-max span. All timings are performed on the IBM p690.*

ratio of specific heats, and R the gas constant. The density u is normalized by the mean density $\overline{u}$.

The governing Eqs. (7.10) and (7.11) are discretized using the rSPH approach [50] with the particles being remeshed using the M_4' kernel function [194] after each time step. Time integration is done with a second-order Runge-Kutta scheme.

Parallel speedup, timing, and efficiency

The speedup and parallel efficiency of the rSPH client are shown in Figs. 7.8 and 7.9 for the scaled and fixed-size problem, respectively. The largest simulation considered in this test case comprises 268 million particles and achieves a parallel efficiency of 91% on 128 processors. The efficiency on 32 processors using 67 million particles is also 91%, which compares well to the 85% efficiency of the *GADGET* SPH code by Springel *et al.* [270] on 32 processors of the same computer model (IBM p690). The efficiency in the fixed-size problem ranges between 100% and 84%. One time step of the simulation using 16.8 million particles takes 196.9 seconds on 4 processors and 7.3 seconds on 128 processors.

The communication overhead, assessed using a fixed-size problem with 16 million particles, is shown in Table 7.3. The fraction of time spent in communication is less than 13% of the total computational time in all cases. Using 4 processors, only 5% of the total time are spent in communication. The communication effort increases by a factor of 2.5 when using 64 times more processors. This demonstrates the high efficiency of the communication routines in the PPM library.

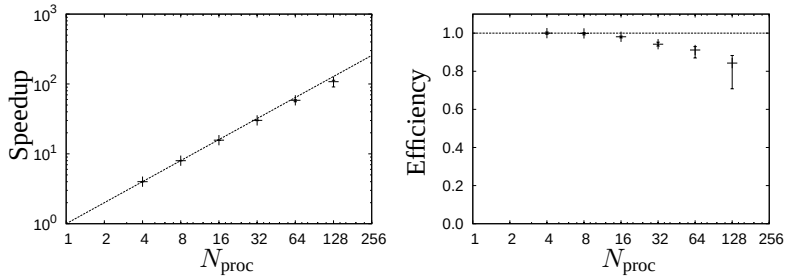


Figure 7.9: Parallel speedup and efficiency of the PPM *rSPH client for the fixed-size problem with 16.8 million particles on 4 to 128 processors. Each point is averaged from 5 samples, error bars indicate the min-max span. All timings are performed on the IBM p690.*

N_{proc}	total time [s]	communication [s]	ratio
4	195	10	5%
16	50	4	8%
64	14	1.2	11%
128	7	0.8	12%

Table 7.3: Communication-to-computation ratio of the PPM *rSPH client solving a fixed-size problem with 16 million particles on the IBM p690.*

7.3.5 Three-dimensional vortex methods

The final test involves simulations using a three-dimensional particle *vortex method* [63]. The client application was implemented by Michael Bergdorf and it demonstrates a large number of the PPM library modules, involving particle convection and diffusion, particle-to-mesh and mesh-to-particle interpolation, particle reinitialization, and the solution of Poisson equations on the mesh.

Hybrid VM [59, 307, 65] solve the incompressible *Navier-Stokes equations* in the Lagrangian vorticity-velocity formulation:

$$\frac{D\boldsymbol{\omega}}{Dt} = (\boldsymbol{\omega}\cdot\nabla)\boldsymbol{v} + \nu\nabla^2\boldsymbol{\omega}, \tag{7.14}$$

and

$$\nabla^2\boldsymbol{\Psi} = -\boldsymbol{\omega}, \tag{7.15}$$

where $\boldsymbol{v} = \nabla\times\boldsymbol{\Psi}$ is the velocity and ν is the viscosity of the fluid. The *vorticity field* $\boldsymbol{\omega}(\boldsymbol{x})$ is discretized using particles that carry *circulation* $\Gamma_p = \boldsymbol{\omega}(\boldsymbol{x}_p)V_p$, and that are convected by the local flow velocity field $\boldsymbol{v}(\boldsymbol{x})$ [63]. The vorticity of the particles is interpolated onto a mesh where it is used as the right-hand side of the vector Poisson Eq. (7.15), which is solved for the *stream function* $\boldsymbol{\Psi}$ using the PPM MG Poisson solver presented in Subsection 7.2.6. Velocities are computed from the stream function using second order finite differences, and the vorticity diffusion and stretching are evaluated at mesh point locations also employing second order finite differences. The time step is completed by interpolating the grid functions $\frac{D\boldsymbol{\omega}}{Dt}$ and $\boldsymbol{v}$ back onto particle locations. Distortion of the particle locations leads to spurious vorticity structures and the flow ceases to be well represented by the particles. Therefore, particles are remeshed onto regular positions after each time step using the *remeshing* routines of the PPM library. The M_4' function [194] is used for all interpolation steps.

To study the parallel performance of the vortex client, we consider the same double shear layer [109] as used for the rSPH tests in Subsection 7.3.4. We use the whole computational domain as vorticity support, so that the number of particles is equal to the number of grid points. All simulations include the solution of a convection-diffusion equation for a passive scalar, and they start from the initial condition proposed by Ghoniem and Knio [109] with $\mathrm{Re} = 990$. The *Reynolds number* is computed as

$$\mathrm{Re} = \frac{\|\boldsymbol{v}\|_\infty\,\delta}{\nu}\,, \tag{7.16}$$

where δ denotes the thickness of the shear layer. All ODEs are integrated using the PPM ODE solver with a second order midpoint Runge-Kutta method.

Parallel speedup, timing, and efficiency

The results for the scaled-size and fixed-size cases are depicted in Figs. 7.10 and 7.11, respectively. The largest system comprises 268 million particles distributed onto 128 processors. For this system, one iteration takes 85 seconds on average with a parallel efficiency of 63%. Vectorization of the code is tested on the NEC SX-5 computer using 8 processors. All major loops vectorize, including 99% of the particle-to-mesh and mesh-to-particle interpolations with an average vector length of 231 words, demonstrating the effectiveness of the colored interpolation scheme described in Subsection 7.2.5.

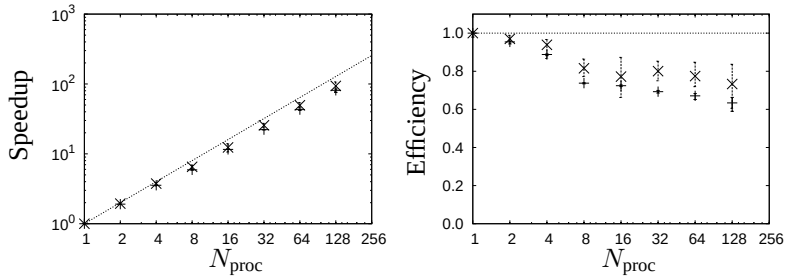


Figure 7.10: Parallel speedup and efficiency of the PPM *VM client (+) and of the particle-mesh interpolation alone (×) for the scaled-size problem. The initial mesh resolution on one processor is* $128 \times 128 \times 128$*. Each point is averaged from 5 samples, error bars indicate the min-max span. All timings are performed on the IBM p690.*

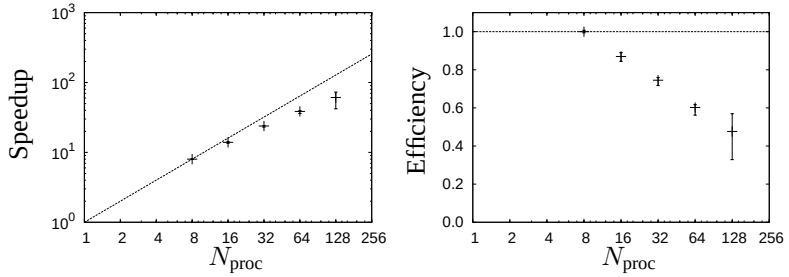


Figure 7.11: Parallel speedup and efficiency of the PPM *VM client for the fixed-size problem with* $256 \times 256 \times 256$ *mesh points on 8 to 128 processors. Each point is averaged from 5 samples, error bars indicate the min-max span. All timings are performed on the IBM p690.*

Interpolating 2 million particles onto a $128 \times 128 \times 128$ mesh takes 3.4 seconds on a single processor of the IBM p690. Interpolating the field back onto the particles takes 1.1 second. Timings for the NEC SX-5 are given in Table 7.2.

Part III

Conclusions and Outlook

Chapter 8

Conclusions

This thesis has addressed a number of issues pertaining to the analysis, modeling, and simulation of diffusion processes in live cells. In particular, we have improved existing single particle tracking techniques to make them suitable for large data sets and fully automated high-throughput assays. Trajectory analysis was presented using novel global and segmentation-based methods, and the fully automatic classification of trajectories using methods from machine learning was investigated. We then considered continuum diffusion processes of abundant particles, where we introduced a novel particle method to simulate reaction-diffusion processes on moving surfaces. Applying geometry-resolving simulations of diffusion in the ER enabled us to determine for the first time molecular diffusion constants from FRAP experiments in complex-shaped organelles, leading to a novel method of FRAP data analysis. The main contributions and conclusions of this thesis are summarized in the following.

8.1 Feature point tracking

In Chapter 1 we have described a computationally efficient and robust method for feature point tracking in quantitative time-resolved studies of particle motion as they appear in several applications in cell biology. The presented method was demonstrated to be of high accuracy and precision even at moderate signal-to-noise ratios, and to provide sub-pixel accuracy in all practical situations. The absence of any intrinsic models regarding the motion of the tracked particles, in combination with its robustness and efficiency, makes the method particularly well suited for biological applications relying on trajectories developed by fluorescence microscopy.

The presented method emphasizes computational efficiency and ease of use. The former goal is motivated by our observation that many available feature point tracking algorithms suffer from poor computational performance or large memory

requirements if long sequences of large images are to be processed [298]. The present implementation is capable of tracking a sequence of 3000 214×214 pixel TIFF images in less than 15 seconds on a 3.06 GHz Intel Pentium 4 desktop computer. The goal of ease of use was approached by minimizing the number of user-set parameters of the algorithm and by providing a user-friendly graphical interface as described in Appendix A.3.

Smoothness assumptions frequently made [55] could be relaxed by reducing the point detection linker to a generic version. In order to recover the robustness of more complicated designs, it was considered an optimization problem to iteratively find the best set of links between R frames such as to minimize a certain cost functional. If available, prior knowledge about the underlying physical processes can be incorporated by suitably choosing this linking cost functional. Extending the linking over several frames allowed to account for temporary occlusion of particles and strong intensity fluctuations, as demonstrated in the quantum dot example in Subsection 4.1.3.

The present tracking algorithm is not intrinsically limited to two dimensions. Its application to three-dimensional data is straightforward, provided such data are available. The following limitations are however present: In the feature point detection, the algorithm is limited to small (compared to background variations) spherical particles or point spread blobs, and the trajectory linking is limited by the specific cost functional one defines. For the cost functional used in this thesis, cf. Eq. (1.14), the limitation is obviously given by the criterion that two equally bright and equally large particles must always be separated by more than the distance they move per frame. Using different cost functionals, this could be relaxed at the expense of other limitations such as a loss of universality due to prior information about the type of motion. Furthermore, we assumed that every detected point corresponds to exactly one particle. The algorithm is thus unable to resolve *particle coalescence* or *division*, and to yield two continuous trajectories if two particles exactly cross in space and time. The non-particle discrimination step is, if used, limited by the assumption that the majority of the detected points corresponds to particles of the desired kind.

The algorithm is presently implemented as a multi-user multi-tier client-server application. In our experience, this implementation is fast and stable even under high load with several concurrent users. The graphical user interface is implemented in Java and runs on various computer platforms.

8.2 Trajectory analysis and classification

In Subsection 2.1.2 we have introduced the MSS analysis as an extension of MSD analysis for biological trajectories. The MSS analysis is valid for more types of motion and provides a second parameter in addition to the diffusion constant. We have shown that the combination of the two parameters enables unambiguous classification of otherwise indistinguishable motion cases [96]. In addition, MSS analysis is more accurate than MSD analysis as was shown by statistical error estimation. This is an important advantage, given the limited lengths of trajectories from biological experiments. Using the pair (ν_2, β) has enabled for the first time systematic studies of the mobility of virus particles prior to internalization.

To increase time resolution in an accuracy-neutral way, trajectories are frequently decomposed into segments of defined patterns. This trajectory segmentation is mostly done by hand, thus introducing human bias and limiting the data throughput. In Section 2.3 we have proposed a method for automatic trajectory segmentation using neural networks. The presented method can be used to render the complete data acquisition and analysis process fully automatic, enabling unbiased high-throughput screens. The segmentation algorithm was for instance used to automatically process the tens of thousands of trajectories from the Adenovirus study presented in Section 4.3. In addition, it enables event-based analyses as described in Section 2.4.

In Section 3.1 we have surveyed different automatic classification algorithms for trajectory data. Using a discrimination capability measure from psycho-physics, we compared their performances among each other as well as to human classification. We found that all methods perform equally well in the case of separable clusters. On non-separable classes of the used data set, GMM performed best, closely followed by SVM. Due to their dynamic character, HMM are the least robust method with the performance of cHMM and dHMM being comparable. Surprisingly, the algorithms on average performed better than human classifiers. This could be due to human bias from prior expectations, fatigue effects, or inaccuracy. The best test person performed about equally well as the best machine learning algorithm, indicating that the latter was able to extract and use most of the relevant information from the data.

The tests have also shown that the representation of the trajectories in data space critically determines the classification performance. Finding a good data representation serves a double purpose: first, it enables accurate classification of the data and, second, it provides important information about the physical properties that distinguish the classes the most. The latter is particularly interesting in biological applications as it contributes to our understanding of the process that created the trajectory. In Section 3.2 we have thus introduced a closed-loop optimizer to adaptively adjust the data encoder such as to maximize classification performance. We have studied the influence of various parameter settings and have tested the algorithm on both synthetic and real biological trajectories. The classification performance that was achieved compares favorably to experience-based manual data encodings.

In summary, Chapters 2 and 3 have demonstrated that automatic, bias free analysis and classification of biological trajectories is possible with near-maximum accuracy, and that machine learning techniques provide a useful tool for estimating the information content and the relevant parameters in a trajectory data set.

8.3 Application to virus motion analysis

The case studies and applications presented in Chapter 4 have demonstrated the utility of the various data analysis and modeling techniques on questions of biological interest. In Section 4.1 we have demonstrated the capability of the feature point tracking program to handle motion of various types – purely random, stationary, fast directed transport – and to reconstruct connected trajectories from intermittent detections of blinking particles. Sub-pixel accuracy was achieved in all cases and the analysis results were in good agreement with published benchmarks and theory.

In Section 4.2 we have reported the main results of a single-particle study on Polyomavirus, done in collaboration with the group of Prof. Helenius [96]. The methods developed in this thesis have hereby enabled the first quantitative study of the earliest steps of virus infection. Using feature point tracking and MSS analysis, four different modes of motion could be identified and unambiguously classified. Perturbation studies and control experiments finally led to the identification of non-trivial trans-membrane interactions, causing actin-mediated confinement of the virus-receptor complex. Moreover, we found that cholesterol depletion leads to decreased mobility of the virus-receptor particles, most likely caused by the formation of stable ordered lipid domains [96].

A second study, done in collaboration with the group of Prof. Greber, considered the motion patterns of human Adenovirus-2 on the plasma membrane of live cells. The automatic data processing techniques and software implementations developed in this thesis have enabled a bias-free high-volume study involving several tens of thousands of experimentally recorded trajectories of virus motion.

As shown in Section 4.3, this enabled highly significant statistical analysis, leading to the conclusion that the secondary integrin receptor influences the mobility of the primary receptor-virus complex. In cells lacking the secondary receptors, mobility was significantly reduced and the residence times in arrest zones were increased. Using event-based analysis, we also found that pass-by events occur super-randomly in wild type cells. This could mean that viruses are transported or biased toward clathrin-coated pits, and that this transport or bias is mediated by integrins.

8.4 Particle method to simulate reaction-diffusion processes on curved moving surfaces

In Section 5.3 we have presented a deterministic adaptive particle method to simulate anisotropic, inhomogeneous diffusion on complex surfaces. The method was demonstrated to be second-order accurate, and it was efficiently parallelized with linear computational cost.

The implicit formulation of the surface as a level set, and its discretization using particles [137], has many advantages over traditional grid-based methods. It is inherently adaptive and can handle moving and deforming surfaces. Furthermore, it allows to use the embedding approach by Bertalmio *et al.* [32] with the usual space discretization schemes, as the metric of the surface is incorporated into the projected diffusion operator.

In Section 5.4 the method was successfully extended to simulating reaction-diffusion processes on moving and deforming surfaces, where the chemical reactions can be treated either deterministically or stochastically. By means of the multi-resolution AGM method by Bergdorf *et al.* [27], particles with locally adapted sizes were used.

8.5 Simulations of diffusion in the ER and FRAP data analysis

In Chapter 6 we have applied the simulation techniques and software implementations to the diffusion of molecules in the lumen and on the membrane of the ER, an organelle of highly convoluted and complex three-dimensional shape. Present methods to derive molecular diffusion constants from FRAP data do however not account for this shape.

Using section images from confocal microscopy we reconstructed the shapes of various ER samples in three dimensions. These reconstructed geometries enabled us to estimate the fractal dimensions of the ER in Section 6.3, allowing to predict the influence of the geometry on the apparent fluorescence recovery. We have shown that the ER is a fractal shape at length scales relevant to FRAP, and that diffusion is thus expected to appear anomalous at these length scales, even if the underlying molecular diffusion is normal. This is in agreement with the simulations described by Ölveczky and Verkman [207] and the experimental results reported by Weiss *et al.* [314, 313]. The observed anomaly is a direct effect of the complexity of the ER geometry and needs not be connected to any molecular events. For membrane-bound molecules, diffusion appears anisotropic if the membrane has different curvatures in different directions [7]. Again, this is a purely geometric effect.

From the point of view of geometry, previous methods of determining diffusion constants from FRAP are only valid when applied to relatively flat surfaces (compared to the size of the bleached region) or compartments that completely fill the bleached volume. If these conditions are not met, uncorrected diffusion constants must be interpreted with caution. The method presented in this thesis allowed for the first time to assess and validate current methods of FRAP analysis, and to directly obtain corrected molecular diffusion constants in the specific organelle geometry at hand.

Using simulations of diffusion in the lumen of reconstructed ER shapes and on their membranes allowed us to quantify the geometry-induced variations in FRAP experiments. We found that FRAP models that do not account for the specific geometry of the individual ER suffer from an uncertainty of at least 250%. Altogether neglecting the fact that the ER does not completely fill the bleached volume results in errors around 400%. Considering that the mass of a particle scales with the third power of the diffusion constant [174], these errors are quite significant. For membrane-bound molecules the situation is even worse as their radius depends exponentially on the diffusion constant [234]. When monitoring diffusion in cellular organelles with boundaries of complex shape, FRAP analysis thus requires proper geometry correction. This correction has to involve information about the shape of the organelle in the vicinity of the bleached region of interest.

Using the present simulation algorithms together with the reconstructed 3D geometries of the ER led to a novel method of FRAP analysis that fully takes the geometry into account. This enables us to determine the corrected value of the molecular diffusion constant from FRAP experiments, both for soluble luminal molecules and for membrane-bound molecules. Since the computational cost and the applicability of the simulation algorithms do not depend on the complexity of the shape, they are also well suited for treating organelles or intracellular structures

other than the ER.

The main advantage of our approach is that it does not need a model geometry or a statistical transport model. This minimizes the number of assumptions and enables uncertainties below 250%. By construction, our method is not hampered by 3D effects, and no assumptions about the connection density of the ER need to be made.

Our approach is however limited by the resolution of light microscopy and, for membrane simulations, by the computational resolution limit imposed by the narrow-band level set formulation. The latter limitation is addressed by using multi-resolution particle methods developed for convection-diffusion equations [27]. The microscopy resolution limit implies that sufficiently detailed ER geometries can only be obtained in peripheral regions of the cell, where the ER is relatively sparse. The bleached ROI of any FRAP experiment to be evaluated must be located in such well-resolved areas of the organelle. The specific shape of the organelle far away from the ROI is of no importance. A further limitation of the present method is that it can not be applied to organelles that move or deform inside the ROI during the time of the FRAP experiment, or between recording the z-stack and performing the FRAP experiment. This is mainly an experimental limitation as the computational method could readily handle moving surfaces, as was shown in Subsection 5.4.3.

8.6 PPM – an efficient universal software framework for hybrid particle-mesh simulations on parallel computers

The lack of efficiently parallelized and easy to use software libraries has so far hindered the wide-spread use of particle methods. We have thus initiated the development of a generic software framework for hybrid particle-mesh simulations. The PPM library described in Chapter 7 provides a complete infrastructure for parallel particle simulations, including adaptive domain decompositions, load balancing, optimized communication scheduling, parallel file I/O, interpolation, data communication, and a set of commonly used numerical solvers. The main features include stacked mappings, completely symmetric evaluations of particle-particle interactions, particle connections, and the concurrent existence of multiple data topologies.

We have demonstrated the library's parallel efficiency and versatility on a number of different physical problems, on various computer architectures, and on up to 242 processors. All applications showed parallel efficiencies reaching or exceeding the present state of the art, and favorable run-times on large systems. By virtue of a coloring scheme for data interpolation, the PPM library also showed excellent vectorization as tested on the NEC SX-5 vector computer.

Based on the PPM library, we have presented a PSE simulation using 1 billion particles on 64 processors, a VM simulation using 268 million particles – to our knowledge the largest VM done so far –, an SPH simulation exceeding the parallel efficiency of the currently fastest domain-specific code, simulations sustaining up to 33% of the machine's peak performance, and a multigrid Poisson routine solving for half a billion unknowns in less than 7 seconds on 64 processors.

Chapter 9

Outlook and Future Work

This chapter outlines possible extensions of the present work as well as potential applications in future research.

