Stochastic models of cell motility

Gradinaru, Cristian; Mølhave, Kristian; Flyvbjerg, Henrik

Publication date:
2012

Document Version
Publisher's PDF, also known as Version of record

Citation (APA):
Gradinaru, C., Mølhave, K., & Flyvbjerg, H. (2012). Stochastic models of cell motility.

Technical University of Denmark

Stochastic models of cell motility

Cristian Grădinaru

15th of December, 2011

Department for Micro- and Nanotechnology, DTU

Abstract

Cell motility and migration are central to the development and maintenance of multicellular organisms, and errors during this process can lead to major diseases. Consequently, the mechanisms and phenomenology of cell motility are currently under intense study. In recent years, a new interdisciplinary field focusing on the study of biological processes at the nanoscale level, with a range of technological applications in medicine and biological research, has emerged.

The work presented in this thesis is at the interface of cell biology, image processing, and stochastic modeling. The stochastic models introduced here are based on persistent random motion, which I apply to real-life studies of cell motility on flat and nanostructured surfaces. These models aim to predict the time-dependent position of cell centroids in a stochastic manner, and conversely determine directly from experimental recordings of cell motility the various motility parameters. This can aid the experimentalist to draw biologically relevant conclusions about cell-substrate interactions.

The need to track cells in a large number of movies has raised the question of automation of cell tracking and that of reproducibility and robustness of cell centroid measurement. To address this, I wrote the PACT cell tracking program, which is optimized for uniform as well as non-uniform backgrounds such as nanostructured surfaces. Rapid progress in the field of the automation of cell tracking steered us into a comparative study of PACT's performance against other cell tracking programs. We find that different programs yield somewhat different results when applied to the same movie of migrating cells but that the differences are not statistically significant.

To introduce persistent random motion, I first present a study of idealized random motion in two dimensions. This finds direct application to experimental studies of cell membrane fluidity and membrane protein dynamics, and I improve on the methodology currently used in that field by showing how to assess the randomness of the motility and how to optimally determine the diffusion coefficient. By adding a persistence component to simple random motion I introduce the standard Ornstein-Uhlenbeck process. I build on this commonly used cell motility model to address the challenges of working with real-life data: positional (centroid coordinate measuring) error and time discretization (due to finite frame rate in a movie of motile cells). This includes optimally measuring the motility parameters and balancing precision of measurement against the mathematical complexity of real-life models of cell motility. Finally, I expanded our understanding of cell response to surface topography by generalizing the Orstein-Uhlenbeck process to study cell motility on anisotropic substrates. I apply the general model to analyze cell motility on a series of anisotropic substrates and discuss the implications of our observations.

This work is potentially useful to cell biologists by addressing their need for precise yet simple tools for studies of cell motility. The advances in the theoretical understanding of motility presented here bear the experimentalists' needs in mind, and can find direct technological applications such as cell guidance and growth using nanotopography.

Resumé

Cellemotilitet og -bevægelse er centrale for multicellulære organismers udvikling og opretholdelse, og fejl i disse processer kan føre til alvorlige sygdomme. Mekanismerne bag cellemotilitet og dens fænomenologi studeres derfor intenst. I de seneste år er et nyt interdisciplinært felt, med fokus på studiet af biologiske processer på nanoskala, og med en række teknologiske anvendelsesmuligheder i medicin og biologisk forskning, opstået.

Arbejdet, der præsenteres i denne afhandling befinder sig på grænsefladen mellem cellebiologi, billedbehandling og stokastisk modellering. Jeg præsenterer stokastiske modeller, baseret på modeller for *vedholdende tilfældig bevægelse* (*persistent random motion*), og anvender dem på eksperimentelle *real-life* studier af cellemotilitet på glatte og nanostrukturerede overflader. Disse modeller bruges til at estimere den tidsafhængige position af cellers geometriske tyngdepunkt (*centroid*), og bestemme de forskellige motilitetsparametre direkte fra eksperimentelle målinger af cellemotilitet. Dette kan hjælpe eksperimentatorer med at drage biologisk relevante konklusioner om vekselvirkninger mellem celle og substrat.

Behovet for at følge celler i stort antal har sat fokus på automatisering af *celletracking* (*cell tracking*) og reproducerbarheden og robustheden af bestemmelsen af positionen af cellers geometriske tyngdepunkt. For at håndtere dette, har jeg skrevet celletracking programmet PACT, som er optimeret for glatte såvel som nanostrukturerede overflader. Den hurtige udvikling indenfor feltet ledte os til et komparativt studie af PACT's præstationsevne sammenlignet med andre celletracking programmer. Vi finder, at de forskellige programmer giver forskellige resultater, når de bruges på samme film af bevægende celler, men at forskellene ikke er statistik signifikante.

For at introducere vedholdende tilfældig bevægelse, præsenterer jeg først et studie af idealiseret tilfældig bevægelse i to dimensioner. Dette finder direkte anvendelse i eksperimentelle studier af cellemembraners fluiditet og membranproteiners dynamik, og jeg forbedrer metodologien, som bruges i dette felt, ved at vise hvordan tilfældigheden i bevægelsen skal tolkes og hvordan diffusionskoefficienten estimeres optimalt. Ved at tilføje et *vedholdende led* (*persistent component*) til den simple tilfældige bevægelse introducerer jeg Ornstein-Uhlenbeck processen, standardmodellen for vedholdende tilfældig bevægelse. Jeg bygger på denne model for at håndtere udfordringerne, som er til stede når eksperimentelle data håndteres: eksperimentel usikkerhed på den måle position og tids-diskretisering (pga. endelig frame rate i film af bevægende celler). Dette omfatter at måle motilitesparametrene optimalt og at vægte målepræcision mod den matematiske kompleksitet af real-life modeller for cellemotilitet. Endelig har jeg udvidet vores forståelse af cellers respons på overfladetopografi ved at generalisere Ornstein-Uhlenbeck processen for studie af cellemotilitet på anisotrope substrater. Jeg anvender den generaliserede model til at analysere cellemotilitet på en række anisotrope substrater og diskuterer betydningen af vores observationer.

Dette arbejde har brugbarhed for cellebiologer, da det henvender sig til deres behov for præcise, men simple værktøjer til studier af cellemotilitet. Fremskridtene indenfor den teoretiske forståelse af cellemotilitet, som præsenteres her, tager hensyn til eksperimentatorers specifikke behov, og har direkte teknologiske anvendelsesmuligheder, såsom styring af cellers bevægelse og vækst vha. nanotopografi.

Acknowledgements

This thesis is one of the requirements towards the Ph.D. degree from the Technical University of Denmark. The Ph.D. work has been performed in the *Stochastic Systems and Signaling* and *Molecular Windows* groups at the Department of Micro- and Nanotechnology of the Technical University of Denmark between September 1st, 2008 and December 15th, 2011. This project has been supervised by Assoc. Prof. Kristian Mølhave and Assoc. Prof. Henrik Flyvbjerg.

This thesis is summarizing the concerted effort of several people over the past three years, beyond my own contribution, and I'd like to extend my thanks to all those involved.

Among those are first and foremost my two advisors, Assoc. Prof. Kristian Mølhave and Assoc. Prof. Henrik Flyvbjerg. Having two advisors provides the opportunity to receive advice from both and offers the freedom to choose. It is also an opportunity to take full charge of the project; writing this thesis has provided plenty of practice for the latter, at Kristian's encouragement. Kristian ended up becoming my main supervisor and has been the leader of DTU's contribution to the *Nanoscale* project, which this work is largely part of. Although I have lived for 8 years in a country that has elected for presidency a man whose slogan was "We can do it," I've never met anyone as flush with optimism as Kristian. Although not a statistician himself, he was very eager to learn and expand his scientific horizons well beyond his nanotechnology training. I appreciated that, I myself coming from a multidisciplinary background.

I am especially grateful to him for having stepped in when my working relationship became strained, as that allowed me to continue the Ph.D. Indeed I came pretty close to quitting a few times. He was also the person who took the initiative to mediate the resumption of my working relationship, which is always a good situation to find yourself in at the end of a Ph.D. degree.

Henrik is a good statistician and an excellent resource for mathematical modeling. I enjoyed the intellectual challenge of my two years of working with him, which allowed me to learn a lot about mathematics in biology. The fact that he enjoys teaching and was very available for debate contributed to that. Perhaps the most important contribution that will remain in my personal life after Henrik was being introduced to David Beerman.

David was a good and supportive friend, whom I have learned a lot from that you don't learn in school.

Special mention is also due to Henrik's lab members: Jonas Pedersen, Christian Vestergård, and Kim Mortensen. My relationship with Christian Vestergård was (and I intentionally took the effort to preserve it so), more personal in scope than scientific. We have nonetheless had a very good scientific relationship (thanks Vester for translating the abstract of this thesis into Danish), but his most important contribution in my life was introducing me to the Danish culture. So thanks Vester and Elise for including me in your events, for putting up with my not learning much Danish, and for letting me sleep on your sofa on those weekend days when it was getting too late to go home.

Being a member of the Molecular Windows group gave me refresher on collegiality, some of which I seemed to have forgotten in the previous two years. So thanks Bilal (you still owe me a tennis game!), Carsten, Eric, and Rafal. Rafal has also provided some of the nanograss substrates used in this project and help with some cell imaging.

Joanna Lopacinska has played a primarily scientific role as my closest collaborator. She is the person who provided all the biological data that I make use of in this thesis. She is a fine biologist, very hard working (as all good biologists are!) and very precise. I was lucky to have such a reliable source of data, and I hope she has forgiven the occasional misbehavior on my part (I know she understands the causes).

Other people have played an intermittent or transient role in providing scientific or personal support, yet thanks are also due to them. Prof. Mogens R. Poulsen owes my thanks for providing support with professional relations within the department. I am glad now that I accepted to be Martin Toft Madsen's master's project co-supervisor. Martin is a smart and motivated guy, a pleasure to interact with, and the person who in a way made it all happen: his master thesis was about building the microscope setup and temperature controller that were needed for all the cell imaging. Cameron Smith provided the occasional cup of coffee in his office and some support and understanding. I wish I'd known him better. Hans Kestler and Johannes Huth became collaborators on our cell tracking work. That led to a 2-day work visit to Ulm, Germany, sponsored by the project. Gertrud Hjørte and Katrine Kiilerich-Hansen helped with processing movies in Autozell. Yong Chen's laboratory provided the grooved substrates used in chapter IV.

Jessica has been a good friend that allowed me to keep some ties to the US and my old life as I was transitioning into the new one. I am glad to see that we've remained friends even as we lost contact over the years and only interacted sporadically since then. Rogier was one of the better roommates I've had, possibly because of the complementarily of our temperaments. He also deserves thanks for reading the introduction chapter of this thesis, and for listening to my stories. Thijs came into our apartment just at the right time, and stayed just the right amount too, when I really needed someone with his insight into life matters, with his social manners, with his laid-back presence. I wonder if he's aware of the amount of help he's provided. Christa and John provided the cheapest housing in Lyngby a student can ask for, despite my initial distrust of how good a deal that was. Anca and Gabi were my contacts to the Romanian community in Denmark. Santimo provided the occasional "hygge" environment in Nørrebro that I came to enjoy, and never gave in as I tried to convince him to finish high school. After all, he is a successful businessman now.

Last, but not least, I have to thank my parents. I have inherited my father's technical intelligence, and this is of course at least part of the reason why I have been around in the sciences for so long. The persistence and resilience that I have I owe instead to my mother, and those two qualities alone are critical prerequisites to finishing any Ph.D. degree, whether spiked with "extra-curricular adventures" or not.

Copenhagen, 15th of December, 2011

Cristian Grădinaru

Table of abbreviations and symbols

Abbreviation/Acronym	**Meaning**
PACT	Program for Automated Cell Tracking
TLA	TimeLapseAnalyzer
OU	Ornstein-Uhlenbeck
LSQ	Least-squares fit
wLSQ	Weighted least-squares fit
gLSQ	Generalized least-squares fit

Symbol	**Meaning**
$\vec{r}$	position vector
m	mss
γ	drag coefficient
$\vec{F}_{therm}$	thermal force
D	diffusion coefficient
$\langle \dots \rangle$	expectation value
$\delta(t)$	delta function
$\vec{\xi}(t)$	generalized white noise: $\langle \vec{\xi}(t) \rangle = 0, \langle \vec{\xi}(t)\vec{\xi}(t') \rangle = 2\delta(t - t')$
δt	interval of time judged small relative to the persistence time
$\mathbf{N}(\mu,\sigma^2)$	Gaussian distribution of mean μ and variance σ^2
$\vec{\eta}_j$	discrete, random, Gaussian-distributed 2D vector ($\vec{\eta}_j \sim \vec{N}(0,1)$)
δ_n	mean squared n-step displacement (see equation 1, section I.2.a)
est(D)	estimator of D (function of measured quantities)
$\mathcal{N}$	number of ensemble elements
$\mathbf{\Sigma}$	$[\delta_{n,m}]$: matrix with elements defined by equation 5, section I.2.a
δ_{est}	$4\delta t \cdot \mathrm{est}_{\underset{n_{max}}{gLSQ}}(D)$, simplifying notation
Δ	$[\delta_n]_{n=1,2..nmax}$: column vector with elements defined by equation 2, section I.2.a
$\mathbf{N}$	$[j]_{j=1,2..nmax}$: column vector
$\vec{r}_i^{\,th}$	theoretical position vector (i.e. free from positional noise)
$\sigma_{exp}{}^2$	variance of positional noise
$\bar{\delta}$	time-average
P	Persistence time
$\vec{v}(t)$	instantaneous velocity: $\vec{v}(t) = d\,\vec{r}(t)\,/dt$
$\phi(t)$	Velocity autocovariance function: $\phi(t) =< \vec{v}(t) \cdot \vec{v}(0) >$
$\vec{u}_j$	secant velocity: $\vec{u}_j = \left(\vec{r}_j - \vec{r}_{j-1}\right)/\,\Delta t$
ϕ_j^u	secant velocity autocovariance function: $\phi_j^u =< \vec{u}(t) \cdot \vec{u}(0) >$
$\delta_{j,k}$	Kroneker-δ: 1 if j=k, 0 otherwise.
P_k	power spectrum
$\delta_{j,k}$	Kroneker-δ: 1 if j=k, 0 otherwise.

List of publications

Grădinaru, C.; Lopacinska, J. M., Flyvbjerg, H.; Huth, J.; Kestler, H. A.; Mølhave K. "Automated Analysis of Cell Migration on Flat and Nanostructured Surfaces", submitted to Plos Computational Biology

Lopacinska, J. M.; **Grădinaru, C.;** Wierzbicki, R.; Schmidt M. S.; Madsen M. T.; Skolimovski M.; Dufva M.; Flyvbjerg H.; Mølhave, K. **2011**, "Cell Motility, Morphology, Viability and Proliferation in response to Nanotopography on Silicon Black" submitted to Nanoscale.

Huth, J.; Buchholz, M.; Krauss, J.; Mølhave K.; **Grădinaru C.;** Wichert, G.; Gress, T. M.; Neumann, H.; Kestler, H. A.; "TimeLapseAnalyzer: Multi-target analysis for live-cell imaging and time-lapse microscopy", **2011**, Comp. Meth. Prog. Biomed. 104(2), pp. 227-234

Table of Contents

Chapter I: Introduction

Cell motility and migration are central to the development and maintenance of multicellular organisms, and errors during this process can lead to major diseases that have been intensely scrutinized over the last decades, such as cancer (tumor formation and metastasis) (Hanahan and Weinberg 2000; Condeelis and Pollard 2006; Hanahan and Weinberg 2011), heart disease (accumulation of cholesterol and white blood cells on the arterial wall)(Saphir and Gore 1950; Prescott et al. 1989; Libby et al. 2002), and mental retardation (Golden 2001). Additionally, tissue formation during embryonic development, wound healing, and various immune responses involve synchronized cell migration (Detrich et al. 1995; Gurtner et al. 2008). It is clear that more insight into the basic mechanisms and phenomenology of cell motility may aid medical research and engineering to develop new therapeutic strategies for controlling conditions where mistakes do occur and thereby lead to disease.

Recently, a new interdisciplinary field focusing on the study of biological processes at the nanoscale level, with a range of technological applications in medicine and biological research, has emerged. The work presented in this thesis is at the interface of cell biology, image processing, and stochastic modeling, aiming to theoretically characterize cell motility. Quantitative mathematical modeling of cell motility is a field that only recently has seen significant developments, both in terms of mathematical complexity of the models involved (Selmeczi et al. 2005; Tu and Grinstein 2005; Korobkova et al. 2006), but also in terms of discriminating between the motility patterns of different cell types (Grădinaru et al.; Selmeczi et al. 2005). The reason for this is, in part, that cell motility is a stochastic process and it has been challenging to model it based on experimental data (Selmeczi et al. 2005; Li et al. 2011).

The stochastic models introduced here are based on persistent random motion (Li et al. 2008), which I apply to real-life studies of cell motility on flat and nanostructured surfaces. These models aim to predict the time-dependent position of cell centroids in a stochastic manner, and conversely determine directly from experimental recordings of cell motility the various motility parameters. This can aid the experimentalist to draw biological relevant conclusions about cell-substrate interactions.

1. Objective

The objectives of this Ph.D. project have been to theoretically characterize cell motility, modeled as a stochastic process, and to implement this theoretical work in the form of computational tools directly useable by biologists interested in interpreting their experimental results. Additionally, the work presented herein attempts to show mathematicians and statisticians alike the underlying biological relevance of this theoretical study, and how their skills can be used to solve problems of current interest. It also aims to assist biologists in understanding the details and especially the caveats of the mathematical analyses performed, thereby helping them to correctly interpret the results of the statistical analysis performed by the tools provided.

To that end, this work has focused on the following:

- To expand the theoretical understanding of the persistent random motion models commonly employed to describe cell motility
    - by investigating how the positional measurement noise of the experimental data, as well as the discrete nature of the cell trajectories, affect the cell motility models and the parameters determined from the experimental data;
    - by proposing theoretical and computational methods to determine the motility models and related parameters directly from experimental data;
    - given prior observations (Selmeczi et al. 2005) of substrate-independence, but cell type-dependence of such persistent random motion models, by taking a step forward to understand the variation in motility within a group of cells belonging to the same cell type. This was done by assessing phylogenetic (species-of-origin dependent) effects on fibroblast motility models and related parameters; and
    - by determining the effect of substrate topography, and especially nanotopography, on cell motility.
- To write a computer program, PACT, specifically optimized for tracking cells on the types of surfaces that have been investigated in this project. This is one of the computational tools that may be used to obtain quantitative data (cell centroid coordinate measurements) from experimental data (movies of motile cells).
- To perform a comparative assessment of PACT versus other computer programs available commercially (Autozell (Baumann 2006)) or on an open-source basis (TLA (Huth et al. 2010; Huth et al. 2011)). This is a comprehensive study of reproducibility of results by quantitating the effect of using different cell tracking programs on the cell motility analysis and corresponding motility parameters values.

2. Modeling cell motility

a. Biological mathematical modeling

While mathematical applications in biology have a long history, there has been a significant increase in interest in the recent years. Most notable is the field of bioinformatics, fueled by the recent availability of whole-genome sequences and especially by the Human Genome Project (Venter et al. 2001). These advances provided data-rich information sets which are difficult to understand without the use of analytical tools. Recent developments in the field of mathematics, especially chaos theory, have helped understanding complex and nonlinear biological processes (Skinner et al. 1992; Waldrop 1992). The intertwining of these two apparently separate fields has also been catalyzed by rapid developments in computer technology (both hardware and software), which enabled calculations and simulations that would have been prohibitive in the past. Finally, biology has seen a surge of interest in *in silico* experimentation due to various ethical concerns and unreliability or risk associated with animal and human research.

b. State of the art in modeling cell motility

Cell motility modeling has a long history, and has especially developed in the case of prokaryotic organisms (e.g. bacteria) (Berg and Brown 1972; Berg 2000; Gerbal et al. 2000; Mavroidis et al. 2004; Tu and Grinstein 2005; Korobkova et al. 2006; Lauga et al. 2006; Leoni et al. 2009). Part of the reason for that is that the biochemical mechanisms underlying prokaryote cell motility are very well established, and backed in some cases by a complete understanding of the structural and biochemical details of the molecular motors involved in flagellar movement, which in turn drives bacterial motility (Charon and Goldstein 2002; Berg 2003).

On the other hand, the mechanisms of eukaryotic motility are less well understood, despite a stronger interest from the medical community to elucidate its mechanisms. In parallel, modeling eukaryotic motility has generally remained simple, partly because of the lack of understanding of the underlying mechanistic details (Doob 1942; Gail and Boone 1970; Schienbein and Gruler 1993; Shenderov and Sheetz 1997). However, recent advances in the field led to the development of increasingly complex motility models (Selmeczi et al. 2005; Li et al. 2008; Li et al. 2011).

The majority of the models proposed to model cell motility, be it prokaryotic or eukaryotic, are stochastic processes largely based on simple Brownian motion (Ornstein 1919; Uhlenbeck and Ornstein 1930; Doob 1942). As described in section I.2.a, statistical analysis of biological data finds increasingly more applications as high-throughput, high-sensitivity technologies become available. One such application is described in (Selmeczi et al. 2005), where the experimental time series for trajectories of motile cells are found to contain so much information that a systematic analysis yields cell-type specific motility models.

c. Advances made and outline

This thesis presents research of an interdisciplinary nature, addressed to those with an interest and expertise in mathematical modeling of biological processes. As such, it has a strong statistics and modeling component, although I left most abstract concepts and various proofs in the appendices, so as to make it accessible to those in the biological sciences who have a primarily experimental interest. This work has been done constantly bearing in mind the experimentalists' needs and indeed part of it has been done in collaboration with experimentalist colleagues at DTU and elsewhere. Consequently, I believe it would also be of value and a straightforward read to those in the biological sciences that have a minimal background in mathematics, but share an interest in modeling cell motility.

Chapter II sets the theoretical foundations for the types of cell motility studies described in the remainder of the thesis. It is structured by starting with the basic concepts of stochastic modeling (with an emphasis to biological applications and to the challenges encountered by experimentalists when dealing with real-life data). A layer of mathematical complexity is added to present the simplest stochastic process that is commonly used to describe cell motility, the standard Ornstein-Uhlenbeck (OU) process (Ornstein 1919; Uhlenbeck and Ornstein 1930). Again, I address the typical experimental challenges and propose several methods for handling them. These mathematical results are confirmed by *in silico* simulations of experimental-like data, which highlights the limitations and the strengths of the methods that I propose.

Chapter III takes a step forward into experimental reality by presenting a program for automated cell tracking, PACT, which I wrote specifically for the type of cell motility data encountered by experimentalists. I make a comparative assessment of PACT against several other cell tracking programs available and draw conclusions on how the choice of program may affect the cell motility analysis results. I also take a step forward in this chapter to apply the purely theoretical concepts of chapter II to experimental data obtained with the use of PACT. The cell motility data (coordinates of cell centroids vs. time) allows for direct testing of the hypothesis that cell motility follows the standard OU process. I find, consistent with prior work in the field (Selmeczi et al. 2005), that deviations from the simplicity of the standard process exist, and given the large amount of experimental data available, I propose a variation of the standard OU process that models the observed data exactly and quantitate the relevant motility parameters. Following this proof-of-concept section, this method was applied to study cell motility on a range of nanostructured surfaces, and I draw preliminary conclusions on the effect of nanotopography on cell motility.

Finally, in chapter IV I have generalized the standard OU process into a tensor form, which allows treatment of cell motility on anisotropic substrates. I develop the necessary theoretical concepts in parallel with analyzing experimental data of cell motility on substrates featuring nanogrooves. I determine the relevant motility parameters and draw a range of conclusions regarding differences between the isotropic motility encountered in chapters II and III and anisotropic motility. However, I underline that this research is preliminary and is purely meant to show the reader the future directions and to make recommendations for future analysis of experimental data of this type.

Chapter V contains the concluding remarks, which sum up all of the results presented, take a step back to the Objective section of this thesis to show what has been accomplished, and present a number of other research avenues that this work opens.

3. *Future applications*

The work presented in this thesis has originally been intended to aid researchers in cell biology to better characterize the cellular response to a range of substrates (chemically modified or not, patterned with nanostructured elements or not, etc.), with the aim of identifying how the cell culture dishes used for cell culturing may affect their behavior. This research also aimed to ultimately lead to designing new micro- and nano-tools for mammalian cell cultures, such as microfluidic cell culture devices, which exhibit many advantages over 2-D monolayer cell cultures (Amatore et al. 2007-2011). Long term applications on this type of research may be found in the field of medical instrumentation engineering, where the basic research results of the type presented here can have direct and commercial applications.

4. References

Amatore C, Chen Y, Cojoc D, Flyvbjerg H, Grădinaru C et al. (2007-2011) Understanding interactions between cells and nanopatterned surfaces. European Union.

Baumann H (2006) Image-Blackboards und Algorithmen zur Bearbeitung von Zellkollisionen im Zelltracking-System AUTOZELL.

Berg HC (2000) Motile Behavior of Bacteria. Physics Today 53(1): 24.

Berg HC (2003) THE ROTARY MOTOR OF BACTERIAL FLAGELLA. Annual Review of Biochemistry 72(1): 19-54.

Berg HC, Brown DA (1972) Chemotaxis in Escherichia coli analysed by Three-dimensional Tracking. Nature 239(5374): 500-504.

Charon NW, Goldstein SF (2002) GENETICS OF MOTILITY AND CHEMOTAXIS OF A FASCINATING GROUP OF BACTERIA: The Spirochetes. Annual Review of Genetics 36(1): 47-73.

Condeelis J, Pollard JW (2006) Macrophages: Obligate Partners for Tumor Cell Migration, Invasion, and Metastasis. Cell 124(2): 263-266.

Detrich HW, Kieran MW, Chan FY, Barone LM, Yee K et al. (1995) Intraembryonic hematopoietic cell migration during vertebrate development. Proceedings of the National Academy of Sciences 92(23): 10713-10717.

Doob JL (1942) The Brownian Movement and Stochastic Equations. The Annals of Mathematics 43(2): 351-369.

Gail MH, Boone CW (1970) The Locomotion of Mouse Fibroblasts in Tissue Culture. Biophysical Journal 10(10): 980-993.

Gerbal F, Chaikin P, Rabin Y, Prost J (2000) An Elastic Analysis of Listeria monocytogenes Propulsion. Biophysical journal 79(5): 2259-2275.

Golden JA (2001) Cell migration and cerebral cortical development. Neuropathology and Applied Neurobiology 27(1): 22-28.

Grădinaru C, Łopacińska JM, Huth J, Kestler HA, Flyvbjerg H et al. Assessment of Automated Analyses of Cell Migration on Flat and Nanostructured Surfaces. Plos Computational Biology submitted.

Gurtner GC, Werner S, Barrandon Y, Longaker MT (2008) Wound repair and regeneration. Nature 453(7193): 314-321.

Hanahan D, Weinberg R (2000) The Hallmarks of Cancer. Cell 100(1): 57-70.

Hanahan D, Weinberg R (2011) Hallmarks of Cancer: The Next Generation. Cell 144(5): 646-674.

Huth J, Buchholz M, Kraus JM, Schmucker M, von Wichert G et al. (2010) Significantly improved precision of cell migration analysis in time-lapse video microscopy through use of a fully automated tracking system. BMC Cell Biology 11(24): 24.

Huth J, Buchholz M, Kraus JM, MÃ¸lhave K, Gradinaru C et al. (2011) TimeLapseAnalyzer: Multi-target analysis for live-cell imaging and time-lapse microscopy. Computer Methods and Programs in Biomedicine 104(2): 227-234.

Korobkova EA, Emonet T, Park H, Cluzel P (2006) Hidden Stochastic Nature of a Single Bacterial Motor. Physical Review Letters 96(5): 058105.