9.1 Automated feature point tracking

Most biological applications only require two-dimensional tracking since the motion either is two-dimensional (e.g. on the plasma membrane) or is observed using "two-dimensional" microscopy techniques such as TIRF or confocal microscopy. The observation method could however be extended to three dimensions in two different ways: first, the particles can be imaged out of focus and the diameter of the diffraction ring pattern can be used as a measure of depth [268] or, second, different observations can be combined and the depth information computationally extracted, e.g. by combining epifluorescence and TIRF microscopy [273], or by using confocal stacks [229].

Although the algorithm is not limited to two-dimensional tracking, the software implementation currently is. With the imminent availability of time-resolved three-dimensional microscopy data, the extension of the software to 3D tracking will be considered.

The algorithm itself could possibly be extended to tracking the shape outlines of larger and deforming objects. This could for example be done by combining the level set techniques of Section 5.3 with the present tracking algorithm. Several open issues pertaining to the fusion and fission of objects, the robustness against noise, and the computational efficiency of such a procedure however need to be addressed.

9.2 Trajectory analysis and classification

Anomalous diffusion is of central importance in biological systems. This is related to the prevalence of fractal structures in nature [184] and is not restricted to intracellular diffusion processes. It is for example known that motor-mediated transport processes on cytoskeletal networks, as well as cell migration, can be modeled by anomalous diffusion [285]. The MSS analysis presented in this thesis is not limited to a particular physical process or system that generated the trajectories.

The trajectory segmentation, encoding optimization, and automatic classification procedures are also applicable to any type of trajectories. Possible future applications range from the analysis of molecular dynamics simulation trajectories, over the classification of credit card usage trajectories for fraud detection [182], to the analysis and segmentation of trajectories in social and political systems [143].

Possible extensions of the algorithms include the use of different classifiers and optimizers in the self-optimizing encoder of Section 3.2. On the theoretical side, several open questions relating to convergence and stability of such closed-loop systems can be tackled. For the trajectory segmentation method of Section 2.3, the use of more elaborate network structures, higher-dimensional input spaces, and different pattern recognition schemes are worthwhile considering.

9.3 Diffusion on surfaces

The computational method presented in Section 5.3 to simulate diffusion on curved surfaces has many applications beyond biology. Surface processes and surfactant evolution are, e.g., important in image processing [323, 32, 31, 250, 91, 296] or in combustion engineering [133, 134, 217].

The present method can be improved by faster and more robust level set algorithms, higher-order operator discretizations, and multi-level adaptive schemes with better parallel efficiency.

9.4 Direct numerical simulations in real cell geometries

As the resolution of live-cell microscopy, the flexibility of software, and the speed of computers increase, the direct numerical simulation methods presented in this thesis will form an ideal tool for the reverse-engineering and understanding of cellular systems and mechanisms. Computer simulations are expected to become in integral part of the scientific inference loop, hence complementing laboratory

bench experiments. Simulations that include all relevant physics, and the real geometry, allow to test hypothetical mechanisms by comparison to experiments. Computer simulations make accessible length and time scales that are difficult to reach experimentally, and they provide complete control over the system structure and its parameters. The resulting method of identifying essential system components that can be tested for experimentally is a powerful tool that has already proven its utility in a conceptual application [177].

Potential applications of computer simulations of reaction-diffusion processes range from the investigation of molecular sorting and endocytic dynamics, over testing whether *Turing patterns* [293] could explain ER exit site localization, to simulations of Golgi transport dynamics. Further possible applications include the investigation of *morphogenesis* [128], as well as simulations of *cell motility* [123] and *cell signaling* [172]. When experimental techniques for time-resolved three-dimensional tracking of organelle shapes become available, the present method can also be used for simulations that involve moving and deforming organelles.

Regarding the specific application of FRAP experiments in the ER, open questions about the homogeneity of the ER lumen can be addressed. More realistic initial conditions for FRAP [35, 312] will enable more accurate analysis and allow quantification of the influence of the initial condition on the result. This will be of increasing importance as better microscopy techniques become available, or when light interference microscopy is used.

9.5 The PPM library

The PPM library and the numerical methods implemented therein will help addressing the current computational challenges in whole-cell computer simulations [279].

Present work in the PPM library is concerned with providing C++ bindings for all user-callable functions, and with adding interfaces to additional external libraries such as Hypre (parallel pre-conditioners), Fishpack (fast Helmholtz solver), and HDF5 (platform-independent binary I/O). Future developments will include the implementation of higher-order methods, P^3M and SPME algorithms [138, 306], immersed interface methods [308], parallel FMM [122] for far-field boundary conditions in the mesh-based solvers, and a parallel boundary element solver. The development of the FFT part will enable free-space boundary conditions using FFTs, differentiation in Fourier space, and general frequency domain operators with user-defined Green's function.

The next major version of the PPM library will also include support for multi-level structures with selectively allocated memory patches that can be arbitrarily placed in the computational domain. This will enable the implementation of multi-domain and multi-level particle schemes [27] in the spirit of mesh-based AMR libraries such as *CHOMBO* [11] and *SAMRAI* [320]. Solvers based on AMR [29] and heterogeneous multiscale methods [2, 3] can then also be added to the library architecture. Open issues pertaining to the validation, maintainability, flexibility, and performance of such codes however need to be addressed by using modern *software engineering* principles.

Part IV

Appendix

Appendix A

Feature Point Tracking Software Resources

A.1 Server and text-mode client users manual

This section describes installation and use of the SPT software described in Subsection 1.2.3, implementing the algorithm presented in Section 1.2. Only the text mode (console) software is described in this section. Usage of the graphical front-end is explained in Appendix A.3. The text-mode software has been successfully tested on the following platforms:

- Microsoft Windows 2000 Professional
- Microsoft Windows XP
- Linux Debian/GNU
- Linux RedHat
- Linux Mandrake
- FreeBSD 4.1
- MacOS X 10.3 and 10.4.

A.1.1 Server

Installation

The particle tracking server is distributed in source code, written in standard ANSI C. After unpacking the distribution, there are the `Server` and `Client` subdirectories that contain the source code for the respective parts of the software. Before compilation, three parameters have to be defined in `config.h` in the `Server` subdirectory. Defining either `DOUBLE_PRECISION` or `SINGLE_PRECISION` declares the floating point precision used by the program. Furthermore, the default TCP communication port and the maximum number of allowed concurrent clients can be set. The comments in the file describe the details.

Windows:
In order to compile the server application under Windows, Microsoft Visual Studio 6 or later is required and your system has to support the Winsock2 API. In Visual Studio, open the file `server.dsw` in the `Server` directory. Be sure to change the compiler settings from `Debug` to `Release` using the menu entry *Build → Set Active Configuration...* . To compile the software, select the menu item *Build → Build server.exe*, or press `F7`. The final executable is called `server.exe` and is located in `Server/bin`. Copy the executable to wherever you want to install it on your system. Make sure the program has read and write permissions for the directory in which it is installed.

Linux/UNIX/MacOS X:
In a terminal/console window, change to the directory `Server` and type `make`. The software will be compiled and the executable placed into `Server/bin`. Copy the executable to wherever you want to install it on your system. Make sure the program has read and write permissions for the directory in which it is installed.

Usage

To start the server application, change to the directory where it has been copied to and start the program from a terminal/console (Windows: DOS command window) by typing `server`. It is important that the program is started from a console and not by clicking it. While starting, to program prints something like

```
Starting server ...
Using defaults*:
        Listening on port 1138
        Max. number of connections: 128

* To specify your own values for the port and the maximum number
  of connections please start the server as follows:
        server [port max_connections]

Server is running on laptop:1138 (192.168.1.33:1138)
```

The values for the port and the maximum number of allowed concurrent client connections may vary depending on the settings in `config.h` (see above under "Installation"). These default values can be overridden using command line arguments. Starting the program with `server 1234 42`, e.g., causes the server to listen to TCP port 1234 and allow 42 connections at most.

Starting the server fails if the program does not have write permissions for the directory in which it is installed. An error message indicates this condition.

The server is stopped (terminated) by hitting `CTRL+C` in the console window where it is running. Under Linux/UNIX/MacOS X, the `kill` command can be used alternatively. Make sure to kill the parent process.

A.1.2 Text-mode client

Installation

The text-mode client is also distributed as ANSI C source code, located in the `Client` directory after unpacking the software. No pre-compilation parameters need to be set. Building the executable is done in a similar way to the server application:

Windows:
In order to compile the client application under Windows, Microsoft Visual Studio 6 or later is required and your system has to support the Winsock2 API. In Visual Studio, open the file `client.dsw` in the `Client` directory. Be sure to change the compiler settings from `Debug` to `Release` using the menu entry *Build* → *Set Active Configuration...* . To compile the software, select the menu item *Build* → *Build client.exe*, or press `F7`. The final executable is called `client.exe` and is located in `Client/bin`. Copy the executable to wherever you want to install it on your system. Make sure the program has read and write permissions for the directory in which it is installed.

Linux/UNIX/MacOS X:
In a terminal/console window, change to the directory `Client` and type `make`. The software will be compiled and the executable placed into `Client/bin`. Copy the executable to wherever you want to install it on your system. Make sure the program has read and write permissions for the directory in which it is installed.

Usage

To start the client, change to the directory where it is installed and type `client myfile.in`, where `myfile.in` is the name of the input file that specifies the tracking job to be executed (see below). The client then connects to the server (specified in the input file), uploads the image data, and receives and stores the results.

Input file syntax

This section explains the syntax and contents of the input file that defines the tracker parameters and the tracking job. An example of an input file can be found in the `Client` directory of the software. The same input files can also be imported by the GUI client (see Appendix A.3) as parameter files. Each parameter is set on one line of the input file by a key word followed by an equality sign and the parameter value, e.g. `host = localhost`. The key words are case-insensitive. Only one command per line is allowed and no line must be longer than 1024 characters. Lines beginning with a hash character (#) are treated as comment lines and are ignored by the program. The tracker operates as a state machine and processes the input file top-down. This means that parameter settings are valid until they are overwritten by a subsequent line containing the same key word. The following key words exist:

Key word	Description	Default
`host`	Either the name or the IP address of the host computer where the server application is running.	`localhost`
`port`	TCP port the server is listening to.	`1138`
`radius`	Particle radius in pixels (w in Subsection 1.2.1). This value should be slightly larger than the apparent particle radius in the images, but smaller than the smallest inter-particle spacing.	`3`
`cutoff`	Cutoff threshold for non-particle discrimination (T_s in Subsection 1.2.1).	`3.0`
`percentile`	Points have to be in this upper percentile of the image intensity distribution in order to be accepted as particles (see r in Subsection 1.2.1). Unit is percent.	`0.1`
`displacement`	Maximum allowed displacement (in pixel) of any particle between two subsequent frames (L in Subsection 1.2.2).	`10.0`
`linkrange`	Number of frames to use for determining the optimal trajectory linking (R in Subsection 1.2.2).	`1`

`results`	The path and name of the file where the resulting trajectories are stored.	`results.txt`
`color`	For each image, the color channel used for tracking can be specified. The setting is valid for all following images until a different channel is specified. Valid values are: `r` (red), `g` (green), `b` (blue), and `i` (intensity). For grayscale images, this parameter has no effect. Invalid channel specifications are ignored.	`i`
`file`	Specifies the files containing the movie or images to be processed (see below for further explanation).	–
`type`	Defines the data type for the following files. The specification is valid until it is overwritten. The following values are accepted: `TIFF` for TIFF images and `MPEG` for MPEG-1 movie streams.	`TIFF`
`list`	A *for-loop* like construct specifying the numbering sequence of the image files (see below for details).	–
`verbose`	Determines whether the output file contains additional information and intermediate results. Possible values are: `0` for no additional output, or `1` for verbose mode.	`0`

The commands `file` and `list` are used to specify the image data to be processed. Suppose we want to track particles in a sequence of 20 TIFF frames. The input file would then contain the lines:

```
file = frame1.tif
file = frame2.tif
file = frame3.tif
...
file = frame20.tif
```

While this allows very flexible file naming conventions, and parameter re-

definitions within the file sequence, it becomes cumbersome for large numbers of files. The `list` command is provided to loop through systematically named files. The above example thus equivalently becomes:

```
list = 1, 20, 1
    file = frame%d.tif
```

The value of `list` contains `[start], [stop], [stepsize]` for the loop. The place-holder `%d` in the file name is replaced by the appropriate frame number. If the file name itself contains a percent sign, use `%%`. All C format definitions are allowed as listed in the documentation of the `printf` function in any C reference book. Only the one `file` command that immediately follows the `list` line is considered a part of the loop construct. Therefore, `list` and `file` commands can be freely intermixed:

```
file = frame1.tif
list = 2, 9, 1
    file = frame%d.tif
file = frame10.tif
file = frame11.tif
list = 12, 20, 1
    file = frame%d.tif
```

Between individual `file` commands, parameters can be re-defined to, e.g., select a different color channel for the subsequent images or to change the file type.

Result file syntax

After successfully completing a tracking job, the result file contains the reconstructed trajectory data as ASCII text in six columns. The first column indicates the frame number. The second and third columns contain the x- and y-coordinates of the particle, respectively. The x-axis points top-down and the y-axis is oriented left-right in the image plane. The 4$^{\text{th}}$ and 5$^{\text{th}}$ columns contain the intensity moments of order 0 and 2, respectively (cf. m_0, m_2 in Subsection 1.2.1), and the 6$^{\text{th}}$ column contains the non-particle discrimination score (S_p in Subsection 1.2.1). Individual trajectories are separated by blank lines, and a trajectory is always at least two frames in length. The file header contains general information about the tracking job such as the used parameter settings and the total processing time.

A.2 Programming and API reference

This section documents the particle tracking API and the client-server communication protocol, complementing the comments that are contained in each source code file to describe the purpose of all functions and subroutines. The source code of the particle tracking server application consists of the following files:

`config.h`	Configuration file for compile-time parameters.
`convolve.c`	Optimized convolution routines for different kernel radii (see Subsection 1.3.1).
`dilate.c`	Optimized gray-scale dilation routines for different kernel radii.
`filelist.c`	File list handling API.
`import.c`	Routines for reading image and movie files.
`messages.c`	The messages of the communication protocol.
`server.c`	The main server program.
`sing.c`	Subroutines for computing FFTs.
`tracker.c`	The particle tracker API.
`mpeg/`	This directory contains the files of *mpeglib*.
`tiff/`	This directory contains the files of *libtiff*.

The source code of the client application consists of only one file, `client.c`, which can be found in the `Client` directory.

A.2.1 Naming conventions

All constants and function names are marked by one of the following prefixes:

`SERVER`	for the server part,
`FL`	for the file list API,
`PT`	for the particle tracker API,
`IMPORT`	for image and movie import,
`SM`	for server messages,
`CLIENT`	for the client part.

A.2.2 External libraries

The software includes the external libraries *libtiff* and *mpeglib* for reading TIFF images and MPEG-1 movie streams. These libraries are open-source, and included in the distribution since specific proprietary changes to them were necessary as detailed below.

libtiff

Importing TIFF images is done using the libtiff v3.5.7 [286]. All common TIFF file formats and compressions are supported. The only proprietary adaptation consisted in setting the two global variables `_TIFFerrorHandler` and `_TIFFwarningHandler` of *libtiff* to `NULL`, in order to avoid unwanted file output.

mpeglib

Importing MPEG-1 video streams is done by the mpeglib v1.3.1 [283]. Only MPEG-1 video streams without audio are supported. Since the mpeglib uses global variables, only one process can use it at a time. Access to mpeglib is thus controlled by a *semaphore* in the tracker server. In addition, the mpeglib was modified to not include any `exit()` calls any more, as these would cause the whole server application to terminate.

FFT

FFTs are computed using the split radix algorithm [264]. An existing code was ported from Fortran to C. Since it makes use of global variables, access is again controlled by a *semaphore* and only one process at a time is allowed to use the FFT routine.

A.2.3 Particle tracking API documentation

The particle tracking algorithm itself is implemented as an API, allowing it to be used by other programs in the future. The main data structure of the API is the `PTSequence`, defined as follows:

```
typedef struct PTSequence
{
    int radius; /* Kernel radius */
    real cutoff; /* Cutoff radius */
```

```
    real percentile; /* Percentile */
    real displacement; /* Maximum displacement */
    int verbose; /* verbose mode */
    real lambda; /* Filter correlation lenght */

    /* Image sequence parameter */
    int width, height;
    real min, max;
    int number_of_frames;
    real *frame;
    PTFrame *framelist; /* List of all frames */

    int linkrange;
    ParticleList *particlelist; /* List of all particles */

    /* Function pointer to the appropriate routine */
    real *(*Convolve)(struct PTSequence *pts, real *filtered, \\
        real *input);
    real *(*Dilate)(struct PTSequence *pts, real *dilated, \\
        real *input);

    real *kernel; /* Holds the kernel */
    int kernel_width;

    int *mask; /* Holds the dilation mask */

    real *filtered; /* Holds the filtered image */
    real *dilated; /* Holds the dilated image */

    /* Structure for the FFT calculation */
    FFT fft;

    /* Result handling */
    int result;
    char result_file[256];
    FILE *result_fp;

    /* Error handling */
    int error;
    char error_msg[128];
} PTSequence;
```

The data type `real` is defined to be either `float` or `double`, depending on the compile-time settings in `config.h`. The definitions of the types `PTFrame`, `ParticleList`, and `FFT` can be found in `tracker.h`.

The following functions are implemented by the *particle tracking API*:

- `PTSequence *PT_CreateSequence(void)`
  Creates a `PTSequence` structure, allocates memory for it, and provides all variables within the structure with default values. Returns a pointer to the created structure. This function should be called before any other, in order to initialize the API.

- `int PT_SetParameter(PTSequence *pts, const char *params)`
  Is used to set the user-defined parameter in a `PTSequence`. The variable `*params` contains a string consisting of a code defining the parameter type, and the value for the parameter. If `*params` e.g. contains `"1 4"`, the kernel radius is set to 4. The codes are:

  ```
  #define PT_PARAM_KERNELRADIUS   1
  #define PT_PARAM_CUTOFF         2
  #define PT_PARAM_PERCENTILE     3
  #define PT_PARAM_DISPLACEMENT   4
  #define PT_PARAM_LINKRANGE      5
  #define PT_PARAM_VERBOSE        6
  ```

  The function returns either 1 for success or 0 on failure (the error variable is set to the appropriate error message).

- `int PT_InitSequence(PTSequence *pts, FileList *filelist)`
  Takes a pointer to a `PTSequence` structure and a pointer to a `FileList` structure. The `FileList` structure contains the list of images or movies that are to be processed. All images in the file list are checked for validity before they are imported. While reading the images (for MPEG streams every frame is extracted as an image), the user-defined color channel is stored in a temporary file. These temporary files are used to save main memory. Upon return of this function, the image parameter of the `PTSequence` is defined and all memory needed for the actual tracking is allocated. The function pointers to the proper convolution and dilation functions are set according to the kernel radius and the image size as outlined in Subsection 1.3.1. The dilation mask and the Fourier transform of the kernel are also computed and stored.

  The function returns either 1 for success or 0 on failure (the error variable is set to the appropriate error message). If an error occurs, the

PTSequence is reset to its empty state, as it would be after calling PT_CreateSequence().

- int PT_FindTrajectories(PTSequence *pts)
  This function takes a pointer to an initialized PTSequence (created by PT_InitSequence) and executes the actual particle tracking algorithm. The results of the tracking procedure are written to a temporary file (see PT_GetResults below).

  The function returns either 1 for success or 0 on failure (the error variable contains the appropriate error message). If an error occurs, the PTSequence is reset to its empty state, as it would be after calling PT_CreateSequence().

- void PT_DestroySequence(PTSequence *pts)
  Frees all memory used by the PTSequence *pts and deletes all temporary files that were created by it.

- void PT_ResetSequence(PTSequence *pts)
  Resets the PTSequence *pts to the empty state, as if it were newly created by PT_CreateSequence.

- char *PT_GetResults(PTSequence *pts)
  Returns a string containing the name of the temporary result file that was created by a successful run of PT_FindTrajectories(). If no file exists, or the run was not successful, a NULL pointer is returned.

- char *PT_GetError(PTSequence *pts)
  Returns a string containing the current error message. If no error occurred, a NULL pointer is returned.

Some of the above functions use other PT_ and IMPORT_ functions that are not intended for direct use. A typical sequence of API function calls is:

```
PT_CreateSequence
PT_SetParameter
PT_InitSequence
PT_FindTrajectories
PT_GetResults
PT_DestroySequence
```


Once a PTSequence has been created, it can be used more than once. This makes it possible to use loops like:

```
PT_CreateSequence
do while(...) {
    PT_SetParameter
    PT_InitSequence
    PT_FindTrajectories
    PT_GetResults
    PT_ResetSequence
}
PT_DestroySequence
```

The tracker server application uses such loops to minimize the overhead of memory allocation and deallocation.

A.2.4 Server software structure

The particle tracking server is built around the following data structure, from which a separate instance is created for each active client connection:

```
typedef struct Client
{
    unsigned int clientsocket;

    FILE *fp;
    FileList *filelist;
    FileListEntry *current_file;

    int fileupload_ack;
    Packet *in, *out;

    PTSequence *pts;
} Client;
```

The server is a user of the particle tracking API described above. The main components of the Client structure are the clientsocket, used for TCP communication with the client, the PTSequence to communicate to the tracking API, and two Packets for incoming and outgoing communication messages (see below).

The FileList for the image files is defined by the structures:

```
typedef struct FileListEntry
```

```
{
    char path[256];
    int type;
    int special;
    struct FileListEntry *prev, *next;
} FileListEntry;

typedef struct FileList
{
    int number_of_files;
    FileListEntry *root;
} FileList;
```

It has the form of a *doubly linked list* with pointers to both the previous and the next file entry. Access to the file list is implemented by the following functions:

```
FileList *FL_CreateFileList(void);
void FL_DestroyFileList(FileList *filelist);
FileListEntry *FL_AddFile(FileList *filelist, int type, \\
    int special);
void FL_RemoveFile(FileList *filelist, FileListEntry *entry);
void FL_RemoveAllFiles(FileList *filelist);
int FL_FileExists(const char *filename);
```

`*FL_AddFile(FileList *filelist, int type, int special)` adds a file to the list. The first parameter is the file list to which to add the entry, the second is the file type (`PT_FILE_TIFF` or `PT_FILE_MPEG`), and the third argument selects the color channel for this file (0: intensity, 1: red, 2: green, 3: blue). The function returns a pointer to the newly created `FileListEntry`. If an error occurs, a `NULL` pointer is returned.

The server allows multiple concurrent clients. This is done using multi-threading (on Windows) or multi-processing (on Linux/UNIX/MacOS X) to take advantage of CPU scheduling and multi-processor machines. Multi-processing should be preferred for its higher stability, memory protection (cf. global variables and semaphores above), and parallelizability.

A.2.5 Communication protocol documentation

The communication between the client and the server is based on a *packet protocol* that uses TCP/IP for packet transport. Communication packets are defined by the structure

```
typedef struct Packet
{
```



```
    int type;
    int len;
    int curlen;
    int buflen;
    unsigned char *body;
    unsigned char *buffer;
} Packet;
```

Each packet consists of a header and a body. A packet has a maximum length of 4096 bytes, 7 of which are reserved for the packet header. The remaining 4089 bytes can be used for the message body. The minimum packet length is 7 bytes, thus a header only. The header is split into 3 + 4 bytes, where the first 3 bytes define the type of the packet and the following 4 bytes declare the length of the body. The *packet type* is a number between 100 and 999 as enlisted below. The body may contain arbitrary data. The protocol uses human-readable ASCII encoding, so a sample packet of type 200, length 2, and body `OK` would read `"2000002OK"`.

All packet types and names are defined in `server.h` and carry the prefix `SERVER_CODE_`. The server knows 9 different packet types that can be received from clients. These types are called *request types*. In reply, the server can send one of 26 possible *response type* packets. 7 of them are *acknowledgment types*, 3 are *special types*, and 16 are *error types*. Error packets contain the error description string in their body. The server *response types* are:

Type	Name	Description
acknowledgment types		
200	OK	Okay
201	PARAMSET	Parameter was successfully set
202	UPLOADINITACK	File upload acknowledged
203	UPLOADOK	File upload completed successfully
204	ABORTEDUPLOAD	File upload aborted
205	FILESDELETED	Files successfully deleted
206	CALCDONE	Tracking procedure done
special types		
300	RESULT	Result of the tracking run
301	RESULTFINISH	All results have been sent
302	STATUS	Current server status
error types		
400	ERROR	General error
401	NOOPENFILE	Temporary file can not be opened
402	CALCFAILED	Tracking procedure failed

403	UPINPROGRESS	File upload in progress
404	NOUPINPROGRESS	No file upload in progress
405	UPTYPENOTSUP	Upload type is not supported
406	EXTNOTSUP	File type is not supported
407	FILELISTERROR	Failed to create `FileList` entry
408	FILEOPENFAILED	Failed to open file
500	UNKNOWNERROR	Unknown error
501	UNKNOWNCODE	Unknown packet type received
502	OUTOFMEMORY	Server is out of memory
503	THREADERROR	Failed to create a new thread
504	TOOMANYCLIENTS	Maximum number of clients exceeded
505	FORKERROR	Failed to fork a new process
506	INVALIDLENGTH	Invalid packet body length

In all packets except `RESULT`, the body is optional. `RESULT` packets always have to include a body with the tracking results. For the error packets, the body contains the error description string, if such a description is available.

The client *request packet types* are:

- `100 SETPARAM` Sets the value of a parameter. Both the parameter type and the new value are given in the packet body. The packet is processed by `PT_SetParameter` as described above. In response, the server sends either `PARAMSET`, if the parameter has been successfully set, or `ERROR` (error message from the tracking API is contained in the packet body).

- `101 UPLOADINIT uploadtype filetype color` Requests a file upload transaction from the server. The three parameters in the body are mandatory and separated by white-spaces: `uploadtype` is always 1, `filetype` is either 1 for a TIFF file or 2 for an MPEG stream, and `color` is 0 for intensity, 1 for red, 2 for green, or 3 for blue.

  The response from the server is `UPLOADINITACK` if the client may begin to upload the data, or one of the following if an error occurred: `FILEOPENFAILED` if the server could not create the temporary file, `FILELISTERROR` if the entry could not be added to the `FileList`, `EXTNOTSUP` if the file type is unknown, `UPTYPENOTSUP` if the upload type is not supported, or `UPINPROGRESS` if another upload from the same client is still in progress.

- `102 UPLOADDATA` Contains the data to be uploaded in the packet body. The upload is binary and no escape sequences are needed. A file can be split among several packets of this type that can be sent in arbitrary order. The response from the server is `OK` if the data were successfully received and written to the temporary file. The following errors are also possible: `NOOPENFILE` if no temporary file is open to write the data to, or `NOUPINPROGRESS` if the upload has not been properly initialized using `UPLOADINIT` (see above).

- `103 UPLOADFINISH` Tells the server that the current upload is finished. No more `UPLOADDATA` can be sent after this request. This packet has no body. The server replies `UPLOADOK` if all files were successfully received and stored, or `NOUPINPROGRESS` if no upload was initialized.

- `104 UPLOADABORT` Aborts the current upload process and causes the server to delete all files received so far. The response of the server is `ABORTEDUPLOAD` if the abort was successful, or `NOUPINPROGRESS` if no upload has been initialized.

- `105 DELETEFILES` Causes the server to delete all temporary files uploaded by this client so far. This can only be done after all uploads are finished or aborted. The response is `FILESDELETED` if all files have been successfully deleted, or `UPINPROGRESS` if an upload is still in progress.

- `106 EXECCALC` Causes the server to initialize the particle tracking API (see Appendix A.2.3) and to execute the actual tracking process (`PT_FindTrajectories`). The response is `CALCDONE` if the particle tracker has successfully finished the calculation. The body of this response packet contains the consumed computational time in seconds. `UPINPROGRESS` is returned if an upload is still in progress, and `CALCFAILED` indicates that the execution failed, in which case the error message from the API is returned in the packet body.

- `107 SENDRESULTS` After successfully executing a tracking procedure, this request asks the server to send the resulting trajectory data. The data can be split among several packets of type `RESULT`. The end of the result transfer is marked by a `RESULTFINISH` response packet. This last packet contains no data body. If the function call to `PT_GetResults()` fails, the server responds `ERROR`; if the result file written by the API can not be read by the server, the response is `FILEOPENFAILED`.

- `108 DISCONNECT` Closes the connection to the server and causes the server to free all memory and delete all files used by the terminated client connection. The corresponding thread or process is terminated and the server replies `OK`.

A.3 GUI client users manual

This section describes the use of the graphical (GUI) point tracking client. The GUI client requires the Java 2 virtual machine (Standard Edition J2SE, version 1.4 or higher) to be installed. The currently installed version can be checked by typing `java -version` in the console/command window of your computer. If J2SE is not installed, it can be downloaded from http://java.sun.com/j2se/. If you plan to run your own tracking server, you also need to install the server application as described in Appendix A.1.1. Otherwise, you need network access to a machine where the server application is running.

The GUI client software has successfully been tested in the following platforms:

- Microsoft Windows 2000 Professional with Java 2 SDK 1.4.1, 1.4.2, and 1.5 beta
- Microsoft Windows XP with Java 2 SDK 1.4.1
- UNIX / Solaris with J2SE 1.4.1
- Linux with J2SE 1.4.0 and 1.4.1
- MacOS X with Java 2 SDK 1.4.2.

A.3.1 Installation and start

To install the program, copy the `jar` file to the location of your choice. Make sure the program has read and write permissions for the target directory.

Windows:
The `jar` file is an executable and can be started by double-clicking onto it. An alternative possibility is to start the application from the command window. This is done by changing into the directory where the `jar` file has been installed and typing

```
java -cp GUI-Client.jar client.Client
```

Linux/UNIX:
The application is started from a console/terminal window by typing

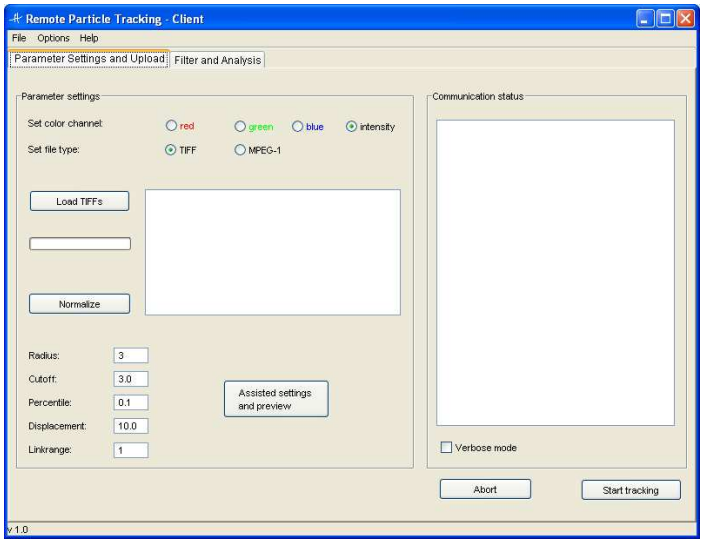


Figure A.1: GUI-Client after start-up.

```
java -jar GUI-Client.jar
```

MacOS X:
The program can be directly started by double-clicking the `jar` file. Alternatively, the program can be run from within a terminal window by typing

```
java -jar GUI-Client.jar
```

After starting the program, a screen like the one shown in Fig. A.1 is presented. The two tabs "*Parameter Settings and Upload*" and "*Filter and Analysis*" are described below.

A.3.2 The Parameter Settings and Upload tab

General settings

The menu entry *Options* → *General Settings* (see Fig. A.2) allows to adjust the location of the directories where temporary files and tracker result files are stored. By default, the temporary directory is created where the executable resides. The

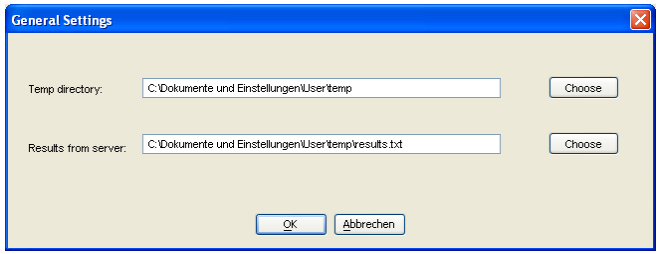


Figure A.2: General settings panel.

Figure A.3: Panel to set the connection parameters.

file to store the raw tracker results as received from the tracking server is usually chosen to be in the temporary directory.

Under the menu item *Options* → *Connection Settings*, the details of the server connection can be set (Fig. A.3). The field *Host name/IP* should contain the name or the IP address of the computer where the server is running. The TCP port for the connection is set in the field *Port*. If the tracker server can not be contacted, check that the machine address and port entries are correct, and that the tracker server application is running on the target machine. If both are the case, check that the firewalls of the client and server machines do not block access to the selected port.

Setting tracker parameters

Before a tracking job can be started, the algorithm parameters need to be set. This starts by choosing the file type of the image data in the *Parameter Settings and Upload* tab as shown in Fig. A.1:

- **TIFF:** A sequence consecutively numbered TIFF image files can be loaded either using the menu *File* → *Load Image Sequence*, or the button *Load TIFFs* to the left of the image list. This causes the selected images to be copied into the temporary directory to protect your original data from unwanted changes. If necessary, the images can be normalized by pushing the *Normalize* button. This changes the pixel values in all images of the sequence according to:

$$I_{\text{new}} = \frac{I - I_{\text{min}}}{I_{\text{max}} - I_{\text{min}}} \cdot 255\,, \tag{A.1}$$

  where I_{min} and I_{max} are the global (over all images of the sequence) minimum and maximum pixel values.

- **MPEG-1:** An MPEG-1 movie file can be imported directly using the menu item *File* → *Load MPEG*, or the button *Load MPEG* to the left of the image list (only visible when the file type radio button is set to *MPEG-1*). All frames contained in the MPEG movie are extracted and stored in the temporary directory. Normalizing the frames of an MPEG movie (see above) only affects the preview. The actual tracking is still done on the original MPEG movie file.

The other parameters in this tab are:

- **Color channel:** Selects the color channel that is to be used for tracking. The intensity channel is computed as the mean of the three color channels. For grayscale images, this parameter has no effect.

- **Radius:** Approximate radius of the particles in the images in units of pixels (w in Subsection 1.2.1). The value should be slightly larger than the visible particle radius, but smaller than the smallest inter-particle separation. The GUI program offers assistance in choosing a good value for this parameter (see next paragraph).

- **Cutoff:** The score cut-off for the non-particle discrimination filter (T_s in Subsection 1.2.1).

- **Percentile:** The percentile r (cf. Subsection 1.2.1) that determines which bright pixels are accepted as points. All local maxima in the upper r^{th} percentile of the image intensity distribution are considered candidate points. *Percentile* is given in units of percent (0% to 100%).

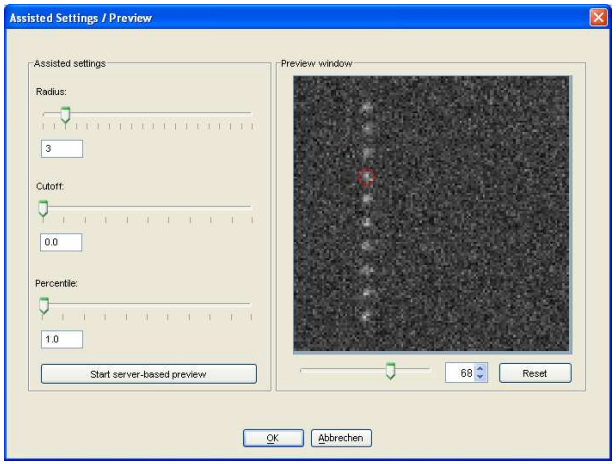


Figure A.4: Assisted settings and preview.

- **Displacement:** The maximum number of pixels a particle is allowed to move between two subsequent frames (L in Subsection 1.2.2).
- **Linkrange:** The number of subsequent frames that are taken into account to determine the optimal trajectory linking (R in Subsection 1.2.2).
- **Verbose:** If checked, this causes the server to send additional information and intermediate results. They can be inspected in the raw tracker result file (cf. "General settings" above).

Assisted settings and preview

The *Assisted settings and preview* button opens a preview window as shown in Fig. A.4. This can be used to find good tracker parameter settings or to test the effect of a parameter change on the tracking outcome.

The preview window on the right-hand side shows the frames of the movie or the image sequence that has been imported. The slider below it allows to navigate through the frames, click-and-drag within the image zooms the view. The *Reset* button restores the original view.

The red circle shows the currently chosen *Radius* value. Its center can be set by CTRL+*click* in the preview image. Placing it directly over a particle allows

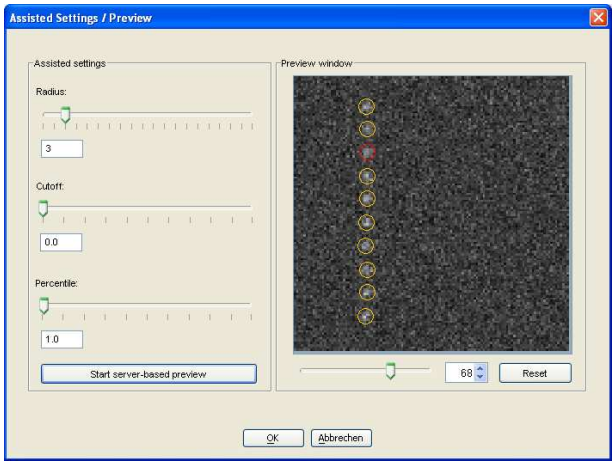


Figure A.5: Recognized particles in a server-based preview.

to easily find the proper setting for the parameter *Radius* using the slider on the left-hand side of the window.

The settings of the *Cutoff* and *Percentile* parameters can be evaluated in a tracker test-run by clicking the button *Start server-based preview*. Particles that are recognized using the current parameter settings are marked by an orange circle in the preview window as shown in Fig. A.5. The parameter values can be adjusted iteratively until the outcome matches your expectations. Doing so for different frames throughout the movie sequence ensures constant tracking quality along the movie.

Clicking *OK* copies the chosen parameter values into the main window, *Cancel* discards the changes.

Saving and reading parameter settings

Once a good set of parameters is found for a particular type of data, they can be stored in a parameter file for later re-use. This is done using the menu item *File* → *Export Settings* and choosing the location for the exported file.

Exported parameter settings can later be read again using the menu item *File* → *Import Settings*. The format of the parameter files is compatible with the text-mode client and is described in Appendix A.1.2 under "Input file syntax".

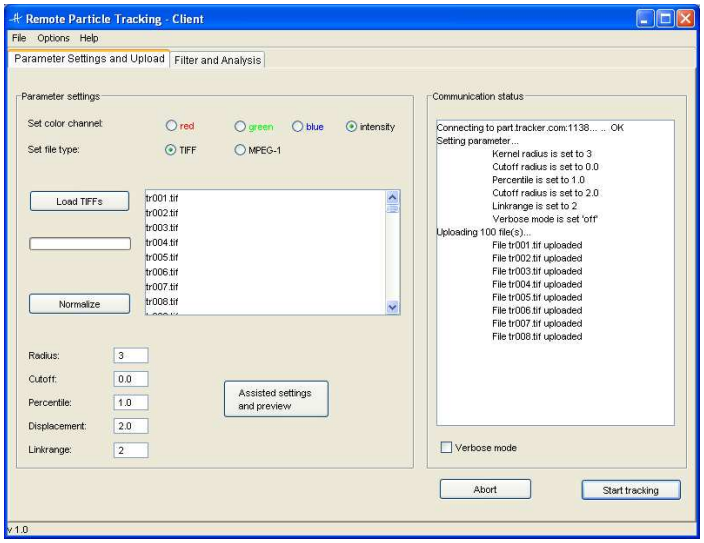


Figure A.6: Image upload to the tracking server.

Starting a tracking job

After setting all parameters, tracking can be started by clicking the button *Start tracking* in the main window. The client now establishes a connection to the server. An error message appears if the connection can not be opened. In this case, check the server settings and the server machine as described above under "General settings". Once a connection has been established, the image data are uploaded to the server. Depending on the data volume and the speed of the network connection, this may take several minutes (Fig. A.6). The upload process can be interrupted any time by pushing the *Abort* button. The transfer is aborted as soon as the current file has finished transmitting.

After completing the particle tracking, the server sends the raw result file back to the GUI client. The location of storage of this raw result file can be set under *Options* → *General Settings* → *Results from server*. The format of the result file is described in Appendix A.1.2 under "Result file syntax". After successfully downloading the results, the connection to the server is closed and the results can be inspected as described in the following section.

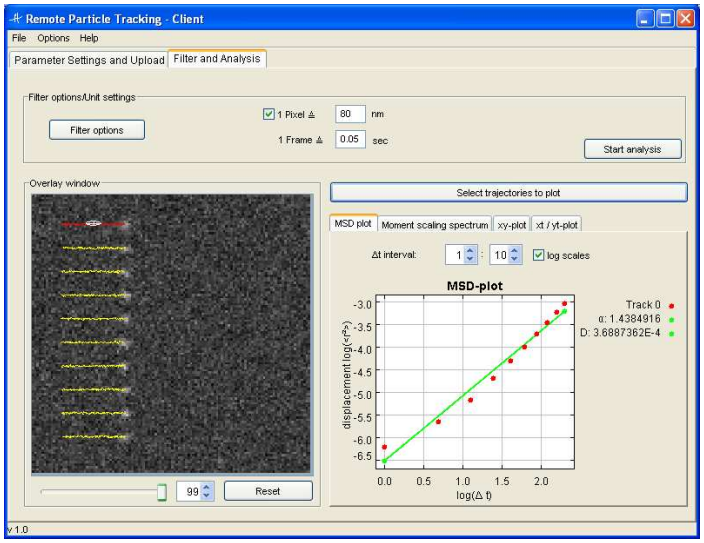


Figure A.7: The Filter and Analysis tab.

A.3.3 The Filter and Analysis tab

Switching to the *Filter and Analysis* tab as shown in Fig. A.7 allows to inspect the tracking results and to perform basic global trajectory analyses as described in Section 2.1.

The *Filter options/Unit settings* section allows to filter the results and to convert from pixels and frames to nanometers and seconds. After changing the filter options or the physical units, the *Start analysis* button has to be pushed for the changes to take effect.

Filter Options

Clicking the *Filter options* button opens a window with four different filters to reduce the number of trajectories (Fig. A.8).

The first filter causes trajectories below a certain length to be excluded from analysis. The second filter also operates on the trajectory length by selecting only the m longest for analysis. The third filter discards all trajectories with a diffusion constant below a certain limit. This is useful to exclude stationary particles from

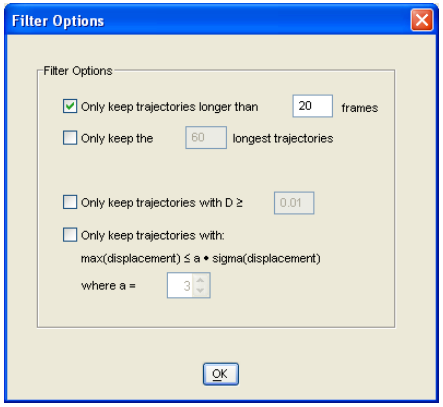


Figure A.8: Filters to reduce the number of trajectories.

subsequent analysis. The fourth filter finally can be used to discard trajectories that contain leaps, probably corresponding to tracking errors. With this filter, all tracks where the longest step displacement is more than a-times larger than the standard deviation of all step displacements are excluded from the analysis. Selecting multiple filters causes them to be applied sequentially in the order in which they appear in the window (cf. Fig. A.8).