Lauga E, DiLuzio WR, Whitesides GM, Stone HA (2006) Swimming in Circles: Motion of Bacteria near Solid Boundaries. Biophysical Journal 90(2): 400-412.

Leoni M, Kotar J, Bassetti B, Cicuta P, Lagomarsino MC (2009) A basic swimmer at low Reynolds number. Soft Matter 5(2): 472-476.

Li L, NÃ¸rrelykke SF, Cox EC (2008) Persistent Cell Motion in the Absence of External Signals: A Search Strategy for Eukaryotic Cells. PLoS ONE 3(5): e2093.

Li L, Cox EC, Flyvbjerg H (2011) "Dicty Dynamics": Dictyostelium motility as persistent random motion. Phys Biol 8: 12.

Libby P, Ridker PM, Maseri A (2002) Inflammation and Atherosclerosis. Circulation 105(9): 1135-1143.

Mavroidis C, Dubey A, Yarmush ML (2004) MOLECULAR MACHINES. Annual Review of Biomedical Engineering 6(1): 363-395.

Ornstein LS (1919) On the Brownian Motion. Koninklijke Nederlandse Akademie van Weteschappen Proceedings Series B Physical Sciences 21: 96-108.

Prescott M, McBride C, M. C (1989) Development of intimal lesions after leukocyte migration into the vascular wall. Am J Pathol 135(5): 835-846.

Saphir O, Gore I (1950) Evidence for an inflammatory basis of coronary arteriosclerosis in the young. Arch Pathol Lab Med 49: 418-426.

Schienbein M, Gruler H (1993) Langevin equation, Fokker-Planck equation and cell migration. Bulletin of Mathematical Biology 55(3): 585-608.

Selmeczi, Mosler, Hagedorn, Larsen, Flyvbjerg (2005) Cell motility as persistent random motion: theories from experiments. Biophysics Journal 89: 912-931.

Shenderov AD, Sheetz MP (1997) Inversely correlated cycles in speed and turning in an ameba: an oscillatory model of cell locomotion. Biophysical Journal 72(5): 2382-2389.

Skinner J, Molnar M, Vybiral T, Mitra M (1992) Application of chaos theory to biology and medicine. Integrative Physiological and Behavioral Science 27(1): 39-53.

Tu Y, Grinstein G (2005) How White Noise Generates Power-Law Switching in Bacterial Flagellar Motors. Physical Review Letters 94(20): 208101.

Uhlenbeck GE, Ornstein LS (1930) On the Theory of the Brownian Motion. Physical Review 36(5): 823.

Venter JC, Adams MD, Myers EW, Li PW, Mural RJ et al. (2001) The Sequence of the Human Genome. Science 291(5507): 1304-1351.

Waldrop M (1992) Complexity: The Emerging Science at the Edge of Order and Chaos: {Simon & Schuster}.

Chapter II: Developments to the theory of 2D motion

This chapter is intended to give the reader an overview of some of the motility models most commonly used to study cell motility, and to present a methodology for extracting meaningful information from *in silico*-generated experimental-like data. This sets the foundation for dealing with real-life experimental data, which I will address in the ensuing part of this thesis.

I will start with an introduction to the simplest stochastic process encountered in Nature, simple random motion (Doob 1942), which finds direct applications in biology in e.g. studies of membrane protein diffusion (Mueller et al. 2003) and cell membrane fluidity (Shinitzki and Henkart 1979). I digress on this topic to show how the method currently employed in the literature (Anderson et al. 1992a; Mueller et al. 2003) to determine the diffusion coefficients is suboptimal, and take a step forward by proposing an improved (and simpler!) method to determine the diffusion coefficient in 2D. Some of this research parallels work by Christian Vestergård (Vestergård 2009) which has focused on 1D diffusion. However, the extension from 1D to 2D diffusion is mathematically nontrivial, especially when particle positional read-out noise is taken into account (section II.3).

Upon addition of a persistence term to simple random motion, I introduce simple persistent random motion (Campos et al.). This stochastic process has been commonly used to model cell motility where it is known as the standard Ornstein-Uhlenbeck (OU) process (Ornstein 1919; Uhlenbeck and Ornstein 1930). I introduce common challenges that occur when dealing with experimental data, such as positional measuring noise of the cell centroids and discretization of cell tracks in time, and develop a method to account for these factors in the standard OU process. I also take a step forward by showing how to optimally measure the motility parameters from experimental-like data (i.e. generated *in silico* and subject to positional noise and time discretization effects) using power spectra analysis. While efficient, this method is mathematically cumbersome and not intuitive, so I develop and test on *in silico*-generated data a simplified (but suboptimal) method of extracting motility parameters from experimental data, which I will make use of in the remainder of this thesis (sections III.2.a, III.2.c, III.3.c, III.4, IV.2.c.iv).

1. Brownian Motion

Below I introduce Brownian motion as a starting point for modeling the (more complex) process of cell motility on surfaces. Brownian motion is known since the 19th century, its discovery credited to the botanist Robert Brown. Mathematically it is one of the simplest stochastic processes, and the solution to the problem was first proposed by Albert Einstein (Einstein 1905; Xiaodong Yang et al. 2006) and Marian Schmuluchovski (von Smoluchowski 1906).

Since its discovery, Brownian motion found several practical applications, ranging from cosmology (stellar dynamics), economics (stock market fluctuations) to chemical technology (osmosis) and biological processes. In the ensuing section I introduce the mathematical formalism of Brownian motion and I take a step forward in the field by developing the methods currently used for determining the diffusion coefficient of a particle. In the biological sciences, this finds direct practical applications in studies of cell membrane fluidity and diffusion of membrane proteins (Shinitzki and Henkart 1979; Anderson et al. 1992a).

a. Langevin Equation

To mathematically define Brownian motion (which is the motion of a particle in a fluid under the approximation of low Reynolds numbers), I turn to Newtonian mechanics. The forces acting on the particle are the drag from the fluid, proportional to the particle velocity, and a stochastic term owing to the molecular collisions between the fluid molecules and the particle. Please refer to the table of symbols for the various notations below.

$$m\frac{d^2\vec{r}}{dt^2} = -\gamma\frac{d\vec{r}}{dt} + \vec{F}_{therm}(t)$$

Equation 1

At low Reynolds numbers the inertial forces are negligible relative to the drag forces (i.e. idealization of a massless particle) and since the fluid is isotropic and uniform, the collective of molecular collisions experiences by the particle with the fluid molecules (the thermal force) is simply a multiple of white noise. This defines Langevin's equation in the Schmulukovski approximation (von Smoluchowski 1906):

$$\frac{d\vec{r}}{dt} = \sqrt{2D}\,\vec{\xi}(t)$$

Equation 2

where $\vec{\xi}(t)$ is a generalized white noise (see table of symbols).

Since on the average the particle does not move away from the origin:

$$\left\langle \vec{r}(t) - \vec{r}(0) \right\rangle = \left\langle \int_0^t \sqrt{2D}\,\vec{\xi}(t)\,dt \right\rangle = \sqrt{2D}\int_0^t \left\langle \vec{\xi}(t) \right\rangle dt = 0$$

Equation 3

a different measurable property is used to relate to the value of D to be determined, namely the mean squared displacement of the particle from the starting point. This defines the scatter of particle positions from the origin:

$$\left\langle \left(\vec{r}(t) - \vec{r}(0)\right)^2 \right\rangle = \left\langle \int_0^t \sqrt{2D}\,\vec{\xi}(t')\,dt' \int_0^t \sqrt{2D}\,\vec{\xi}(t'')\,dt'' \right\rangle = 2D\int_0^t\int_0^t \left\langle \vec{\xi}(t')\vec{\xi}(t'') \right\rangle dt'dt''$$

which evaluates to

$$\left\langle \left(\vec{r}(t) - \vec{r}(0)\right)^2 \right\rangle = 2D\int_0^t\int_0^t \left\langle \vec{\xi}(t')\vec{\xi}(t'') \right\rangle dt'dt'' = 2D\int_0^t\int_0^t \delta(t'-t'')\,dt'dt'' = 4Dt$$

Equation 4

b. Simulating Brownian motion

To compare and add to the different methods proposed in the literature for extracting the value of the diffusion coefficient from a given set of particle positions, I have simulated a particle undergoing Brownian motion. This controlled experiment has the advantage over experimental data of knowing with infinite precision the value of the diffusion coefficient used to generate the dataset, which becomes the gold standard in assessing the quality of the various estimators of the diffusion coefficients used to determine it. Since Langevin's equation can be integrated analytically, less numerically inexact Monte Carlo simulations are not needed. If the position measurement is performed at a fixed rate, δt*:*

$$\vec{r}(t_j + \delta t) = \vec{r}(t_j) + \sqrt{2D\,\delta t}\;\vec{\eta}_j$$

Equation 1

Note that in this integral version of Langevin's equation $\vec{\xi}(t)$ is no longer a continuous quantity (function of time). Instead, it is a discrete, random, Gaussian-distributed 2D vector ($\vec{\eta}_j \sim \mathbf{N}(0,1)$), which makes Equation 1 useable for *in silico* data production.

The MATLAB code used to generate one such dataset may be found in Appendix 4.a, and a sample trajectory (N=1000 data points recorded) is shown in figure 1 below.

Each individual particle progresses slowly (as $n^{1/2}$ as seen previously) away from the starting position at (x,y)=(0,0), although the ensemble average of the particle displacement is zero.

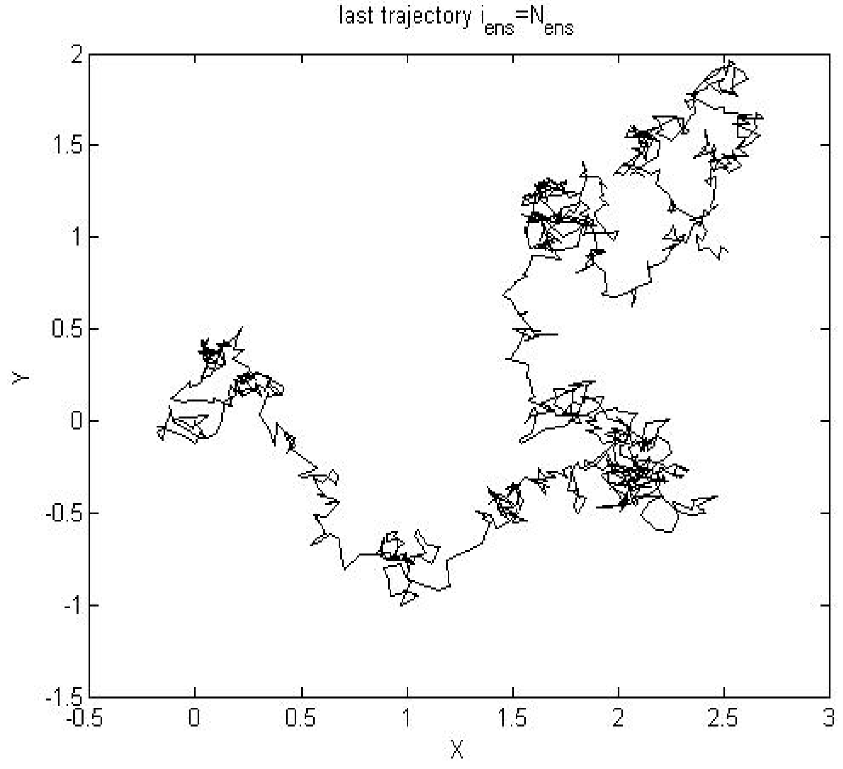


Figure 1. Sample trajectory of a bead undergoing Brownian motion in 2D (N=1000)

2. *Least squares method for determining the diffusion coefficient*

The current methodology used to determine the diffusion coefficient is briefly reviewed, and improved ways to estimate it are introduced. To determine the diffusion coefficient D, the mean squared displacements of the particle are commonly determined:

$$\left\langle \left(\vec{r}(t_j + \delta t) - \vec{r}(t_j)\right)^2 \right\rangle = 2D\,\delta t \left\langle \vec{\xi}_j^{\,2} \right\rangle = 4D\delta t$$

Equation 1

and in general for an *n*-step displacement:

$$\left\langle \left(\vec{r}(t_j + n\delta t) - \vec{r}(t_j)\right)^2 \right\rangle = 4nD\,\delta t$$

Equation 2

From this results that D can be determined from the noise amplitude of the data by plotting the expectation value of the mean squared displacement as a function of *n* and measuring the slope of the resulting line, and this has traditionally been used to determine the diffusion coefficient (Anderson et al. 1992a; Mueller et al. 2003). This is the approach currently employed in the literature, which I show to be suboptimal. By comparing the variances of various LSQ-based estimators, I determine that to determine optimally the diffusion coefficient it is sufficient to use the mean squared one-step displacement of positions only, and that adding the two-step, three-step etc. displacement terms into a LSQ fit as done currently in the literature is actually detrimental to the procedure. Moreover, this method of estimation also reaches the limit that maximizes the use of the information provided by the dataset (the Cramer-Rao limit) (Rao 1973).

These results are tested with and supported by data generated *in silico* (section II.1.b).

a. Simple least squares fitting

In real life many repetitions of the experiment are not always practical, and a more usual way (Anderson et al. 1992a; Mueller et al. 2003) to estimate the expectation value of the mean squared displacement from the origin is to calculate a different quantity, which I will refer to here as the mean squared n-step displacement, from a single experiment:

$$\delta_n = \frac{1}{N-n-1}\sum_{j=1}^{N-n-1}\left(\vec{r}(t_j + n\delta t) - \vec{r}(t_j)\right)^2$$

Equation 1

To estimate D from this type of data, a simple one-parameter LSQ fit of a straight line passing through the origin to δ_n is typically done (Anderson et al. 1992a; Mueller et al. 2003). Namely, the quantity below must be minimized with respect to *D*:

$$\sum_{n=1}^{n_{max}}\left(\delta_n - 4nD\,\delta t\right)^2 = \min$$

Equation 2

This is done using as many as all of the N-1 δ_n values available (*n*=1,2..N-1), where N is the number of coordinate pairs recorded. The caveat is that as *n* increases and approaches N there are increasingly fewer distinct n-step displacements that can be included in the average used to evaluate δ_n which leads substantial deviation from linearity at large values of *n* as seen in Figure 2. Since an unweighted least-squares fitting is performed at this point, the error bars are not shown below.

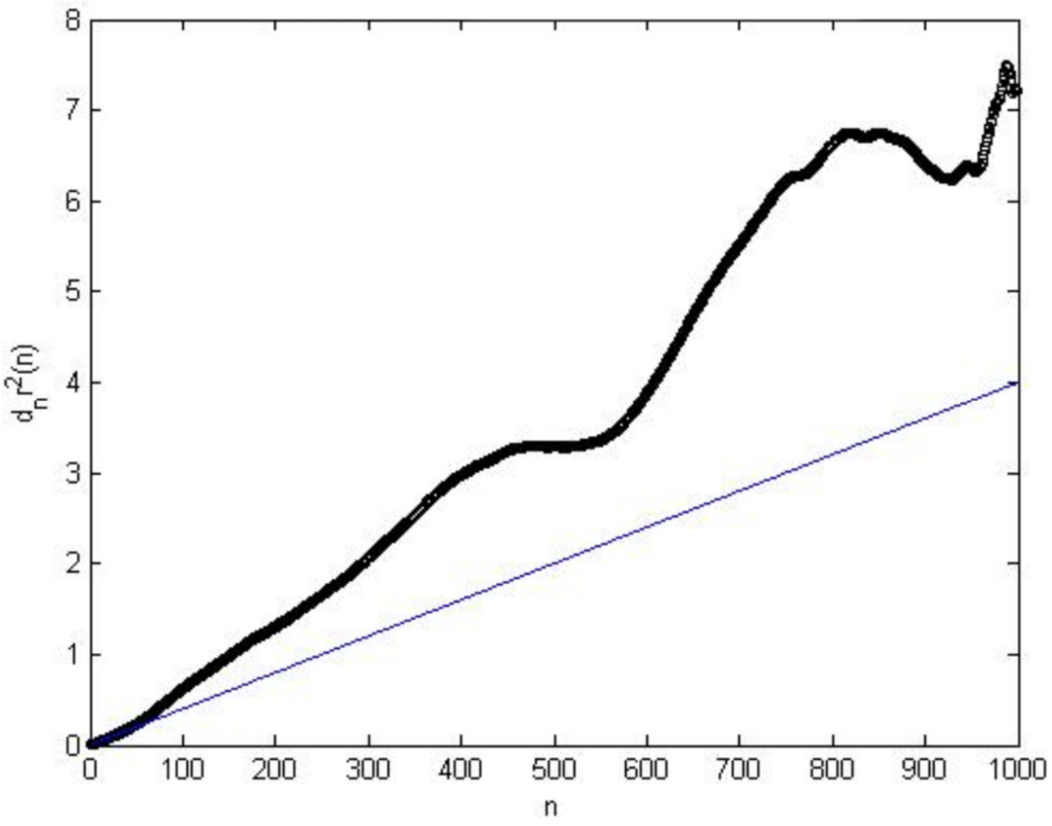


Figure 2. Black circles: mean n-th order displacement δ_n versus *n*. Blue line: line passing through origin and the first point, which provides the best estimate of *D*.

For generality, I have derived the expression for the estimator of *D* (Appendix 1.a) by fitting up to the first n_{max}-step displacement:

$$\mathrm{est}(D)_{n_{max}} = \frac{3\sum_{n=1}^{n_{max}} n\,\delta_n}{n_{max}(n_{max}+1)(2n_{max}+1)2\delta t}$$

Equation 3

This is an unbiased estimator of D (Appendix 1.a). In order to assess the efficiency of this estimator, its variance was determined (Appendix 1.a):

$$\mathrm{var}\,\mathrm{est}(D)_{n_{max}} = \left(\frac{3}{n_{max}(n_{max}+1)(2n_{max}+1)2\delta t}\right)^2 \cdot \sum_{m,n=1}^{n_{max}} mn \cdot \mathrm{cov}\,\delta_{n,m}$$

Equation 4

where

$$\operatorname{cov}\delta_{n,m} = \frac{16D^2\delta t^2}{N-m}\cdot\begin{cases} \frac{1}{6}\cdot(N-n)^3 - \frac{2}{3}m(N-n)^2 + \left(m^2 - \frac{1}{6}\right)\cdot(N-n) & \\ + m\left(nm + \frac{2}{3} - m^2\right) & \text{for } N \le m+n \\ m\left(-\frac{m^2}{3} + \frac{1}{3} + nm\right) - \frac{m^2}{6}\left(m^2 - 1\right)\cdot\frac{1}{(N-n)} & \text{otherwise} \end{cases}$$

Equation 5

While the expression above is only applicable for $n \geq m$, due to the symmetry of the covariance matrix one can simply obtain the $n<m$ elements by exchanging the values of m and n. Note that var δ_n can be identified in the above by setting m=n.

To illustrate and check the accuracy of this formula, a simulation of Brownian motion (N=1000) was run $\mathcal{N}$=10000 times, thus generating an ensemble large enough to accurately ($1/\mathcal{N}^{1/2} \approx 1\%$ error) calculate actual expectation values such as $\langle \delta_n \rangle$ with the corresponding $\pm 1\sigma$ (red traces in figure 3). As seen, these are precisely overlaid (i.e. within the 1% error expected) to the blue traces which are the theoretically-derived expectation values (equation 1, section II.2.a).

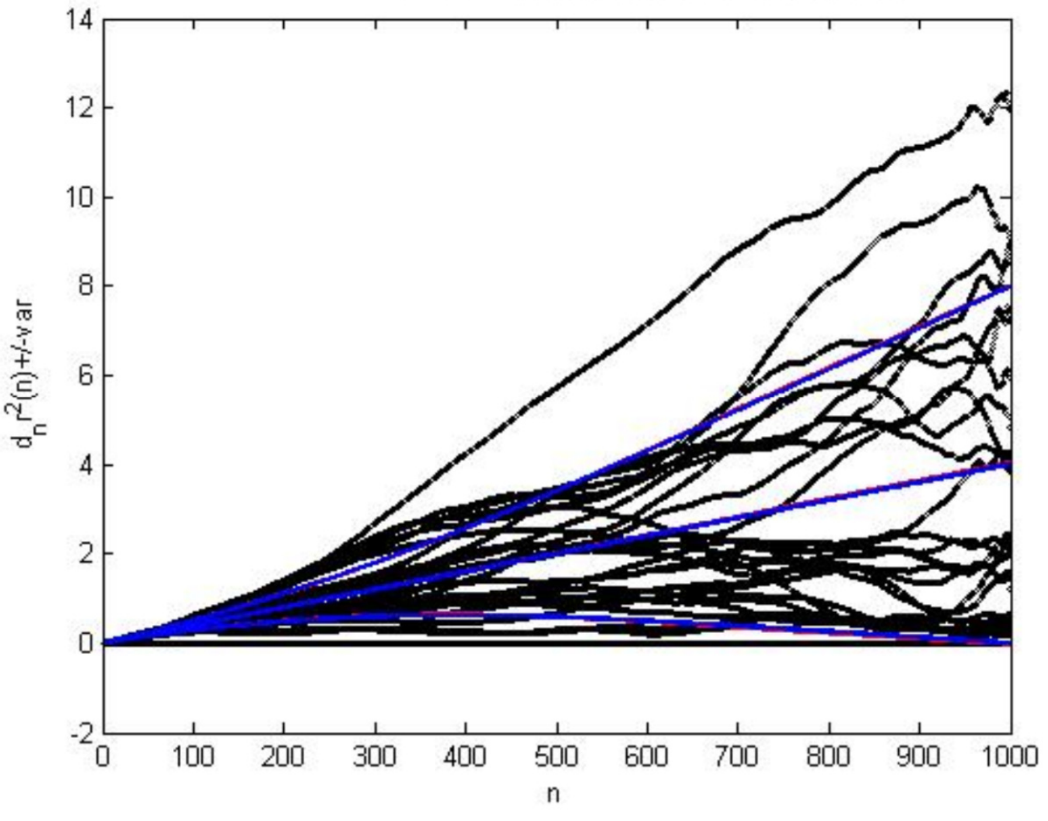


Figure 3. Black: several ensemble elements (δ_n's from select experiments); Blue: expectation values and $\pm$ 1σ determined analytically using input value of the D parameter; Red: ensemble-averaged δ_n and $\pm$ 1σ.

The expression for the variance of the estimator of D may seem daunting; however, it is perhaps more informative to look at family of curves showing var est(D)$_{\text{nmax}}$ as a function of N for increasingly large values of of n_{max} (figure 4). The variance of this estimator of D increases with larger n_{max}, partly due to the larger error at large n_{max}, partly due to correlation of the δ_n values (which means that the larger-order displacements don't bring much more additional information to the problem). Thus, only several small values of n_{max} are shown.

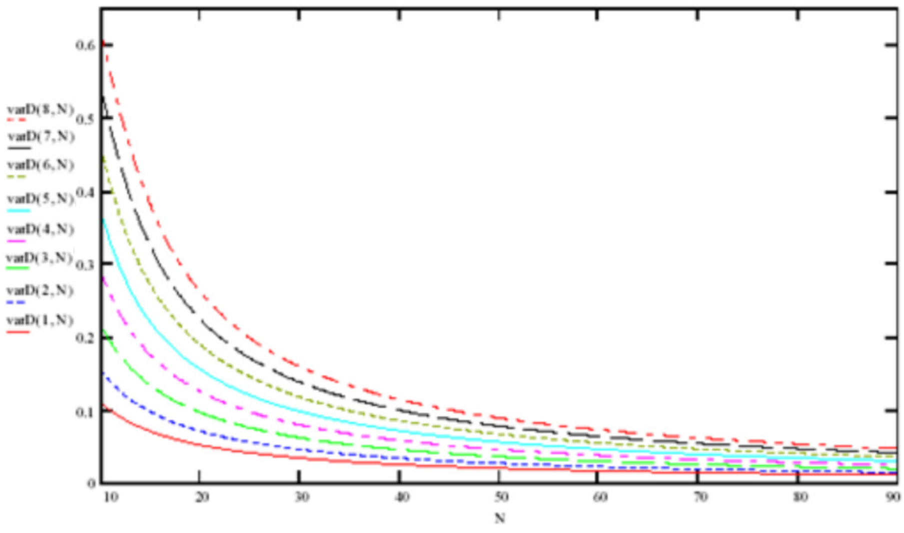


Figure 4. Family of curves showing var est(D)$_{\text{nmax}}$ as a function of N for increasingly large values of of n_{max}.

It is clear from the graph that the most accurate estimation of D is achieved for n_{max}=1 i.e. by only using the one-step displacements (only one data point: δ_1) which really amounts to not doing an LSQ fit at all. From equation 2.a.3 this estimator is

$$\operatorname{est}(D)_1 = \frac{\delta_1}{4\delta t}; \quad \operatorname{var}\operatorname{est}(D)_1 = \frac{D^2}{N-1}$$

Equation 6

This surprising result shows that using the larger-step displacement information actually worsens the result, as a consequence of redundancy (correlation of δ_n values) and of increased error introduced with larger values of n.

b. Weighted least squares fitting

The unweighted LSQ method introduced in the previous section disregards the actual error bars of the δ_n values and effectively assigns equal weight to them. An improvement to the method is weighted LSQ (wLSQ), which uses the theoretical expression for the variance of δ_n (equation 5, section II.2.a, setting m=n). I note here that due to the data redundancy involved in calculating the δ_n values (equation 1, section II.2.a), these variances are underestimated .

$$\sum_{n=1}^{n_{max}} \frac{1}{\operatorname{var}\delta_n} \cdot (\delta_n - 4nD\,\delta t)^2 = \min$$

Equation 1

from which follows (Appendix 1.a):

$$\mathrm{est}_{\underset{n_{max}}{wLSQ}}(D) = \frac{\displaystyle\sum_{n=1}^{n_{max}} \frac{n\delta_n}{\mathrm{var}\,\delta_n}}{\displaystyle 4\delta t \sum_{n=1}^{n_{max}} \frac{n^2}{\mathrm{var}\,\delta_n}}$$

Equation 2

This is an unbiased estimator as well (Appendix 1.a). The sums needed to evaluate the estimator above do not have a closed form and they must therefore be evaluated numerically. I have determined the expression for the variance of this estimator of D (Appendix 1.a):

$$\mathrm{var\ est}_{\underset{n_{max}}{wLSQ}}(D) = \frac{1}{\left(\displaystyle 4\delta t \sum_{n=1}^{n_{max}} \frac{n^2}{\mathrm{var}\,\delta_n}\right)^2} \cdot \sum_{m,n=1}^{n_{max}} \frac{mn}{\mathrm{var}\,\delta_n \cdot \mathrm{var}\,\delta_m} \cdot \mathrm{cov}\,\delta_{n,m}$$

Equation 3

which also contains sums that may only be evaluated numerically.

A plot of var est(D)$_{nmax}$ using these two methods (unweighted LSQ, red trace, and wLSQ, blue trace) is shown in figure 5. While they are expected to and do intersect at n_{max}=1 (no fit performed, system not overdetermined), there is a visible and substantial reduction in the variance of the estimator as the weights are included (from ~0.7 in the unweighted case, n_{max}=N-1, to ~0.1 in the weighted case). The variance of est$_{wLSQ}$ $(D)_{nmax}$ still increases monotonically, though slowly, with n_{max}, which indicates that the correlation between the δ_n values still overshadows the effect of potentially having more information by including the larger-step displacements.