Overlay window

The *Overlay window* is similar to the preview window described above. It visualizes the trajectories and the original movie frames in an overlay, for the user to check the correctness of the results. The slider below the image can be used to scroll through the frames, click-and-drag zooms in to a specific region. The *Reset* button restores the original view.

With the button *Select trajectories to plot*, individual trajectories can be manually chosen for analysis. The selected trajectories are highlighted in red in the *Overlay window*. Ellipses mark confinement zones in the selected tracks (see below).

Global analysis plots

The selected and/or filtered trajectories can be analyzed in a number of diagrams. All graphics are created using ptplot 5.3[1]. Portions of a plot can be enlarged by drawing a rectangle into the plot, starting from the upper-left corner and ending at the lower-right corner. Drawing the rectangle the opposite way (i.e. lower-right to upper-left) zooms out. All plots can be exported as EPS graphics files by typing `CTRL+S` within an active plot window. This is a proprietary function that is not present in the original ptplot package. Replacing the ptplot part of the GUI client with a standard version thus removes this functionality. The plot data can also be exported into text files by choosing the menu item *File* → *Export Plot Data*. This allows to archive the analysis results and to re-create the plots in any application of your choice.

The following trajectory analysis plots are available in the GUI client:

- **xy-plot:** This is the direct visualization of the data received from the particle tracker. The path traces of the trajectories are shown in the xy-plane. Notice that they are mirrored with respect to the *Overlay window*, due to different matrix indexing conventions.

- **xt/yt-plot:** These plots separately show the trajectory's x and y components versus time. A common use of such plots is to detect phases of immobility. If the moving-window standard deviation (cf. Section 2.2) of both the x and the y position simultaneously fall below a threshold σ for a time duration of more than τ, the particle is considered immobile during that time. The parameters σ and τ can be entered by the user in the plot window. All detected confinement areas are highlighted in the xt/yt-plot and also in the *Overlay window*. This is done by ellipses where the two half-axes are given by the position standard deviations in the x and y direction, respectively, stretched by a factor of 5 for better visibility.

- **MSD plot:** The global MSD analysis of the trajectories as described in Subsection 2.1.1 is shown in this plot. The plot can be displayed using either linear of logarithmic axes. The program automatically performs a linear least squares regression through the data in the plot and displays the diffusion constant and the MSD slope determined from this fit in the plotting window.

[1] Ptolemy project II: http://ptolemy.eecs.berkeley.edu/java/ptplot5.3/ptolemy/plot/doc/main.htm

- **Moment scaling spectrum:** The global MSS analysis as introduced in Subsection 2.1.2 is shown in this plot. The linear regression line to determine the MSS slope β is automatically determined and displayed.

Importing data for analysis

The described trajectory analysis functionality of the GUI can also be used on imported trajectory data that was generated earlier or using a different tracking program. Using the menu item *File → Import Tracks For Analysis* allows to select one or several trajectory files (text files in t–x–y column format with individual trajectories separated by a blank line). The data from these files are imported as if they were generated by the current run, except that the *Overlay window* is unavailable, since the original frame images are not present. The imported data are overwritten as soon as a new tracking job is started.

Export functionality

The GUI client can export the generated data in multiple ways to enable further processing and analysis in other applications. All export functions are located in the *File* menu:

- *Export unfiltered tracks.* Exports an exact copy of the raw result data as received from the tracking server.
- *Export filtered tracks.* All trajectories that have not been excluded from the analysis by the *Filter options* are exported to a file.
- *Export selected tracks.* Only the trajectories that were specifically selected using the *Select trajectories to plot* button are written to the file. They are marked red in the *Overlay window*.
- *Export analysis results.* Data from the global MSD and MSS analyses are exported. The first column is the trajectory number, second and third columns contain the MSD slope (from the log plot) and the diffusion constant, respectively. The fourth column contains the global MSS slope β of the trajectory (cf. Subsection 2.1.2).
- *Export plot data.* Exports all data from all plots to a file. This allows to archive the plots or to re-create them using other programs such as Matlab, gnuplot, or Excel.

Appendix B

Summary of Classification Methods

This appendix summarizes the classification methods that are used in this thesis. Brief mathematical and algorithmic descriptions are given. For more detailed descriptions of the methods, the reader is referred to the cited literature or the book by Cherkassky and Mulier [54].

B.1 k–nearest neighbors (KNN)

If the data come from a set $\mathcal{X} \subseteq \mathbb{R}^d$, classification can be done using the *k–nearest neighbor* (KNN) clustering algorithm. A previously unseen pattern $\boldsymbol{x} \in \mathcal{X}$ is hereby assigned to the same class $y \in \mathcal{Y}$ to which the majority of its k (to be chosen) nearest neighbors belongs. The algorithm constitutes the simplest form of a *self-organizing map* [158] with fixed connections.

B.2 Gaussian mixtures with expectation maximization (GMM)

Gaussian mixture models (GMM) are clustering algorithms in $\mathcal{X} \subseteq \mathbb{R}^d$. They assume Gaussian probability distributions on $\mathbb{R}^d$ and try to approximate the unknown distribution $P(\boldsymbol{x}, y)$ on $\mathcal{X} \times \mathcal{Y}$ by a mixture of n Gaussians $\mathcal{N}_i(\boldsymbol{x}, y, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ with means $\boldsymbol{\mu}_i \in \mathbb{R}^d$, $i = 1, \ldots, n$, and covariance matrices $\boldsymbol{\Sigma}_i \in \mathbb{R}^{d \times d}$, $i = 1, \ldots, n$. The parameters $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are chosen so as to maximize the *log-likelihood* that the given training sample has actually been drawn $i.i.d.$ from the probability distribution $P(\boldsymbol{x}, y) = \sum_{i=1}^{n} \mathcal{N}_i(\boldsymbol{x}, y, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. The algorithm proceeds as follows:

Algorithm 3 (Expectation maximization on a mixture of Gaussians)

Step 1: Choose a set of initial means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_n$ using the k–means *clustering algorithm [129]. All covariances are initialized to identity: $\boldsymbol{\Sigma}_i = \mathbb{1}_d$.*

Step 2: Assign the m training samples to the n clusters Γ_i using the minimum Mahalanobis distance rule*: Sample $\boldsymbol{x}$ belongs to cluster Γ_i if the corresponding log-likelihood measure becomes minimum, i.e. $i = \arg\min_i \left[\log\left(\det\left(\boldsymbol{\Sigma}_i\right)\right) + \left(\boldsymbol{x}-\boldsymbol{\mu}_i\right)^\top \left(\boldsymbol{\Sigma}_i\right)^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}_i\right)\right]$.*

Step 3: Compute new means $\boldsymbol{\mu}_i \leftarrow \sum_{\boldsymbol{x}\in\Gamma_i} \boldsymbol{x}/|\Gamma_i|$ and new covariance estimates $\boldsymbol{\Sigma}_i \leftarrow \sum_{\boldsymbol{x}\in\Gamma_i}\left(\boldsymbol{x}-\boldsymbol{\mu}_i\right)\left(\boldsymbol{x}-\boldsymbol{\mu}_i\right)^\top/|\Gamma_i|$ where $|\Gamma_i|$ denotes the number of vectors $\boldsymbol{x}$ assigned to cluster Γ_i.

Step 4: If the changes in the means and covariances are smaller than a certain tolerance, stop, otherwise go to Step 2.

B.3 Support Vector Machines (SVM)

Support Vector Machines (SVM) are kernel-based classifiers [199, 251] for binary classification in $\mathcal{X} \subseteq \mathbb{R}^d$. They are successfully used in time series prediction [198], gene expression analysis [39], and DNA or protein analysis [330]. SVM make use of a fundamental theorem from *statistical learning theory* that gives an upper bound for the expected risk [299]:

Theorem 1 *Let d_v denote the* Vapnik-Chervonenkis dimension *(VC) of the function class $\mathcal{F}$ and let $R_e[f]$ be the empirical risk for the 0/1-loss of a given classifier function $f \in \mathcal{F}$, evaluated on m test samples. It holds with probability of at least $1-\varepsilon$, that*

$$R[f] \leq R_e[f] + \sqrt{\frac{d_v\left(\log\frac{2m}{d_v}+1\right) - \log\left(\frac{\varepsilon}{4}\right)}{m}} \tag{B.1}$$

for all $\varepsilon > 0$, for $f \in \mathcal{F}$, and $m > d_v$.

The VC dimension d_v of a function class $\mathcal{F}$ measures how many points $\boldsymbol{x} \in \mathcal{X}$ can be separated in all possible ways using only functions of the class $\mathcal{F}$. Kernel methods use a mapping $\boldsymbol{\Phi}(\boldsymbol{x})$ of the training data $\boldsymbol{x}$ onto a higher-dimensional *feature space* $\mathcal{K}$ where the data can be separated by a hyper-plane $f(\boldsymbol{x}) = (\boldsymbol{w}\cdot\boldsymbol{\Phi}(\boldsymbol{x})) + b$. In $\mathcal{K}$, the optimal separating hyper-plane is determined such that the points $\boldsymbol{\Phi}(\boldsymbol{x})$ closest to it (called the *support vectors*) have maximum distance from it, i.e. such that the "safety margin" is maximized. This is done by solving the *quadratic programming problem* $(\boldsymbol{w}, b) = \arg\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|_2^2$ subject to the

condition that $\boldsymbol{w}\cdot\boldsymbol{\Phi}(\boldsymbol{x}) + b$ is a separating hyper-plane. Solving the dual optimization problem, the Lagrange multipliers α_i, $i = 1, \ldots, s$, are obtained, where s is the number of support vectors. The *classification function* f in $\mathcal{K}$ is then given by

$$f(\boldsymbol{x}) = \frac{3}{2} + \frac{1}{2}\cdot\operatorname{sign}\left(\sum_{i=1}^{s} y_i\alpha_i\left(\boldsymbol{\Phi}(\boldsymbol{x})\cdot\boldsymbol{\Phi}(\boldsymbol{x}_i)\right) + b\right). \tag{B.2}$$

Since f only depends on the scalar product of the data in feature space, the mapping $\boldsymbol{\Phi}$ does not need to be explicitly known. Instead, a *kernel function* $\eta(\boldsymbol{x}, \boldsymbol{x}_i)$ is introduced such that $\eta(\boldsymbol{x}, \boldsymbol{x}_i) = \boldsymbol{\Phi}(\boldsymbol{x})\cdot\boldsymbol{\Phi}(\boldsymbol{x}_i)$. The support vector classifier $f : \mathcal{X} \mapsto \{1, 2\}$ to be evaluated for any new observation thus is

$$f(\boldsymbol{x}) = \frac{3}{2} + \frac{1}{2}\cdot\operatorname{sign}\left(\sum_{i=1}^{s} y_i\alpha_i\eta(\boldsymbol{x}, \boldsymbol{x}_i) + b\right). \tag{B.3}$$

Notice that the sum only runs over all support vectors. Since generally $s \ll m$, this allows efficient classification of a new observation by comparing it to a small relevant subset of the training data. The assumed functional form of the kernel η determines the function space of the map $\boldsymbol{\Phi}$ and thereby the performance of the particular classifier.

B.4 Hidden Markov Models (HMM)

Hidden Markov Models (HMM) are stochastic signal source models, i.e. they do not require observations $\boldsymbol{x} \in \mathbb{R}^d$, but can treat discrete dynamic time series $\boldsymbol{x} = \{O_1, \ldots, O_T\} \in \mathcal{X}$, $O_i \in \mathbb{R}$. In the past, their most successful application was in speech recognition [225]. An HMM attempts to model the source producing the signal $\boldsymbol{x}$ as a dynamic system that can be described at any time t as being in one of r distinct discrete states, $Q_1, \ldots, Q_r$. The states are hidden in the sense that they can not be observed. At regularly spaced discrete time points $t_i = i\Delta t$, $i = 1, \ldots, T$, the system changes its internal state, possibly back to the same state. The process is assumed to be *Markovian*, i.e. its probabilistic description is completely determined by the present and the previous state. Let q_i denote the state of the system at time t_i. The Markov property then states that $P\left[q_i = Q_j | q_{i-1} = Q_k, q_{i-2} = Q_l, \ldots\right] = P\left[q_i = Q_j | q_{i-1} = Q_k\right]$. The *state transitions* are described by probabilities $a_{jk} = P\left[q_i = Q_k | q_{i-1} = Q_j\right]$ forming the elements of the state *transition matrix* $\boldsymbol{A}$ under the constraints $a_{jk} \geq 0$

$\forall\ j,k$ and $\sum_{k=1}^{r} a_{jk} = 1$. At each time point t_i the system produces an observable output O_i, drawn from an output probability distribution $b_{Q_i}(O)$ associated with state Q_i; $\mathcal{B} = \{b_{Q_j}\}_{j=1}^{r}$. The model is completed with the initial state probabilities $\Pi = \{\pi_j = P[q_1 = Q_j]\}_{j=1}^{r}$ and the complete HMM is denoted by $\Lambda = (\boldsymbol{A}, \mathcal{B}, \Pi)$.

Given the form of HMM described above, there are three problems of interest to be solved [225]:

(1) Given an observation $\boldsymbol{x} = \{O_1, \ldots, O_T\}$ and a model $\Lambda = (\boldsymbol{A}, \mathcal{B}, \Pi)$, compute the probability $P[\boldsymbol{x}|\Lambda]$ that the observation $\boldsymbol{x}$ has been produced by a signal source described by Λ.

(2) Given an output sequence $\boldsymbol{x} = \{O_1, \ldots, O_T\}$ and a model $\Lambda = (\boldsymbol{A}, \mathcal{B}, \Pi)$, determine the most probable internal state sequence $\{q_1, \ldots, q_T\}$ of the model Λ that produced $\boldsymbol{x}$.

(3) Determine the model parameters $\Lambda = (\boldsymbol{A}, \mathcal{B}, \Pi)$ to maximize $P[\boldsymbol{x}|\Lambda]$ for a given observation $\boldsymbol{x}$.

B.4.1 Discrete hidden Markov models (dHMM)

If the set of possible distinct values $\{v_k\}$ of any output O_i is finite, we call the HMM *discrete* (dHMM). The output probability distribution of any state Q_j is thus discrete: $b_{Q_j} = \left\{b_{Q_j}(k) = P[O_i = v_k | q_i = Q_j]\right\}$ for $k = 1, \ldots, M$. Direct solution of problem (1) would involve a sum over all possible state sequences: $P[\boldsymbol{x}|\Lambda] = \sum_{\forall\{q_1,\ldots,q_T\}} P[\boldsymbol{x}|\{q_1, \ldots, q_T\}, \Lambda] P[\{q_1, \ldots, q_T\}|\Lambda]$. The computational cost of this evaluation is $\mathcal{O}(2Tr^T)$, which is about 10^{50} for an average dHMM and thus clearly unfeasible. The *forward backward algorithm* [23, 24] solves this problem efficiently in $\mathcal{O}(r^2T)$. The solution of problem (2) is given by the *Viterbi algorithm* [302, 103], and the "training problem" (3) is solved using the iterative *Baum-Welch expectation maximization method* [82].

B.4.2 Continuous hidden Markov models (cHMM)

If the observations O_i are drawn from a continuum, b_{Q_i} is a continuous probability density function and the HMM is called *continuous* (cHMM). The most general case for which the above three problems have been solved is a finite mixture of n Gaussians $\mathcal{N}_i$, thus $b_{Q_j}(O) = \sum_{i=1}^{n} c_{jk}\mathcal{N}_k(O, \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk})$ [173, 150].

Appendix C

Converting a Triangulated Surface to a Level Set

Any triangulated surface that is, e.g., read from a geometry description file can robustly be converted to a smooth, regular level set using the following procedure. The level function is only stored within the narrow band, and the function value indicates the distance to the surface (signed distance function, cf. Subsection 5.3.2). Smoothness and regularity of the level function are important properties for the stability of the diffusion algorithms described in Sections 5.3 and 5.4.

Algorithm 4 (Triangulation to level set conversion)

Step 1: *Determine for each grid point if it is inside or outside of the closed surface described by the triangulation. This can be done using the point-in-domain algorithm [244] as, e.g., implemented in the* GNU Triangulated Surface Library *(GTS, http://gts.sourceforge.net).*

Step 2: *Each point outside of the surface is assigned the level function value $\psi = 1$, points inside are set to $\psi = -1$.*

Step 3: *Several sweeps (about 5) of a $3 \times 3 \times 3$ boxcar average low-pass filter with uniform weights are applied to the field ψ to regularize the level function and enable gradient computations.*

Step 4: *Reinitialize the level function (cf. Subsection 5.3.4) to produce a signed distance function with $\|\nabla\psi\|_2 = 1$.*

Appendix D

Diffusion on Domains with Complex Boundaries may Appear Anomalous

We consider a diffusion process that is confined to a complex-shaped domain. The process is observed at a larger length scale, not fully resolving the shape of the domain. Using volume averaging theory, we show in the following that the geometric complexity of the boundary shape has both qualitative and quantitative effects on the apparent observed diffusion. The isotropic, linear, strongly self-similar [101] diffusion process on the small scale is governed by the equation

$$\left(\frac{\partial}{\partial t} - \nabla^2\right) u(\boldsymbol{x}, t) = 0 \qquad , \boldsymbol{x} \in \Omega \tag{D.1}$$

with a function $u : \Omega \times \mathbb{R}_0^+ \mapsto \mathbb{R}$, $u \in C^\ell(\Omega \times \mathbb{R}_0^+)$, $\ell \geqslant 2$. This process takes place inside a closed and connected domain $\Omega \subset \mathbb{R}^d$ with boundary $\partial\Omega$. The $d-1$ dimensional manifold $\partial\Omega$ may have a complex shape with geometric features on a small length scale e. The whole process is observed by an imaginary observer with a resolution limit $E \gg e$ such that the ratio $\varepsilon = e/E \ll 1$ is negligible. We denote the order of magnitude of the values of $u(\boldsymbol{x}, t)$ by k. The limited resolution of the observer makes it impossible to fully resolve the shape of Ω on the observation length scale. All that the observer can see is a *volume average* of the field $u(\boldsymbol{x}, t)$, thus

$$\langle u(\boldsymbol{x}, t)\rangle = \frac{1}{|V(\boldsymbol{x})|} \int_{V_\Omega(\boldsymbol{x})} u(\boldsymbol{y}, t)\, d^d\boldsymbol{y}\,. \tag{D.2}$$

The average at each point $\boldsymbol{x}$ is taken over an *observation volume* $V(\boldsymbol{x})$, centered at $\boldsymbol{x}$, with $|V(\boldsymbol{x})| = \mathcal{O}(E^d)$. $V_\Omega(\boldsymbol{x}) = V \cap \Omega$ is the part of the averaging volume that is inside the domain (we did not require $u(\boldsymbol{x}, t)$ to be defined outside of Ω). $|\cdot|$ is the *volume measure* which can for example be the d-dimensional *Lebesgue measure*. To simplify the notation, we write V instead of $V(\boldsymbol{x})$ in the following.

The volume average in Eq. (D.2) amounts to convolving $u(\boldsymbol{x}, t)$ with the indicator function $\chi(V_\Omega)$. Since convolution and differentiation are commutative (M. Bergdorf, personal communication, 2005), the resulting averaged function is again $\in C^\ell$ and all differential operators can be applied to it.

The *volume averaging theorem* [218] for the gradient of a scalar quantity $\psi \in C^\ell$, $\ell \geqslant 1$, on Ω is:

$$\langle\nabla\psi\rangle = \nabla\langle\psi\rangle + \frac{1}{|V|}\int_{\mathcal{M}} \psi\boldsymbol{n}\, dA\,. \tag{D.3}$$

$\mathcal{M}$ is the surface of the domain inside the averaging volume, $\mathcal{M} = \partial(V \cap \Omega)$, $\boldsymbol{n}$ is the outward (i.e. out of Ω) unit normal on $\mathcal{M}$, and dA is the surface element on $\mathcal{M}$. Again the averaging volume V is not restricted to Ω.

For the divergence of a vector field $\boldsymbol{\Psi}$, the corresponding theorem reads:

$$\langle\nabla\cdot\boldsymbol{\Psi}\rangle = \nabla\cdot\langle\boldsymbol{\Psi}\rangle + \frac{1}{|V|}\int_{\mathcal{M}} \boldsymbol{n}\cdot\boldsymbol{\Psi}\, dA\,. \tag{D.4}$$

The volume average of the time derivative of a scalar quantity is

$$\langle\frac{\partial\psi}{\partial t}\rangle = \frac{\partial\langle\psi\rangle}{\partial t} - \frac{1}{|V|}\int_{\mathcal{M}} \psi\boldsymbol{v}\cdot\boldsymbol{n}\, dA\,, \tag{D.5}$$

where $\boldsymbol{v}$ is the velocity of the surface $\mathcal{M}$. This expression remains formally unchanged for vector quantities. Elegant proofs of these averaging theorems can be found in the 1977 paper by Gray [119].