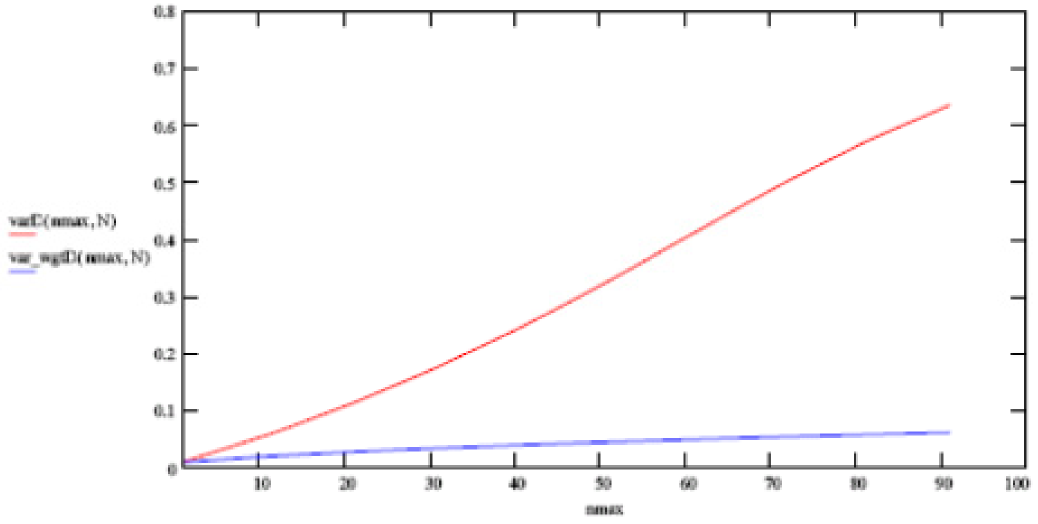


Figure 5. var est$_{wLSQ}$$(D)_{nmax}$ as a function of n_{max} as obtained with unweighted LSQ (red trace) and respectively weighted LSQ (blue trace).

c. Generalized least squares fitting

Finally, the last improvement to the LSQ estimator that I will present uses the generalized LSQ (gLSQ) method (Takeaki and Kurata 2004). The Gauss-Markov theorem ensures this approach always finds the best linear unbiased estimator (i.e. that with the lowest variance among all linear unbiased estimators). While that minimal variance may or may not equal the Cramér-Rao bound (the minimal variance theoretically attainable of all estimators) (Rao 1973), since a non-linear estimator may perform better than gLSQ, it is instructive to see how this method compares to the two other LSQ estimators discussed above. For practical purposes, the gLSQ method is equivalent to decorrelating the δ_n values by use of the full covariance matrix (matrix elements defined by Equation 1, section I.2.a), then by performing a LSQ fit on the resulting decorrelated data. The resulting equation that needs to be solved to determine the estimator is:

$$\begin{pmatrix} 1 \\ 2 \\ \vdots \\ n_{\max} \end{pmatrix}^{T} \cdot \mathbf{\Sigma}^{-1} \cdot \begin{pmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_{n_{\max}} \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ \vdots \\ n_{\max} \end{pmatrix}^{T} \cdot \mathbf{\Sigma}^{-1} \cdot \begin{pmatrix} 1 \\ 2 \\ \vdots \\ n_{\max} \end{pmatrix} \cdot \delta_{est}$$

Equation 1

Using exclusively matrix notation, this expression can be rewritten as (see table of symbols for notations):

$$\mathbf{N}^{T} \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{\Delta} = \mathbf{N}^{T} \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N} \cdot \delta_{est}$$

Equation 2

Following mathematical manipulation to simplify this expression (Appendix 1.a), this becomes:

$$\operatorname{est}_{\substack{gLSQ \\ n_{\max}}}(D) = \frac{\delta_1}{4\delta t}$$

Equation 3

Since this is identical to the unweighted LSQ fit estimator ($n_{max} = 1$), it is also unbiased. Its variance remains:

$$\operatorname{var} \operatorname{est}_{\substack{gLSQ \\ n_{\max}}}(D) = \frac{D^2}{N-1}$$

Equation 4

d. Comparison of least squares fitting methods and concluding remarks.

It is noteworthy that the generalized LSQ estimator is identical with the unweighted LSQ estimator of D for $n_{max} = 1$. As stated before, the generalized LSQ method yields the best linear unbiased estimator, and these results indicate that the best estimate of D will not be obtained by doing a linear fit to this plot, but rather by taking the more simplistic, yet more accurate, approach of using the $\delta_1/4\delta t$ as the estimator of choice. Certainly, the asymptotic linearity of a plot similar to that shown in figure 2 is a clear indication of Brownian behavior and should still be used to prove it. Work by Christian Vestergård (Vestergård 2009) also confirms that this estimator is also efficient as defined by the Cramér-Rao criterion (i.e. it has the least variance of all, not just linear, unbiased estimators).

3. Handling real-life data

a. Introduction

An additional complication to the model, which nonetheless brings it more in line with the experimental reality, is the inclusion of experimental noise (Mortensen et al.; Anderson et al. 1992b, 1992a; Mueller et al. 2003). As an approximation, this is modeled as a Gaussian-distributed misreading of the particle coordinates:

$$\vec{r}_i = \vec{r}_i^{\,th} + \vec{\eta}_i$$

Equation 1

thus:

$$\delta_n \vec{r}_i = \delta_n \vec{r}_i^{\,th} + \vec{\eta}_{i+n} - \vec{\eta}_i$$

Equation 2

where $\delta_n \vec{r}_i^{\,th}$ is the n-step displacement discussed above in the absence of measurement error, and $\vec{\eta}_i$ is the measurement error, presumed Gaussian distributed with mean zero and variance σ_{exp}^2:

$$\vec{\eta}_i = \begin{pmatrix} \eta_{x,i} \\ \eta_{y,i} \end{pmatrix} \quad \text{with both } \eta_{x,i},\ \eta_{y,i} \sim N(0, \sigma_{exp}^2)\ \forall i = 1,2..N$$

As before,

$$\left\langle \left(\delta_n r^{th}\right)^2 \right\rangle = \left\langle \left(\vec{r}^{\,th}(t + n\delta t) - \vec{r}^{\,th}(t)\right)^2 \right\rangle = 4nD\,\delta t$$

but

$$\left\langle \delta_n r^2 \right\rangle = \left\langle \left(\delta_n \vec{r}_i^{\,th} + \vec{\xi}_{i+n} - \vec{\xi}_i\right)^2 \right\rangle = 4nD\,\delta t + 2\sigma_{exp}^2$$

Equation 3

Hence, just as before, when δ_n is plotted against *n* a straight line with a slope of $4D\delta t$ is expected. Due to the measurement error, however, the y-intercept is no longer 0, but rather it is $2\sigma_{exp}^2$.

b. Comparison of the various LSQ methods

As before I will consider increasingly complex LSQ estimators of *D*: unweighted (uwLSQ), weighted (wLSQ), and generalized (gLSQ). They are easily shown to be unbiased, but to determine the variance (and thus the efficiency of the estimator) the covariance matrix is first needed. This is just the covariance matrix derived above in the $\sigma_{exp}=0$ case with a few extra σ_{exp}-dependent terms added (Appendix 1.a):

$$\mathrm{cov}(\delta_{n,m}) = \begin{cases} \mathrm{cov}(\delta^{th}_{n,m}) + \dfrac{4\sigma^4_{exp}}{N-m}\left(4+2\dfrac{\delta_{m,n}-m}{N-n}\right) + \dfrac{32mD\delta t\sigma^2_{exp}}{N-m} & \text{if } N > m+n \\ \mathrm{cov}(\delta^{th}_{n,m}) + \dfrac{8\sigma^4_{exp}+32mD\delta t\sigma^2_{exp}}{N-m} + \dfrac{8\sigma^4_{exp}\delta_{m,n}}{(N-m)(N-n)} & \text{otherwise} \end{cases}$$

Equation 1

As before, this expression only provides the covariance matrix elements for *m≤n*; the symmetry of the covariance matrix is to be used to derive the other half. As expected, this expression reduces to the previously derived covariance matrix when σ_{exp} is set to zero. To calculate the unweighted LSQ estimators for *D* and σ_{exp}^2, it follows from Equation 3 that the expression to be minimized is:

$$\sum_{n=1}^{n_{max}}\left(\overline{\delta_n r^2} - 4nD_{est}\,\delta t - 2\sigma^2_{exp}\right)^2 = \min$$

thus (Appendix 1.a):

$$\begin{cases} \mathrm{est}_{\substack{uwLSQ\\n_{max}}}(D) = 3\dfrac{\sum\limits_{n=1}^{n_{max}} n\overline{\delta_n r^2} - \dfrac{n_{max}+1}{2}\sum\limits_{n=1}^{n_{max}}\overline{\delta_n r^2}}{(n_{max}-1)n_{max}(n_{max}+1)\delta t} \\ \mathrm{est}_{\substack{uwLSQ\\n_{max}}}(\sigma^2_{exp}) = \dfrac{(2n_{max}+1)\sum\limits_{n=1}^{n_{max}}\overline{\delta_n r^2} - 3\sum\limits_{n=1}^{n_{max}} n\overline{\delta_n r^2}}{(n_{max}-1)n_{max}} \end{cases}$$

Equation 2

Similarly (Appendix 1.a):

$$\begin{cases} \underset{n_{max}}{est_{uwLSQ}}(D) = \dfrac{\displaystyle\sum_{m,n=1}^{n_{max}} n \frac{\overline{\delta_n r^2} - \overline{\delta_m r^2}}{var\left(\overline{\delta_n r^2}\right) var\left(\overline{\delta_m r^2}\right)}}{\displaystyle 4\delta t \sum_{m,n=1}^{n_{max}} n \frac{n-m}{var\left(\overline{\delta_n r^2}\right) var\left(\overline{\delta_m r^2}\right)}} \\ \underset{n_{max}}{est_{uwLSQ}}(\sigma^2_{exp}) = \dfrac{\displaystyle\sum_{m,n=1}^{n_{max}} n \frac{\overline{\delta_n r^2} + \overline{\delta_m r^2}}{var\left(\overline{\delta_n r^2}\right) var\left(\overline{\delta_m r^2}\right)}}{\displaystyle 2 \sum_{m,n=1}^{n_{max}} n \frac{n+m}{var\left(\overline{\delta_n r^2}\right) var\left(\overline{\delta_m r^2}\right)}} \end{cases}$$

Equation 3

Finally, using the matrix notation to derive the expression for the gLSQ estimators of *D* and $\sigma_{exp}{}^2$:

$$\begin{pmatrix} 1 & 1 \\ 1 & 2 \\ \vdots & \vdots \\ 1 & n_{max} \end{pmatrix}^T \cdot \mathbf{\Sigma}^{-1} \cdot \begin{pmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_{n_{max}} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ \vdots & \vdots \\ 1 & n_{max} \end{pmatrix}^T \cdot \mathbf{\Sigma}^{-1} \cdot \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ \vdots & \vdots \\ 1 & n_{max} \end{pmatrix} \cdot \begin{pmatrix} 2\sigma^2_{exp} \\ \delta_{est} \end{pmatrix}$$

As before,

$$\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{\Delta} = \mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N} \cdot \mathbf{P}_{est}$$

thus:

$$\mathbf{P}_{est} = \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N}\right)^{-1} \cdot \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{\Delta}\right)$$

Equation 4

The variances of the estimators of *D* above are:

$$var\, \underset{n_{max}}{est_{uwLSQ}}(D) = \left(\frac{3}{(n_{max}-1)n_{max}(n_{max}+1)\delta t} \right)^2 \cdot \sum_{m,n=1}^{n_{max}} cov\, \delta_{n,m} \left[mn - n(n_{max}+1) + \left(\frac{n_{max}+1}{2} \right)^2 \right]$$

Equation 5

and

$$\operatorname{var}\underset{n_{\max}}{\operatorname{est}}_{wLSQ}(D) = \left(4\delta t \sum_{m,n=1}^{n_{\max}} n \frac{n-m}{\operatorname{var}(\delta_n)\operatorname{var}(\delta_m)}\right)^2 \sum_{m,n=1}^{n_{\max}} \frac{\operatorname{cov}\delta_{n,m}}{\operatorname{var}(\delta_n)\operatorname{var}(\delta_m)} \cdot$$
$$\left[mn\left(\sum_{p=1}^{n_{\max}} \frac{1}{\operatorname{var}(\delta_p)}\right)^2 - 2n\sum_{p=1}^{n_{\max}} \frac{1}{\operatorname{var}(\delta_p)}\sum_{p=1}^{n_{\max}} \frac{p}{\operatorname{var}(\delta_p)} + \left(\sum_{p=1}^{n_{\max}} \frac{p}{\operatorname{var}(\delta_p)}\right)^2\right]$$

Equation 6

Finally,

$$\operatorname{var}\underset{n_{\max}}{\operatorname{est}}_{gLSQ}(D) = \frac{1}{16\,\delta t^2}\begin{pmatrix}0\\1\end{pmatrix}^T \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N}\right)^{-1}$$

Equation 7

c. *Conclusion*

To conclude, in a practical setting, the various estimates of *D* and σ_{exp} and their corresponding variances can only be evaluated numerically due to the dependence of the covariance matrix on these two parameters. However, as in the previous sections, it remains true that a full-scale LSQ fit to the entire dataset is unnecessary, and that the slope and the intercept of a straight line passing through δ_1 and δ_2 is sufficient to optimally determine the σ_{pos} and *D* parameters from an unweighted LSQ fit (based on numerical simulations).

4. *Simple persistent random motion*

In this section I introduce persistent random motion, which adds to the simple Brownian motion discussed above a layer of complexity: persistent motion. This simply means that overlaid to the random displacements due to thermal forces, which characterize with Brownian motion, is an additional non-random, systematic displacement due to inertia. Below I introduce the mathematical formalism of the standard Orstein-Uhlenbeck (OU) process, proposed in 1919 (Ornstein 1919; Uhlenbeck and Ornstein 1930). This process has been used with great success in modeling cell motility and current studies add various subtleties to it to account for cell- or surface-specific behavior (Grădinaru et al.; Selmeczi et al. 2005). The types of cell motility models proposed in the remainder of this thesis all build up on the standard OU process.

a. *The standard Ornstein-Uhlenbeck (OU) process*

Mathematical motility models of cell behaviour have remained relatively simple over many decades. The standard model still used today is the Ornstein-Uhlenbeck (OU) process (Ornstein 1919; Uhlenbeck and Ornstein 1930), an old model inspired by Brownian motion. It is formally equivalent to the Langevin equation describing Brownian motion [equation 1, section II.1.a] in the case where the inertial forces are no longer negligible:

$$m\frac{d^2\vec{r}}{dt^2} = -\gamma\frac{d\vec{r}}{dt} + \vec{F}_{therm}(t)$$

Equation 1

Dividing by γ yields the differential stochastic equation that defines the OU process:

$$P\frac{d\vec{v}(t)}{dt} = -\vec{v}(t) + \sqrt{2D}\vec{\xi}(t)$$

Equation 2

Here *P* is the persistence time of the motion and D is the diffusion coefficient of the OU process. In the context of cell motility, where the random component of the motion is due to factors internal to the cell (cytoskeleton rearrangements such as those associated with fillopodia and lamellipodia extensions), D is called the motility coefficient of the microorganism. Similarly, the first term of the right-hand side describes how the cell is slowed down in the direction opposite to that of its motility due to adhesion and retraction of focal adhesion sites. Unlike the process described by equation 1, this is also not due to factors external to the cell (such as friction or drag force), but due to a "memory kernel" of the cell cytoskeleton (Selmeczi et al. 2005). In this interpretation, the persistence time *P* refers to the rate of cytoskeleton "memory" loss during cell migration.

A quantity frequently encountered when dealing with persistent random motion is the velocity autocovariance (autocovariance) function, defined by:

$$\phi(t) = <\vec{v}(t)\cdot\vec{v}(0)>$$

Equation 3

By integrating Equation 2 twice the velocity autocovariance for an OU process is (Selmeczi et al. 2005):

$$\phi(t) = <v^2> e^{-|t|/P} = \frac{nD}{P}e^{-|t|/P}$$

Equation 4

Here *n* is the dimension of the space which supports motility, which in the case of this study is 2, and ϕ [m^2/s^2] is the auto-correlation, which describes the correlation between the velocity at time 0 and the velocity at time *t*.

5. *Estimation of motility parameters from experimental-like data*

This section bridges the theory presented insofar with real-life experiments and shows how experimental data should be handled, and what type of information can be extracted from it. As before, real-life experimental effects such as the discretization of time and positional noise are taken into account. A method for optimally determining the motility parameters, which makes use of power spectra analysis, is introduced. Additionally, a simplified procedure (however, not as accurate as the power spectra method), is described in detail. For simplicity, as well as to allow for the possibility to check the results against simulations, most of the results below refer directly to the standard OU process. The last item of this section introduces some of the variants of the OU process that best describe real-life cell motility data, and how to handle them.

a. Time discretization and positional noise

Since much of the parameter estimation is done using velocity, rather than positional data (see sections b. and c. below), care must be taken when considering another effect: time discretization. Experimental data is available in the form of particle positions recorded in time. Typically this is done by collecting time-lapse videos of a set of particles with a constant recording rate (e.g. a finite Δt between successive frames). In this case, the instantaneous velocities of the particle, v(t) are not available and they can be determined only to an approximation:

$$\vec{u}_j = \frac{\vec{r}_j - \vec{r}_{j-1}}{\Delta t}$$

I call these quantities *secant velocities* and use the letter *u* to denote them. The use of secant, instead of instantaneous velocity, has a direct effect on the velocity autocovariance function. Defining the discrete (and experimentally available) quantity:

$$\phi_j^u = \left\langle \vec{u}_{j+1} \cdot \vec{u}_1 \right\rangle$$

Equation 1

and noting that $\vec{u}_j = \int_{t_{j-1}}^{t_j} \vec{v}(t)dt$ and $\vec{u}_1 = \int_0^{\Delta t} \vec{v}(t)dt$,

$$\phi_j^u = \left\langle \vec{u}_{j+1} \cdot \vec{u}_1 \right\rangle = \int_{t_j}^{t_{j+1}} \int_0^{\Delta t} \left\langle \vec{v}(t')\vec{v}(t) \right\rangle dt dt' = \int_{t_j}^{t_{j+1}} \int_0^{\Delta t} \phi(t) dt dt'$$

Equation 2

Since for the standard OU process the analytical expression for ϕ(t) is given by eq. 4, section I.4.a, the integral can be evaluated analytically.

$$\phi_j^u = \frac{2nPD}{(\Delta t)^2} \cdot \left(\cosh\frac{\Delta t}{P} - 1 \right) \cdot e^{-t_j/P} - \frac{2nD}{\Delta t}\left(1 - \frac{P}{\Delta t}\sinh\frac{\Delta t}{P} \right) \cdot \delta_{j,0}$$

Equation 3

Here $\delta_{j,0}$ is the Kroneker-δ. Plots of φ_j^u and $\varphi(t_j)$ are shown in figure 1, which illustrates how φ_j^u is proportional to $\varphi(t_j)$, with a proportionality factor directly related to the c=Δt/P ratio: $2(\cosh c - 1)/c^2$. The only exception arises for the first point of the velocity autocovariance, where a constant offset of -2(c - sinh c)/c^2 is applied. Indeed, in the limit $\Delta t \rightarrow 0$, $u \rightarrow v$ and $\varphi_j^u \rightarrow \varphi(t_j)$. This remains true in the case of non-OU processes where φ(t) follows monoexponential behavior, as that is all that is needed to evaluate the integral in equation 2.

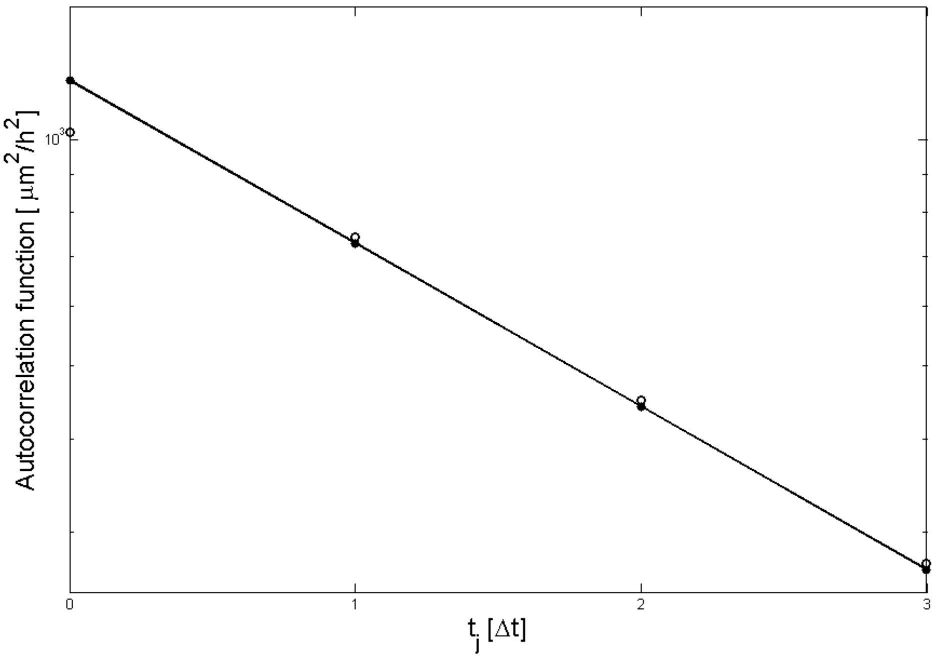


Figure 1: Instantaneous velocity autocovariance function (black dots and connecting solid line) and secant velocity autocovariance function (hollow dots).

The complication of positional noise has already been introduced in section II.3.a. Here, too, this is modeled as a Gaussian-distributed misreading of the particle coordinates:

$$\vec{r}_i = \vec{r}_i^{\,th} + \vec{\eta}_i$$

Equation 4

I describe below a simple method for estimating the positional noise for a standard OU process which involves the use of Fürth's formula (Fürth 1920). This is an expression for the mean square displacement of a motile cell, which has been routinely used to describe cell motility. It is not limited to motility models following the standard OU process; instead, it is generally applicable to motility models whose velocity autocovariance function follows a simple mono-exponential dependence such as those encountered in chapter III of this thesis.

On a two-dimensional surface, Fürth's formula takes the following form:

$$\left\langle (\vec{r}(t) - \vec{r}(0))^2 \right\rangle = 4D\left(t - P\left(1 - e^{-t/P}\right)\right)$$

Equation 5

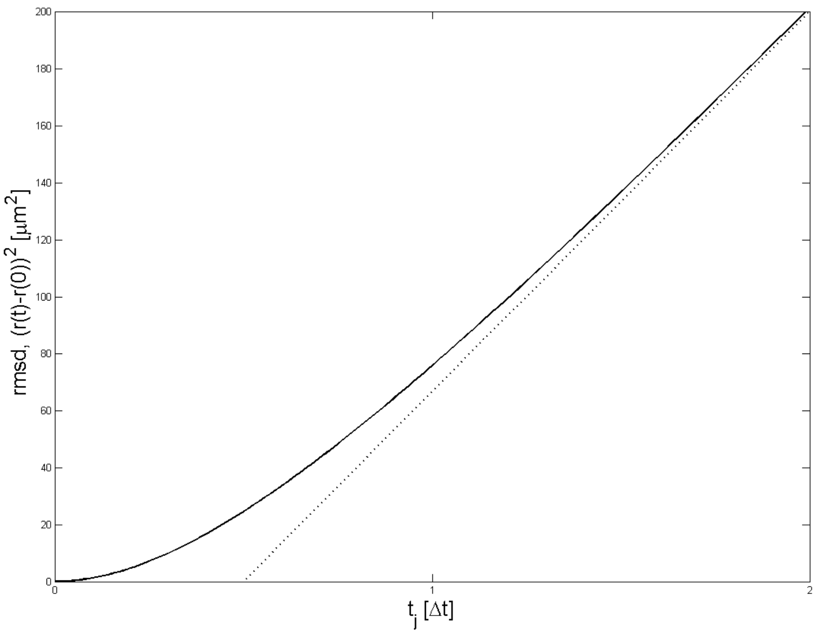


Figure 2: Fürth's formula for the mean squared displacement of persistent random motion described by the OU-process. Full line: Mean squared displacement from equation 2; Dotted line: Asymptotic behavior. The dotted line intersects the time axis at t = P.

Here I expand Fürth's formula to account for the effect of positional noise on the motility data:

$$\left\langle \left(\vec{r}_{exp}(t_j) - \vec{r}_{exp}(0)\right)^2 \right\rangle = \left\langle \left(\vec{r}(t_j) - \vec{r}(0) + \vec{\eta}_j - \vec{\eta}_0\right)^2 \right\rangle = \left\langle \left(\vec{r}(t_j) - \vec{r}(0)\right)^2 \right\rangle + 4\sigma_{pos}^2$$

thus:

$$\left\langle \left(\vec{r}_{exp}(t_j) - \vec{r}_{exp}(0)\right)^2 \right\rangle = 4D\left(t_j - P\left(1 - e^{-t_j/P}\right)\right) + 4\sigma_{pos}^2$$

Equation 6

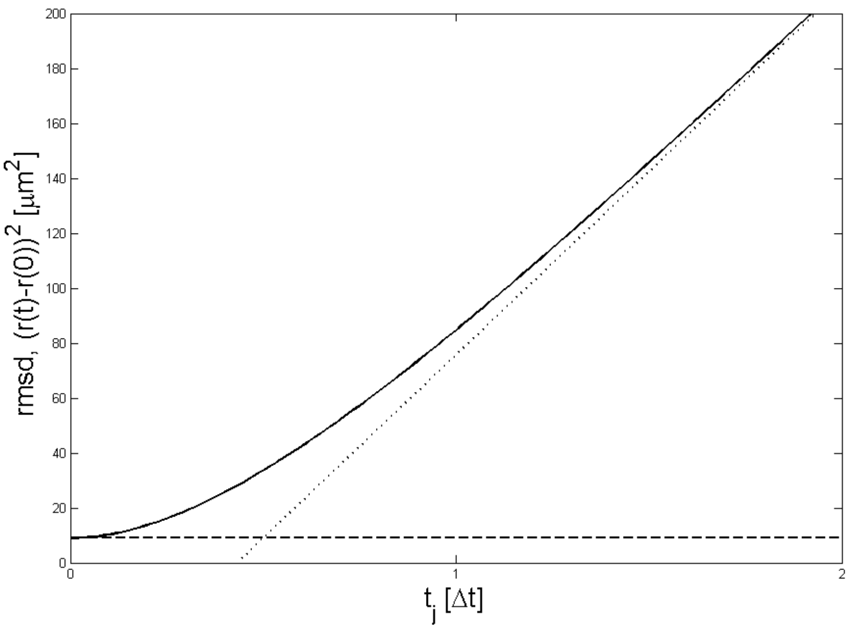


Figure 3: Fürth's formula for the mean squared displacement of persistent random motion described by the OU-process. Full line: Mean squared displacement from equation 2; Dotted line: Asymptotic behavior at t→∞. The dotted line intersects the time axis at t = P. Dashed line: Asymptotic behavior at t→0. The dashed line intersects the rmsd axis at $4\sigma_{pos}{}^2$.

Weigthed LSQ-fitting a plot of mean square displacement of a motile cell to Equation 3 provides a reliable measure of the positional noise. To test this, the trajectories of 50 independent motile cells following a standard OU process with parameters typical of fibroblasts have been simulated over 2 hours as described in Appendix D of (Nørrelykke and Flyvbjerg 2010), yielding experimental-like data from 128 frames. The MATLAB code used to generate this dataset may be found in Appendix 4.b. While the estimates for P and D are imprecise (25% error for P, 28% for D), the positional error can be determined remarkably accurately from this fit (0.9978 ± 0.0008 μm vs. 1 μm input). This is because the error bars of $\left(\vec{r}_{exp}(t)-\vec{r}_{exp}(0)\right)^2$ are correlated, thus the fit to Fürth's formula is not correctly weighed. As an asymptote of Fürth's formula as t→0 (where the data is the least correlated and thus the fit most precise), σ_{pos} can be determined much more accurately.

The positional measurement error also has an effect on the autocovariance velocity function. From equation 4,

$$\vec{u}_j = \frac{\vec{r}_j - \vec{r}_{j-1}}{\Delta t} = \frac{\vec{r}_j^{\,th} - \vec{r}_{j-1}^{\,th}}{\Delta t} + \frac{\vec{\eta}_j - \vec{\eta}_{j-1}}{\Delta t} = \vec{u}_j^{\,th} + \frac{\vec{\eta}_j - \vec{\eta}_{j-1}}{\Delta t}$$

where $\vec{\eta}_i$ is the measurement error:

$$\vec{\eta}_i = \begin{pmatrix} \eta_{x,i} \\ \eta_{y,i} \end{pmatrix} \quad \text{with both } \eta_{x,i},\ \eta_{y,i} \sim N(0,\sigma_{exp}^2)\ \forall i = 1,2..N$$

In this case, the velocity autocovariance function becomes:

$$\phi_j^u = \left\langle \vec{u}_{j+1} \cdot \vec{u}_1 \right\rangle = \left\langle \left(\vec{u}_j^{th} + \frac{\vec{\eta}_j - \vec{\eta}_{j-1}}{\Delta t} \right) \cdot \left(\vec{u}_1^{th} + \frac{\vec{\eta}_1 - \vec{\eta}_0}{\Delta t} \right) \right\rangle$$

$$\phi_j^u = \phi_j^{th,u} + \frac{\left\langle \left(\vec{\eta}_j - \vec{\eta}_{j-1} \right) \cdot \left(\vec{\eta}_1 - \vec{\eta}_0 \right) \right\rangle}{\Delta t^2}$$

thus

$$\phi_0^u = \frac{2nPD}{(\Delta t)^2} \cdot \left(\cosh \frac{\Delta t}{P} - 1 \right) \cdot e^{-t_j/P} - \frac{2nD}{\Delta t} \left(1 - \frac{P}{\Delta t} \sinh \frac{\Delta t}{P} \right) + \frac{4\sigma_{pos}^2}{(\Delta t)^2}$$
$$\phi_1^u = \frac{2nPD}{(\Delta t)^2} \cdot \left(\cosh \frac{\Delta t}{P} - 1 \right) \cdot e^{-t_j/P} - \frac{2\sigma_{pos}^2}{(\Delta t)^2}$$
$$\phi_j^u = \frac{2nPD}{(\Delta t)^2} \cdot \left(\cosh \frac{\Delta t}{P} - 1 \right) \cdot e^{-t_j/P} \quad \text{for } j > 1$$

Equation 7

Plots of φ_j^u, $\varphi_j^{th,u}$ and $\varphi(t_j)$ are shown in figure 4, which differs from figure 1 by the addition of the φ_j^u with positional noise (upward triangles). The positional noise's only effect on the autocovariance function is to raise the first point of by $4\sigma_{pos}^2/(\Delta t)^2$ and to lower the second point by $2\sigma_{pos}^2/(\Delta t)^2$. This is true for the case of non-OU processes, and in general for all forms of the autocovariance function where it is uncorrelated with the positional noise.

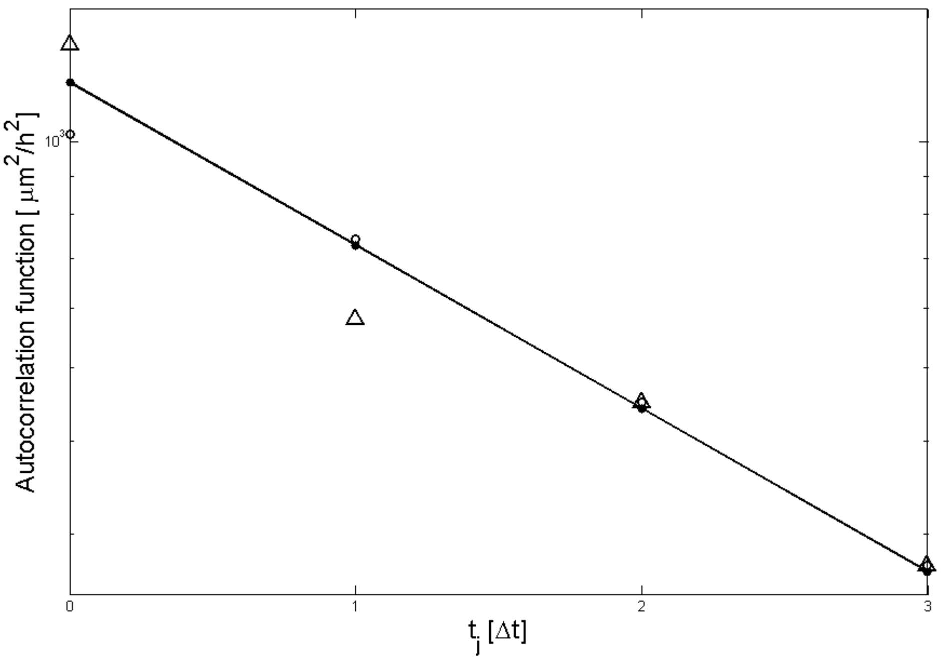


Figure 4: Instantaneous velocity autocovariance function (black dots and connecting solid line) and secant velocity autocovariance function (hollow dots). The secant velocity autocovariance function for data subject to positional noise is shown as hollow triangles.

It is important to note here that, the first two points of the autocovariance function aside, all the remaining points follow the behavior of the instantaneous velocity autocovariance function, except for a factor which depends on Δt/P:

$$\phi_j^u = 2\left(\frac{P}{\Delta t}\right)^2 \cdot \left(\cosh\frac{\Delta t}{P} - 1\right) \cdot \phi(j \cdot \Delta t) \text{ for } j > 1$$

Equation 8

Since the amplitude of φ(t) is $<v^2>$, the mean squared velocity of the cells (equation 4.a.4), the amplitude of φ^u is proportional to $<v^2>$, the proportionality constant dependent on the Δt/P ratio. As this ratio vanishes, $u_j \rightarrow v(t_j)$ and $\varphi^u \rightarrow \varphi(j \cdot \Delta t)$, the proportionality factor becomes 1.

b. Optimizing the estimators: Power spectral analysis

I have developed a new method that provides optimal accuracy and precision for extracting the motility parameters P and D from experimental-like motility data that follows the simple Ornstein-Uhlenbeck process. Since the experimentally measured data is cell position, $\vec{r}(t)$, a theoretical expression relating these positions with the parameters of interest D and P is needed. The defining equation of the standard OU process (equation 2, section II.4.a) and the definition of instant velocity $\vec{v} = d\vec{r}/dt$ form a system of differential equations.

Appendix D of (Nørrelykke and Flyvbjerg 2010) shows the general solutions for the trajectory of a Einstein-OU process, namely one that involves Hookean as well as Stokes forces and Brownian motion, and that shown here adapted for the specific case of a simple OU process (i.e. no Hookean component):

$$\begin{pmatrix} x_{j+1} \\ v_{j+1} \end{pmatrix} = \begin{bmatrix} 1 & P \cdot (1-c) \\ 0 & c \end{bmatrix} \cdot \begin{pmatrix} x_j \\ v_j \end{pmatrix} + \begin{pmatrix} \Delta x_j \\ \Delta v_j \end{pmatrix}$$

Equation 1

A constant recording frequency Δt between successive frames is assumed, and the shorthand notation c=exp(-Δt/P) is also employed. For simplicity a 1-D model has been considered, so the vector signs of the position x has been dropped. Equation 1 is thus a stochastic difference equation which relates the position-velocity vector in frame j with that in the following frame j+1. The stochastic component is present in the last term of equation 1:

$$\begin{pmatrix} \Delta x_j \\ \Delta v_j \end{pmatrix} = \Delta x_{j,+} \cdot \begin{pmatrix} -1 \\ 1/P \end{pmatrix} + \Delta x_{j,-} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Equation 2

where $\Delta x_{j,\pm}$ are stochastic variables defined as follows:

$$\Delta x_{j,+} = \sqrt{2D}\int_{t_j}^{t_{j+1}} e^{-(t_{j+1}-t')}\xi(t')dt'$$

$$\Delta x_{j,-} = \sqrt{2D}\int_{t_j}^{t_{j+1}} \xi(t')dt'$$

Hence they are random, correlated Gaussian variables, with expectation value zero and covariance matrix:

$$\left\langle \Delta x_{j,+}, \Delta x_{j,-} \right\rangle = \begin{bmatrix} DP(1-c) & 2DP(1-c) \\ 2DP(1-c) & 2D\Delta t \end{bmatrix}$$

They can be decorrelated by introducing the random uncorrelated Gaussian variables $\eta_j^{(a)}$, $\eta_j^{(b)}$ ~N(0,1):

$$\begin{pmatrix} \Delta x_j \\ \Delta v_j \end{pmatrix} = \sqrt{1+\alpha}\cdot\eta_j^{(a)}\cdot\begin{pmatrix} A_- - A_+ \\ A_+/P \end{pmatrix} + \sqrt{1-\alpha}\cdot\eta_j^{(b)}\cdot\begin{pmatrix} -A_- - A_+ \\ A_+/P \end{pmatrix}$$

Equation 3

where the shorthand notations

$$A_+ = \sqrt{\frac{DP}{2}(1-c^2)};\quad A_- = \sqrt{D\Delta t};\quad \alpha = \sqrt{\frac{2P}{\Delta t}\cdot\frac{1-c}{1+c}}$$

have been introduced. This allows *in silico* numerically exact simulations of a process following the standard OU process (MATLAB code shown in Appendix 4.b).

Since experimental data consists of cell coordinates x_j, the instantaneous velocities v_j are not available, but the secant velocities can be defined as $u_j=(x_{j+1}-x_j)/\Delta t$. Substituting this into the difference equation 1 the following difference stochastic equation in u_j results:

$$u_{j+1} = cu_j + \Delta u_j$$

Equation 4

where Δu_j is now the stochastic term:

$$\Delta u_j = \frac{P(1-c)\Delta v_j + \Delta x_{j+1} - \Delta x_j}{\Delta t}$$

Equation 5

The theoretical power spectrum in terms of secant velocities is derived by noting that

$$P_k^{u,th} = \tilde{\phi}_k^u \qquad \text{and} \qquad P_k^{v,th} = \tilde{\phi}_k^v$$

Applying the Fourier transform to evaluate $P_k^{v,th}$:

$$P_k^{v,th} = \frac{\langle \hat{v}_k^* \hat{v}_k \rangle}{N\Delta t} = \Delta t \frac{(\Delta v)^2}{1 + c^2 + 2\cos(2\pi k / N)}$$

Equation 6

Similarly,

$$P_k^{u,th} = \frac{\langle \hat{u}_k^* \hat{u}_k \rangle}{N\Delta t} = \frac{2P^2}{(\Delta t)^2} \cdot \left(\cosh \frac{\Delta t}{P} - 1 \right) \cdot P_k^{v,th} - \frac{2nD}{\Delta t} \left(1 - \frac{P}{\Delta t} \sinh \frac{\Delta t}{P} \right)$$

Equation 7

This shows the effect of discretization on the shape of the power spectrum curve. The original Lorentzian is multiplied by a Δt/P-dependent factor (the same that marks the difference between φ^u (t) and φ^u (t), equation 8, section II.5.a) and shifted up by an amount proportional to D. In this case, since discretization effects are modeled completely, as I do here, they are no longer "errors." As can be seen in figure 1 below, based on a simulation of the standard OU process, ignoring the discretization errors can lead to significant bias in estimating the parameters D and P, especially when the persistence time is comparable to the measurement frequency Δt (as is the case in all of the preliminary data presented above).

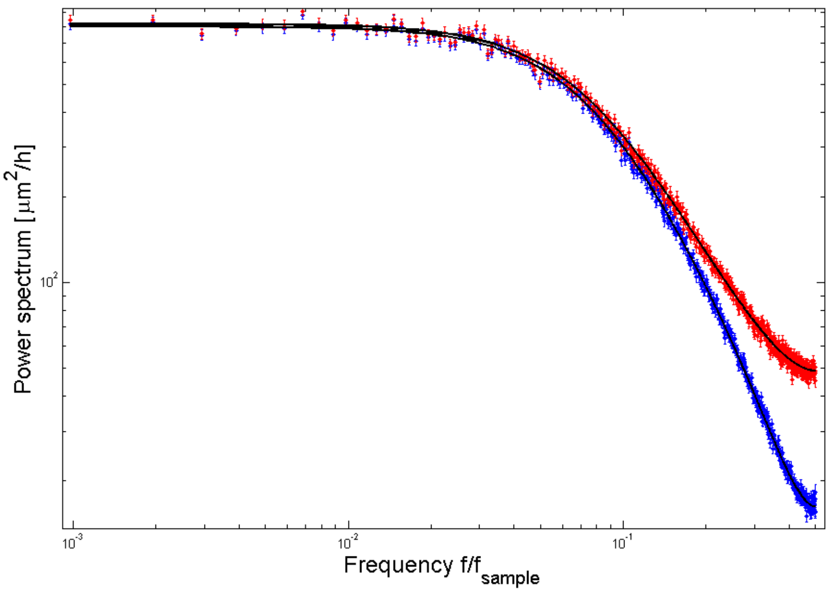


Figure 1: Comparison of $P_u(f_k)$ with $P_v(f_k)$ for a simulation of an OU-process with $\Delta t/P = 1/2$. The frequency axis is discrete because the measurement time is finite, $\Delta f = 1{=}t_{msr}$. The black traces are the theoretical expressions which overlay perfectly (noise aside) to the simulated data (red ($P_u(f_k)$) and blue ($P_v(f_k)$) traces)

The fact that positions are not recorded continuously does not affect Furth's formula in any way, except that the experimental data with is available only at discrete lapses of time, integer multiples of Δt. This is an advantage of the mean squared displacement: This statistics has no discretization "errors." On the other hand, if discretization effects can be modeled completely, then discretization effects are no longer to be seen as "errors." In that situation, one can do better than using the mean squared displacement. As can be seen in figure 1 above, ignoring the discretization errors can lead to significant bias in estimating the parameters D and P, especially when the persistence time is comparable to the measurement frequency Δt (as is the case in all of the preliminary data presented above).

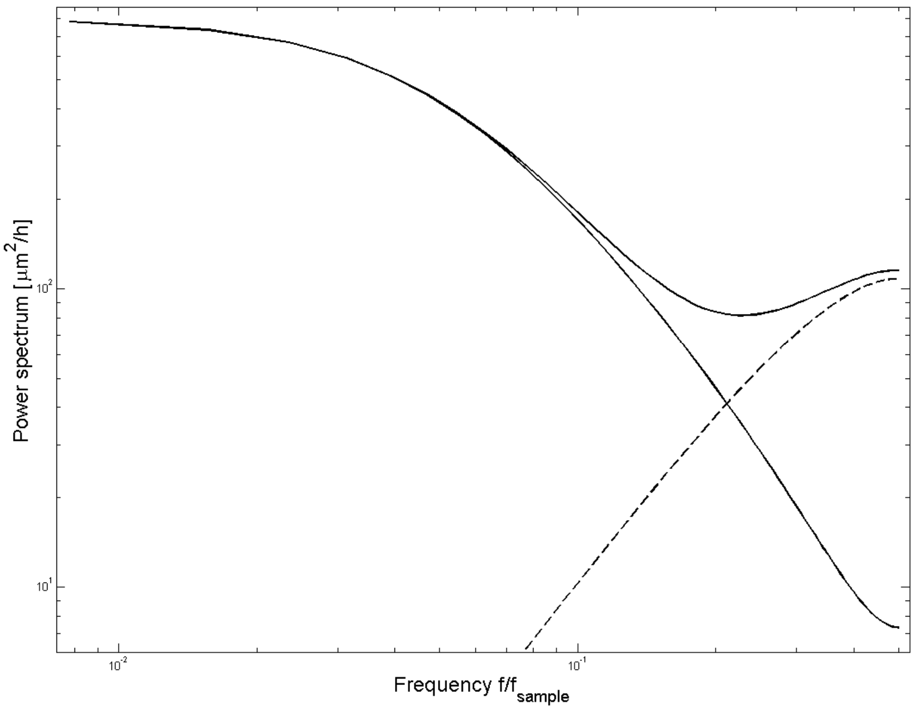


Figure 2: Effects of positional noise on the power spectra of a standard OU process. As expected, since positional noise is a high-frequency noise, it only affects the high-frequency domain of the power spectrum.

c. Simplified method for estimating motility parameters

Although the power spectra method for determining the motility parameters provides the optimal estimators, it suffers from mathematical complexity. Below I propose a more intuitive and simplified data analysis procedure, which entails estimating the motility parameters by a weighted fit of a simple exponential function to the velocity autocovariance function. This has the advantage of simplicity and time of analysis over the iterative method proposed in (Selmeczi et al. 2005). However, since the values of the autocovariance velocity calculated from experimental data are correlated and their error bars are underestimated, this procedure will necessarily yield a less accurate estimate of the persistence time.

To assess the loss of precision when using this simplified method, I generated experimental-like motility data *in silico* following the procedure described in Appendix D of (Norrelykke and Flyvbjerg 2009; Nørrelykke and Flyvbjerg 2010), and estimated the motility parameters as described above. The dataset from section 5.a. was reused, where the motility of 50 independent cells was simulated in accordance to the standard OU process, and with input parameters similar to those characteristic of the motility of real life NIH 3T3 fibroblasts on glass (see section III.2.a) : $P = 40$ min, $D = 7.3\ \mu m^2/min$ (or $\sigma_0 = (2D)^{1/2}/P = 0.0955\ \mu m/min^{3/2}$ and $\sigma_{1\perp} = \sigma_{1||} = 0$), $\Delta t = 1$ min, and $\sigma_{pos} = 1\ \mu m$. The motility of these cells was followed over nearly 2 hours, yielding experimental-like data from 128 frames. Note that in the remainder of this section I will be using a set of parameters $\sigma_0 = (2D)^{1/2}/P$ and $\sigma_{1\perp} = \sigma_{1||} = 0$ in preparation for characterizing the motility of NIH 3T3 done in section III.2.a. For the purposes of this chapter, however, the usage of σ_0 instead of D may only be regarded as change of notations.

The statistical analysis of cell motility been automated by way of a MATLAB program, data_analysis_exp.m, whose MATLAB source code is listed in Appendix 6. It reads in a list of cell centroid coordinates and calculates the motility parameters.

Below I describe how the data analysis program calculates the motility parameters in detail, as well as the other information that it outputs. The significance of these results, as well as their interpretation and possible caveats are also discussed.

As a note to the user, it is important that two parameters be changed in the body of the data_analysis_exp.m file before data processing can start. These may be found under the "Conversion parameters" section and are dt, which should be set to the frame rate of the movies, in minutes, and um2pix, which should be set to the size of a pixel in μm (this is microscope-dependent).

The filename containing the post-processed version of the cell trajectories is requested as an input. If a file named "trajectories-fixed.txt" which contains coordinates of fiduciary marks exists in the current directory, it will be used to perform a stage drift correction on the data. The corrected data (or the original data if no stage drift correction is performed) is then plotted as shown in Figure 1.

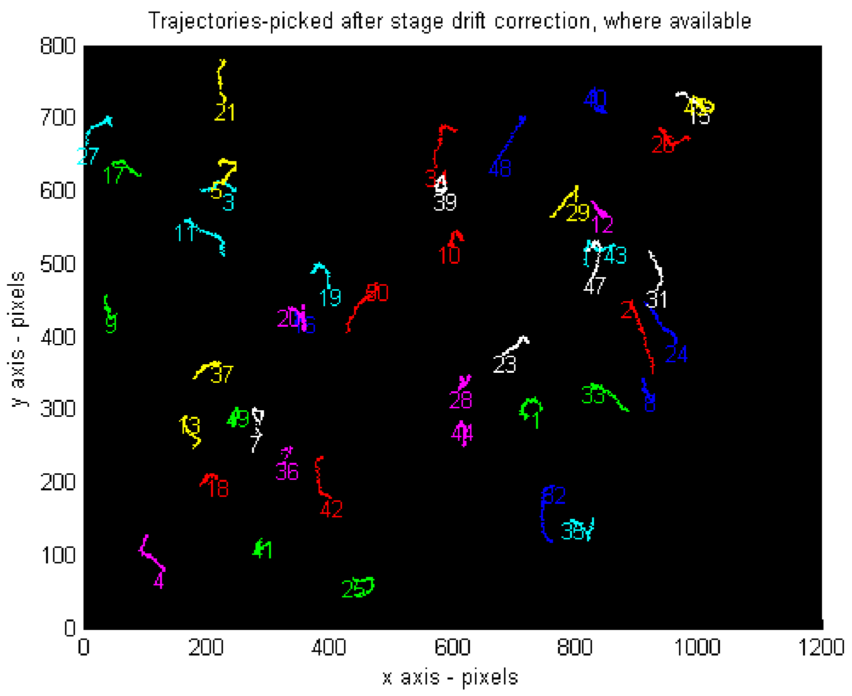


Figure 1: Graphical representation of the trajectories simulated using a standard OU process.

The value of the rsd filter is requested from the user, and I have used 15-20 µm (about half of a typical cell's diameter) as an rsd cutoff. All the trajectories passed the rsd filter. Another plot of all remaining trajectories is generated, which in this case is identical to the one shown in Figure 2.

An additional test to check that the data is drift-free is performed by calculating a frame-average secant velocity ($<u_x^2>$ and $<u_y^2>$). A plot of each $<u_x^2>$ and $<u_y^2>$ with their respective error bars vs. time is shown such as the one shown in Figure 2.

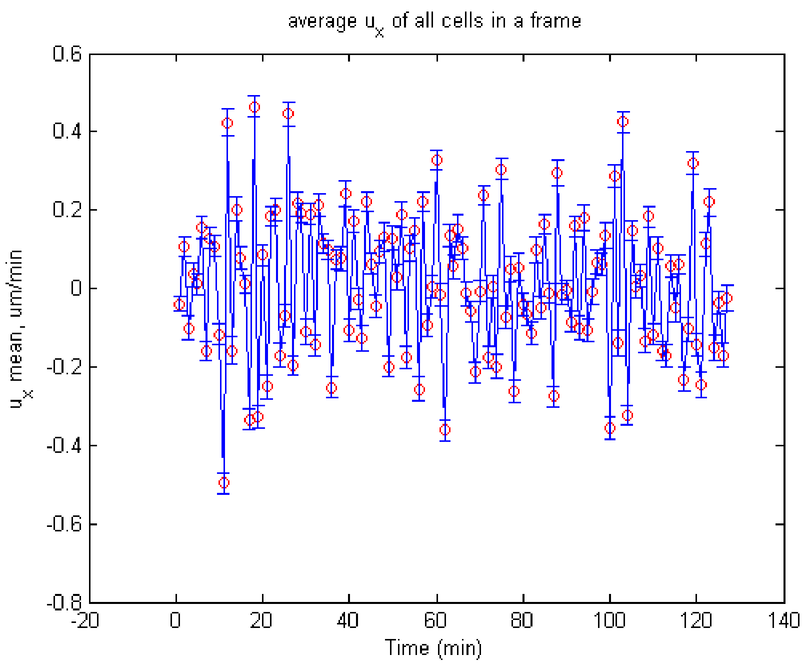
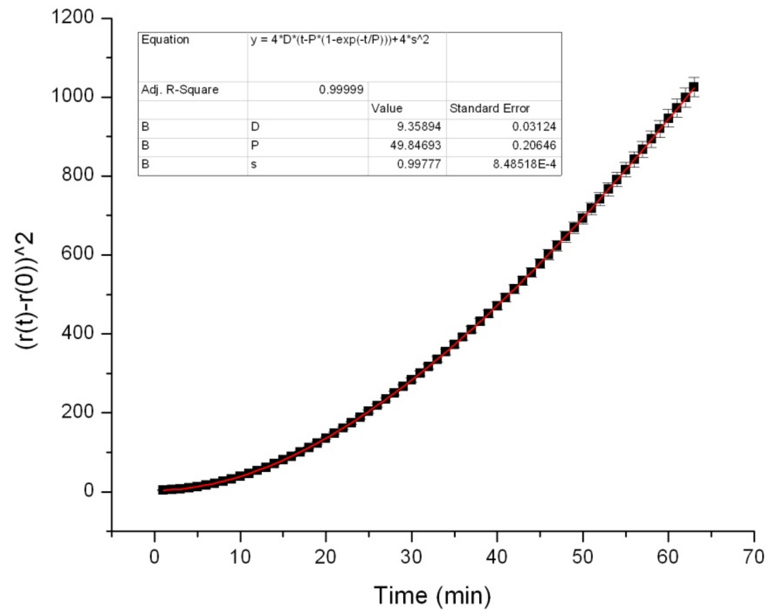


Figure 2: Average secant velocities (x component shown) in each frame, data simulated using a standard OU process.

Provided sufficiently many cells are in each frame, and given that they all move independently of one another, this should average out to zero, as seen in Figure 3. The program outputs the mean $<u_x^2>$ and $<u_y^2>$ over all frames and this is indeed reasonably close to zero within several standard deviations. Note that these are secant velocities calculated from data affected by positional noise, so it is not uncommon to see a deviation from zero by 3-4σ:

```
Time-averaged ux_mean =  0.0101 ± 0.0275 um/min
Time-averaged uy_mean =  0.0119 ± 0.0283 um/min
```

A preliminary analysis of the dataset involves plotting and fitting $\overline{\left(\vec{r}(t_j)-\vec{r}(0)\right)^2}$ to Fürth's formula. While the estimates for P and D are somewhat imprecise (24.5% error for P, 28.2% for D), the positional error can be determined remarkably accurately from this fit (0.9978 ± 0.0008 vs. 1 input). This is because the error bars of $\overline{\left(\vec{r}(t_j)-\vec{r}(0)\right)^2}$ are correlated and thus the fit to Fürth's formula is not correctly weighed. As an asymptote of Fürth's formula as t→0 (where the data is least correlated and thus the fit most precise), σ_{pos} can be determined much more accurately from this fit.



Equation	y = 4*D*(t-P*(1-exp(-t/P)))+4*s^2		
Adj. R-Square	0.99999		
		Value	Standard Error
B	D	9.35894	0.03124
B	P	49.84693	0.20646
B	s	0.99777	8.48518E-4

Figure 3: Average rmsd of cell centroids (black) and fit to Fürth's formula (red), data simulated using a standard OU process.

The compliance of the data set with the OU model is tested by plotting the perpendicular and respectively parallel components of the acceleration vs. speed. As per the standard OU process $\langle a_\perp \rangle = 0$ and $\langle a_\parallel \rangle = -v/P$ are expected, which can be seen to hold true in both Figure 4 and 5.

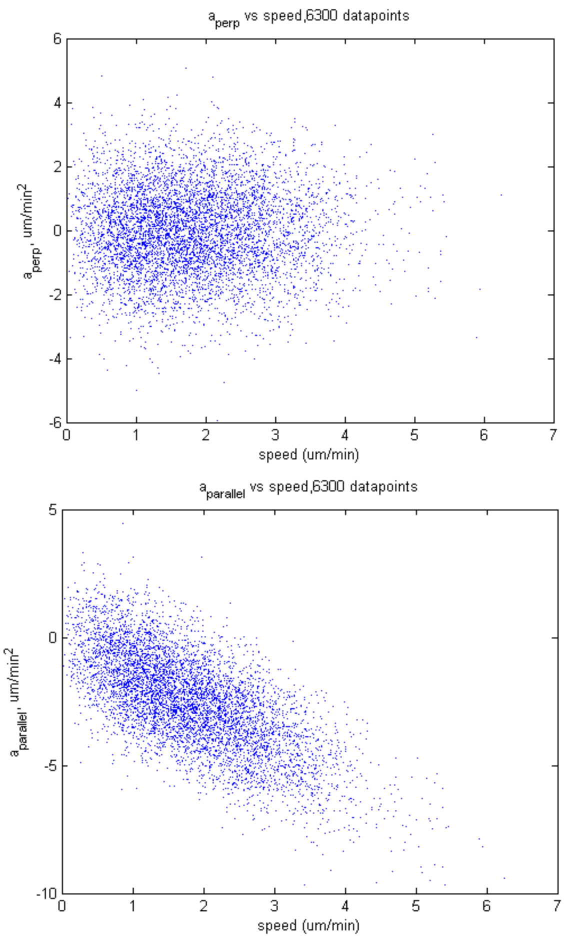


Figure 4: Acceleration vs. speed: components perpendicular (top) and parallel (bottom), data simulated using a standard OU process.