Using the theorems in Eqs. (D.4) and (D.5), and assuming the boundary to be at rest, the volume average of equation Eq. (D.1) becomes

$$\frac{\partial\langle u(\boldsymbol{x}, t)\rangle}{\partial t} = \nabla\cdot\langle\nabla u(\boldsymbol{x}, t)\rangle + \frac{1}{|V|}\int_{\mathcal{M}} \nabla u(\boldsymbol{x}, t)\cdot\boldsymbol{n}\, dA\,. \tag{D.6}$$

Using Eq. (D.3) to expand the average of the gradient, this becomes

$$\frac{\partial\langle u(\boldsymbol{x}, t)\rangle}{\partial t} = \nabla^2\langle u(\boldsymbol{x}, t)\rangle + \frac{1}{|V|}\nabla\cdot\int_{\mathcal{M}} u(\boldsymbol{x}, t)\boldsymbol{n}\, dA + \frac{1}{|V|}\int_{\mathcal{M}} \nabla u(\boldsymbol{x}, t)\cdot\boldsymbol{n}\, dA\,. \tag{D.7}$$

The fully resolved solution $u(\boldsymbol{x}, t)$ can always be written as the sum of the averaged solution plus *small-scale fluctuations*, thus

$$u(\boldsymbol{x}, t) = \langle u(\boldsymbol{x}, t)\rangle + \widetilde{u}(\boldsymbol{x}, t)\,. \tag{D.8}$$

All terms in this equation are $\mathcal{O}(k)$ in their value. The length scales of variation in u and $\widetilde{u}$ are $\mathcal{O}(e)$, the ones of $\langle u \rangle$ are $\mathcal{O}(E)$. The governing equation for $\widetilde{u}(\boldsymbol{x}, t)$ is obtained by substituting Eq. (D.8) into Eq. (D.7), and subtracting it from the full Eq. (D.1):

$$\frac{\partial \widetilde{u}(\boldsymbol{x}, t)}{\partial t} = \nabla^2 \widetilde{u}(\boldsymbol{x}, t) - \frac{1}{|V|} \nabla \cdot \int_{\mathcal{M}} u(\boldsymbol{x}, t) \boldsymbol{n} \, dA - \frac{1}{|V|} \int_{\mathcal{M}} \nabla u(\boldsymbol{x}, t) \cdot \boldsymbol{n} \, dA \,. \quad \text{(D.9)}$$

Expanding the integral terms using Eq. (D.8) we find

$$\frac{\partial \widetilde{u}(\boldsymbol{x}, t)}{\partial t} = \nabla^2 \widetilde{u}(\boldsymbol{x}, t) - \frac{1}{|V|} \nabla \cdot \int_{\mathcal{M}} \langle u(\boldsymbol{x}, t) \rangle \boldsymbol{n} \, dA - \frac{1}{|V|} \nabla \cdot \int_{\mathcal{M}} \widetilde{u}(\boldsymbol{x}, t) \boldsymbol{n} \, dA - \frac{1}{|V|} \int_{\mathcal{M}} \nabla \langle u(\boldsymbol{x}, t) \rangle \cdot \boldsymbol{n} \, dA - \frac{1}{|V|} \int_{\mathcal{M}} \nabla \widetilde{u}(\boldsymbol{x}, t) \cdot \boldsymbol{n} \, dA \,. \quad \text{(D.10)}$$

Consider the first integral term on the right-hand side. The function $\langle u(\boldsymbol{x}, t) \rangle$ is – by construction – approximately constant over $\mathcal{M}$ and is pulled out of the integral. The remaining integral is the surface area of $\mathcal{M}$ and constant as well. The first term thus vanishes (divergence of a constant). The third term can be neglected, since the gradient of $\langle u(\boldsymbol{x}, t) \rangle$ is approximately zero. These qualitative statements can be made more formal by order analysis. The second and fourth integral terms are $\mathcal{O}(k/(eE))$[1], the first and third ones are $\mathcal{O}(k/E^2)$. The ratio of these orders is ε, which means that we can neglect the first and third integral. The final equation for the small-scale fluctuation hence becomes

$$\frac{\partial \widetilde{u}(\boldsymbol{x}, t)}{\partial t} = \nabla^2 \widetilde{u}(\boldsymbol{x}, t) - \frac{1}{|V|} \nabla \cdot \int_{\mathcal{M}} \widetilde{u}(\boldsymbol{x}, t) \boldsymbol{n} \, dA - \frac{1}{|V|} \int_{\mathcal{M}} \nabla \widetilde{u}(\boldsymbol{x}, t) \cdot \boldsymbol{n} \, dA \,. \quad \text{(D.11)}$$

In order to obtain an equation for the averaged field, we proceed in a similar way

[1] Integration over the surface $\mathcal{M}$ followed by division by the volume introduces a scale factor of $\mathcal{O}(1/E)$ in value.

by substituting Eq. (D.8) into Eq. (D.7):

$$\frac{\partial \langle u(\boldsymbol{x}, t) \rangle}{\partial t} = \nabla^2 \langle u(\boldsymbol{x}, t) \rangle + \frac{1}{|V|} \nabla \cdot \int_{\mathcal{M}} \langle u(\boldsymbol{x}, t) \rangle \boldsymbol{n} \, dA + \frac{1}{|V|} \nabla \cdot \int_{\mathcal{M}} \widetilde{u}(\boldsymbol{x}, t) \boldsymbol{n} \, dA + \frac{1}{|V|} \int_{\mathcal{M}} \nabla \langle u(\boldsymbol{x}, t) \rangle \cdot \boldsymbol{n} \, dA + \frac{1}{|V|} \int_{\mathcal{M}} \nabla \widetilde{u}(\boldsymbol{x}, t) \cdot \boldsymbol{n} \, dA \,. \quad \text{(D.12)}$$

By the same reasoning as before, we neglect the first and third integral terms on the right-hand side. For $\widetilde{u}(\boldsymbol{x}, t)$, we make the ansatz

$$\widetilde{u}(\boldsymbol{x}, t) = \boldsymbol{b}(\boldsymbol{x}, t) \cdot \nabla \langle u(\boldsymbol{x}, t) \rangle \,. \quad \text{(D.13)}$$

The value of $\widetilde{u}$ is $\mathcal{O}(k)$ and the one of $\nabla \langle u \rangle$ is $\mathcal{O}(k/E)$ (gradient on the large scale). The value of $\boldsymbol{b}$ thus is $\mathcal{O}(E)$, with the length scales of variation being $\mathcal{O}(e)$. Substituting this ansatz yields

$$\frac{\partial \langle u(\boldsymbol{x}, t) \rangle}{\partial t} = \nabla^2 \langle u(\boldsymbol{x}, t) \rangle + \frac{1}{|V|} \nabla \cdot \int_{\mathcal{M}} \left(\boldsymbol{b}(\boldsymbol{x}, t) \cdot \nabla \langle u(\boldsymbol{x}, t) \rangle \right) \boldsymbol{n} \, dA + \frac{1}{|V|} \int_{\mathcal{M}} \nabla \left(\boldsymbol{b}(\boldsymbol{x}, t) \cdot \nabla \langle u(\boldsymbol{x}, t) \rangle \right) \cdot \boldsymbol{n} \, dA \,. \quad \text{(D.14)}$$

The first and second integral terms are both $\mathcal{O}(k/(eE))$ [2]. Using the linearity of the divergence operator and the fact that $\nabla^2 = \nabla \cdot \nabla$, above equation can thus be approximated by

$$\frac{\partial \langle u(\boldsymbol{x}, t) \rangle}{\partial t} = \nabla \cdot \left(\left[\mathbb{1} + \frac{1}{|V|} \int_{\mathcal{M}} \boldsymbol{b}(\boldsymbol{x}, t) \otimes \boldsymbol{n} \, dA \right] \nabla \langle u(\boldsymbol{x}, t) \rangle \right) + \frac{1}{|V|} \int_{\mathcal{M}} \nabla \left(\boldsymbol{b}(\boldsymbol{x}, t) \cdot \nabla \langle u(\boldsymbol{x}, t) \rangle \right) \cdot \boldsymbol{n} \, dA \,, \quad \text{(D.15)}$$

[2] The integrand is $\mathcal{O}(k)$, the integration introduces a factor of $\mathcal{O}(E^{d-1})$, the divergence is of $\mathcal{O}(1/e)$ (small length scales in $\boldsymbol{b}$ cause large derivatives), and the division by the averaging volume is $\mathcal{O}(E^{-d})$.

where we have again used that $\langle u(\boldsymbol{x},t)\rangle$ is almost constant over $\mathcal{M}$. Both integrals are still $\mathcal{O}(k/(eE))$.

This can not be simplified any further without assuming specific boundary conditions for the small-scale process. Let Eq. (D.1) have homogeneous Neumann boundary conditions on $\partial\Omega$, thus $\boldsymbol{n}\cdot\nabla u(\boldsymbol{x},t)=0,\ \boldsymbol{x}\in\partial\Omega$. Using Eq. (D.8), this translates into the following condition for the fluctuations:

$$\boldsymbol{n}\cdot\nabla\widetilde{u}(\boldsymbol{x},t) = -\boldsymbol{n}\cdot\nabla\langle u(\boldsymbol{x},t)\rangle \qquad ,\ \boldsymbol{x}\in\partial\Omega\,. \tag{D.16}$$

Using this together with Eq. (D.13), the last integral term in Eq. (D.15) can be written as

$$\frac{1}{|V|}\int_{\mathcal{M}}\nabla\left(\boldsymbol{b}(\boldsymbol{x},t)\cdot\nabla\langle u(\boldsymbol{x},t)\rangle\right)\cdot\boldsymbol{n}\,dA = -\frac{1}{|V|}\int_{\mathcal{M}}\nabla\langle u(\boldsymbol{x},t)\rangle\boldsymbol{n}\,dA\,. \tag{D.17}$$

The term on the right-hand side is of $\mathcal{O}(k/E^2)$. The second integral in Eq. (D.15) is thus ε times smaller than the first one and can be neglected. The final governing equation for the averaged process, observed on the length scale E, under homogeneous Neumann boundary conditions thus becomes

$$\frac{\partial\langle u(\boldsymbol{x},t)\rangle}{\partial t} = \nabla\cdot\left(\left[\mathbb{1}+\frac{1}{|V|}\int_{\mathcal{M}}\boldsymbol{b}(\boldsymbol{x},t)\otimes\boldsymbol{n}\,dA\right]\nabla\langle u(\boldsymbol{x},t)\rangle\right)\,. \tag{D.18}$$

Both terms inside the parentheses are $\mathcal{O}(1)$, the divergence introduces a scale factor of $\mathcal{O}(1/e)$ (small scales in $\boldsymbol{b}$), and the gradient of the averaged field is $\mathcal{O}(k/E)$. The whole equation thus is of $\mathcal{O}(k)$ in value. This equation describes a diffusion process on the observation length scale E. The effectively observed *apparent diffusion tensor* on this scale is $\mathcal{O}(1)$ in value and given by

$$\boldsymbol{D}_{\text{app}} = \left[\mathbb{1}+\frac{1}{|V|}\int_{\mathcal{M}}\boldsymbol{b}(\boldsymbol{x},t)\otimes\boldsymbol{n}\,dA\right]\,. \tag{D.19}$$

The process thus appears *anisotropic* if the tensor $\boldsymbol{D}_{\text{app}}$ is not proportional to $\mathbb{1}$. It furthermore appears *anomalous* if $\boldsymbol{D}_{\text{app}}$ is a function of the time scale[3]. From Eq. (D.19) we see that diffusion can macroscopically appear anisotropic or anomalous (if $\boldsymbol{b}$ depends on time), even if the microscopic process is normal and isotropic.

[3] In this case, the mean square displacement no longer scales linearly with time, cf. Subsection 2.1.2.

Appendix E

Experimental Protocols

All experimental work used in Chapter 6 was carried out by members of the group of Prof. Ari Helenius at the Institute of Biochemistry at ETH Zürich. The original protocols are reproduced here for reference.

E.1 FRAP experiments in the ER lumen

E.1.1 Cells and DNA construct

VERO cells were grown on coverslips at 37°C in Dulbecco's Minimal Essential Medium supplemented with 10% fetal calf serum, 2 mM glutamine, 100 g/ml penicillin, 100 U/ml streptomycin (GibcoBRL; Life Technologies, Eggstein, Germany) at 37°C in a 5% CO_2 incubator and were used in all experiments. Cells were transiently transfected with a reporter gene containing the ER targeting signal sequence fused to GFP and the ER retention sequence (ssGFP–KDEL; derived from pCMV/myc/ER/GFP, Invitrogen) using Superfect (Sigma). Alternatively, cells were transfected using Nucleofactor by amaxa (Köln, Germany) according to the protocol for COS-7 cells (Kit V, program A24). Briefly, 1×106 VERO cells were pelleted, resuspended in 100 μl of solution V, and electroporated with 1 – 2.5 μg of DNA. The electroporated cells were resuspended in 350 μl MEM. Of this solution, 100 μl were seeded on one 18 mm coverslip and incubated over night (15 h) at 37°C and 5% CO_2. 12–16 hours post transfection cells were imaged live on a temperature-controlled stage at 37°C.

E.1.2 Photobleach Experiments

FRAP experiments were performed on an inverted Zeiss LSM510 confocal microscope, using the 488-nm line of a 30 mW Argon/2 laser with a 100×, 1.4 NA objective. A defined region of interest (ROI; 4 μm×4 μm) was photobleached at

full laser power (100% power, 100% transmission, 20 iterations); recovery of fluorescence was monitored by scanning the ROI at low laser power (50% power, 3% transmission). The scanning laser intensity did not significantly photobleach the specimen over the time course of the experiment. Images were acquired as 8-bit TIFF files (512×512 pixel frame; $0.18\,\mu$m/pixel) and processed using NHI Image 1.62. Image series with little or no apparent motion of ER structures within the ROI were selected. The average fluorescence in the ROI and the average background were determined from the images. After subtracting the background, the fluorescence values were normalized according to Phair and Misteli [216] to correct for the loss in fluorescence caused by imaging. To be able to compare FRAP curves from different cells, these values $F(t)$ were further normalized by their respective asymptotic value $F_\infty = F(t \to \infty)$, determined as outlined in Appendix F.1. FRAP$(t) = F(t)/F_\infty$ is shown in all the figures.

E.2 FRAP experiments on the ER membrane

E.2.1 Cell line, DNA construct and expression of VSVG-GFP

VERO cells were maintained in MEM (plus Earle's plus GlutaMAX^{TM}I) supplemented with 10% fetal calf serum and non essential amino acids (Gibco BRL, San Diego, CA, USA) at 37°C/5% CO_2. The cDNA plasmid VSVG3-SP-GFP [154] encoding GFP-tagged temperature sensitive vesicular stomatitis virus glycoprotein (tsO45-VSV-G) was kindly provided by Dr. Kai Simons. Cells on 18 mm glass coverslips at 80–90% confluence were transfected with 0.5 μg plasmid DNA per coverslip using FuGENE 6 Transfection Reagent (Roche Diagnostics, Indianapolis, IN, USA) and incubated for 12–14h at the non-permissive temperature (40°C), at which VSVG-GFP is incompletely folded and retained in the ER [107].

E.2.2 Live cell microscopy and FRAP analyses

For live cell microscopy, transfected cells on 18 mm glass coverslips were transferred to a custom-built metal microscope coverslip chamber in CO_2-independent medium supplemented with 10% FCS (Gibco BRL, SanDiego, CA, USA). FRAP analyses were performed at 40°C on an inverted Zeiss LSM510 confocal microscope (Oberkochen, Germany) equipped with a temperature-controlled stage and a 100x 1.4 NA objective. A defined region of interest (ROI; 4 μm×4 μm) was bleached using the 488 nm line of a 30 mW Argon laser at high laser intensity (100% power, 100% transmission) and fluorescence recovery was recorded by

scanning at low laser intensity (100% power, 10% transmission). Images were acquired as 12 bit LSM files at 512×512 pixels/frame and 0.09 μm/pixel lateral resolution. Image series with little or no apparent motion of ER structures within the ROI were selected and imported into ImageJ 1.34 (http://rsb.info.nih.gov/ij/) for processing. The average fluorescence intensity of the ROI was determined after background subtraction and normalization according to Phair and Misteli [216]. All FRAP curves were normalized by their asymptotic value as outlined in Appendix F.1.

Appendix F

Simulations of Diffusion in the Endoplasmic Reticulum

This appendix describes the details of the diffusion simulations of fluorescence recovery in the ER for both the lumen and the membrane as used for the results in Sections 6.5. All computer simulations are performed using the numerical methods described in Chapter 5, implemented in Fortran 90 and parallelized using the PPM library presented in Chapter 7 [248].

F.1 Simulations in the ER lumen

The diffusive motion of a fluorescently labeled soluble protein in the ER lumen is simulated and the total fluorescence intensity inside the originally bleached ROI B is monitored over time. We assume the molecules of interest to diffuse normally (i.e. no anomalous diffusion) and freely within the confines of the ER lumen Ω. In the following, the *FRAP value* at time $t_n = n\delta t$ is defined as

$$F(t_n) = \frac{1}{N_B} \sum_{p \in B} \omega_p^h(t_n)\,, \tag{F.1}$$

where N_B is the total number of particles inside the ROI B and $\omega_p^h(t_n)$ is the PSE particle strength. In order to focus on the influence of organelle geometry, we use the idealized *initial condition*

$$u_0(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \in \{\Omega \cap B\} \\ C & \text{if } \boldsymbol{x} \in \{\Omega \setminus B\}\,. \end{cases} \tag{F.2}$$

More realistic initial conditions [35, 312] can readily be accommodated by setting the initial strengths of the particles accordingly. They would however leave our conclusions unchanged as their effects would equally apply to all simulations. The

ROI B is taken to be the square cylinder defined by:

$$B = [p,q] \times [r,s] \times [0,L_z] \qquad \begin{cases} 0 \leqslant p < q \leqslant L_x \\ 0 \leqslant r < s \leqslant L_y\,, \end{cases} \tag{F.3}$$

with (L_x, L_y, L_z) the extent of the bounding box of the ER in all spatial directions. Without loss of generality, the constant initial concentration outside the bleached area is chosen to be $C = 1$ as this simply corresponds to normalizing the FRAP curves with respect to their pre-bleach value. The assumption of a homogeneous initial concentration distribution outside the bleached area seems feasible due to the following facts:

1. After transfection, the cells are incubated for at least 12 hours. During this time, they express the green fluorescent protein which is assumed to freely diffuse within the ER lumen and to completely fill it. Experiments show that a protein can easily move across the whole ER in about 30 seconds. Therefore, a homogeneous distribution inside the ER is assumed after 12 hours.

2. The experimenter chooses "healthy" cells, i.e. cells that exhibit a more or less homogeneous fluorescence inside the ER.

The geometric domain Ω for the simulations is a reconstructed representation of a real ER. The reconstruction technique is described in Section 6.2. The PSE simulations solve the isotropic, homogeneous diffusion equation in the lumen of the reconstructed ER shapes, using the second-order accurate isotropic PSE kernel proposed by Cottet (G.-H. Cottet, personal communication, 1999):

$$\eta(\boldsymbol{x}) = \frac{15}{\pi^2} \frac{1}{|\boldsymbol{x}|^{10} + 1}\,. \tag{F.4}$$

All simulations are run for the same value of the computational diffusion constant $\nu_{\text{sim}} = 3 \cdot 10^{-5}\, b^2/\delta t$ (scaled with the lateral edge length $b = 50$ of the ROI and the simulation time step $\delta t = 0.01$) in order to be able to study the influences of geometry. Time integration is done using the explicit Euler scheme with a time step of $\delta t = 0.01$ until a final time of $T = 2000\,\delta t$.

Since the bleached volumes of the different ER samples contain different numbers of particles, and since the total number of particles also varies among samples, the different FRAP curves have different asymptotic levels for $t \to \infty$. Moreover,

the FRAP curves do normally not recover to 1.0, even if the protein is fully mobile. This is due to the homogeneous Neumann boundary condition and the fact that the total mass in the domain is conserved. In order to be able to compare the FRAP curves of the simulation runs among each other, they are normalized by their respective steady-state value F_∞. They thus all asymptotically recover to 1.0, leaving the different geometries as the only source of variation. Initially, the total mass in the system is given by

$$m_t = \sum_{p=1}^{N} V_p u_p^h = V_p(N - N_B)\,. \tag{F.5}$$

The latter equality makes use of the initial condition as given in Eq. (F.2) and the choice $C = 1$. The asymptotic value of the fluorescence in the ROI is given by homogeneously distributing this mass among all particles. For particles with constant volume V_p we thus have

$$F_\infty = \frac{m_t}{NV_p} = \frac{N - N_B}{N}\,, \tag{F.6}$$

and we normalize the FRAP curves as $F(t)/F_\infty$.

F.2 Simulations on the ER membrane

The triangulated surfaces from the 3D reconstruction are first converted to level sets as described in Appendix C. This is not an intrinsic necessity of the method, but is required by the specific data output format available from the 3D reconstruction software (cf. Section 6.2).

The simulations solve the intrinsic diffusion equation on the membrane of the reconstructed ER shapes using the method presented in Section 5.3 and the operator discretization of Eq. (5.58). All simulations use a computational diffusion constant of $\nu_{\text{sim}} = 10^{-4}\,\mu\text{m}^2/\text{s}$, a band half-width of $k = 3h$ (h between 0.042 μm and 0.047 μm), and employ between 800'000 and 2 million particles concentrated in a $14\,\mu\text{m}\times 14\,\mu\text{m}$ neighborhood around the ROI of $4\,\mu\text{m}\times 4\,\mu\text{m}$. Since only the geometry in the vicinity of the ROI influences the fluorescence recovery, an ER cut-out around the bleached region is considered in the simulations. The finite reservoir of the rest of the ER can be modeled using Dirichlet boundary conditions of value F_∞ as given by Eq. (F.6). This is important for larger times and to recover the correct asymptotic level.

Time integration is done using a 9-step STS scheme [9] with an elementary Euler step size of $\delta t = 10^{-4}$ s. The concentration is initially set to 1 everywhere outside the bleached ROI, where it is set to zero (Eq. (F.2)). More accurate initial conditions [35, 312] can easily be used if they are experimentally available. The ROI is again given by Eq. (F.3). The total mass of fluorescent molecules in the ROI is determined from all particles adjacent to the membrane by linearly interpolating their strengths along inter-particle lines that cross the membrane. Prior to analysis, all FRAP curves are normalized by their steady-state value to make them comparable (cf. Appendix F.1).

The narrow-band level set method imposes a scale constraint on the geometry that can be resolved: the bands from two opposite surfaces must never overlap, i.e., the smallest "feature" of the surface must be at least $2k$ in diameter. In the present simulations, this amounts to $2k = 6h \approx 300\,\text{nm}$, which is more than 10 times larger than the curvature radius limit for biological membranes. In order to avoid under-resolved regions, the level function ψ is thus low-pass filtered prior to the simulations. This can be done without loss of information since the wavelength of the light used to record the geometry is larger (488 nm, cf. Appendix E).