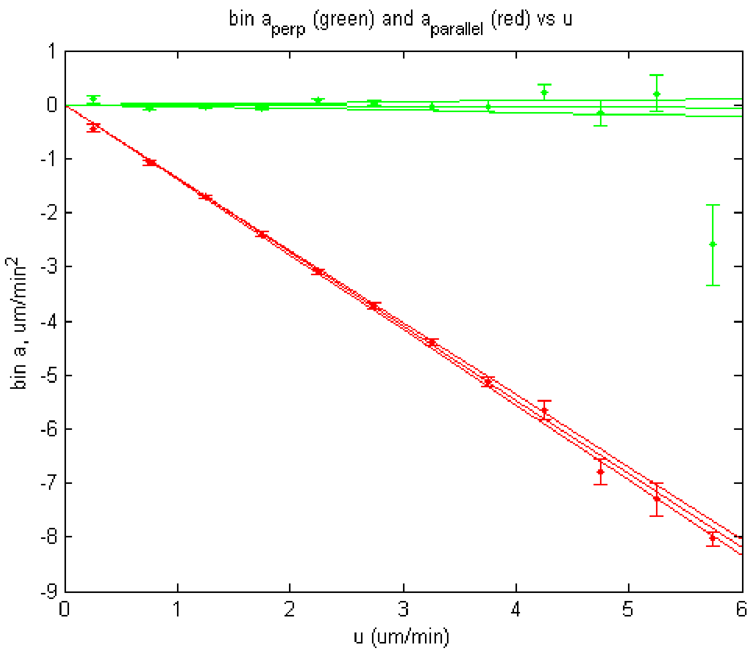


Figure 5: Acceleration vs. speed: components perpendicular (top) and parallel (bottom), binned in bins of 0.5 µm/min for visualization purposes; data simulated using a standard OU process.

The spread of the acceleration from its expectation value is related to σ_0 and as expected to be a constant, velocity-independent dispersion of the acceleration is seen in Figure 6. A linear fit to this data may be used to determine the σ parameters.

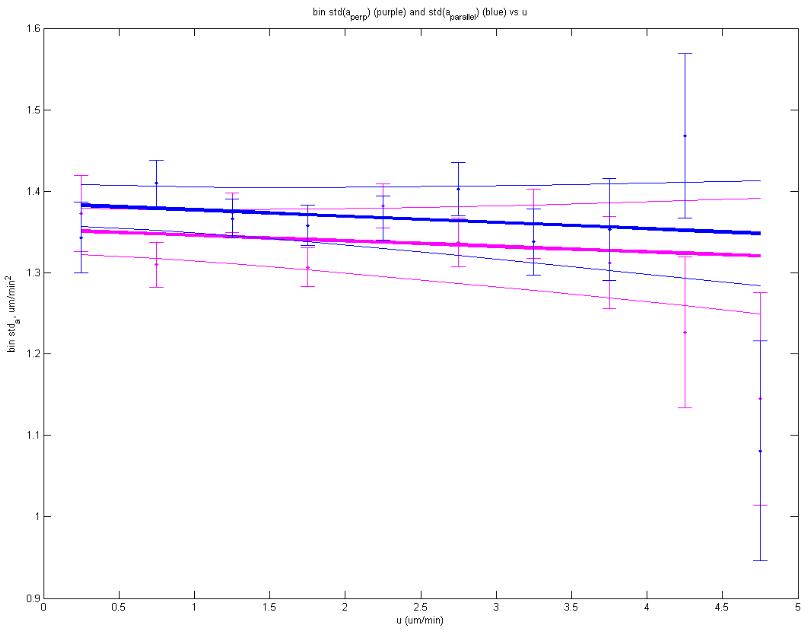


Figure 7: Standard deviations of acceleration vs. speed: components perpendicular (purple) and parallel (blue), binned in bins of 0.5 µm/min, and fit with ±1σ shown in continuous lines; data simulated using a standard OU process.

As expected for a standard OU process, $\sigma_{1\perp}$ and $\sigma_{1||}$ are zero within the error bars (see equation 5, section III.2.a). The y-intercepts are also equal within the error bars; however, they are substantially different from the input value of 0.0955.

```
std(a_perp): y-intercept sigma0 =  1.3527 ± 0.0284 um/min^(3/2)
std(a_perp) slope sigma1= -0.0068 ± 0.0137 1/min^(1/2)
sigma_perp(u) =  1.3527 + -0.0068*u um/min^(3/2)
Theoretical sigma_perp(u) =  0.0955 + 0*u um/min^(3/2)

std(a_parallel) y-intercept sigma0=  1.3847 ± 0.0253 um/min^(3/2)
std(a_parallel) slope sigma1= -0.0077 ± 0.0124 1/min^(1/2)
sigma_par(u) =  1.3847 + -0.0077*u um/min^(3/2)
Theoretical sigma_perp(u) =  0.0955 + 0*u um/min^(3/2)
```

This is to be expected since I am not plotting instantaneous velocities vs. accelerations, and the data is subject to positional noise. As a test of the relative contributions of each of these factors, I have re-run this simulation with the same parameters except for σ_{pos} which has now been set to zero.

```
std(a_perp): y-intercept sigma0 =  0.0763 ± 0.0015 um/min^(3/2)
std(a_perp) slope sigma1=  0.0006 ± 0.0023 1/min^(1/2)
sigma_perp(u) =  0.0763 +  0.0006*u um/min^(3/2)
Theoretical sigma_perp(u) =  0.0955 + 0*u um/min^(3/2)

std(a_parallel) y-intercept sigma0=  0.0790 ± 0.0000 um/min^(3/2)
std(a_parallel) slope sigma1= -0.0054 ± 0.0000 1/min^(1/2)
sigma_par(u) =  0.0790 + -0.0054*u um/min^(3/2)
Theoretical sigma_perp(u) =  0.0955 + 0*u um/min^(3/2)
```

While some discrepancy remains (-20% error) which I attribute to finite, but small $\Delta t/P = 1/40$, it is significantly improved from the previous result. Indeed, this is an expected result because adding positional noise to the data increases the variance of the secant accelerations by a constant term proportional to $\sigma_{pos}^2/(\Delta t)^2$ in the absence of discretization effects.

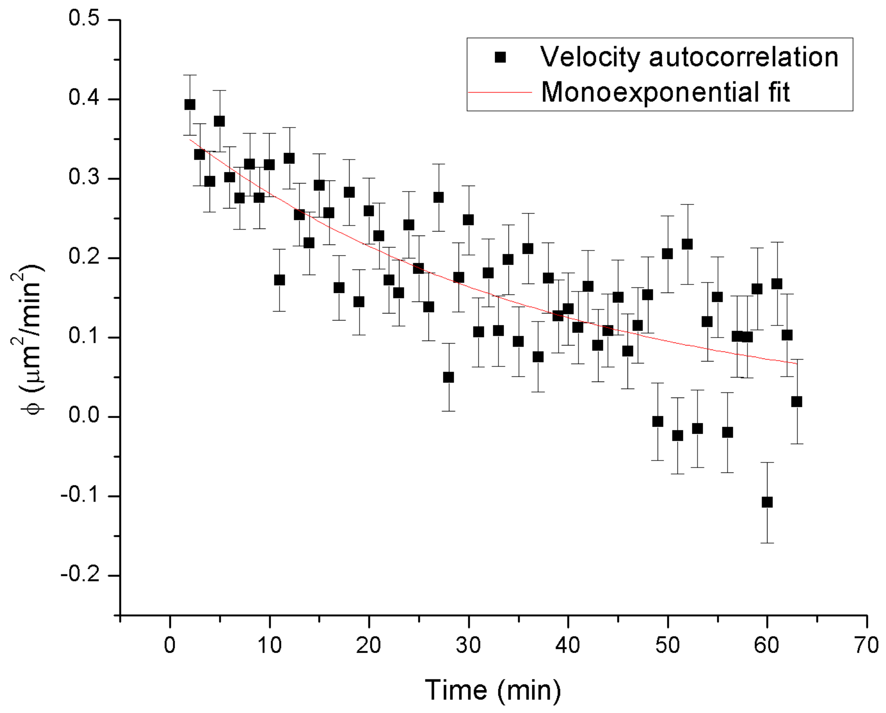
Additionally, for larger $\Delta t/P$ values such as those typically used in these experiments (Δt = 4 min, P = 40 min) and in the absence of positional noise, the σ parameters can also be significantly affected:

```
std(a_perp): y-intercept sigma0 =  0.3194 ± 0.0081 um/min^(3/2)
std(a_perp) slope sigma1= -0.0051 ± 0.0036 1/min^(1/2)
sigma_perp(u) =  0.3194 + -0.0051*u um/min^(3/2)
Theoretical sigma_perp(u) =  0.0955 + 0*u um/min^(3/2)

std(a_parallel) y-intercept sigma0=  0.3134 ± 0.0047 um/min^(3/2)
std(a_parallel) slope sigma1= -0.0001 ± 0.0021 1/min^(1/2)
sigma_par(u) =  0.3134 + -0.0001*u um/min^(3/2)
Theoretical sigma_perp(u) =  0.0955 + 0*u um/min^(3/2)
```

A fitting method based on power spectral analysis would allow optimal estimation of all parameters. The current version of the data analysis software, however, is unable to obtain more reliable estimates of the σ parameters.

An initial weighted fit of the velocity autocovariance to a function of the form $\varphi_0 \bullet \exp(-t_j/P)$+baseline was performed. This method provides an additional check as the baseline is expected to be zero within its own error bars, and the fit result of -0.0 ± 0.1 $\mu m^2/min^2$ confirms this. Following this check, the baseline was set to 0 and a weighted LSQ fit of the autocovariance function to a simple mono-exponential function was done to determine the persistence time (Figure 8). The result of the fit was P = 42 ± 4 min which agrees well with P value used to generate this dataset, 40 minutes. Similarly, the input value of $\varphi_0 = 0.362$ $\mu m^2/min^2$ agrees with the fit results of 0.37 ± 0.02 $\mu m^2/min^2$. As noted before (equation 8, section II.5.a), this parameter is nearly equal to the expectation value of the squared instantaneous velocity $<v^2>$ (the slight difference is dependent on the magnitude of $\Delta t/P$ and is less than 10% for $\Delta t/P<1/2$, which is typically encountered in experimental data).



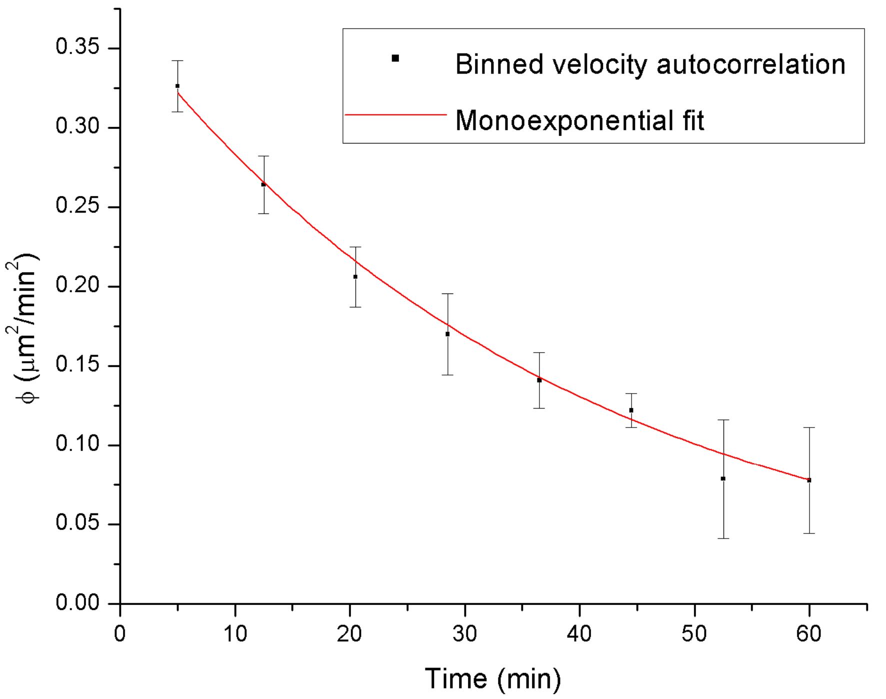


Figure 1 - Simulated data. Top: Mono-exponential fit (red trace) to the autocovariance velocity function of 50 cells following a standard OU process generated *in silico*. This data was generated with a frame rate of 1 min and a positional noise of 1 μm. Bottom: Binned autocovariance velocity (black) and monoexponential fit (red trace).

To conclude, I recommend using this fitting method when ~10% errors in the value of P and φ_0 are acceptable. I conclude that the data analysis software in its current form can provide highly reliable measures of the persistence time of cells following the standard OU process; however, the cell motility coefficient is extremely sensitive to positional noise and discretization effects, and a more involved theoretical method is necessary to reliably determine that parameter.

6. Conclusions and outlook

This section has provided the reader with the theoretical tools needed to characterize cell motility, which will be done in chapters III and IV of this thesis. Simple Brownian motion has been introduced first, and I have highlighted my contribution to optimizing the methodology of determining the diffusion coefficient from 2D diffusion data, with potential applications in membrane protein mobility and cell membrane fluidity studies.

A layer of mathematical complexity has been added by considering the case where the inertial effects are not negligible (as it is the case with simple Brownian motion), and the mathematical formalism of the standard OU process is built upon this. I have developed the theoretical tools needed for modeling cells that follow the standard OU process by taking into account experimentally-relevant factors, such as cell centroid positional measurement noise and trajectory discretization effects. I have also proposed an optimal method for determining the motility parameters P and D from experimental-like motility data, thereby setting the path (and validating the method against computer simulations of motility) for interpreting real experimental data. Further research in this area is needed, however, to deal with the velocity-dependence of the sigma parameters introduced in section III.2.a of this thesis.

Finally, a trade-off between mathematical complexity and computing speed on the one hand, and precision on the other, was needed for practical reasons, which led me to develop a simplified method of analysis of experimental data, again validated by testing on computer simulations of motility.

7. References

Anderson CM, Georgiou GN, Morrison IE, Stevenson GV, Cherry RJ (1992a) 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. Journal of Cell Science 101(2): 415-425.

Anderson CM, Georgiou GN, Morrison IE, Stevenson GV, Cherry RJ (1992b) 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. Journal of cell science 101 (Pt 2): 415-425.

Campos D, Méndez V, Llopis I Persistent random motion: Uncovering cell migration dynamics. Journal of Theoretical Biology 267(4): 526-534.

Doob JL (1942) The Brownian Movement and Stochastic Equations. The Annals of Mathematics 43(2): 351-369.

Einstein A (1905) Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen. Annalen der Physik 322(8): 549-560.

Fürth R (1920) Die Brownsche Bewegung bei Berücksichtigung einer Persistenz der Bewegungsrichtung. Mit Anwendungen auf die Bewegung lebender Infusorien. Zeitschrift für Physik A Hadrons and Nuclei 2(3): 244-256.

Grădinaru C, Łopacińska JM, Huth J, Kestler HA, Flyvbjerg H et al. Assessment of Automated Analyses of Cell Migration on Flat and Nanostructured Surfaces. Plos Computational Biology submitted.

Mortensen KI, Churchman LS, Spudich JA, Flyvbjerg H Optimized localization analysis for single-molecule tracking and super-resolution microscopy. Nat Meth 7(5): 377-381.

Mueller DJ, Engel A, Matthey U, Meier T, Dimroth P et al. (2003) Observing Membrane Protein Diffusion at Subnanometer Resolution. Journal of Molecular Biology 327(5): 925-930.

Norrelykke SF, Flyvbjerg H (2009) Power spectrum analysis with least-squares fitting: Amplitude bias and its elimination, with application to optical tweezers and atomic force microscope cantilevers. Review of Scientific Instruments 81(7).

Nørrelykke SF, Flyvbjerg H (2010) Power spectrum analysis with least-squares fitting: Amplitude bias and its elimination, with application to optical tweezers and atomic force microscope cantilevers. Review of Scientific Instruments 81(7).

Ornstein LS (1919) On the Brownian Motion. Koninklijke Nederlandse Akademie van Weteschappen Proceedings Series B Physical Sciences 21: 96-108.

Rao CR (1973) Linear Statistical Inference and Its Applications: John Wiley and sons.

Selmeczi, Mosler, Hagedorn, Larsen, Flyvbjerg (2005) Cell motility as persistent random motion: theories from experiments. Biophysics Journal 89: 912-931.

Shinitzki M, Henkart P, editors (1979) International review of cytology. 121-148 p.

Takeaki K, Kurata H (2004) Generalized Least Squares:
Wiley. 313 p.

Uhlenbeck GE, Ornstein LS (1930) On the Theory of the Brownian Motion. Physical Review 36(5): 823.

Vestergård C (2009) Maximum likelihood estimation of the diffusion coefficient of a brownian particle in the presence of measurement error. Copenhagen: University of Copenhagen.

von Smoluchowski M (1906) Zur kinetischen Theorie der Brownschen Molekularbewegung und der Suspensionen. Annalen der Physik 326(14): 756-780.

Xiaodong Yang, Houqiang Li, Zhou X (2006) Nuclei Segmentation Using Marker-Controlled Watershed, Tracking Using Mean-Shift, and Kalman Filter in Time-Lapse Microscopy. IEEE Transactions on Circuits and Systems Part I: Regular Papers 53: 2405.

Chapter III: Cell tracking and motility on isotropic substrates

Parts of this chapter have been directly adapted from a paper intended to be submitted to Plos computational biology, the software section, on which I figure as first author: Grădinaru, C.; Lopacinska, J. M., Flyvbjerg, H.; Huth, J.; Kestler, H. A.; Mølhave K. "Automated Analysis of Cell Migration on Flat and Nanostructured Surfaces."

1. Cell tracking and centroid measurement

a. Introduction

Cell migration plays an essential role in many biological processes: e.g. wound healing, embryogenesis, inflammation, and metastasis where uncontrolled cell migration can lead to tumor spreading and hence can cause cancer progression. Studying these processes frequently require cell tracking, and most motility studies of monolayer cultures involve fluorescent labeling of cells, which allows for fluorescence microscopy studies (Van Haastert and Devreotes 2004; Dufour et al. 2005). This technique either requires extensive mutagenesis to have fluorescent protein expressed by the studied cell-type, or is limited by the fact that membrane-attached or permeable fluorescent drugs often alter cell behavior (Presley 2005). In sparse cultures, cells can have sufficiently good contrast against the background that their boundaries can be identified with bright field microscopy without labeling (Keren et al. 2008). This can be done manually with point-and-click methods at great expense of time and labor (Michl et al. 2005; Selmeczi et al. 2005; Tang et al. 2006; Wolf et al. 2007; Boldajipour et al. 2008; Henrickson et al. 2008). When cells in only a few images have to be tracked, this is not a challenge. However, when long time-lapse sequences of motile cells need to be analyzed, this approach is impractical, raising the need for a reliable automated cell tracking program.

Table 1 - Overview of representative programs for cell tracking by time-lapse light microscopy.

Program	*Publisher/Seller*	*Commercial*	*Open source*	*Coordinate output format*
Volocity	Improvision	Yes	No	Text
Imaris	Bitplane	Yes	No	Excel
Autozell (Baumann 2006)	Universität Bremen	Yes	No	Text
TLA (Huth et al. 2010)	University of Ulm	No	Yes	Excel/Text
CellTrack (Sacan et al. 2008)	Ohio State University	No	Yes	Text
ImageJ plugins (Stuurman; Sbalzarini and Koumoutsakos 2005)	ETH & UCSF	No	Yes	Text

Representative programs are listed in Table 1. The majority of such programs (not included in Table 1) are designed for tracking fluorescently labeled cells (Pal and Pal 1993; Van Haastert and Devreotes 2004; Dufour et al. 2005; Dormann and Weijer 2006;; Ali et al. 2008; Meijering et al. 2009). The programs designed for use with light microscopy images track either the cell nucleus (Xiaodong Yang et al. 2006; Xiaowei et al. 2006) or the entire cell (O. Debeir 2005; Baumann 2006; Li et al. 2008; Huth et al. 2010). The cell position can be defined as: (i) the center of the nucleus (e.g. (Selmeczi et al. 2005)); (ii) the centroid of the cell's perimeter as seen in the light microscope (Li et al.2011); (iii) the centroid of the cell's footprint as seen in the light microscope (Keren et al. 2008); and (iv) the centroid of the actin cytoskeleton of fluorescently labeled cells (Yumura and Fukui 1998). Most of these programs are not open source, and may be difficult to adapt to the specific purposes of a given experiment. In other cases the complexity of the mathematical procedures used for boundary identification may be a hurdle to adapting the code to a specific purpose (Dufour et al. 2005; Xiaowei et al. 2006; Ali et al. 2008).

There are two main approaches to cell tracking in the current state-of-the-art (Gerlich et al. 2003; O. Debeir 2005; Dormann and Weijer 2006; Xiaowei et al. 2006; Hand et al. 2009; Meijering et al. 2009). One approach is frame-by-frame image segmentation and tracking (Pal and Pal 1993; Althoff 2005). In the first step, the object candidates are detected in a given frame on the basis of their specific properties (border, texture, color). This approach is efficient when object borders are sharp, and it is commonly used with fluorescently labeled cells and other high-contrast images. The other approach consists in optimizing a parameterized model shape to fit the model to the cells in a frame. Instead of tracking all objects in the frame, this method focuses on those candidates which correspond to the chosen model shape (Leymarie and Levine 1993; Sethian 1999). As with the first approach, detected objects are paired between consecutive frames in order to produce tracks.

In this chapter I introduce a new cell tracking program, PACT (**P**rogram for **A**utomated **C**ell **T**racking). PACT is suitable for tracking motile cells on flat and nanostructured surfaces, and is simple enough for users to freely modify it according to their experimental needs. Since nanostructured surfaces are currently of great interest as cell-culture substrates and can appear as a highly non-uniform background in the image, I emphasize the use of a reliable spatial image filter here – see details in the materials and methods section, and supp. info 2 for the code.

Test results are presented for the performance of PACT, which is also compared to other programs's performance (*TLA* (Huth et al. 2010) and *Autozell* (Baumann 2006)): efficiency with regard to object detection, accuracy of centroid positioning, and segmentation performance in the context of time-lapse analysis. A statistical analysis is then performed on the ensemble of individual tracks of cells to determine the overall cell population motility statistics in a movie of NIH 3T3 fibroblasts on glass. From the tracks, the velocity auto-covariance function (or auto-covariance of the velocity) is estimated (Equation S1, Appendix 3.a). It is well described by a simple exponential function with characteristic time, *P*, the *persistence time* of the motility (data shown in Appendix 3.a). We also determine the amplitude of the velocity autocovariance function, φ_0, which is approximately equal to the mean squared velocity of the cells. Results from PACT, TLA and Autozell are compared by this analysis.

Despite the need for cell tracking programs and algorithms, there are only few studies evaluating their performance (Baumann 2006; Huth et al. 2010). In this chapter I make an extensive comparative analysis of tracking programs. I find considerable sensitivity of results to the specific algorithm/software employed. Thus, the tracking algorithm and its effect on results should be well documented in future studies. I conclude that cell biologists intending to track independent migrating cells on a wide variety of surfaces may find in PACT an efficient choice.

b. PACT – Program for Automated Cell Tracking

To assess the reliability of automated tracking, I evaluated the cell segmentation performance at the single-frame level. To that end I did the following:

(i) Tested the segmentation efficiency of our code, namely how well it locates individual cells in comparison to the other programs available.
(ii) Assessed how changes in focus can affect the accuracy of cell centroid positioning.
(iii) Compared PACT's accuracy of centroid positioning in bright-field imaging and fluorescence imaging.
(iv) Evaluated the background removal efficiency on nanostructured surfaces by comparing segmentation and centroid position results from bright-field and fluorescence microscopy of the same sample.

In the subsections below, I elaborate on these points and the ensuing discussion section deals with how these observations relate to each other.

i. Segmentation efficiency

I tested segmentation on a movie frame of HeLa cells on a glass substrate, NIH 3T3 on a flat silicon substrate, and NIH 3T3 on a silicon black substrate with PACT, TimeLapseAnalyzer (TLA) (Huth et al. 2010), Autozell (Baumann 2006), and CellTrack (Sacan et al. 2008). A comparison of the number of objects identified by each program vs. the actual number, as determined by manual counting i.e. segmentation efficiency, is shown in Table 2.

PACT has a cell segmentation efficiency comparable to that of TLA, and both are better than Autozell and CellTrack for HeLa/glass, while CellTrack performs better for 3T3/Si. The cell selection efficiency is very dependent on the software used and on the settings used in the segmentation. With training in how to use software, users can achieve 80-90% count rates compared to manual counting.

Table 2 – Number of objects segmented by the various cell tracking programs in bright field images before manual post-processing

	PACT	Autozell	TLA	CellTrack	Manual count
HeLa/glass	199	124	197	157[1]	206
NIH 3T3/flat Si	107[2]	104	132[3]	160[1]	158
NIH 3T3/Si black	40	0	75[4]	0	48

The tracking efficiency was assessed by counting the objects picked in the representative images used in Table 2. A section of such a frame processed in the different programs is shown in Figure 1 for visual comparison.

[1] In my experience, Celltrack often picks spurious objects in the background, which are not cells, and this artificially inflates the cell count in Table 2.

[2] The cell count for NIH 3T3/flat Si by PACM is underestimated due to this data being cumulated from five different images, as the band -pass filter of PACM removes the edge of the image prior to processing. This effect makes the cells on the edge inaccessible to segmentation. This is less obvious in the case of the HeLa/glass or NIH 3T3/Si black samples where the count was performed on a single image.

[3] TLA occasionally assigns multiple centroids to one cell, and I have counted such instances as just one object in Table 2.

[4] Wiener-filter noise-parameter lowered to 0.03 from the default of 0.05 for non-uniform background removal.

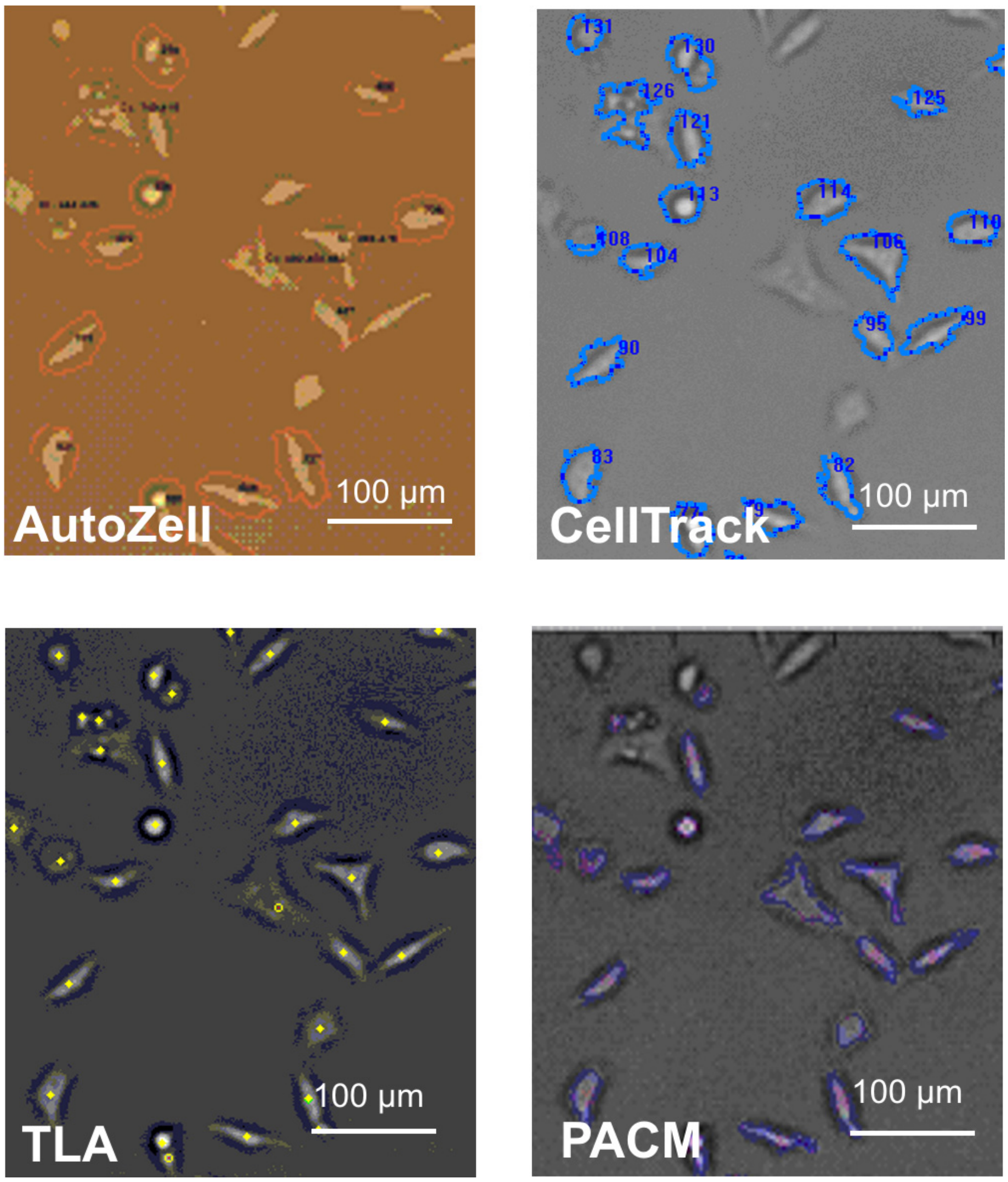


Figure 1 - Section of a typical frame of HeLa/glass after segmentation in Autozell, CellTrack, TLA and PACT. Autozell shows red contours around the tracked objects, CellTrack shows blue contours around the tracked objects, TLA marks the centroid of the object with a yellow marker, and PACT shows blue contours around the tracked object and a red cross at the brightest spot in the image. The settings for each program were optimized so as to reach a balance between minimizing the number of cells missed by the tracking process and the number of false positives (segmented objects that were not cells).

ii. Focus Influence on Centroid Positioning

PACT was able to identify cell boundaries even when the cells were imaged slightly out of focus. Imaging out of focus in bright field microscopy has the advantage of enhancing the contrast between cells and background. This is due to light interference effects and blurring of details of cell organelles, which are visible in focus and can interfere with cell identification. We did not employ a blurring method (such as a

Gaussian blur) to remove the fine-structure of cells, as that procedure would also negatively affect the sharpness of contrast between cell and background.

Long-term time-lapse recordings may lead to drift and variable focus. Another source of noise can be the light conditions and also variations in threshold and filter settings for the image processing which in PACT also can influence the cell centroid measurement. The following experiment estimates the combined effect of these influences by varying focus, which changes the cell contour and light levels

NIH 3T3 fibroblasts fixed on a glass slide were imaged in overfocus at 2, 4, 6, 8, 10, 15, and 20 µm, respectively, as well as in focus (z=0 µm), which we consider a fairly wide range of focus fluctuations for a time lapse experiment. After segmentation and post-processing, 92 cells were found in all frames. I calculated the individual cell centroids throughout the stack. From these positions I subtracted average centroid value of each frame to remove stage drift, and then evaluated the average root- mean-square displacements of all 92 cells segmented, as a function of z (Figure 2). Typical samples of segmented cells are shown in Figure 2 next to the corresponding points: z=0, 4, and 20 µm, respectively. The overall average root mean square displacement from the z-stack mean was 1.0 ± 0.1 µm.

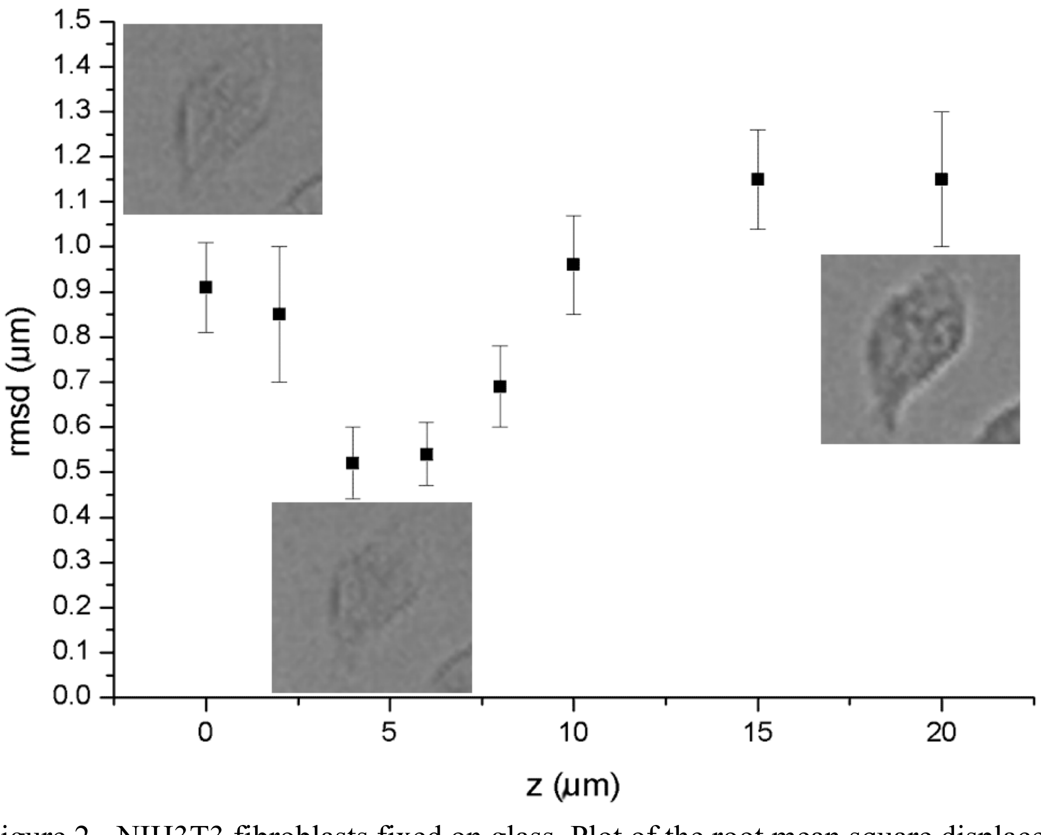


Figure 2 - NIH3T3 fibroblasts fixed on glass. Plot of the root mean square displacement of the centroids from their mean in the stack of images of NIH 3T3 imaged increasingly out of focus shows the effect of defocusing on the centroid tracking accuracy. For illustration, a representative cell as viewed in the light microscope is shown at z=0 (in focus), z=4 µm and z=20 µm.

iii. Precison of Centroid Positioning

In order to test the precision of the method, I compared the performance of our program with bright field images to images of dyed cells visualized by fluorescence microscopy. Images of fluorescently dyed NIH 3T3 fibroblasts fixed on glass and flat Si were recorded in both the bright field and fluorescence mode, Fig. 4. 22 cells on flat Si were identified in both bright-field and fluorescence images after post-processing. A pair-wise comparison of centroids between the two imaging modes was done by translational alignment (applying a constant offset to all centroids from one imaging mode such that the mean of all centroids common to both imagining modes is exactly the same). This procedure was necessary to account for the slight light path offset due to inserting the fluorescence filter, and was followed by calculating the pair-wise RMSD of all centroids common to both imaging modes. This yielded an RMSD of 2.8 µm for the flat Si substrate (Table 3). I tested whether the orientations of the segments that connect the centroid pairs are random by plotting a histogram of the distribution of their polar angles (data not shown) which was indeed uniform within error bars. I thus conclude that the source of this discrepancy is random (white) noise.

Table 3 – Fixed cell results. The columns *PACT* and *Manual Count* show the fraction of objects tracked after/before post-processing in bright-field images. *Co-identified* shows pairs of cells that were co-identified in fluorescence and bright field post-processed data and their corresponding RMSD

	PACT	Manual count	Co-identified
NIH 3T3/flat Si	26 / 32 (81%)	35 / 42 (83%)	22 pairs: RMSD 2.8 µm
NIH 3T3/Si black	21 / 40 (52%)	40 / 48 (83%)	14 pairs: RMSD 4.3 µm

iv. Nanostructure Background Removal

To process images with grainy backgrounds, such as images recorded on a substrate of silicon black, the band-pass filter used by PACT (see materials and methods) to process images is of critical importance. By setting the high-frequency limit to 5-10 pixels, i.e., higher than the recommended value of 1-3 pixels that is appropriate for removal of pixel noise, grainy images can be analyzed. On this substrate, cells appear as shadows on a very noisy background and I was unable to detect cells with other cell tracking programs (Table 1), with the exception of TLA, which can employ a Wiener filter as a pre-processing step to cell detection. I compared the fluorescence image with the bright-field images of individual NIH 3T3 fibroblasts fixed on silicon black; see Figure 3 and Table 3. The mismatch was larger than what was found on flat substrates, as expected on these highly grainy images: 4.3 µm for the 14 cells found co-localized after post-processing in both the bright-field and fluorescence out of 21 cells in bright-field and 25 cells in fluorescence. The mismatch of centroid pairs was randomly oriented with uniform distribution of directions within errors due to finite statistics (data not shown).

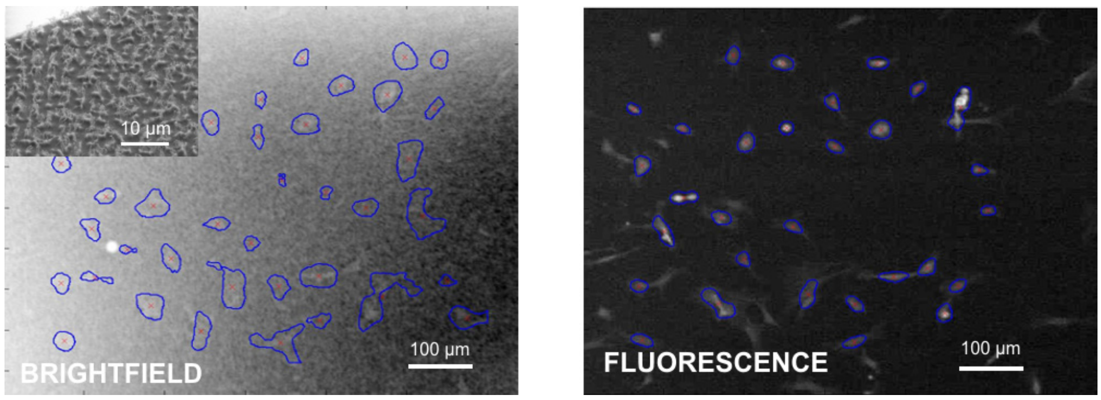


Figure 3 - Fixed NIH 3T3 cells on silicon black. Left: Bright field image of NIH3T3/Si black after tracking in PACT and SEM image of the Si black substrate used viewed at a 30º angle (inset). Right: Fluorescence image of the same sample imaged in the exact same position, showing the actin cytoskeleton, also processed in PACT.

2. *Analyzing real-life data*

a. *Modeling real-life data using OU-like processes*

Chapter II of this thesis dealt extensively with the theoretical aspects of persistent random motion and explored methods for coping with the challenges presented by experimental data, such as positional noise and time discretization. Using PACT introduced in section 1 above, which provides a method for reliably determining cell centroids from experimental data, I take a step forward to explore how the theoretical tools introduced in the earlier chapter can be applied to analyze real-life data.

Several extensions of the standard OU process have been proposed in the literature to model the observed motility of various cells on a range of substrates (Grădinaru et al.; Selmeczi et al. 2005). Here I will introduce some of the features of one such model which is applicable to the motility of NIH 3T3 fibroblasts. The cell motility parameters were extracted from five movies recorded at different positions on the same NIH3T3/glass sample. The cumulative motility data collected from these movies (58 trajectories) as accelerations vs. velocity is shown in Figure 1. The procedure described by (Selmeczi et al. 2005) was followed to assess whether NIH 3T3 fibroblast motility can be described by the standard OU process or an extension thereof.

The dataset consists of time lapse movies of the cells, with images recorded at constant time intervals Δt. The individual cell positions in images $\vec{r}_j$=[x(t$_j$), y(t$_j$)] were determined using PACT. From the observed positions, the secant velocities $\vec{u}_j$ and secant accelerations $\vec{\alpha}_j$ can be determined:

$$\vec{u}_j = \frac{\vec{r}_{j+1} - \vec{r}_j}{\Delta t} \quad \text{and} \quad \vec{\alpha}_j = \frac{\vec{u}_{j+1} - \vec{u}_j}{\Delta t}$$

Equation 1

I remind the reader here of the distinction between these quantities and the instantaneous velocities and accelerations, for which I keep the traditional notation $\vec{v}(t)$ and $\vec{a}(t)$. The secant velocities $\vec{u}_j$ and secant accelerations $\vec{\alpha}_j$ are discrete time series, whereas $\vec{v}(t)$ and $\vec{a}(t)$ are continuous functions of time. Only the secant quantities are available from a real experiment and differ from $\vec{v}(t)$ and $\vec{a}(t)$ due to discretization effects owing to the finite frame rate Δt (II.5.a). They also include contributions from the positional noise, which is the inherent measurement error that affects each cell centroid values output by the tracking program. In the limit where the positional noise and Δt become vanishingly small, the secant quantities equal their instantaneous counterparts.

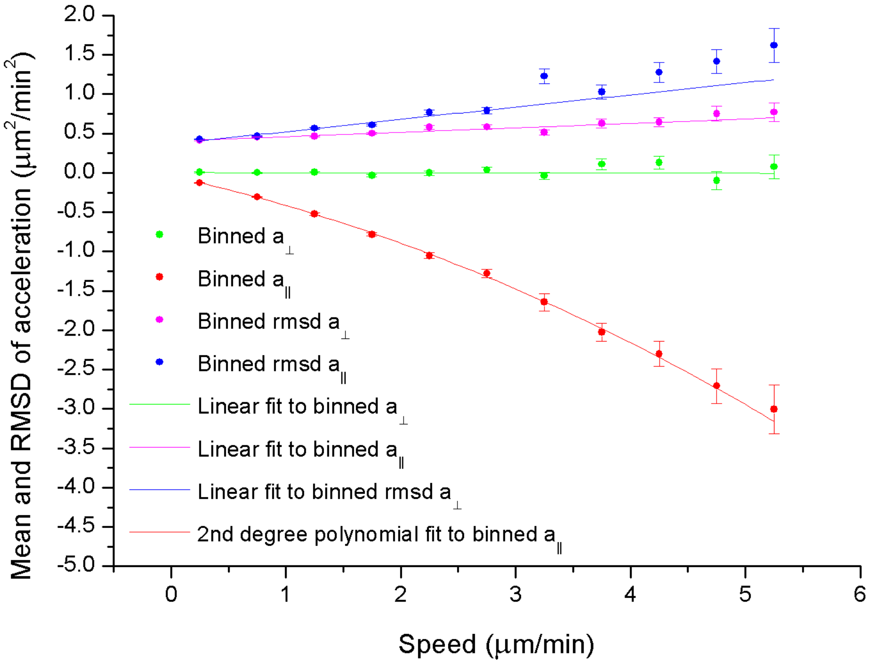


Figure 1 - Plot of components of acceleration and of RMSD of accelerations versus speed. Red and green data points are the mean values of the parallel and perpendicular components of the secant acceleration, respectively, as a function of secant speed. Blue and magenta data points show RMSD of the same quantities, parallel and perpendicular components respectively. The lines show theoretical expectation values of the same quantities, according to the NIH 3T3 model, fitted to these data points.

Below I will describe how to analyze the data to derive an OU-like model that best describes the corresponding cell motility. Cell motility models are often described as OU-like processes. The standard OU process follows the following equation (Equation 2, II.4.a), rewritten here with the shorthand notation √2D = s:

$$\frac{d\vec{v}}{dt}(t) = -\frac{1}{P}\vec{v}(t) + s\vec{\xi}(t)$$

Equation 2

where $\vec{\xi}$ is a normalized generalized Gaussian white noise as described before (I.1.a).

In the standard OU model the parameter *P* is interpreted as the persistence time of the motility and *s* is the cell motility coefficient. Using secant quantities, this becomes:

$$\vec{\alpha}_j \overset{\text{def}}{=} \frac{\Delta\vec{u}_j}{\Delta t} = -\beta\vec{u}_j + \sigma\vec{\eta}_j$$

Equation 3

where $\vec{\eta}_j$ is a vector whose components are Gaussian distributed random numbers of mean 0 and variance 1 as described before (II.1.b). In the absence of discretization effects and positional noise equation 3 can be derived directly from equation 1 by integration followed by division by Δt. However, in the case of data subject to this type of effects, such as real experimental data, this is only an approximation:

$$\beta^{-1} = P + \mathcal{O}(\sigma_{pos}) + \mathcal{O}'(\Delta t)$$

Equation 4

and similarly

$$\sigma\sqrt{\Delta t} = s + \mathcal{O}(\sigma_{pos}) + \mathcal{O}'(\Delta t)$$

Equation 5

The β and σ parameters can be determined directly from plots of α and RMSD(α) vs. *u*, and depend on the experimental setup through the positional noise and discretization of time (III.5.a). A system-specific set of parameters, *P* and *s*, require indirect methods to determine them (II.5.b and II.5.c).

As a reminder, for the OU process, the secant velocities can be used to calculate P as the characteristic time of the velocity autocorrelation function. This can be estimated from the cell tracks as follows, essentially replacing the expectation value in equation 3, section II.4.a with an averaging over cell trajectories and time:

$$\varphi_j^{\exp} = \frac{\sum_{\alpha=1}^{last\ track} \sum_{k=0}^{N_\alpha - j} \vec{u}(t_j + t_k) \cdot \vec{u}(t_k)}{\sum_{\alpha=1}^{last\ track} N_\alpha}$$

where N_α denotes the number of points in track α, $t_j = j\Delta t$ and $t_k = k\Delta t$ as all of our movies were recorded at a fixed frame rate. By fitting a monoexponential function to the remaining values of φ, the persistence time P can be determined accurately as the characteristic time of the monoexponential fit (II.5.c). I also point out to the reader that the first two points of the autocorrelation velocity function are also affected by more subtle effects (e.g. the fine oscillations in trajectory due to pseudopod action) (Li et al., Phys Biol), which makes using them to determine positional noise an imprecise method.

The standard OU model predicts that the perpendicular projection of the cell acceleration to the trajectory would be zero, the noise term aside (equation 2). This can be seen to hold true in Figure 1 (green trace). Similarly, the parallel component is expected to show linear dependence on velocity for cells following the standard OU process and the data in Figure 1 (red trace) shows deviation from linearity of $a_{||}$ at high speeds. Additionally, in the standard OU process the RMSD of the two components of the acceleration are expected to be v-independent, which is clearly not the case (purple and blue traces). These deviations from the behavior expected of the standard OU process led us to propose the following extension of the OU process to describe the motility of NIH 3T3 fibroblasts:

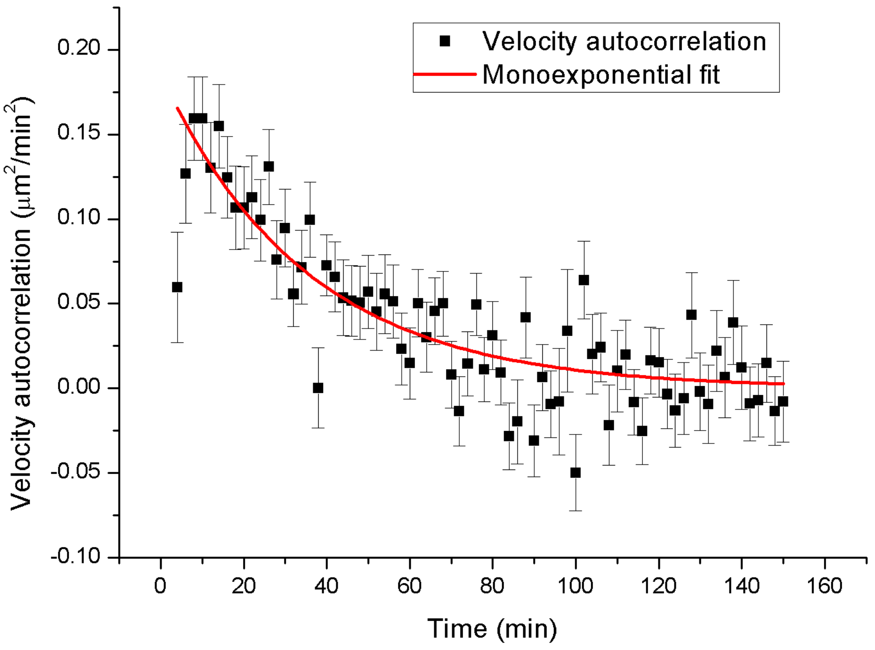
$$\frac{\Delta \vec{u}_j}{\Delta t} = -(\beta_0 + \beta_1 u_j) \cdot \vec{u}_j + \overline{\overline{\boldsymbol{\sigma}}}(\vec{u}_j)\vec{\eta}_j$$

Equation 6

It is important to highlight at this point the distinction between the β parameters and the persistence time, P, on the one hand, and between the σ parameters and the cell motility coefficient s, on the other hand.

The process described by equation 6 is similar to the normal human derman fibroblast (NHDF) model proposed by Selmeczi et al (Selmeczi et al. 2005). The noise term was modeled in the same manner as in that paper, namely, as a velocity-dependent tensor times a white noise term. Similarly, to account for the non-linearity of $a_{||}$, the P^{-1} factor in equation 2 was replaced with a linear function of speed, $\beta_0+\beta_1 v$, which is consistent with a 2nd degree polynomial dependence of $a_{||}$ on speed as our data indicate (Figure 1, red trace). Following fitting these parameters were $\beta_0 = 0.32 \pm 0.02$ min^{-1} and $\beta_1 = 0.051 \pm 0.007$ μm^{-1}.

The notable difference between the behavior of NIH 3T3 cells presented here and that of NHDF cells is that the former do not display the initial fast decrease of the velocity autocorrelation function (defined by equation 2) to its slower exponential decrease at larger time-separations (Figure 2, top panel), which is characteristic of the NHDF model. This is more easily seen in the binned autocorrelation velocity, shown in Figure 2 (bottom panel).



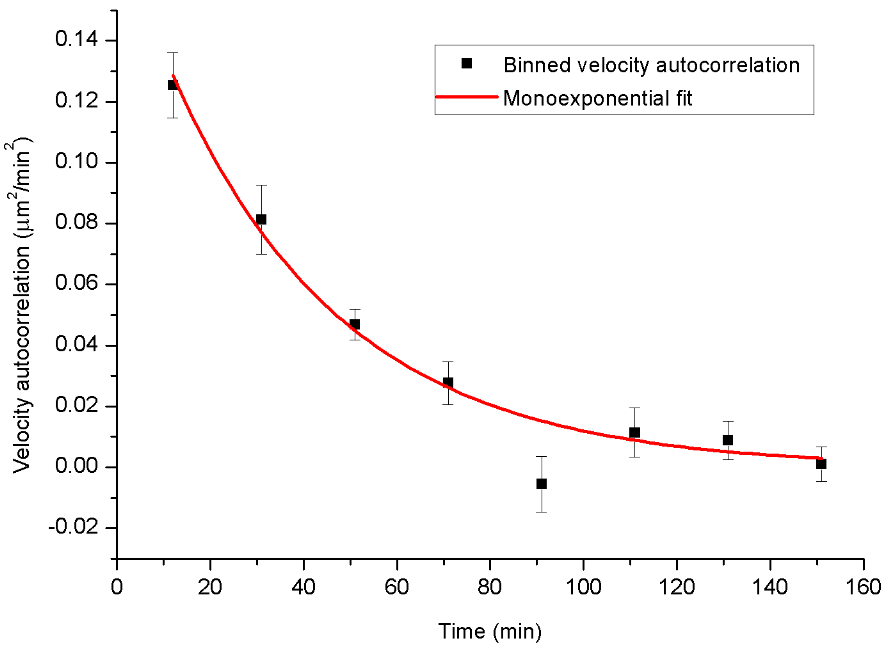


Figure 2 - Motility of NIH 3T3 on glass Top: Data points are the experimental velocity autocorrelation function. The error bars underestimate the true scatter as they were computed from experimental data that are correlated due to persistence of cell motion. The red line shows a mono-exponential fit to these data. Bottom: Binned velocity autocorrelation (black) and monoexponential fit (red).

One possible explanation for this difference between human (NHDF) and mouse (NIH 3T3) fibroblasts, which brings the motility of NIH 3T3 fibroblasts closer to the standard OU process, is that the $\beta_1 v$ term in equation 6, although present, is relatively small as compared to the β_0 term. In this case, equation 6 can be approximated to:

$$\frac{\Delta \vec{u}_j}{\Delta t} \approx -\beta_0 \cdot \vec{u}_j + \overline{\overline{\boldsymbol{\sigma}}}(\vec{u}_j)\vec{\eta}_j$$

Equation 8

which is the same as the difference form of the standard OU process (equation 3), noise term aside. Hence, its solution $\varphi(t)$ is expected to be mono-exponential. The perturbation induced by the presence of a small $\beta_1 v$ term may be buried in the noise in the case of NIH 3T3 cells, but not NHDF. Indeed, the $\beta_1 \langle v \rangle / \beta_0$ ratio is 0.156 for NIH 3T3, but significantly larger, 0.429, for NHDF (Selmeczi et al. 2005). I also underline here the fundamental difference between the persistence time, P, defined here as the lifetime of the velocity autocorrelation function, and the β_0 parameter. Since the β_0 parameter is determined from a polynomial fit to the parallel component of the acceleration as a function of speed as in Figure 1, and the instantaneous cell acceleration and speed are unavailable in real experimental settings due to the finite frame rate time ($\Delta t = 2$ min), the relationship $P^{-1} = \beta_0$ only holds in the limit $\Delta t/P \rightarrow 0$ and $\sigma_{pos} \rightarrow 0$. Indeed, in the case of NIH 3T3 cells, $P = 36 \pm 4$ min, which is substantially larger than $\beta_0^{-1} \approx 3$ min. The significant discrepancy between the two parameters highlights the importance of positional noise and trajectory discretization effect in interpreting these results (equations 4, 5). To conclude, the NIH 3T3 model's most significant departure from the NHDH model proposed in [Selmeczi et al.] is a mono-exponential dependence of the velocity autocorrelation function on time. Thus in the NIH 3T3 model I omit the memory kernel of the NHDF model.