Bibliography

[1] S. Abarbanel and A. Ditkowski. Asymptotically stable fourth-order accurate schemes for the diffusion equation on complex shapes. *J. Comput. Phys.*, 133:279–288, 1997.

[2] A. Abdulle and W. E. Finite difference heterogeneous multi-scale method for homogenization problems. *J. Comput. Phys.*, 191(1):18–39, 2003.

[3] A. Abdulle and C. Schwab. Heterogeneous multiscale FEM for diffusion problems on rough surfaces. *Multiscale Model. Simul.*, 3(1):195–220, 2005.

[4] M. Adams, H. H. Bayraktar, T. M. Keaveny, and P. Papadopoulos. Applications of algebraic multigrid to large-scale finite element analysis of whole bone micro-mechanics on the IBM SP. In *Proc. SC03 Conf. High Perf. Network. & Comput.* ACM/IEEE, 2003.

[5] M. F. Adams, H. H. Bayraktar, T. M. Keaveny, and P. Papadopoulos. Ultrascalable implicit finite element analyses in solid mechanics with over half a billion degrees of freedom. In *Proc. SC04 Conf. High Perf. Network. & Comput.*, page 34. ACM/IEEE, Gordon Bell Award paper, 2004.

[6] B. M. Aizenbud and N. D. Gershon. Diffusion of molecules on biological membranes of nonplanar form – a theoretical study. *Biophys. J.*, 38:287–293, 1982.

[7] B. M. Aizenbud and N. D. Gershon. Diffusion of molecules on biological membranes of nonplanar form. II. Diffusion anisotropy. *Biophys. J.*, 48:543–546, 1985.

[8] B. Alberts, D. Bray, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Essential Cell Biology*. Garland Publication, Inc., New York, 1997.

[9] V. Alexiades, G. Amiez, and P.-A. Gremaud. Super-time-stepping acceleration of explicit schemes for parabolic problems. *Comm. Numer. Meth. Eng.*, 12(1):31–42, 1996.

[10] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Clarendon Press, Oxford, 1987.

[11] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome. A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations. *J. Comput. Phys.*, 142:1–46, 1998.

[12] C. M. Anderson, G. N. Georgiou, I. E. Morrrison, G. V. Stevenson, and R. J. Cherry. Tracking of cell surface receptors by fluorescence digital imaging microscopy using a charge-coupled device camera. Low-density lipoprotein and influenza virus receptor mobility at 4 degrees C. *J. Cell Sci.*, 101(Pt2):415–425, 1992.

[13] W. Andreoni and A. Curioni. New advances in chemistry and material science with CPMD and parallel computing. *Parallel Comput.*, 26:819, 2000.

[14] M. Aon and S. Cortassa. On the fractal nature of cytoplasm. *FEBS Lett.*, 344:1–4, 1994.

[15] D. Axelrod, D. E. Koppel, J. Schlessinger, E. Elson, and W. W. Webb. Mobility measurement by analysis of fluorescence photobleaching recovery kinetics. *Biophys. J.*, 16:1055–1069, 1976.

[16] M. E. Bachlechner, A. Omeltchenko, A. Nakano, R. K. Kalia, P. Vashishta, I. Ebbsjö, and A. Madhukar. Dislocation emission at the silicon/silicon nitride interface: A million atom molecular dynamics simulation on parallel computers. *Phys. Rev. Lett.*, 84(2):322–325, 2000.

[17] G. Bader and R. Deiterding. A distributed memory adaptive mesh refinement package for inviscid flow simulations. In P. Jonas and V. Uruba, editors, *Proc. of Colloquium on Fluid Dynamics*, pages 9–14, Prague, Oct. 19–20 1999. Institute of Thermodynamics, Academy of Sciences of the Czech Republic.

[18] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 – Revision 2.1.5, Argonne National Laboratory, 2004.

[19] D. S. Banks and C. Fradin. Anomalous diffusion of proteins due to molecular crowding. *Biophys. J.*, 89:2960–2971, 2005.

[20] E. Bänsch, P. Morin, and R. H. Nochetto. A finite element method for surface diffusion: the parametric case. *J. Comput. Phys.*, 203:321–343, 2005.

[21] M. T. Barlow and R. F. Bass. Transition densities for Brownian motion on the Sierpinski carpet. *Probab. Theory Rel. Fields*, 91:307–330, 1992.

[22] M. T. Barlow and E. A. Perkins. Brownian motion on the Sierpinski gasket. *Probab. Theory Rel. Fields*, 79:543–623, 1988.

[23] L. E. Baum and J. A. Egon. An inequality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology. *Bull. Amer. Meteorol. Soc.*, 73:360–363, 1967.

[24] L. E. Baum and G. R. Sell. Growth functions for transformations on manifolds. *Pac. J. Math.*, 27(2):211–227, 1968.

[25] R. D. Benguria and M. C. Depassier. Speed of fronts of the reaction-diffusion equation. *Phys. Rev. Lett.*, 77(6):1171–1173, 1996.

[26] H. Berestycki, F. Hamel, and N. Nadirashvili. The speed of propagation for KPP type problems. I: Periodic framework. *J. Eur. Math. Soc.*, 7(2):173–213, 2005.

[27] M. Bergdorf, G.-H. Cottet, and P. Koumoutsakos. Multilevel adaptive particle methods for convection-diffusion equations. *Multiscale Model. Simul.*, 4(1):328–357, 2005.

[28] J. M. Bergelson, J. A. Cunningham, G. Droguett, E. A. Kurt-Jones, A. Krithivas, J. S. Hong, M. S. Horwitz, R. L. Crowell, and R. W. Finberg. Isolation of a common receptor for Coxsackie B viruses and Adenoviruses 2 and 5. *Science*, 275:1320–1323, 1997.

[29] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 191(1):18–39, 1984.

[30] J. Bernasconi. Informationsverarbeitung in neuronalen Netzwerken. Lecture notes, ETH Zürich 2004.

[31] M. Bertalmio, A. L. Bertozzi, and G. Sapiro. Navier-Stokes, fluid dynamics, and image and video inpainting. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, December 9–14 2001.

[32] M. Bertalmio, L. T. Cheng, S. Osher, and G. Sapiro. Variational problems and partial differential equations on implicit surfaces. *J. Comput. Phys.*, 174(2):759–780, 2001.

[33] U. S. Bhalla. Models of cell signaling pathways. *Curr. Opin. Genetics & Devel.*, 14:375–381, 2004.

[34] J. A. Board Jr., J. W. Causey, J. F. Leathrum Jr., A. Windemuth, and K. Schulten. Accelerated molecular dynamics simulation with the parallel fast multipole algorithm. *Chem. Phys. Lett.*, 198(1,2):89–94, 1992.

[35] J. Braga, J. M. P. Desterro, and M. Carmo-Fonseca. Intracellular macromolecular mobility measured by fluorescence recovery after photobleaching with confocal laser scanning microscopes. *Mol. Biol. Cell*, 15:4749–4760, 2004.

[36] D. Bray, R. B. Bourret, and M. I. Simon. Computer simulation of the phosphorylation cascade controlling bacterial chemotaxis. *Mol. Biol. Cell*, 4:469–482, 1993.

[37] P. P. Brieu and A. E. Evrard. P4M: a parallel version of P3M. *New Astron.*, 5:163–180, 2000.

[38] I. N. Bronstein and K. A. Semendyayev. *Handbook of Mathematics*. Van Nostrand Reinhold, 20th edition, 1991.

[39] M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. S. Furey, M. Ares, and D. Haussler. Knowledge-based analysis of microarray gene expression data using support vector machines. *Proc. Natl. Acad. Sci. USA*, 97(1):262–267, 2000.

[40] R. Brown. A brief account of microscopical observations made in the months of June, July, and August, 1827, on the particles contained in the pollen of plants; and on the general existence of active molecules in organic and inorganic bodies. *Phil. Mag.*, 4:161–173, 1828.

[41] A. Brünger, R. Peters, and K. Schulten. Continuous fluorescence microphotolysis to observe lateral diffusion in membranes. Theoretical methods and applications. *J. Chem. Phys.*, 82(4):2147–2160, 1985.

[42] J. M. Buhmann. Machine learning I. Lecture notes, ETH Zürich 2004.

[43] B. Burlando. The fractal geometry of evolution. *J. Theor. Biol.*, 163(2):161–172, 1993.

[44] A. Canning, L. W. Wang, A. Williamson, and A. Zunger. Parallel empirical pseudopotential electronic structure calculations for million atom systems. *J. Comput. Phys.*, 160:29–41, 2000.

[45] A. Caprara and R. Rizzi. Improving a family of approximation algorithms to edge color multigraphs. *Inform. Process. Lett.*, 68:11–15, 1998.

[46] E. A. Carmona and L. J. Chandler. On parallel PIC versatility and the structure of parallel PIC approaches. *Concurrency: Pract. Ex.*, 9(12):1377–1405, 1997.

[47] N. Carriero and D. Gelernter. Linda in context. *Commun. ACM*, 32(4):444–458, 1989.

[48] S. Chandrasekhar. Stochastic problems in physics and astronomy. *Rev. Mod. Phys.*, 15:1–89, 1943.

[49] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines. Tech. note, National Taiwan University, Department of Computer Science and Information Engineering, March 2003. http://www.csie.ntu.edu.tw/∼cjlin.

[50] A. K. Chaniotis, D. Poulikakos, and P. Koumoutsakos. Remeshed smoothed particle hydrodynamics for the simulation of viscous and heat conducting flows. *J. Comput. Phys.*, 182(1):67–90, 2002.

[51] M. A. J. Chaplain, M. Ganesh, and I. G. Graham. Spatio-temporal pattern formation on spherical surfaces: numerical simulation and application to solid tumour growth. *J. Math. Biol.*, 42:387–423, 2001.

[52] S. R. Chapple and L. J. Clarke. The parallel utilities library. In *Proceedings of the IEEE Scalable Parallel Libraries Conference*, pages 21–30. IEEE, 1994.

[53] M. K. Cheezum, W. F. Walker, and W. H. Guilford. Quantitative comparison of algorithms for tracking single fluorescent particles. *Biophys. J.*, 81:2378–2388, 2001.

[54] V. S. Cherkassky and F. Mulier. *Learning from data*. J. Wiley & Sons, 1998.

[55] D. Chetverikov and J. Verestóy. Feature point tracking for incomplete trajectories. *Computing*, 62(4):321–338, 1999.

[56] D. L. Chopp. Some improvements on the fast marching method. *SIAM J. Sci. Comput.*, 23(1):230–244, 2001.

[57] A. J. Chorin. Numerical study of slightly viscous flow. *J. Fluid Mech.*, 57(4):785–796, 1973.

[58] M. Christensen. How to simulate anisotropic diffusion processes on curved surfaces. *J. Comput. Phys.*, 201:421–438, 2004.

[59] J. P. Christiansen. Numerical simulation of hydrodynamics by the method of point vortices. *J. Comput. Phys.*, 13:363–379, 1973.

[60] N. B. Cole, C. L. Smith, N. Sciaky, M. Terasaki, M. Edidin, and J. Lippincott-Schwartz. Diffusional mobility of Golgi proteins in membranes of living cells. *Science*, 273(5276):797–801, 1996.

[61] W. J. Conover, M. E. Johnson, and M. M. Johnson. A comparative study of tests for homogeneity of variances, with applications to the outer continental-shelf bidding data. *Technometrics*, 23(4):351–361, 1981.

[62] C. Cornwell and L. Wille. Parallel molecular dynamics for short-ranged many-body potentials. *Comp. Phys. Commun.*, 128:477–491, 2000.

[63] G.-H. Cottet and P. Koumoutsakos. *Vortex Methods – Theory and Practice*. Cambridge University Press, New York, 2000.

[64] G.-H. Cottet, P. Koumoutsakos, and M. L. Ould Salihi. Vortex methods with spatially varying cores. *J. Comput. Phys.*, 162:164–185, 2000.

[65] G.-H. Cottet and P. Poncet. Advances in direct numerical simulations of 3D wall-bounded flows by Vortex-in-Cell methods. *J. Comput. Phys.*, 193(1):136–158, 2004.

[66] R. Couturier and C. Chipot. Parallel molecular dynamics using OpenMP on a shared memory machine. *Comp. Phys. Commun.*, 124:49–59, 2000.

[67] CPMD V3.7. © IBM Corp. 1990-2003, © MPI für Festkörperforschung Stuttgart 1997-2001.

[68] J. C. Crocker and D. G. Grier. Methods of digital video microspopy for colloidal studies. *J. Colloid Interf. Sci.*, 179:298–310, 1996.

[69] S. S. Cross. The application of fractal geometric analysis to microscopic images. *Micron*, 25:101–113, 1994.

[70] M. Dahan, S. Lévi, C. Luccardini, P. Rostaing, B. Riveau, and T. Antoine. Diffusion dynamics of glycine receptors revealed by single-quantum dot tracking. *Science*, 302(5644):442–445, 2003.

[71] S. B. Dalziel. Decay of rotating turbulence: some particle tracking experiments. *Appl. Scien. Res.*, 49:217–244, 1992.

[72] S. B. Dalziel. Decay of rotating turbulence: some particle tracking experiments. In F. T. M. Nieuwstadt, editor, *Flow Visualization and Image Analysis*, pages 27–54. Kluwer, Dordrecht, 1993.

[73] S. B. Dalziel. Rayleigh-Taylor instability: experiments with image analysis. *Dyn. Atmos. Oceans*, 20:127–153, 1993.

[74] E. A. Dari, G. C. Buscaglia, and A. J. Lew. A parallel general purpose finite element system. In *IX SIAM Conference on Parallel Processing for Scientific Computing*, 1999.

[75] M. J. Dayel, E. F. Hom, and A. S. Verkman. Diffusion of green fluorescent protein in the aqueous-phase lumen of the endoplasmic reticulum. *Biophys. J.*, 76:2843–2851, 1999.

[76] M. De Brabander, G. Geuens, R. Nuydens, M. Moeremans, and J. De Mey. Probing microtubule-dependent intracellular motility with nanometre particle video ultramicroscopy (nanovid ultramicroscopy). *Cytobios.*, 43(174S):273–283, 1985.

[77] C. De Duve. *Une visite guidée de la cellule vivante*. Pour la Science, 1987.

[78] V. K. Decyk. Skeleton PIC codes for parallel computers. *Comp. Phys. Commun.*, 87:87–94, 1995.

[79] P. Degond and S. Mas-Gallic. The weighted particle method for convection-diffusion equations. Part 1: The case of an isotropic viscosity. *Math. Comput.*, 53(188):485–507, 1989.

[80] P. Degond and S. Mas-Gallic. The weighted particle method for convection-diffusion equations. Part 2: The anisotropic case. *Math. Comput.*, 53(188):509–525, October 1989.

[81] R. Deiterding. Construction and application of an AMR algorithm for distributed memory computers. *Lect. Notes Computational Sci. Engrg.*, 41:361–372, 2005.

[82] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Stat. Soc.*, 39(1):1–38, 1977.

[83] Y. Deng, R. F. Peierls, and C. Rivera. An adaptive load balancing method for parallel molecular dynamics simulations. *J. Comput. Phys.*, 161:250–263, 2000.

[84] B. Di Martino, S. Briguglio, G. Vlad, and P. Sguazzero. Parallel PIC plasma simulation through particle decomposition techniques. *Parallel Comput.*, 27:295–314, 2001.

[85] G. L. Djordjević and M. B. Tošić. A heuristic for scheduling task graphs with communication delays onto multiprocessors. *Parallel Comput.*, 22:1197–1214, 1996.

[86] J. Dubinski. A parallel tree code. *New Astron.*, 1:133–147, 1996.

[87] D. Durand, R. Jain, and D. Tseytlin. Parallel I/O scheduling using randomized, distributed edge coloring algorithms. *J. Parallel Distrib. Comput.*, 63:611–618, 2003.

[88] M. Edidin, S. C. Kuo, and M. P. Sheetz. Lateral movements of membrane glycoproteins restricted by dynamic cytoplasmic barriers. *Science*, 254(5036):1379–1382, 1991.

[89] M. Edidin, Y. Zagyansky, and T. J. Lardner. Measurement of membrane protein lateral diffusion in single cells. *Science*, 191:466–468, 1976.

[90] M. Edidin, M. C. Zuniga, and M. P. Sheetz. Truncation mutants define and locate cytoplasmic barriers to lateral mobility of membrane–glycoproteins. *Proc. Natl. Acad. Sci. USA*, 91(8):3378–3382, 1994.

[91] R. Eils and C. Athale. Computational imaging in cell biology. *J. Cell Biol.*, 161(3):477–481, 2003.

[92] A. Einstein. Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen. *Annalen der Physik*, 17:549–560, 1905.

[93] J. D. Eldredge, A. Leonard, and T. Colonius. A general deterministic treatment of derivatives in particle methods. *J. Comput. Phys.*, 180:686–709, 2002.

[94] J. Ellenberg, E. D. Siggia, J. E. Moreira, C. L. Smith, J. F. Presley, H. J. Worman, and J. Lippincott-Schwartz. Nuclear membrane dynamics and reassembly in living cells: Targeting of an inner nuclear membrane protein in interphase and mitosis. *J. Cell Biol.*, 138(6):1193–1206, 1997.

[95] K. Esselink, B. Smit, and P. A. J. Hilbers. Efficient parallel implementation of molecular-dynamics on a toroidal network. Part I. Parallelizing strategy. *J. Comput. Phys.*, 106:101–107, 1993.

[96] H. Ewers, A. E. Smith, I. F. Sbalzarini, H. Lilie, P. Koumoutsakos, and A. Helenius. Single-particle tracking of murine polyoma virus-like particles on live cells and artificial membranes. *Proc. Natl. Acad. Sci. USA*, 102(42):15110–15115, 2005.

[97] K. J. Falconer. *Techniques in Fractal Geometry*. John Wiley & Sons, Chichester, UK, 1997.

[98] R. D. Falgout, J. E. Jones, and U. M. Yang. The design and implementation of Hypre, a library of parallel high performance preconditioners. Technical report UCRL-JRNL-205459, Lawrence Livermore National Laboratory, 2004.

[99] T. J. Feder, I. Brust-Mascher, J. P. Slattery, B. Baird, and W. W. Webb. Constrained diffusion or immobile fraction on cell surfaces: a new interpretation. *Biophys. J.*, 70:2767–2773, 1996.

[100] D. P. Felsenfeld, P. L. Schwartzberg, A. Venegas, R. Tse, and M. P. Sheetz. Selective regulation of integrin-cytoskeleton interactions by the tyrosine kinase Src. *Nature Cell Biol.*, 1(4):200–206, 1999.

[101] R. Ferrari, A. J. Manfroi, and W. R. Young. Strongly and weakly self-similar diffusion. *Physica D*, 154:111–137, 2001.

[102] R. A. Fisher. The advance of advantageous genes. *Ann. Eugenics*, 7:335–369, 1937.

[103] G. D. Forney. The Viterbi algorithm. *Proc. IEEE*, 61:268–278, 1973.

[104] R. Fraile and S. J. Maybank. Vehicle trajectory approximation and classification. Report, University of Reading, Dept. of Computer Science, 1998.

[105] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, J. A. Montgomery, Jr., T. Vreven, K. N. Kudin, J. C. Burant, J. M. Millam, S. S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G. A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J. E. Knox, H. P. Hratchian, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, P. Y. Ayala, K. Morokuma, G. A. Voth, P. Salvador, J. J. Dannenberg, V. G. Zakrzewski, S. Dapprich, A. D. Daniels, M. C. Strain, O. Farkas, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. V. Ortiz, Q. Cui, A. G. Baboul, S. Clifford, J. Cioslowski, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, C. Gonzalez, and J. A. Pople. Gaussian 03, Revision C.02, 2004. Gaussian, Inc., Wallingford, CT, www.gaussian.com.

[106] T. Fujiwara, K. Ritchie, H. Murakoshi, K. Jacobson, and A. Kusumi. Phospholipids undergo hop diffusion in compartmentalized cell membrane. *J. Cell Biol.*, 157(6):1071–1081, 2002.

[107] C. J. Gallione and J. K. Rose. A single amino acid substitution in a hydrophobic domain causes temperature-sensitive cell-surface transport of a mutant viral glycoprotein. *J. Virol.*, 54:374–382, 1985.

[108] J. Gelles, B. J. Schnapp, and M. P. Sheetz. Tracking kinesin-driven movements with nanometre-scale precision. *Nature*, 331:450–453, 1998.

[109] A. Ghoniem and O. Knio. The development and application of the transport element method to three dimensional reacting shear layers. In *Vortex Dynamics and Vortex Methods*, pages 165–218. American Mathematical Society, 1991.

[110] R. N. Ghosh and W. W. Webb. Automated detection and tracking of individual and clustered cell surface low density lipoprotein receptor molecules. *Biophys. J.*, 66(5):1302–1318, 1994.

[111] A. Gierer and H. Meinhardt. A theory of biological pattern formation. *Kybernetik*, 12:30–39, 1972.

[112] J. Gilbert and T. Benjamin. Uptake pathway of polyomavirus via ganglioside GD1a. *J. Virol.*, 78:12259–12267, 2004.

[113] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comput. Phys.*, 22:403–434, 1976.

[114] H. Gish and K. Ng. Parametric trajectory models for speech recognition. In *Proc. Fourth International Conference on Spoken Language (ICSLP 96)*, volume 1, pages 466–469. IEEE, 1996.

[115] S. Gleiter and H. Lilie. Coupling of antibodies via protein Z on modified polyoma virus-like particles. *Protein Sci.*, 10:434–444, 2001.

[116] M. Goulian and S. M. Simon. Tracking single proteins within cells. *Biophys. J.*, 79:2188–2198, 2000.

[117] A. Grama, V. Kumar, and A. Sameh. Scalable parallel formulation of the Barnes-Hut method for n-body simulations. *Parallel Comput.*, 24:797–822, 1998.