Because of the monoexponential nature of the velocity autocovariance function, Fürth's formula will remain formally identical to its standard OU counterpart, inclusive of the case where positional noise is accounted for, making equation 6, section II.5.a generally applicable not only to the standard OU process but also to the NIH 3T3 model. I will make use of that feature in the following chapter to assess the positional noise in various datasets. Likewise, the simplified fitting method proposed in II.5.c is also applicable here, unlike in the case of the NHDF model. It is important to note, however, that the amplitude of that exponential is no longer 2D/P as in the standard OU process, as the D parameter is no longer defined. Instead, its relationship to the other motility parameters can be determined by numerical integration.

The method presented in this section is generally applicable to any sparse cell culture undergoing isotropic motility, and I have employed the data_analysis_exp.m program (Appendix 6), which allows for automating the plot generation (see section II.5.c). Following the proof-of-the-method test on NIH3T3 fibroblasts described above, this program has also been extensively used for analyzing cell motility on Si black substrates (section 4 of this chapter). All the fits presented above have been performed in Origin 8.0 using data output by data_analysis_exp.m.

b. Effect of RMSD filtering

In order to remove all fixed objects automatically, an additional, RMSD-based filter is applied to the tracks prior to their analysis. For each track, both data_analysis_exp.m (in case a program other than PACT is used for cell tracking) and

PACT calculate the root-mean-squared deviation of the tracked point from its mean (i.e., time averaged) value:

$$RMSD = \sqrt{\frac{\sum_{i=1..N}\left(\vec{r}_i - \vec{r}_{mean}\right)^2}{N-1}}$$

Equation 1

This is a small quantity if a cell does not move. In our experience, an RMSD threshold of approximately half a typical cell diameter (15 pixels in our images) is optimal for the exclusion of non-moving cells. This ensures that only cells that move at least one cell diameter over the course of a time-lapse experiment are used for further analyses. Most of the cells discarded this way (~30% of all tracked immobilized objects) appear to be mechanically immobilized, attached permanently to the surface, or lysed, and they should not be included in the motility analysis. The few cells that are not lysed or immobile, but simply happen to move very little for the duration of the observation, are also removed by this filter. This causes negligible error because their contribution to the velocity autocovariance function would be nearly zero if they were allowed to contribute to its statistics, while inclusion of a large number of non-moving cells and other objects would skew the statistics. Consequently, the RMSD-filter's elimination of these cells has minimal effect on our estimates for persistence times. Specifically, in our movies the RMSD filter only discarded two to three biologically relevant cells out of a full dataset of 70. I found that the convenience of having all fixed objects removed automatically greatly outweighs the negligible effect of their exclusion on statistics. In general, however, since the parameter φ_0 is nearly equal to the mean squared velocity of the cells, one should clearly state the use of the RMSD filter and estimate its effect, since its elimination of the slow-moving cells has the potential of artificially inflating φ_0.

c. *Reproducibility of Statistical Data Analysis*

Before testing whether different cell tracking programs yield the same velocity autocovariance statistics for the tracks that they find for a given movie of motile cells, we assessed the reproducibility of the results obtained using PACT. Five movies were recorded on different positions on the same sample and each movie and the combined dataset were statistically analyzed. PACT found 8 to 15 tracks per movie, each movie recording ~12 hours of NIH 3T3 motility on glass, with Δt=2 min between successive frames. Results are shown in Figure 4 below and in Table S1 in Appendix 3.a.

The 5 pairs of parameters were determined from a weighted least-squares fit to velocity autocovariance function data points. We note that the error bars of the autocovariance data points, shown in Figure S2, Appendix 3.a, become increasingly underestimated as recording time goes by, due to data redundancy in the method used to calculate them (Equation S1, Appendix 3.a). This leads to artificially low weighing factors in the least-squares algorithm, which lowers the error bars of the fit parameters P and especially φ_0. Additionally, the autocovariance data points themselves are correlated (i.e. not statistically independent) as they are calculated from the same set of cell centroid coordinates. This factor also has the effect to artificially deflating the error bars of the fit parameters P and φ_0 (fit results reported in Table S1, Appendix 3.a and Figure 4). To account for these data overdispersion effects while assessing the reproducibility of our results, we quantitated the degree of this artifact in our data (Appendix 3.b). This procedure allows the estimation of the motility parameters from all 5 movies as a weighted average of the parameters measured from individual movies: $\overline{P} = 35 \pm 4$ min and $\overline{\varphi_0} = 0.18 \pm 0.02$ $\mu m^2/min^2$. This matches nearly exactly the parameters determined by combining the 5 sets of centroid coordinates and analyzing those coordinates as one set ("combined dataset," Table S1, Appendix 3.a): $P = 36 \pm 4$ min and $\varphi_0 = 0.18 \pm 0.02$ $\mu m^2/min^2$. We conclude that the observed discrepancy between the motility parameters determined from the five movies is statistically insignificant: 4/5 parameter pairs match within 1σ (68.3% expected to match) after correction. This shows that this method of analysis is reproducible.

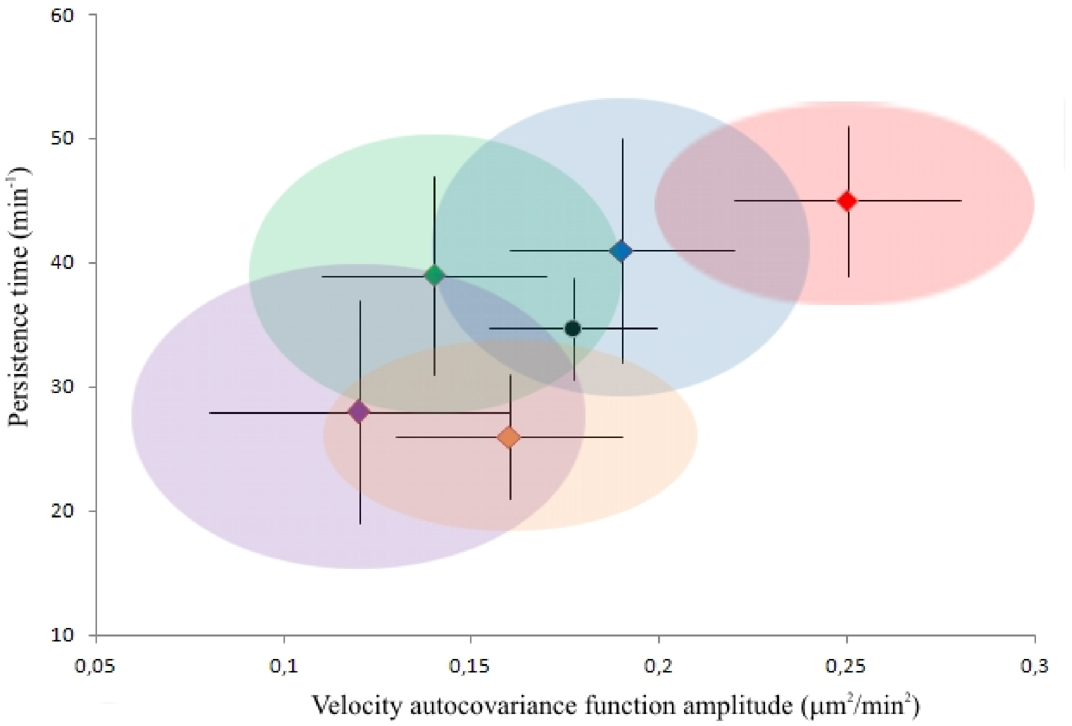


Figure 4 - Motility parameters for NIH 3T3 cells on glass analyzed with PACT. Each of the 5 movies is depicted as a point in the associated P-φ_0 space. The black error bars show the overdispersion-*un*corrected 1σ (i.e. original fit results), while the corrected 1σ levels are shown as color-matched ellipsoids around each point. Color code: blue - movie 1 (9 tracks after post-processing); purple - movie 2 (12 tracks); orange - movie 3 (15 tracks); green - movie 4 (8 tracks); and red - movie 5 (14 tracks). The black point shows the weighted mean of parameters from the 5 movies and corresponding overdispersion-corrected 1σ error bars.

3. *On the choice of cell centroid*

I have tested the performance of PACT and other programs at tracking motile cells by performing a statistical analysis of tracks:

a. I compared pair-wise the statistics obtained by tracking with different programs.

b. I compared the positional noise as function of software.

c. I compared the velocity autocovariance functions calculated from tracks obtained with different programs

a. Pair-wise Comparison of Tracks

Following the post-processing step, for the purpose of assessing the relative performance of the various programs with regards to centroid positioning error, I have compared pair-wise the tracks of cells tracked by each of the three programs (forming 3 datasets), TLA, Autozell, and PACT, so as to avoid cell selection bias due to the differing tracking methods implemented. The difference between the tracks of the same cell obtained with two different tracking programs is due to a conglomerate of errors arising from of differences in the spatial filtering, thresholding and tracking algorithm employed by each program. To remove systematic error, the tracks common to all three datasets (the subsets) were first translationally aligned to minimize the pair-wise RMSD (Eq. 3) of the three subsets. This minimal pair-wise RMDS was used as a measure of the mismatch between the coordinates output by each program.

$$RMSD_{1,2} = \sqrt{\frac{\sum_{i=1..N}\left(x_i^{PROGRAM\_1} - x_i^{PROGRAM\_2}\right)^2}{N}}$$

Equation 2

The magnitude of pair-wise mismatches between the three programs may be found in Table 4 (off-diagonal elements). We attribute the higher RMSD between the Autozell subset and the subsets generated by the other two programs to the fact that Autozell rounds the coordinates of centroids to the nearest integer, which necessarily introduces an additional random error. Also note that the default settings of TLA, which have been used in this and the following section unless explicitly stated otherwise, involve a special smoothing procedure of the tracks which slightly alters centroid coordinates (see section vii below).

b. Centroid Positional Error

I have also evaluated the mean square displacements of cell centroids as a function of time for each of the three subsets to assess the centroid positional error for each of the three cell tracking programs. By fitting to the extended Fürth's formula (equation 6, section II.5.a), we determined the centroid positional measurement error, σ_{pos}. These values are listed in Table 4.

Table 4 – Positional error effects. The diagonal elements show the centroid positional measurement error as determined from a fit to Fürth's formula. The off-diagonal elements show the pair-wise RMSD between the programs. All values in µm.

Positional error/µm	PACT	TLA	Autozell
PACT	1.40 ± 0.05	1.9	2.6

TLA	-	0.36 ± 0.03	2.7
Autozell	-	-	1.36 ± 0.04

The results in Table 4 are interpretable in light of Equation 3 which relates each tracking algorithm's intrinsic noise (denoted as the white noise term ξ_i with variance $2\sigma_{pos,i}^2$) with the measured centroid coordinate value r_i. The i subscript refers to the algorithm/program used:

$$\vec{r}_i = \vec{r}_i^{\,th} + \vec{\eta}_i$$

Equation 3

In the limit of large N, the measured value of the centroid is biased by the choice of algorithm due to positional noise (random error, variance $2\sigma_{pos,i}^2$), as well as any systematic, algorithm-specific, centroid positioning error. Thus the pair-wise RMSD is:

$$\mathrm{RMSD}_{i,j}^2 = \langle (\vec{r}_i - \vec{r}_j)^2 \rangle = \langle (\vec{r}_i^{\,th} - \vec{r}_j^{\,th})^2 \rangle + 2\sigma_{pos,i}^2 + 2\sigma_{pos,j}^2 + (cross\ terms)$$

Equation 4

Indeed, the difference between the mean squared pair-wise displacement (square of RMSD from Table 4) and the corresponding sum of the variances ($2\sigma_{pos,i}^2 + 2\sigma_{pos,j}^2$) is very similar to the variances themselves (-0.6 to 3.3 μm^2). Thus, the relative centroid tracking accuracy of the different programs does not differ significantly from the positional noise level itself.

While Autozell and PACT appear to track cells with similar positional noise levels, the TLA dataset has a much lower positional noise. I attribute this effect to the Kalman filter employed for noise reduction and subsequent smoothing of cell tracks using a moving average filter as provided in the standard procedure for processing bright field images in TLA (Kalman 1960). Indeed, reprocessing this movie in TLA with the moving average filter switched off and the Kalman filter adjusted so that no prediction is made, yields $\sigma_{pos} = 0.61 \pm 0.07$ µm, nearly twice that determined with this program's standard procedure, but still lower than PACT and Autozell's.

c. Comparing Velocity Autocovariance Functions

I have assessed the reliability of various cell tracking programs by testing them on NIH 3T3 cells recorded on a glass substrate (movie #3). The cells in this movie were tracked by Autozell, MATLAB, and TLA, and following post-processing, the tracks found by each of the three programs (the original datasets) were analyzed. I have named the resulting tracks the *original datasets*: 15 tracks identified by PACT following post-processing, 9 tracks identified by Autozell, and 14 tracks identified by TLA using the default settings. I have also included the motility parameters obtained from TLA with the moving average smoothing option turned off and adjusted to prevent any Kalman filter prediction ("TLA/raw") such that the position measurement is accepted as the true object position (15 tracks). The results are shown in figure 5 and in Table S3 in Appendix 3.a.

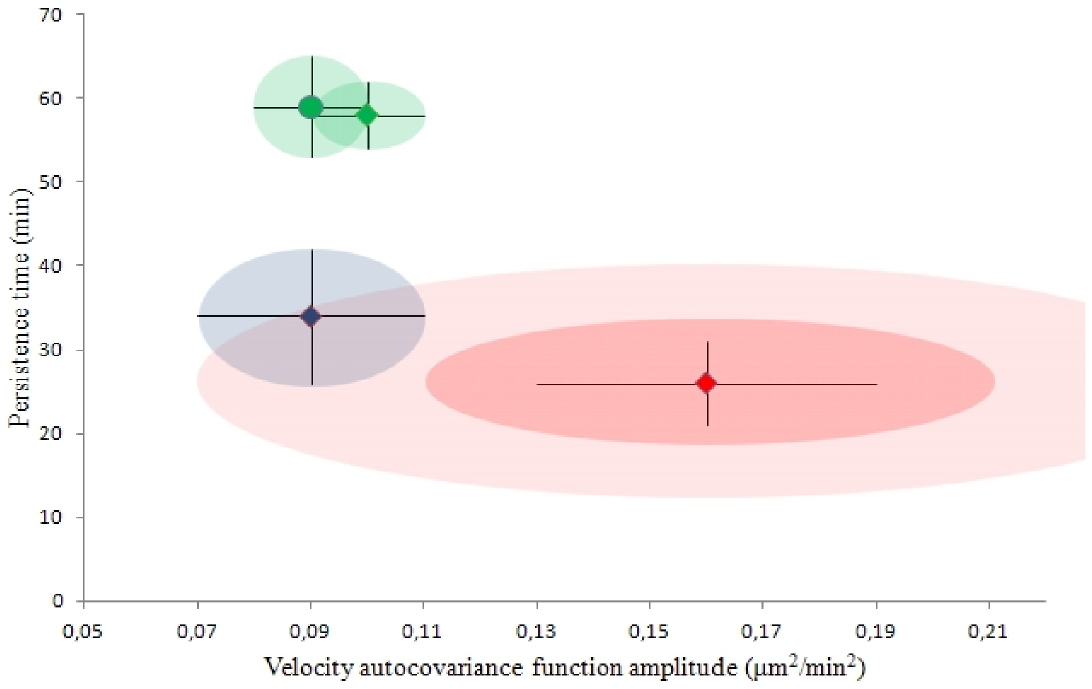


Figure 5A - Comparison of PACT, Autozell, and TLA for the purpose of cell motility analysis. Each of the datasets is depicted as a point in the associated P-φ_0 space. The black error bars show the overdispersion-*un*corrected 1σ (i.e. original fit results). Both the corrected 1σ (red ellipsoid) and 2σ levels (light red ellipsoid) are shown around the PACT point. In the absence of overdispersion information, the *un*corrected 1σ levels are shown as color-matched ellipsoids around the Autozell (blue) and TLA (green) points. The Kalman-filtered and moving-average smoothed P-φ_0 point is shown as a green circle, to distinguish it from the unfiltered, unsmoothed result, shown as a green square.

The motility parameters thus determined with Autozell and/or TLA differ substantially (φ_0 nearly halved, P nearly doubled) from the results obtained using PACT. The Autozell and PACT results are statistically equivalent as there is a clear overlap between the blue (Autozell, uncorrected 1σ) and light red ellipsoid (PACT, corrected 2σ), and a possible overlap within 1σ between the two programs if the Autozell results were data overdispersion-corrected. However, both TLA ellipsoids (1σ) remain well outside the range of overlap with the other two programs. We speculated that the Kalman filter or the moving average filter was the culprit; however, the results do not change much upon removal of these effects. This is perhaps the best illustration of the increase in fit imprecision due to the correlation of the velocity autocovariance data points. Indeed, the velocity autocovariance functions determined using the TLA datasets show significant structure (Table S4, Appendix 3.a), which would indeed lead to significant underestimation of the error bars, manifested as the substantially smaller green ellipsoids in figure 5. I am uncertain of the cause of the high correlation of the velocity autocovariance values as determined with TLA.

To account for the possible effects of not having used the tracks of the exact same cells in this analysis, we have also separately analyzed the common subset of tracks of the cells detected by all three programs (hereafter referred to as the subset). The results are shown in Figure 5B below and in Table S4 in Appendix 3.a.

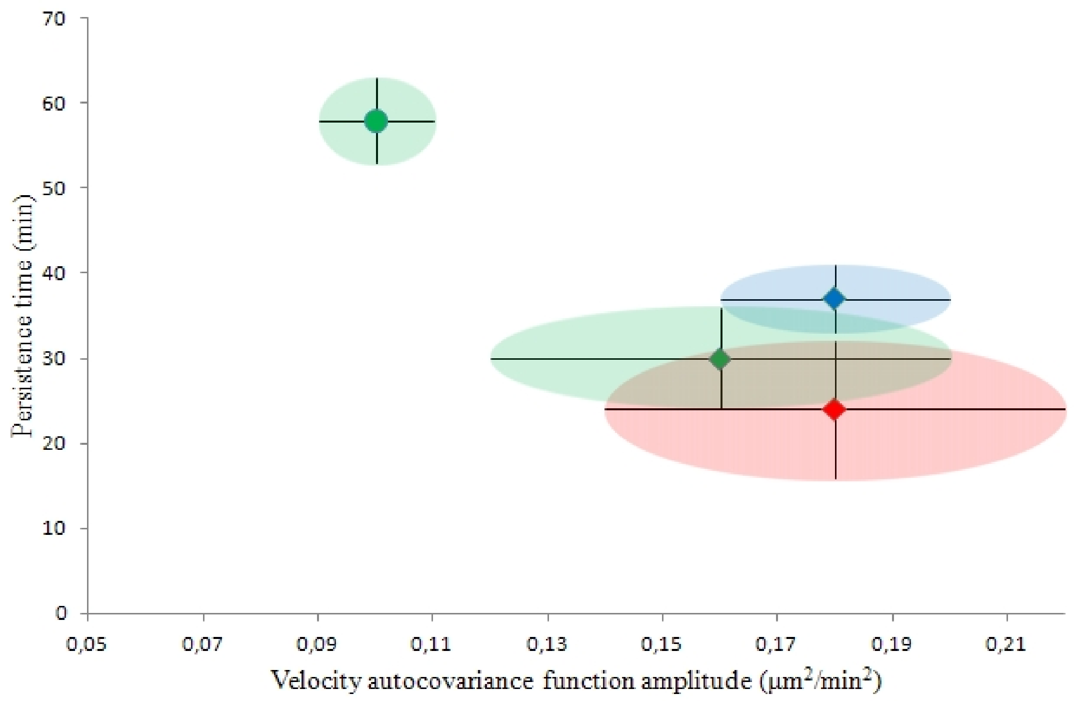


Figure 5B - Comparison of PACT, Autozell, and TLA for the purpose of cell motility analysis. Each of the subsets is depicted as a point in the associated P-φ_0 space. The black error bars show the overdispersion-*un*corrected 1σ (i.e. original fit results). In the absence of overdispersion information, the *un*corrected 1σ levels are shown as color-matched ellipsoids around the PACT (red), Autozell (blue), and TLA (green) points. The Kalman-filtered and moving-average smoothed P-φ_0 point is shown as a green circle, to distinguish it from the unfiltered, unsmoothed result, shown as a green square.

Only 3 of the tracks correspond to cells that were identified and tracked by all three programs. A comparison of cell motility parameters determined from raw data (i.e. without any filtering/smoothing procedure applied to the data) in fig 5B shows that there is no significant variation in the motility parameters, φ_0 and P. I conclude that the observed discrepancy between the motility parameters determined with the three programs is statistically insignificant: 2/3 persistence times match within 1σ and all (3/3) φ_0 parameters do as well (at most 68.3% expected to match), despite the shown error bars being underestimates of the real error bar levels.

I note, however, that the Kalman filtered- and moving average smoothed-subset yields a very different P-φ_0 set of results. This is at least in part due to the substantial underestimation of the respective error bars, owing to the highly structured velocity autocovariance functions (Figure S4, Appendix 3.a). Finally, to test the the possible effect of Autozell's tracking method, which involved truncating the centroid coordinate to the nearest pixel, we have similarly rounded the coordinates generated by PACT to the nearest pixel to generate the PACT/cut common dataset (Table S4, Appendix 3.a). I found that the motility parameters determined from this subset had not changed, and conclude that the systematic centroid truncation performed by Autozell is unlikely to affect the motility analysis results.

These observations raise the point of the importance of the definition of the cell centroid and the effect this definition has on cell motility statistical analysis. I elaborate on this below.

4. Biological Applications

To assess the influence of surface nanotopography on cell motility, we have imaged NIH 3T3 fibroblast on a range of Si black surfaces. The details of sample preparation have been described elsewhere (Lopacinska et al. 2011), and the cell handling and imaging has been done exactly as for the NIH 3T3/glass experiments (section III.5.a and III.5.b). The characteristics of the samples used are summarized in Figure 1 and Table 1 below.

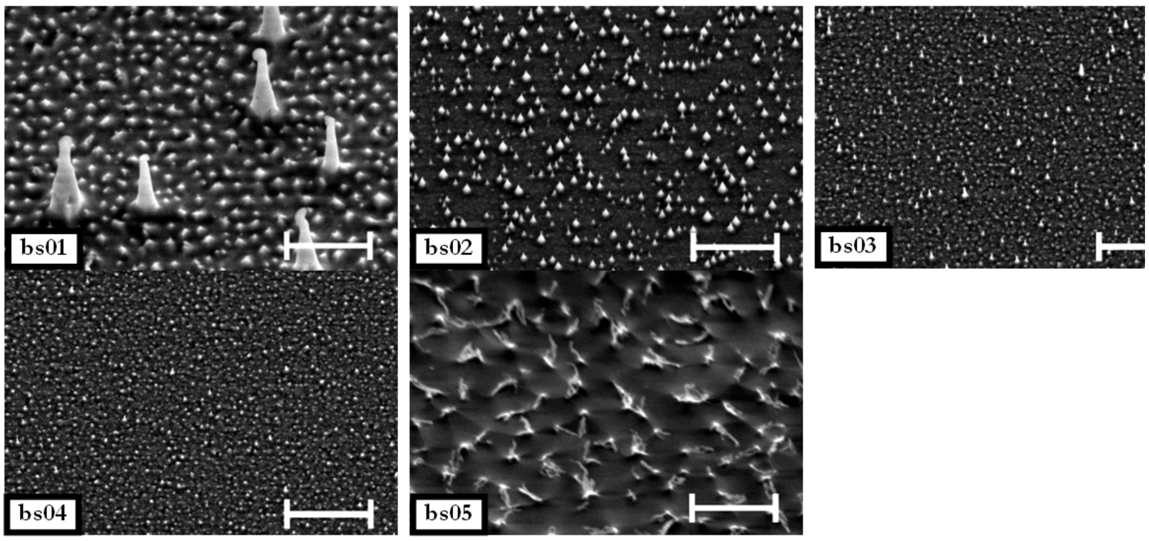


Figure 1: Geometrical characteristics of nanograss surface morphologies that were tested for NIH3T3 response. Substrate topography measured on SEM images. All images taken at 30º angle. Scale bars: 2 µm. (From Lopacinska et al. 2011).

Table 1 Characteristics of different nanograss surface morphologies tested for NIH3T3 response (Lopacinska et al. 2011).

Sample	Density	Pitch	Diameter	Height
	[part/um^2]	[um]	[um]	[um]
bs01A	5.30	0.43	0.198	0.333
bs01B	0.04	5.00	1.743	4.216
bs02	7.40	0.37	0.074	0.296
bs03A	24.3	0.20	0.071	0.061
bs03B	2.16	0.70	-	0.286
bs04	14.80	0.26	0.071	0.102

The motility parameters are shown schematically in figure 2.

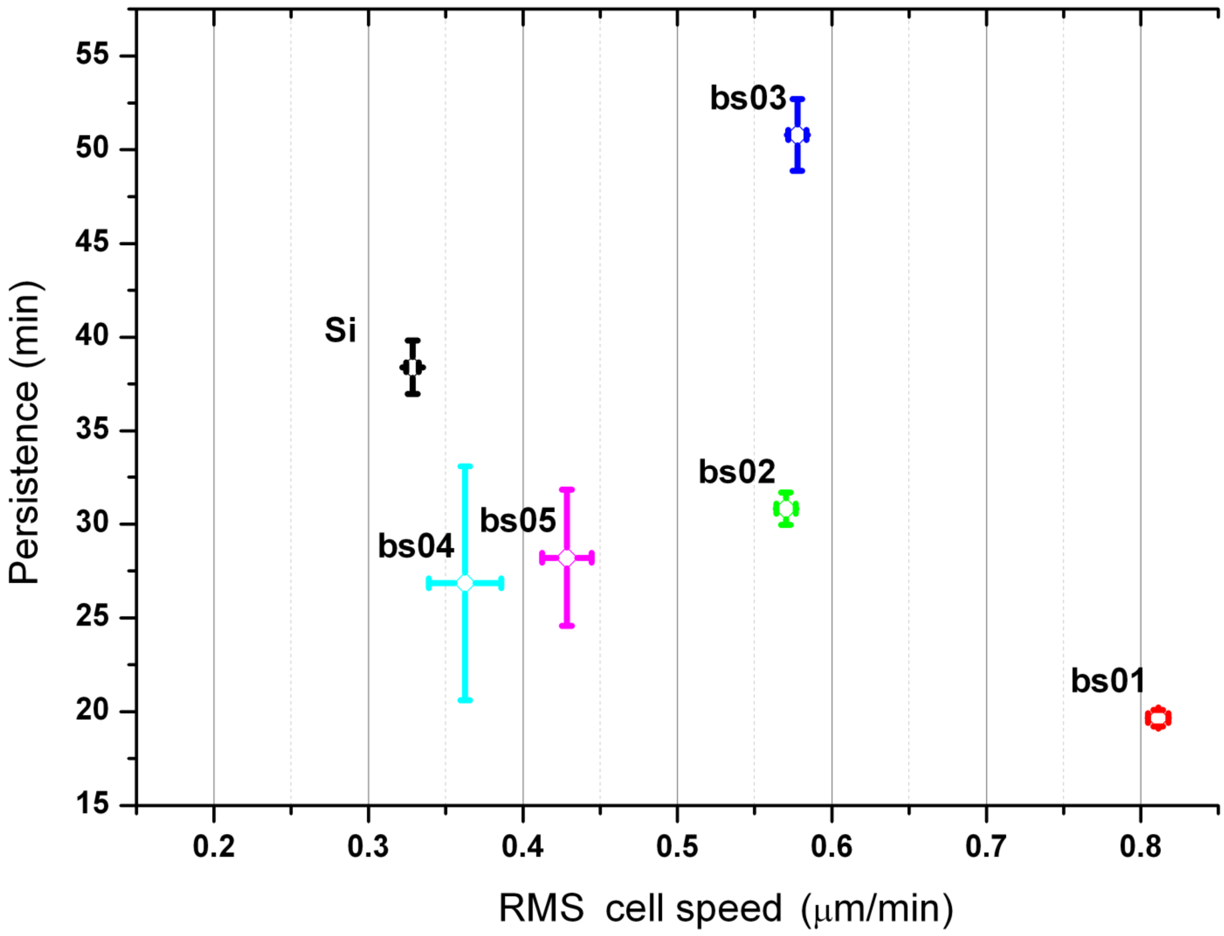


Figure 2 Persistence times P of cells on the different surfaces plotted against the values of the RMS cell speed. Both values were obtained by fitting a monoexponential function to the auto-covariance $\langle \vec{v}(t_1) \cdot \vec{v}(t_2) \rangle$ of velocities in trajectories. These results show that nanotopography alone can lead to large variations in the RMS cell speed and persistence time. This type of motility analysis provides a new and different parameterization of differences in cell responses to various surfaces. (From Lopacinska et al. 2011).