[118] J. Grasman, J. W. Brascamp, J. L. Van Leeuwen, and B. Van Putten. The multifractal structure of arterial trees. *J. Theor. Biol.*, 220:75–82, 2003.

[119] W. G. Gray. On the theorems for local volume averaging of multiphase systems. *Int. J. Multiphase Flow*, 3:333–340, 1977.

[120] D. M. Green and J. A. Sweets. *Signal detection theory and psychophysics*. Krieger, New York, 1966.

[121] L. Greengard and W. D. Gropp. A parallel version of the fast multipole method. *Computers Math. Applic.*, 20(7):63–71, 1990.

[122] L. Greengard and V. Rokhlin. The rapid evaluation of potential fields in three dimensions. *Lect. Notes Math.*, 1360:121–141, 1988.

[123] H. P. Grimm, A. B. Verkhovsky, A. Mogilner, and J.-J. Meister. Analysis of actin dynamics at the leading edge of crawling cells: implications for the shape of keratocyte lamellipodia. *Eur. Biophys. J.*, 32:563–577, 2003.

[124] J. Haber, F. Zeilfelder, O. Davydov, and H.-P. Seidel. Smooth approximation and renderung of large scattered data sets. In *IEEE Visualization*, pages 341–347. IEEE, 2001.

[125] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of the 1996 IEEE Conference on Evolutionary Computation (ICEC '96)*, pages 312–317, 1996.

[126] M. Hao, S. Mukherjee, and F. R. Maxfield. Cholesterol depletion induces large scale domain segregation in living cell membranes. *Proc. Natl. Acad. Sci. USA*, 98:13072–13077, 2001.

[127] F. H. Harlow. Particle-in-cell computing method for fluid dynamics. *Methods Comput. Phys.*, 3:319–343, 1964.

[128] L. G. Harrison, S. Wehner, and D. M. Holloway. Complex morphogenesis of surfaces: theory and experiment on coupling of reaction-diffusion patterning to growth. *Faraday Discuss.*, 120:277–294, 2001.

[129] J. Hartigan and M. Wong. A k-means clustering algorithm. *Appl. Stat.*, 28:100–108, 1979.

[130] F. Hedman and A. Laaksonen. Data parallel large-scale molecular dyamics for liquids. *Int. J. Quant. Chem.*, 46:27–38, 1993.

[131] G. S. Heffelfinger. Parallel atomistic simulation. *Comp. Phys. Commun.*, 128:219–237, 2000.

[132] A. Helenius. Biochemistry II. Lecture notes, ETH Zürich 2001.

[133] M. Herrmann. A domain dcomposition parallelization of the fast marching method. In *Center for Turbulence Research Annual Research Briefs*, pages 213–225. CTR Stanford University, 2003.

[134] M. Herrmann. On mass conservation and desingularization of the level set/vortex sheet method. In *Center for Turbulence Research Annual Research Briefs*, pages 15–30. CTR Stanford University, 2004.

[135] F. L. Hichcock. The distribution of a product from several sources to numerous localities. *J. Math. Phys.*, 20:224, 1941.

[136] B. W. Hicks and K. J. Angelides. Tracking movements of lipids and Thy1 molecules in the plasmalemma of living fibroblasts by fluorescence video microscopy with nanometer scale precision. *J. Membr. Biol.*, 144(3):231–244, 1995.

[137] S. E. Hieber and P. Koumoutsakos. A Lagrangian particle level set method. *J. Comput. Phys.*, 210:342–367, 2005.

[138] R. W. Hockney and J. W. Eastwood. *Computer Simulation using Particles*. Institute of Physics Publishing, 1988.

[139] I. Holyer. The NP-completeness of edge-coloring. *SIAM J. Comput.*, 10:718–720, 1981.

[140] R. Holyst, D. Plewczyński, A. Aksimentiev, and K. Burdzy. Diffusion on curved periodic surfaces. *Phys. Rev. B*, 60(1):302–307, 1999.

[141] M. S. Horwitz. *Virology. Vol. 1.*, chapter Adenoviruses, pages 1723–1740. Raven Press, New York, 1990.

[142] R. Iino, I. Koyama, and A. Kusumi. Single molecule imaging of green fluorescent proteins in living cells: E-cadherin forms oligomers on the free cell surface. *Biophys. J.*, 80:2667–2677, 2001.

[143] A. G. Ivakhnenko, S. F. Kozubovski, and Y. V. Kostenko. Objective system analysis of macroeconomic systems. *Syst. Anal. Model. Simul.*, 1(3):201–206, 1990.

[144] A. G. Ivakhnenko and J.-A. Müller. Recent developments of self-organising modeling in prediction and analysis of stock markets. Technical report, Glushkov Inst. Cybernetics, Kyiv, Ukraina, 1996.

[145] K. Jacobson, A. Ishihara, and R. Inman. Lateral diffusion of proteins in membranes. *Ann. Rev. Physiol.*, 49:163–175, 1987.

[146] A. K. Jain. *Fundamentals of Image Processing*. Prentice Hall, Englewood Cliffs, NJ, 1986.

[147] O. C. Jenkins, M. J. Matarić, and S. Weber. Primitive-based movement classification for humanoid imitation. In *Proc. IEEE-RAS Intl. Conf. on Humanoid Robotics (Humanoids-2000)*, 2000.

[148] G.-S. Jiang and D. Peng. Weighted ENO schemes for Hamilton-Jacobi equations. *SIAM J. Sci. Comput.*, 21(6):2126–2143, 2000.

[149] H. Jönsson, M. Heisler, G. V. Reddy, V. Agrawal, V. Gor, B. E. Shapiro, E. Mjolsness, and E. M. Meyerowitz. Modeling the organization of the WUSCHEL expression domain in the shoot apical meristem. *Bioinformatics*, 21:i232–i240, 2005.

[150] B. H. Juang, S. E. Levinson, and M. M. Sondhi. Maximum likelihood estimation for multivariate mixture observations of Markov chains. *IEEE Trans. Informat. Theory*, IT–32(2):307–309, 1985.

[151] R. K. Kalia, T. J. Champbell, A. Chatterjee, A. Nakano, P. Vashishta, and S. Ogata. Multiresolution algorithms for massively parallel molecular dynamics simulations of nanostructured materials. *Comp. Phys. Commun.*, 128:245–259, 2000.

[152] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.

[153] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *J. Par. Dist. Comp.*, 48:96–129, 1998.

[154] P. Keller, D. Toomre, E. Diaz, J. White, and K. Simons. Multicolour imaging of post-Golgi sorting and trafficking in live cells. *Nat. Cell Biol.*, 3(2):140–149, 2001.

[155] A. K. Kenworthy, B. J. Nichols, C. L. Remmert, G. M. Hendrix, M. Kumar, J. Zimmerberg, and J. Lippincott-Schwartz. Dynamics of putative raft-associated proteins at the cell surface. *J. Cell Biol.*, 165:735–746, 2004.

[156] S. Kim. An $\mathcal{O}(N)$ level set method for Eikonal equations. *SIAM J. Sci. Comput.*, 22(6):2178–2193, 2001.

[157] M. Kinder and W. Brauer. Classification of trajectories – extracting invariants with a neural network. *Neural Networks*, 6(7):1011–1017, 1993.

[158] T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, 3rd edition, 2001.

[159] A. N. Kolmogorov, I. G. Petrovsky, and N. S. Piskunov. Étude de l'équation de la diffusion avec croissance de la quantité de matière et son application à un problème biologique. *Bulletin Université d'Etat à Moscou (Bjul. Moskowskogo Gos. Univ.)*, Série internationale A 1:1–26, 1937.

[160] P. Koumoutsakos. Multiscale flow simulations using particles. *Annu. Rev. Fluid Mech.*, 37:457–487, 2005.

[161] P. Koumoutsakos, A. Leonard, and F. Pépin. Boundary conditions for viscous vortex methods. *J. Comput. Phys.*, 113:52–61, 1994.

[162] J. Kwik, S. Boyle, D. Fooksman, L. Margolis, M. P. Sheetz, and M. Edidin. Membrane cholesterol, lateral mobility, and the phosphatidylinositol 4,5-bisphosphate-dependent organization of cell actin. *Proc. Natl. Acad. Sci. USA*, 100:13964–13969, 2003.

[163] T. Lahiri, A. Chakrabarti, and A. K. Dasgupta. Multilamellar vesicular clusters of phosphatidylcholine and their sensitivity to spectrin: A study by fractal analysis. *J. Struct. Biol.*, 123:179–186, 1998.

[164] L. D. Landau and E. M. Lifshitz. *Statistical Physics*. Pergamon Press, Oxford, 1969.

[165] F. Lerasle, G. Rives, and M. Dhome. Tracking of human limbs by multiocular vision. *Computer Vision and Image Understanding*, 75(3):229–246, 1999.

[166] P. Lévy. *Théorie de l'addition des variables aléatoires*. Gauthier-Villars, Paris, 1937.

[167] H. Li, S. Chen, and H. Zhao. Fat fractal and multifractals for protein and enzyme surfaces. *Intl. J. Biol. Macromolecules*, 13(4):210–216, 1991.

[168] H. Li, Y. Li, and H. Zhao. Fractal analysis of protein chain conformation. *Intl. J. Biol. Macromolecules*, 12(1):6–8, 1990.

[169] L. S. Liebovitch, D. Scheurle, M. Rusek, and M. Zochowski. Fractal methods to analyze ion channel kinetics. *Methods*, 24:359–375, 2001.

[170] K.-T. Lim, S. Brunett, M. Iotov, R. B. McClurg, N. Vaidehi, S. Dasgupta, S. Taylor, and W. A. Goddard III. Molecular dynamics for very large systems on massively parallel computers: The MPSim program. *J. Comput. Chem.*, 18(4):501–521, 1997.

[171] C. S. Lin, A. L. Thring, J. Koga, and E. J. Seiler. A parallel particle-in-cell model for the massively parallel processor. *J. Par. Dist. Comp.*, 8:196–199, 1990.

[172] K. Lipkow, S. S. Andrews, and D. Bray. Simulated diffusion of phosphorylated CheY through the cytoplasm of *Escherichia coli*. *J. Bacteriol.*, 187(1):45–53, 2005.

[173] L. A. Liporace. Maximum likelihood estimation for multivariate observations of Markov sources. *IEEE Trans. Informat. Theory*, IT–28(5):729–734, 1982.

[174] J. Lippincott-Schwartz, E. Snapp, and A. Kenworthy. Studying protein dynamics in living cells. *Nature Rev. Mol. Cell Biol.*, 2:444–456, 2001.

[175] P. H. M. Lommerse, G. A. Blab, L. Cognet, G. S. Harms, B. E. Snaar-Jagalska, H. P. Spaink, and T. Schmidt. Single-molecule imaging of the H-Ras membrane-anchor reveals domains in the cytoplasmic leaflet of the cell membrane. *Biophys. J.*, 86:609–616, 2001.

[176] Q. M. Lu and D. S. Cai. Implementation of parallel plasma particle-in-cell codes on PC cluster. *Comp. Phys. Commun.*, 135:93–104, 2001.

[177] C. Luedeke, S. Buvelot Frei, I. Sbalzarini, H. Schwarz, A. Spang, and Y. Barral. Septin-dependent compartmentalization of the endoplasmic reticulum during yeast polarized growth. *J. Cell Biol.*, 169(6):897–908, 2005.

[178] A. Lyubartsev and A. Laaksonen. Parallel molecular dynamics simulations of biomolecular systems. *Lect. Notes Comput. Sc.*, 1541:296–303, 1998.

[179] T. MacFarland, H. M. P. Couchman, F. R. Pearce, and J. Pichlmeier. A new parallel P^3M code for very large-scale cosmological simulations. *New Astron.*, 3:687–705, 1998.

[180] P. Macklin and J. Lowengrub. Evolving interfaces via gradients of geometry-dependent interior Poisson problems: application to tumor growth. *J. Comput. Phys.*, 203:191–220, 2005.

[181] A. Madzvamuse, A. J. Wathen, and P. K. Maini. A moving grid finite element method applied to a model biological pattern generator. *J. Comput. Phys.*, 190:478–500, 2003.

[182] S. Maes, K. Tuyls, B. Vanschoenwinkel, and B. Manderick. Credit card fraud detection using Bayesian and neural networks. In *Proceedings of NF2002*, Havana Cuba, 2002.

[183] D. Makris and T. Ellis. Spatial and probabilistic modeling of pedestrian behaviour. In *British Machine Conference (BMVC)*, pages 557–566, 2002.

[184] B. B. Mandelbrot. *The fractal geometry of nature*. W. H. Freeman & Co., San Francisco, 1982.

[185] A. F. M. Marée. *From Pattern Formation to Morphogenesis*. PhD thesis, University of Utrecht, 2000.

[186] D. Marguet, E. T. Spiliotis, T. Pentcheva, M. Lebowitz, J. Schneck, and M. Edidin. Lateral diffusion of GFP-tagged $H2L^d$ molecules and of GFP–TAP1. reports on the assembly and retention of these molecules in the endoplasmic reticulum. *Immunity*, 11:231–240, 1999.

[187] D. S. Martin, M. B. Forstner, and J. A. Käs. Apparent subdiffusion inherent to single particle tracking. *Biophys. J.*, 83:2109–2117, 2002.

[188] W. Marwan. Theory of time-resolved complementation and its use to explore the sporulation control network in *Physarum polycephalum*. *Genetics*, 164:105–115, 2003.

[189] D. Marx and J. Hutter. Ab initio molecular dynamics: Theory and implementation. In J. Grotendorst, editor, *Modern Methods and Algorithms of Quantum Chemistry, Proceedings*, pages 329–477, Jülich, 2000. John von Neumann Institute for Computing.

[190] Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics*. Springer, 3rd edition, 2002.

[191] F. Milizzano and P. G. Saffman. The calculation of large Reynolds number two-dimensional flow using discrete vortices with random walk. *J. Comput. Phys.*, 23:380–392, 1977.

[192] S. Miller and S. Luding. Event-driven molecular dynamics in parallel. *J. Comput. Phys.*, 193:306–316, 2003.

[193] T. Miura and P. K. Maini. Speed of pattern appearance in reaction-diffusion models: Implications in the pattern formation of limb bud mesenchyme cells. *Bull. Math. Biol.*, 66:627–649, 2004.

[194] J. J. Monaghan. Extrapolating B splines for interpolation. *J. Comput. Phys.*, 60:253–262, 1985.

[195] B. Moon and J. Saltz. Adaptive runtime support for direct simulation Monte Carlo methods on distributed memory architectures. In *Proceedings of the IEEE Scalable High-Performance Computing Conference*, pages 176–183. IEEE, 1994.

[196] A. D. Morgan, E. L. Dagless, D. J. Milford, and B. T. Thomas. Road edge tracking for robot road following: a real-time implementation. *Image Vision Comput.*, 8(3):233–240, 1990.

[197] P. M. Morse and H. Feshbach. *Methods of Theoretical Physics, Part I*. McGraw-Hill, New York, 1953.

[198] S. Mukherjee, E. Osuna, and F. Girosi. Nonlinear prediction of chaotic time series using a support vector machine. In J. Principe, L. Gile, N. Morgan, and E. Wilson, editors, *Neural Networks for Signal Processing VII – Proceedings of the 1997 IEEE Workshop*, pages 511–520. IEEE, 1997.

[199] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Trans. Neural Networks*, 12(2):181–202, 2001.

[200] F. Müller-Plathe, W. Scott, and W. F. van Gunsteren. PARALLACS: a benchmark for parallel molecular dynamics. *Comp. Phys. Commun.*, 84:102–114, 1994.

[201] S. Munro and H. R. Pelham. A C-terminal signal prevents secretion of luminal ER proteins. *Cell*, 48:899–907, 1987.

[202] S. B. Nadler Jr. *Continuum Theory*. M. Dekker, Basel and Hong Kong, 1992.

[203] S. Nehls, E. L. Snapp, N. B. Cole, K. J. M. Zaal, A. K. Kenworthy, T. H. Roberts, J. Ellenberg, J. F. Presley, E. Siggia, and J. Lippincott-Schwartz. Dynamics and retention of misfolded proteins in native ER membranes. *Nature Cell Biol.*, 2:288–295, 2000.

[204] L. Neiberg and D. P. Casasent. Feature space trajectory (FST) classifier neural network. In D. P. Casasent, editor, *Proc. SPIE Intelligent Robots and Computer Vision XIII: Algorithms and Computer Vision*, volume 2353, pages 276–292. SPIE, 1994.

[205] T. J. Newman, J. Antonovics, and H. M. Wilbur. Population dynamics with a refuge: Fractal basins and the suppression of chaos. *Theor. Popul. Biol.*, 62:121–128, 2002.

[206] C. Nieter and J. R. Cary. VORPAL: a versatile plasma simulation code. *J. Comput. Phys.*, 196(2):448–473, 2004.

[207] B. P. Ölveczky and A. S. Verkman. Monte Carlo analysis of obstructed diffusion in three dimensions: Application to molecular diffusion in organelles. *Biophys. J.*, 74:2722–2730, 1998.

[208] J. Owens and A. Hunter. Application of the self-organizing map to trajectory classification. In *Third IEEE International Workshop on Visual Surveillance (VS'2000)*, pages 77–83, 2000.

[209] C. Pan, J. F. Prins, and C. T. Miller. A high-performance lattice Boltzmann implementation to model flow in porous media. *Comp. Phys. Commun.*, 158:89–105, 2004.

[210] A. Partikian, B. Ölveczky, R. Swaminathan, Y. Li, and A. S. Verkman. Rapid diffusion of green fluorescent protein in the mitochondrial matrix. *J. Cell Biol.*, 140:821–829, 1998.

[211] F. R. Pearce and H. M. P. Couchman. Hydra: a parallel adaptive grid code. *New Astron.*, 2:411–427, 1997.

[212] L. Pelkmans, J. Kartenbeck, and A. Helenius. Caveolar endocytosis of simian virus 40 reveals a new two-step vesicular-transport pathway to the ER. *Nature Cell Biol.*, 3:473–483, 2001.

[213] L. Pelkmans, D. Püntener, and A. Helenius. Local actin polymerization and dynamin recruitment in SV40-induced internalization of caveolae. *Science*, 296:535–539, 2002.

[214] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. A PDE-based fast local level set method. *J. Comput. Phys.*, 155:410–438, 1999.

[215] R. Peters, A. Brünger, and K. Schulten. Continuous fluorescence microphotolysis: A sensitive method for study of diffusion processes in single cells. *Proc. Natl. Acad. Sci. USA*, 78(2):962–966, 1981.

[216] R. D. Phair and T. Misteli. High mobility of proteins in the mammalian cell nucleus. *Nature*, 404:604–609, 2000.

[217] C. D. Pierce and P. Moin. Progress-variable approach for large-eddy simulation of non-premixed turbulent combustion. *J. Fluid Mech.*, 504:73–97, 2004.

[218] K. M. Pillai. Governing equations for unsaturated flow through woven fiber mats. Part 1. Isothermal flows. *Composites A*, 33:1007–1019, 2002.

[219] S. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *J. Comput. Phys.*, 117:1–19, 1995.

[220] S. J. Plimpton, M. F. Seidel, D. B. Pasik, R. S. Coats, and G. R. Montry. A load-balancing algorithm for a parallel electromagnetic particle-in-cell code. *Comp. Phys. Commun.*, 152:227–241, 2003.

[221] S. J. Plimpton and A. Slepoy. Microbial cell modeling via reacting diffusive particles. *J. Phys.: Conference Series*, 16:305–309, 2005.

[222] M. Pütz and A. Kolb. Optimization techniques for parallel molecular dynamics using domain decomposition. *Comp. Phys. Commun.*, 113:145–167, 1998.

[223] H. Qian, M. P. Sheetz, and E. L. Elson. Single particle tracking. Analysis of diffusion and flow in two-dimensional systems. *Biophys. J.*, 60(4):910–921, 1991.

[224] J. Qiang, R. D. Ryne, S. Habib, and V. Decyk. An object-oriented parallel particle-in-cell code for beam dynamics simulation in linear accelerators. *J. Comput. Phys.*, 163:434–451, 2000.

[225] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, 1989.

[226] D. C. Rapaport. Large-scale molecular dynamics simulations using vector and parallel computers. *Comput. Phys. Rep.*, 9:1–53, 1988.

[227] E. A. Reits and J. J. Neefjes. From fixed to FRAP: measuring protein mobility and activity in living cells. *Nature Cell Biol.*, 3:E145–E147, 2001.

[228] A. Renyi. *Probability Theory*. North-Holland, Amsterdam, 1970.

[229] J. Rink, E. Ghigo, Y. Kalaidzidis, and M. Zerial. Rab conversion as a mechanism of progression from early to late endosomes. *Cell*, 122:735–749, 2005.

[230] J. Riordan, C. R. Doering, and D. ben Avraham. Fluctuations and stability of Fisher waves. *Phys. Rev. Lett.*, 75(3):565–568, 1995.

[231] D. Roccatano, R. Bizzarri, G. Chillemi, A. Sanna, and N. Di Nola. Development of a parallel molecular dynamics code for SIMD computers: Algorithm for use of pair list criterion. *J. Comput. Chem.*, 19(7):685–694, 1998.

[232] D. Rumelhart, G. Hinton, and R. Williams. *Parallel Distributed Processing: Explorations in the microstructure of cognition. Volume 1: Foundations*, chapter Learning internal representations by error propagation. MIT Press, 1986.

[233] T. A. Ryan, P. J. Millard, and W. W. Webb. Imaging [Ca^{2+}] dyanmics during signal transduction. *Cell Calcium.*, 11:145–155, 1990.

[234] P. G. Saffman and M. Delbrück. Brownian motion in biological membranes. *Proc. Natl. Acad. Sci. USA*, 72:3111–3113, 1975.

[235] Y. Sako and A. Kusumi. Compartmentalized structure of the plasma membrane for receptor movements as revealed by a nanometer-level motion analysis. *J. Cell Biol.*, 125(6):1251–1264, 1994.

[236] J. K. Salmon, M. S. Warren, and G. S. Winckelmans. Fast parallel tree codes for gravitational and fluid dynamical n-body problems. *Internat. J. Supercomput. Appl.*, 8(2):129–142, 1994.

[237] J. M. Sanchiz and F. Pla. Feature correspondence and motion recovery in vehicle planar navigation. *Pattern Recognition*, 32:1961–1977, 1999.

[238] C. Sas, G. O'Hare, and R. Reilly. Online trajectory classification. *Lect. Notes Comput. Sc.*, 2659:1035–1044, 2003.

[239] M. J. Saxton. Lateral diffusion in a mixture of mobile and immobile particles. *Biophys. J.*, 58:1303–1306, 1990.

[240] M. J. Saxton. Anomalous diffusion due to obstacles: a Monte Carlo study. *Biophys. J.*, 66:394–401, 1994.

[241] M. J. Saxton. Anomalous subdiffusion in fluorescence photobleaching recovery: a Monte Carlo study. *Biophys. J.*, 81:2226–2240, 2001.

[242] M. J. Saxton and K. Jacobson. Single-particle tracking: Applications to membrane dynamics. *Annu. Rev. Biophys. Biomol. Struct.*, 26:373–399, 1997.

[243] I. F. Sbalzarini. On protein diffusion in the endoplasmic reticulum: A computational approach using particle simulations. Semester thesis, Institute of Computational Science, ETH Zürich, July 2001.

[244] I. F. Sbalzarini. Diffusion in the endoplasmic reticulum: Theoretical foundation, computer simulations, models. Diploma thesis, Institute of Computational Science, ETH Zürich, February 2002.

[245] I. F. Sbalzarini, A. Hayer, A. Helenius, and P. Koumoutsakos. Simulations of (an)isotropic diffusion on curved biological surfaces. *Biophys. J.*, 90(3):878–885, 2006.

[246] I. F. Sbalzarini, A. Mezzacasa, A. Helenius, and P. Koumoutsakos. Effects of organelle shape on fluorescence recovery after photobleaching. *Biophys. J.*, 89(3):1482–1492, 2005.

[247] I. F. Sbalzarini, J. Theriot, and P. Koumoutsakos. Machine learning for biological trajectory classification applications. In *Proceedings of the Summer Program 2002*, pages 305–316. Center for Turbulence Research, Stanford University/NASA Ames, 2002.

[248] I. F. Sbalzarini, J. H. Walther, M. Bergdorf, S. E. Hieber, E. M. Kotsalis, and P. Koumoutsakos. PPM – a highly efficient parallel particle-mesh library for the simulation of continuum systems. *J. Comput. Phys.*, in print, 2006.

[249] B. A. Scalettar and J. R. Abney. Molecular crowding and protein diffusion in biological membranes. *Comments Moll. Cell Biophys.*, 7:79–107, 1991.

[250] H. Scharr and D. Uttenweiler. 3D anisotropic diffusion filtering for enhancing noisy actin filament fluorescence images. *Lect. Notes Comput. Sc.*, 2191:69–75, 2001.

[251] B. Schölkopf and A. J. Smola. *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, Massachusetts, 2002.

[252] B. Schram, J. F. Tocanne, and A. Lopez. Influence of obstacles on lipid lateral diffusion: computer simulation of FRAP experiments and application to proteoliposomes and biomembranes. *Eur. Biophys. J.*, 23:337–348, 1994.

[253] P. Schwartz, D. Adalsteinsson, P. Colella, A. P. Arkin, and M. Onsum. Numerical computation of diffusion on a surface. *Proc. Natl. Acad. Sci. USA*, 102(32):11151–11156, 2005.

[254] N. Sciaky, J. Presley, C. Smith, K. J. M. Zaal, N. Cole, J. E. Moreira, M. Terasaki, E. Siggia, and J. Lippincott-Schwartz. Golgi tubule traffic and the effects of Brefeldin A visualized in living cells. *J. Cell Biol.*, 139(5):1137–1155, 1997.

[255] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Natl. Acad. Sci. USA*, 93:1591–1595, 1996.

[256] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, Cambridge, UK, 1999.

[257] J. Shadbolt and J. G. Taylor. *Neural networks and the financial markets: predicting, combining, and portfolio optimization*. Springer, 2002.

[258] M. P. Sheetz, M. Schindler, and D. E. Koppel. Lateral mobility of integral membrane-proteins is increased in spherocytic erythrocytes. *Nature*, 285:510–511, 1980.

[259] T. Shenk. *Fundamental Virology*, chapter Adenoviridae, pages 979–1016. Lippincott-Raven, New York, 1996.

[260] R. Shibata. *Biometrika*, volume 68, chapter An optimal selection of regression variables. 1968.

[261] C.-W. Shu and S. Osher. Efficient implementation of essentially nonoscillatory shock-capturing schemes. *J. Comput. Phys.*, 77(2):439–471, 1988.

[262] E. D. Siggia, J. Lippincott-Schwartz, and S. Bekiranov. Diffusion in inhomogeneous media: Theory and simulations applied to whole cell photobleach recovery. *Biophys. J.*, 79:1761–1770, 2000.

[263] K. Simons and E. Ikonen. Functional rafts in cell membranes. *Nature*, 387:569–572, 1997.

[264] R. C. Singleton. An algorithm for computing the mixed radix fast Fourier transform. *IEEE Trans. Audio Electroacoust.*, 17(2):93–103, 1969.

[265] G. D. Smith. *Numerical Solution of Partial Differential Equations: Finite Difference Methods*. Oxford Appl. Math. Comput. Sci. Ser., Oxford, 3rd edition, 1985.

[266] T. G. J. Smith, W. B. Marks, G. D. Lang, W. H. J. Sheriff, and E. A. Neal. A fractal analysis of cell images. *J. Neurosci. Meth.*, 27:173–180, 1989.

[267] F. S. Soo and J. A. Theriot. Large-scale quantitative analysis of sources of variation in the actin polymerization-based movement of Listeria monocytogenes. *Biophys. J.*, 89:703–723, 2005.

[268] M. Speidel, A. Jonas, and E. L. Florin. Three-dimensional tracking of fluorescent nanoparticles with subnanometer precision by use of off-focus imaging. *Optics Letters*, 28(2):69–71, 2003.

[269] K. R. Spring and M. W. Davidson. Depth of field and depth of focus. Technical report. www.microscopyu.com/articles/formulas/ formulasfielddepth.html.

[270] V. Springel, N. Yoshida, and S. D. M. White. GADGET: a code for collisionless and gasdynamical cosmological simulations. *New Astron.*, 6:79–117, 2001.

[271] W. A. Stahel. *Statistische Datenanalyse*. Friedr. Vieweg & Sohn, 1995.

[272] T. Stehle, Y. Yan, T. L. Benjamin, and S. C. Harrison. Structure of murine polyomavirus complexed with an oligosaccharide receptor fragment. *Nature*, 369:160–163, 1994.

[273] J. A. Steyer and W. Almers. Tracking single secretory granules in live chromaffin cells by evanescent-field fluorescence microscopy. *Biophys. J.*, 76:2262–2271, 1999.

[274] B. Storrie, R. Pepperkok, E. H. K. Stelzer, and T. E. Kreis. The intracellular mobility of a viral membrane glycoprotein by confocal microscope fluorescence recovery after photobleaching. *J. Cell Sci.*, 107:1309–1319, 1994.

[275] T. P. Straatsma, M. Philippopoulos, and J. A. McCammon. Nwchem: Exploiting parallelism in molecular simulations. *Comp. Phys. Commun.*, 128:377–385, 2000.

[276] M. Suomalainen, M. Y. Nakano, S. Keller, K. Boucke, R. P. Stidwill, and U. F. Greber. Microtubule-dependent minus and plus end-directed motilities are competing processes for nuclear targeting of adenovirus. *J. Cell Biol.*, 144(4):657–672, 1999.

[277] K. Suzuki, K. Ritchie, E. Kajikawa, T. Fujiwara, and A. Kusumi. Rapid hop diffusion of a G-protein-coupled receptor in the plasma membrane as revealed by single-molecule techniques. *Biophys. J.*, 88:3659–3680, 2005.

[278] R. Swaminathan, C. P. Hoang, and A. S. Verkman. Photochemical properties of green fluorescent protein GFP–S65T in solution and transfected CHO cells: analysis of cytoplasmic viscosity by GFP translational and rotational diffusion. *Biophys. J.*, 72:1900–1907, 1997.

[279] K. Takahashi, K. Yugi, K. Hashimoto, Y. Yamada, C. J. F. Pickett, and M. Tomita. Computational challenges in cell simulation: A software engineering approach. *IEEE Intelligent Systems*, pages 64–69, September/October 2002.

[280] M. Terasaki, L. B. Chen, and K. Fujiwara. Microtubules and the endoplasmic reticulum are highly interdependent structures. *J. Cell Biol.*, 103:1557–1568, 1986.

[281] M. Terasaki, L. A. Jaffe, G. R. Hunnicutt, and J. A. Hammer III. Structural change of the endoplasmic reticulum during fertilization: Evidence for loss of membrane continuity using the green fluorescent protein. *Dev. Biol.*, 179:320–328, 1996.

[282] R. Thakur, R. Bordawekar, A. Choudhary, R. Ponnusamy, and T. Singh. PASSION runtime library for parallel I/O. In *Proceedings of the IEEE Scalable Parallel Libraries Conference*, pages 119–128. IEEE, 1994.

[283] The MPEG Library. mpeglib v1.3.1.

[284] R. E. Thompson, D. R. Larson, and W. W. Webb. Precise nanometer localization analysis for individual fluorescent probes. *Biophys. J.*, 82(5):2775–2783, 2002.

[285] S. Thurner, N. Wick, R. Hanel, R. Sedivy, and L. Huber. Anomalous diffusion on dynamical networks: a model for interacting epithelial cell migration. *Physica A*, 320:475–484, 2003.

[286] TIFF Software. libtiff v3.5.7, © 1988–1997 Sam Leffler, © 1991–1997 Silicon Graphics, inc.

[287] D. Toomre and D. J. Manstein. Lighting up the cell surface with evanescent wave microscopy. *Trends Cell Biol.*, 11(7):298–303, 2001.

[288] D. Toomre, J. A. Steyer, P. Keller, W. Almers, and K. Simons. Fusion of constitutive membrane traffic with the cell surface observed by evanescent wave microscopy. *J. Cell Biol.*, 149(1):33–40, 2000.

[289] A.-K. Tornberg and B. Enquist. Numerical approximations of singular source terms in differential equations. *J. Comput. Phys.*, 200:462–488, 2004.

[290] U. Trottenberg, C. Oosterlee, and A. Schueller. *Multigrid*. Academic Press, San Diego, 2001.

[291] B. Tsai, J. M. Gilbert, T. Stehle, W. Lencer, T. L. Benjamin, and T. A. Rapaport. Gangliosides are receptors for murine polyoma virus and SV40. *EMBO J.*, 22:4346–4355, 2003.

[292] M. E. Tuckerman, D. A. Yarnem, S. O. Samuelson, A. L. Hughes, and G. J. Martyna. Exploiting multiple levels of parallelism in molecular dynamics based calculations via modern techniques and software paradigms on distributed memory computers. *Comp. Phys. Commun.*, 128:333–376, 2000.

[293] A. M. Turing. The chemical basis of morphogenesis. *Phil. Trans. R. Soc. London*, B237:37–72, 1952.

[294] G. Turk. Generating textures on arbitrary surfaces using reaction-diffusion. *Computer Graphics*, 25(4):289–298, 1991.

[295] G. Turkiyyah, D. Reed, and J. Yang. Fast vortex methods for predicting wind-induced pressures on buildings. *J. Wind Engng. & Ind. Aerod.*, 58:51–79, 1996.

[296] D. Uttenweiler, C. Weber, B. Jähne, R. H. A. Fink, and H. Scharr. Spatiotemporal anisotropic diffusion filtering to improve signal-to-noise ratios and object restoration in fluorescence microscopic image sequences. *J. Biomed. Optics*, 8(1):40–47, January 2003.

[297] R. Valdarnini. Parallelization of a treecode. *New Astron.*, 8:691–710, 2003.

[298] P. Vallotton, A. Ponti, C. M. Waterman-Storer, E. D. Salmon, and G. Danuser. Recovery, visualization, and analysis of actin and tubulin polymer flow in live cells: A fluorescent speckle microscopy study. *Biophys. J.*, 85:1289–1306, 2003.

[299] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.

[300] A. S. Verkman. Solute and macromolecule diffusion in cellular aqueous compartments. *Trends Biochem. Sci.*, 27(1):27–33, 2002.

[301] L. Verlet. Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. *Phys. Rev.*, 159(1):98–103, 1967.

[302] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. Informat. Theory*, IT–13:260–269, 1967.

[303] V. G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskret. Anal.*, 3:25–30, 1964. in Russian.

[304] M. Vrljic, S. Y. Nishimura, W. E. Moerner, and H. M. McConnell. Cholesterol depletion suppresses the translational diffusion of class II major histocompatibility complex proteins in the plasma membrane. *Biophys. J.*, 88(1):334–347, 2005.

[305] M. Wachsmuth, W. Waldeck, and J. Langowski. Anomalous diffusion of fluorescent probes inside living cell nuclei investigated by spatially-resolved fluorescence correlation spectroscopy. *J. Mol. Biol.*, 298:677–689, 2000.

[306] J. H. Walther. An influence matrix particle-particle particle-mesh algorithm with exact particle-particle correction. *J. Comput. Phys.*, 184:670–678, 2003.

[307] J. H. Walther and P. Koumoutsakos. Three-dimensional particle methods for particle laden flows with two-way coupling. *J. Comput. Phys.*, 167:39–71, 2001.

[308] J. H. Walther and G. Morgenthal. An immersed interface method for the vortex-in-cell algorithm. *J. Turbul.*, 3:1–9, 2002.

[309] Q. X. Wang. Variable order revised binary treecode. *J. Comput. Phys.*, 200:192–210, 2004.

[310] M. S. Warren and J. K. Salmon. A portable parallel particle program. *Comp. Phys. Commun.*, 87:266–290, 1995.

[311] E. R. Weeks and H. L. Swinney. Random walks and Lévy flights observed in fluid flows. *Nonlinear Sci. Today*, 8, 1998. Springer Verlag, New York.

[312] M. Weiss. Challenges and artifacts in quantitative photobleaching experiments. *Traffic*, 5:662–671, 2004.

[313] M. Weiss, M. Elsner, F. Kartberg, and T. Nilsson. Anomalous subdiffusion is a measure for cytoplasmic crowding in living cells. *Biophys. J.*, 87:3518–3524, 2004.

[314] M. Weiss, H. Hashimoto, and T. Nilsson. Anomalous protein diffusion in living cells as seen by fluorescence correlation spectroscopy. *Biophys. J.*, 84:4043–4052, 2003.

[315] T. Werder, J. H. Walther, and P. Koumoutsakos. Hybrid atomistic-continuum methods for the simulation of dense fluid flows. *J. Comput. Phys.*, 205(1):373–390, 2005.

[316] S. T. Wereley, L. Gui, and C. D. Meinhart. Advanced algorithms for microscale particle image velocimetry. *AIAA J.*, 40(6):1047–1055, 2002.

[317] J. White and E. Stelzer. Photobleaching GFP reveals protein dynamics inside live cells. *Trends Cell Biol.*, 9:61–65, 1999.

[318] J. H. Williamson. Low-storage Runge-Kutta schemes. *J. Comput. Phys.*, 35:48–56, 1980.

[319] B. Winckler, P. Forscher, and I. Mellman. A diffusion barrier maintains distribution of membrane proteins in polarized neurons. *Nature*, 397(6721):698–701, 1999.

[320] A. Wissink, R. Hornung, S. Kohn, S. Smith, and N. Elliott. Large scale parallel structured AMR calculations using the SAMRAI framework. In *Proc. SC01 Conf. High Perf. Network. & Comput. Denver CO*, November 10–16 2001.

[321] Y. Xiao, R. Chen, R. Shen, J. Sun, and J. Xu. Fractal dimension of exon and intron sequences. *J. Theor. Biol.*, 175:23–26, 1995.

[322] J.-J. Xu, Z. Li, J. Lowengrub, and H. Zhao. A level-set method for interfacial flows with surfactant. *J. Comput. Phys.*, 212:590–616, 2006.

[323] Y. Yacoob and L. S. Davis. Temporal multi-scale models for flow and acceleration. In *Proc. IEEE Conf. CVPR, Puerto Rico*, pages 921–927. IEEE, 1997.

[324] F. Yang, L. G. Moss, and G. N. Phillips Jr. The molecular structure of green fluorescent protein. Technical report, Dept. of Biochemistry and Cell Biology, Rice University, Houston, TX, 1997.

[325] I. Yavneh. On red-black SOR smoothing in multigrid. *SIAM J. Sci. Comput.*, 17:180–192, 1996.

[326] M. Yokokawa, K. Itakura, A. Uno, T. Ishihara, and Y. Kaneda. 16.4-Tflops direct numerical simulation of turbulence by a Fourier spectral method on the Earth Simulator. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–17. IEEE, 2002.

[327] I. T. Young. Quantitative microscopy. *IEEE Eng. Med. Bio.*, 15:59–66, 1996.

[328] X. Yuan, C. Salisbury, D. Balsara, and R. Melhem. A load balancing package on distributed memory systems and its application to particle-particle particle-mesh (P3M) methods. *Parallel Comput.*, 23:1525–1544, 1997.

[329] F. Zhang, B. Crise, B. Su, Y. Hou, J. K. Rose, A. Bothwell, and K. Jacobson. Lateral diffusion of membrane-spanning and glycosylphosphatidylinositol-linked proteins: toward establishing rules governing the lateral mobility of membrane proteins. *J. Cell Biol.*, 115(1):75–84, 1991.

[330] A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller. Engineering support vector machine kernels that recognize translation initiation sites in DNA. *Bioinformatics*, 16:799–807, 2000.

Own Publications

Parts of this thesis have been published in the following papers:

Refereed journal papers

1. I. F. Sbalzarini, J. H. Walther, M. Bergdorf, S. E. Hieber, E. M. Kotsalis, and P. Koumoutsakos. PPM – a highly efficient parallel particle-mesh library for the simulation of continuum systems. *J. Comput. Phys.*, in press, 2006.
2. I. F. Sbalzarini, A. Hayer, A. Helenius, and P. Koumoutsakos. Simulations of (an)isotropic diffusion on curved biological surfaces. *Biophys. J.* 90(3): 878–885, 2006.
3. H. Ewers, A. E. Smith, I. F. Sbalzarini, H. Lilie, P. Koumoutsakos, and A. Helenius. Single-particle tracking of murine polyoma virus-like particles on live cells and artificial membranes. *Proc. Natl. Acad. Sci. USA*, 102(42): 15110–15115, 2005.
4. I. F. Sbalzarini, A. Mezzacasa, A. Helenius, and P. Koumoutsakos. Effects of organelle shape on fluorescence recovery after photobleaching. *Biophys. J.* 89(3): 1482–1492, 2005.
5. I. F. Sbalzarini and P. Koumoutsakos. Feature point tracking and trajectory analysis for video imaging in cell biology. *J. Struct. Biol.*, 151(2): 182–195, 2005.
6. C. Luedeke, S. Buvelot-Frei, I. Sbalzarini, H. Schwarz, A. Spang, and Y. Barral. Septin-dependent compartmentalization of the endoplasmic reticulum during yeast polarized growth. *J. Cell Biol.*, 169(6): 897–908, 2005.

Other publications

1. I. F. Sbalzarini, H. Ewers, A. Smith, A. Helenius, and P. Koumoutsakos. Heimtückische Viren auf lebenden Zellen. *ETH Bulletin*, 298: 48–50, 2005.

2. A. E. Smith, H. Ewers, I. Sbalzarini, P. Koumoutsakos, and A. Helenius. Alternate endocytic pathways revealed by virus entry into animal cells. *Mol. Biol. Cell*, 15: 195A, 2004.
3. I. F. Sbalzarini, A. Mezzacasa, A. Helenius, and P. Koumoutsakos. Large-scale simulations of diffusion in cell biology. *ERCIM News*, 59: 69–70, 2004.
4. I. Sbalzarini, P. Koumoutsakos, A. Mezzacasa, and A. Helenius. Computing diffusion in intracellular compartments: The influence of organelle geometry. In *New Horizons in Molecular Sciences and Systems: An Integrated Approach*. October 16 - 18, 2003, Nago City, Okinawa, Japan, page 113, 2003.
5. I. F. Sbalzarini, A. Mezzacasa, A. Helenius, and P. Koumoutsakos. Organelle shape has an influence on fluorescence recovery results. *Mol. Biol. Cell*, 13: 276A, 2002.
6. I. F. Sbalzarini, J. Theriot, and P. Koumoutsakos. Machine learning for biological trajectory classification applications. In *Proceedings of the Summer Program 2002*, pages 305–316. Center for Turbulence Research, Stanford University/NASA Ames, 2002.
7. P. Catalano, M. Wang, G. Iaccarino, I. F. Sbalzarini, and P. Koumoutsakos. Optimization of cylinder flow control via actuators with zero net mass flux. In *Proceedings of the Summer Program 2002*, pages 297–303, Center for Turbulence Research, Stanford University/NASA Ames, 2002.

Index

Curriculum Vitae

Name:	Ivo Fabian Sbalzarini
Citizen of:	Switzerland
Born:	16th January 1977, in Arbon, Switzerland
May 1997	Matura Typus C, Kantonsschule Wattwil.
1997 - 2002	Diploma studies in Mechanical Engineering at ETH Zürich, Switzerland, with majors in Control and Computational Science.
Feb 2002	Diploma in Mechanical Engineering at ETH Zürich (Dipl. Masch.-Ing. ETH).
2002 - 2006	Research and teaching assistant at the Computational Science and Engineering Laboratory (CSELab), Institute of Computational Science, ETH Zürich, under the supervision of Prof. Dr. P. Koumoutsakos.