While no consistently clear monotonous relation of either of the two cell parameters on surface topology parameters can be found, this results indicates clearly a strong dependence on motility on surface topography, and paves the way for further investigations in the area.

5. Materials and Methods

a. Cell Culture and Substrate Nanofabrication

HeLa and NIH 3T3 cells were imaged on a microscope glass slide (Thermo Scientific, Menzel-Gläser), and NIH 3T3 cells on flat silicon and silicon black. Silicon black samples were fabricated by reactive ion etching of silicon substrates (Jansen and et al. 1995; Cui 2008).

The cell lines were obtained from Risø National Laboratory, Denmark. The cells were grown in Dulbecco's Modified Eagle Medium: Nutrient Mixture F-12 (DMEM/F12)

+ GlutaMAX (Invitrogen) supplemented with either 10% fetal bovine serum – FBS (Sigma) in the case of HeLa cells, or 10% of newborn calf serum for NIH3T3 cells, 2 mM L-glutamine (Sigma), 100 U/mL penicillin (Sigma), 100 μg/mL streptomycin (Sigma) and grown until confluency. Cells were harvested by a standard trypsinization method, then seeded at a concentration of 5 x 10^4 cells per each well of 24-well plate and cultured on the tested materials (1cmx1cm) for 1 or 3 days. The time-lapse microscopy experiments were performed in a home-made cell culture chamber equipped with a bubble trap and adjustable medium flow, mounted on a temperature-controlled microscope stage.

For the fluorescence microscopy experiments performed to compare brightfield and fluorescence microscopy centroids of the actin cytoskeleton of the cells, the cells were treated with 2% glutaraldehyde in 0.05M cacodylate buffer for 15-20 minutes at the room temperature, washed in 1xPBS containing 0.05% Tween-20, and permeabilized with 0.1% Triton X-100 in 1xPBS for 1-5 minutes at room temperature. After being washed three times with 1xPBS containing 0.05% Tween-20, the cells were incubated in TRITC-conjugated phalloidin (Sigma-Aldrich) for 30 minutes, and rinsed three times with 1xPBS containing 0.05% Tween-20.

b. *Data and Image Sequence Acquisition*

The videos contain 8-bit, grayscale images recorded with a temporal resolution of 2-10 minutes. Each image pixel has a size of 0.977 x 0.977 μm and the resolution of the images was 1024 x 768. The recording device was a Zeiss Axiotech microscope equipped with a 10x Zeiss objective with a 19 mm working distance and a field of view of 1000 x 750 μm2 and a Labview-controlled microscope stage. The acquisition technique was bright field microscopy and reflected light microscopy was used as many of the substrates were not transparent to light. The fluorescence and bright field microscopy experiments with fixed cells were performed on an Olympus BX51 upright microscope. Analysis of bright field images was performed using the TLA setup file provided by the authors. By default, TLA tracks cells imaged in bright field microscopy by first applying a low-pass Gaussian filter of size 25x25 pixels and standard deviation 11 pixels. A Wiener low-pass filter in a 15x15 pixel mask around each pixel is applied to remove the pixel noise the image, and this is followed by the actual segmentation process.

c. *PACT Algorithm, Program Implementation, Structure and Functionality*

PACT was implemented using MATLAB (ver. 2008a) and is a text-based, interactive, open source application for the analysis of cell motility data such as the methods described by Selmeczi et al. (Selmeczi et al. 2005). This analysis requires accurate measurements of cell centroid coordinates throughout the duration of their observation.

Processing a time lapse movie in PACT consists of: *Tracking* the individual moving objects through a sequence of images, where the objects have been localized by an image segmentation routine on images after a suitable image filtering process for optimal segmentation results. PACT employs a combination of two of the simplest and fastest segmentation methods: thresholding and edge detection. The result of a PACT processing is a time-ordered list of centroid coordinates for each object that is tracked.

Root mean square deviation filtering (*RMSD-filtering,* see point (v) above) is an optional last step of processing and removes tracks of objects whose displacement is below a user-defined threshold. This option automatically removes tracks of non-motile cells, such as cells immobilized on the surface and lysed cells.

Post-processing is the manual screening of the processed data using criteria of biological relevance, which results in final identification of biologically relevant *cells* amongst the objects tracked. Human judgment is needed to decide which tracks are of sufficient quality for inclusion in the data analysis. This involves discarding segments of tracks where cells either: (a) undergo division; (b) come into contact with one another; (c) carry foreign objects; (d) and optionally cells that do not display normal motility patterns (i.e. those that do not extend lamellipodia or filopodia).

The source code of PACT is provided in Appendix 5. The program takes the recorded movie frames as input and outputs the cell centroid coordinates. The frames must be located in the same folder as the source code of the program and a new folder ("processed") is created where overlay images of original frames and the contours and cell centroids are stored. The section "*working parameters*" in the main program may be used to change the name of the input files and to invert the images if needed. The frame may be color or gray-scale; PACT will remove the color component if present. 8-bit depth is sufficient if the contrast is good, else 16-, 24-, etc. may be needed.

For image filtering, a band-pass filter that is part of 'IDL (Interactive Data Language) Particle Tracking' package (Crocker et al.) is first applied to remove pixel noise and non-uniform background illumination effects (Figure 6). This is done by convolving (○ operator) the image array *I* in two steps with a Gaussian function *G* and a boxcar function *B* and subtracting the result.

$$\mathit{Filtered}\ I = \left(I^T \circ G^T\right)^T \circ G^T - \left(I^T \circ B^T\right)^T \circ B^T$$

Equation 3

In Fourier space this amounts to applying a radially symmetric mask function which features a Gaussian decline at high frequencies and an abrupt fall at low frequencies. The limits of this band-pass filter are also set up in the *working parameters* section: the high-frequency limit by the standard deviation of *G,* and the low-frequency limit by the width of *B*. We found that a high-frequency limit of 2-3 pixels works well for most flat surfaces, and that as much as eight pixels may be required for the highly noisy backgrounds that we have encountered when imaging cells on silicon black. We typically set the low frequency limit to 30-50 pixels as this corresponds approximately to the size of an average cell in our microscopy setup and efficiently filters inhomogeneous background illumination.

For the segmentation, PACT has a text-based, interactive user interface, which iteratively requests two parameters until the user is satisfied with the selection. The first parameter is the *peak exclusion threshold* which defines the minimal brightness of a spot in the band-pass filtered image required for inclusion on the object list. The peak (brightest pixel) of the object is marked as a red cross on the image. The cell contour shown as a blue line in the image is defined as the isohypse located at a user-defined percentage (*the contour cutoff*) of the peak level (Figure 6). The segmentation is done manually for the first and last frames in the movie, and all intervening frames are processed automatically with an average of the contour cutoffs and linear interpolation of the peak exclusion thresholds set for these two frames. The linear interpolation is especially useful for long movies where the background and/or object brightness may change slowly in time due to factors external to the experiment (see section (ii) above).

Further selection includes restrictions on minimal and maximal cell area (to remove unwanted objects such as small pieces of dust or large air bubbles and cell conglomerates from the analysis) and minimal distance between cells to ensure that only independent cells are detected. These parameters may also be set in the *working parameters* section.

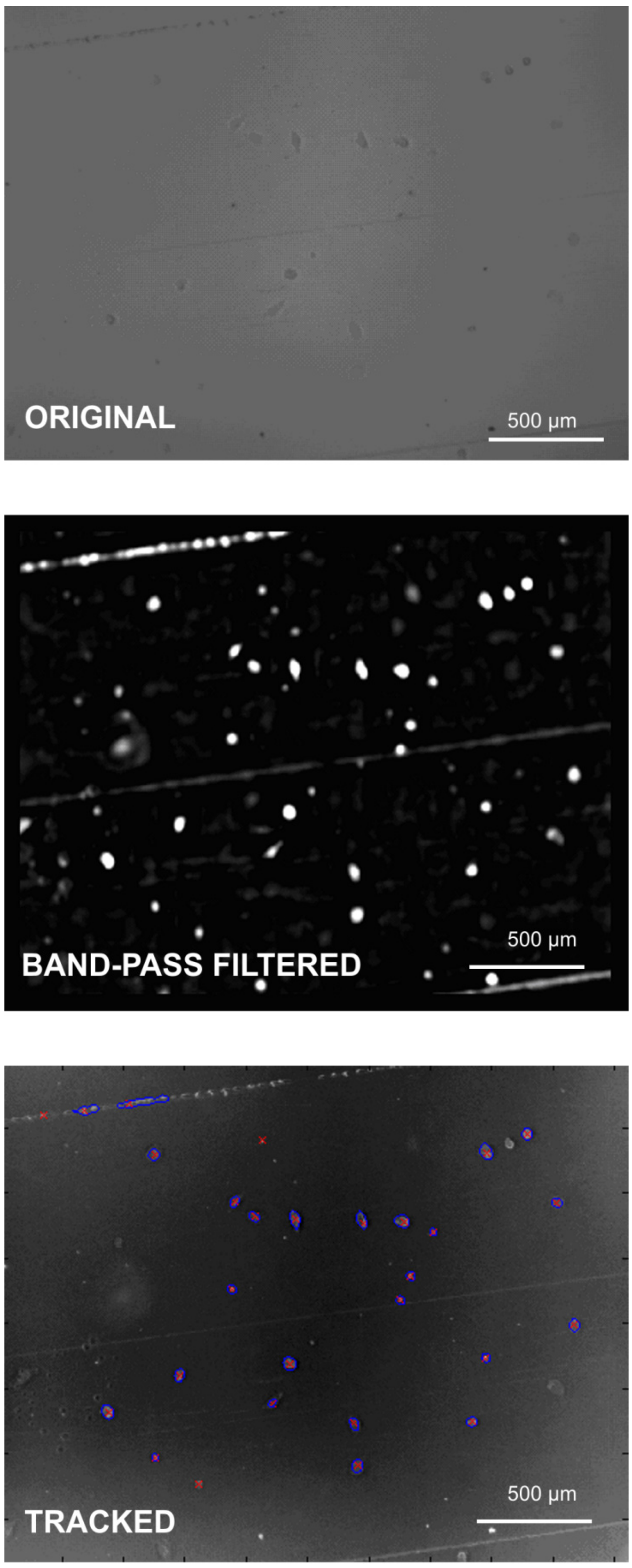


Figure 6 - Image segmentation in PACT: band-pass filtering followed by segmentation by thresholding and edge detection. Image of HeLa/glass (top), band-pass filtered using frequency limits of 3 and 30 pixels (middle), and processed frame (bottom) showing peaks brighter than peak-exclusion threshold of 60 as red crosses and cell contours as blue shapes. Some peaks do not have a blue contour around them, as they did not pass the additional cell selection criteria (size and proximity to other cells).

The tracking process entails sorting this independent list of coordinates (determined at discrete times as the centroid of the cell footprint) into particle tracks using a routine part of 'IDL Particle Tracking' package (Crocker et al.). If desired, it is possible to add a last exclusion criterion for removing the non-moving cells on the basis of an RMSD filter (see RMSD-filtering and data post-processing sections above).

6. *Discussion and Conclusions*

The segmentation efficiency tested in (i) was performed by 4 programs showed in table 2 that perform reasonably well on flat substrates with detection efficiencies up to ranging from 90% of the manual count to about 60% especially for nanostructured substrates. These values were obtained by one user optimizing settings in the present case scenario and may vary between uses and time lapse movies. Image filtering routines were shown to be essential for efficient detection of cells on structured backgrounds (iv), and these routines are not available with all programs (Table 1). Additionally, settings may need to be optimized over time in a movie (ii) due variations in imaging conditions, which may be aided by the linear interpolation possible in PACT. The variations in segmentation combined with track formation algorithms such as the RMSD filter (v) can result in very different yield of tracks from the same data set. This is especially true when looking the complete tracks (ix), where the joint dataset is only three tracks. The overall analysis process may be prone to biased selection of subpopulations of cells with specific features, such as proximity to other cells and specific morphology features, all depending on the program being used, sample details and users choice of settings. For this reason, I recommend that before performing automated time lapse analysis on a specific movie, the users assess how well the program performs compared to manual counting and consider carefully if the undetected cells might result in a bias in the detected cell population.

I developed PACT for tracking cells on non-uniform backgrounds. To that end, I tested PACT on silicon black substrates. Cells on such substrates are often difficult to distinguish from the background even by eye. TLA and PACT are the only programs that successfully tracked cells in these images. The Wiener filter that TLA employs in its bright field tracking method is comparable to the convolution filter implemented in PACT with regard to cell recognition. However, I have specifically designed PACT to reduce operator effort at the post-processing step, which is the most time-consuming step of the analysis. I accomplished this by implementing additional automated post-processing features: restrictions on the area of the object and minimal inter-cell distance. These features make PACT less likely to find false positives and this is reflected in the percentage of objects that pass the post-processing step of PACT in comparison with TLA: 81% versus 72% for flat Si and 52% versus 31% for silicon black. I conclude that PACT is more useful for the sparse cell culture motility analyses we perform in this paper. PACT can, however, be less efficient for applications such as dense cell cultures.

The positional error was shown in section (vii) to be of the order 1-2μm. Compared to this noise level, the cell centroids appear to be fairly reliably determined: They are not strongly influenced by variations in focus and threshold settings, but scatter with an RMSD of 1μm for a 20 μm focus variation in the test performed in section (ii). The brightfield image centroid position correlates with the fluorescence microscopy results with an RMSD on flat surfaces of 2.8μm as shown in (iii), while nanostructured substrates increase the RMSD to 4.3μm. This indicates that bright field microscopy can be used for motility analysis with precision comparable to that of fluorescence microscopy and that the overall process is not strongly influenced by variations in the experimental procedure and image segmentation.

I tested the internal consistency of the velocity auto-covariance function measurements on cell motility in section (viii) and found that the results are consistent, especially when accounting the artificial underestimation of the motility parameters' error bars due to data overdispersion. Comparing the motility parameters obtained with different programs in (ix), Figures 5A and 5B, I find that the different programs give statistically consistent results. PACT and Autozell provide motility parameters that agree well within their respective uncertainty, when data overdispersion effects are accounted for, with the exception of TLA which leads to unusually high correlations in the velocity autocovariance function values and subsequent under- or over-estimation of motility parameters.

Finally, we have implemented the features present in PACT (RMSD filter, track screening) in TLA and we are currently implementing the parameter estimation as an add-on to TLA.

In conclusion, this study shows that cell tracking can be done with different programs, often with reasonable segmentation efficiency and precision: the pair-wise RMSD of tracks output by two different programs is 2-3μm, consistent with a positional noise per dataset of ~1μm. The velocity auto-covariance function analysis demonstrates that different programs provide comparable precision in the motility parameters P and φ_0 in a statistically significant manner.

7. References

Ali R, Gooding M, Christlieb M, Brady M. Advanced phase-based segmentation of multiple cells from bright-field microscopy images; 2008 14-17 May 2008. pp. 181-184.

Althoff K (2005) Segmentation and Tracking Algorithms for in Vitro Cell Migration Analysis: Chalmers University of Technology.

Baumann H (2006) Image-Blackboards und Algorithmen zur Bearbeitung von Zellkollisionen im Zelltracking-System AUTOZELL.

Boldajipour B, Mahabaleshwar H, Kardash E, Reichman-Fried M, Blaser H et al. (2008) Control of Chemokine-Guided Cell Migration by Ligand Sequestration. Cell 132(3): 463-473.

Carlier M-F, Clainche CL, Wiesner S, Pantaloni D (2003) Actin-based motility: from molecules to movement. BioEssays 25(4): 336-345.

Crocker JC, Grier D, Weeks E Particle tracking using IDL. Available: http://www.physics.emory.edu/~weeks/idl/.

Cui Z (2008) Nanofabrication: principles, capabilities and limits. New York: Springer Science+Business Media, LLC.

Dormann D, Weijer CJ (2006) Imaging of cell migration. EMBO J 25(15): 3480-3493.

Dufour A, Shinin V, Tajbakhsh S, Guillen-Aghion N, Olivo-Marin JC et al. (2005) Segmenting and tracking fluorescent cells in dynamic 3-D microscopy with coupled active surfaces. Image Processing, IEEE Transactions on 14(9): 1396-1410.

Gerlich D, Mattes J, Eils R (2003) Quantitative motion analysis and visualization of cellular structures. Methods 29(1): 3-13.

Hand AJ, Sun T, Barber DC, Hose DR, Macneil S (2009) Automated tracking of migrating cells in phase-contrast video microscopy sequences using image registration. Journal of Microscopy 234(1): 62-79.

Henrickson SE, Mempel TR, Mazo IB, Liu B, Artyomov MN et al. (2008) T cell sensing of antigen dose governs interactive behavior with dendritic cells and sets a threshold for T cell activation. Nature Immunology 9(3): 282-291.

Huth J, Buchholz M, Kraus JM, Schmucker M, von Wichert G et al. (2010) Significantly improved precision of cell migration analysis in time-lapse video microscopy through use of a fully automated tracking system. BMC Cell Biology 11(24): 24.

Jansen H, et al. (1995) The black silicon method: a universal method for determining the parameter setting of a fluorine-based reactive ion etcher in deep silicon trench etching with profile control. Journal of Micromechanics and Microengineering 5(2): 115.

Kalman RE (1960) A new approach to linear filtering and prediction problems. Transactions of the ASME - Journal of Basic Engineering 82(Series D): 35-45.

Keren K, Pincus Z, Allen GM, Barnhart EL, Marriott G et al. (2008) Mechanism of shape determination in motile cells. Nature 453(7194): 475-480.

Leymarie F, Levine MD (1993) Tracking deformable objects in the plane using an active contour model. IEEE Transactions on Pattern Analysis and Machine Intelligence 15(6): 617 - 634.

Li K, Miller ED, Chen M, Kanade T, Weiss LE et al. (2008) Cell population tracking and lineage construction with spatiotemporal context. Medical Image Analysis 12(5): 546-566.

Li L, Cox EC, Flyvbjerg H (2011) "Dicty Dynamics": Dictyostelium motility as persistent random motion. Phys. Biol. 8 (2011) 046006 (12 pages)

Li S, Guan J-L, Chien S (2005) Biochemistry and Biomechanics of Cell Motility. Annual Review of Biomedical Engineering 7(1): 105-150.

Lopacinska S, Guan J-L, Chien S (2005) Biochemistry and Biomechanics of Cell Motility. Annual Review of Biomedical Engineering 7(1): 105-150.

Lopacinska J-M, Grădinaru C, Wierzbicki R, Schmidt M-S, Madsen M-T, Skolimovski M, Dufva M, Flyvbjerg H, Mølhave K (2011) Cell Motility, Morphology, Viability and Proliferation in response to Nanotopography on Silicon Black. Nanoscale.

Meijering E, Dzyubachyk O, Smal I, van Cappellen WA (2009) Tracking in cell and developmental biology. Seminars in Cell & Developmental Biology 20(8): 894-902.

Michl P, Ramjaun AR, Pardo OE, Warne PH, Wagner M et al. (2005) CUTL1 is a target of TGF² signaling that enhances cancer cell motility and invasiveness. Cancer cell 7(6): 521-532.

O. Debeir PVH, R. Kiss, C. Decaestecker (2005) Tracking of Migrating Cells Under Phase-Contrast Video Microscopy With Combined Mean-Shift Processes. IEEE Transactions on Medical Imaging 24: 697.

Ornstein LS (1919) On the Brownian Motion. Koninklijke Nederlandse Akademie van Weteschappen Proceedings Series B Physical Sciences 21: 96-108.

Pal NR, Pal SK (1993) A review on image segmentation techniques. Pattern Recognition 26(9): 1277-1294.

Presley JF (2005) Imaging the secretory pathway: The past and future impact of live cell optical techniques. Biochimica et Biophysica Acta (BBA) - Molecular Cell Research 1744(3): 259-272.

Sacan A, Ferhatosmanoglu H, Coskun H (2008) CellTrack: an open-source software for cell tracking and motility analysis. Bioinformatics 24(14): 1647-1649.

Sbalzarini IF, Koumoutsakos P (2005) Feature point tracking and track analysis for video imaging in cell biology. Journal of Structural Biology 151(2): 182-195.

Selmeczi, Mosler, Hagedorn, Larsen, Flyvbjerg (2005) Cell motility as persistent random motion: theories from experiments. Biophysics Journal 89: 912-931.

Sethian JA (1999) Level Set Methods and Fast Marching Methods. Cambridge, U.K.: Cambridge Univ. Press.

Stuurman N Mtrack2. Available: http://valelab.ucsf.edu/~nico/IJplugins/MTrack2.html.

Tang Q, Adams JY, Tooley AJ, Bi M, Fife BT et al. (2006) Visualizing regulatory T cell control of autoimmune responses in nonobese diabetic mice. Nature Immunology 7(1): 83-92.

Uhlenbeck GE, Ornstein LS (1930) On the Theory of the Brownian Motion. Physical Review 36(5): 823.

Van Haastert PJM, Devreotes PN (2004) Chemotaxis: signalling the way forward. Nat Rev Mol Cell Biol 5(8): 626-634.

Wolf K, Wu YI, Liu Y, Geiger J, Tam E et al. (2007) Multi-step pericellular proteolysis controls the transition from individual to collective cancer cell invasion. Nature Cell Biology 9(8): 893-904.

Xiaodong Yang, Houqiang Li, Zhou X (2006) Nuclei Segmentation Using Marker-Controlled Watershed, Tracking Using Mean-Shift, and Kalman Filter in Time-Lapse Microscopy. IEEE Transactions on Circuits and Systems Part I: Regular Papers 53: 2405.

Xiaowei C, Xiaobo Z, Wong STC (2006) Automated segmentation, classification, and tracking of cancer cell nuclei in time-lapse microscopy. IEEE Transactions on Biomedical Engineering 53(4): 762-766.

Yumura S, Fukui Y (1998) Spatiotemporal dynamics of actin concentration during cytokinesis and locomotion in Dictyostelium. Journal of Cell Science 111(15): 2097-2108.

Chapter IV: Cell motility on anisotropic substrates

1. *Anisotropic motility*

a. *Motivation*

Recent work on the interaction between cells and nanostructured substrates has indicated that anisotropic substrates have the potential of directly influencing isotropy of cell morphology and motility. By way of an example, the laboratories of Yong Chen have imaged cells on grooves and lines coated with specific adhesion promoting proteins, as shown in the images below (Hu et al.)
.

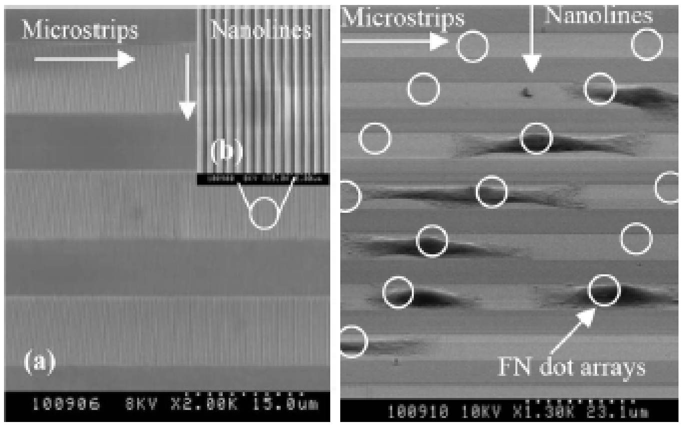


Figure 1: SEM image of nanogratings arrays defined by soft UV-NIL, contact photolithography and reactive ion etch techniques (left) and SEM image of cells on a hybrid pattern, showing elongation and alignment of single HeLa cell cultured on hybrid patterns for 36h (right). (Hu et al.)

Static images like these show the achieved biological structure, but tell little about the dynamics, e.g. for how long time the structure will remain. Here I examine varying types of anisotropic motility of 3T3 fibroblast on substrates patterned with grooves. Guided by various motility statistics, we find a model that describes the observed anisotropic motility.

b. Current advances

Modeling of cell motility on isotropic substrates has reached a significant level of sophistication (Selmeczi, Mosler et al. 2005; Li, Cox, and Flyvbjerg 2011). By contrast, modeling of motility in anisotropic environments has essentially been limited to the study of chemotaxis, which is cell motility in response to chemical gradients (Watkins 1995; Ionides, Fang et al. 2004; Simpson. 2004; Neilson, Veltman et al. 2011).

There are a limited number of studies of motility on translationally invariant substrates, mostly concerned with determining diffusion-like parameters of cell motility on anisotropic surfaces or interpreting the differences in velocity along the axis of anisotropy vs. perpendicular to it without presenting a specific model of motility (Dunn and Brown 1986; Dickinson, Guido et al. 1994; Dickinson 2000). Some of these studies present transport models (i.e., models of cell probability density on the surface) (Patlak 1953; Tranquillo and Lauffenburger 1987; Dickinson 2000) and only a few of those are stochastic models of motility (Matthes and Gruler 1988; Stokes, Lauffenburger et al. 1991).

2. Anisotropic motility: theory and supporting experimental data

The dataset consists of time lapse movies of the cells, with images recorded at constant time intervals Δt. The individual cell positions in images $\vec{r}_j$ =[x(t$_j$), y(t$_j$)] were determined using PACT. From the observed positions, the secant velocities $\vec{u}_j$ and secant accelerations $\vec{\alpha}_j$ can be determined:

$$\vec{u}_j = \frac{\vec{r}_{j+1} - \vec{r}_j}{\Delta t} \text{ and } \vec{\alpha}_j = \frac{\vec{u}_{j+1} - \vec{u}_j}{\Delta t}$$

Equation 1

I remind the reader here of the distinction between these quantities and the instantaneous velocities and accelerations, for which I keep the traditional notation $\vec{v}(t)$ and $\vec{a}(t)$. The secant velocities $\vec{u}_j$ and secant accelerations $\vec{\alpha}_j$ are discrete time series, whereas $\vec{v}(t)$ and $\vec{a}(t)$ are continuous functions of time. Only the secant quantities are available from a real experiment and differ from $\vec{v}(t)$ and $\vec{a}(t)$ due to discretization effects owing to the finite frame rate Δt (II.5.a). They also include contributions from the positional noise, which is the inherent measurement error that affects each cell centroid values output by the tracking program. In the limit where the positional noise and Δt become vanishingly small, the secant quantities equal their instantaneous counterparts.

a. The anisotropic OU process

Below I will describe how to analyze the data to derive an OU-like model that best describes the corresponding cell motility. Cell motility models may be described as OU-like processes (III.2.a). The standard OU process follows equation 2, section III.2.a:

$$\frac{d\vec{v}}{dt}(t) = -\frac{1}{P}\vec{v}(t) + s\vec{\xi}(t)$$

In the standard OU model the parameter *P* is interpreted as the persistence time of the motility and *s* is the cell motility coefficient. Using secant quantities, this becomes equation 3, section III.2.a:

$$\vec{\alpha}_j \overset{\text{def}}{=} \frac{\Delta\vec{u}_j}{\Delta t} = -\beta\vec{u}_j + \sigma\vec{\eta}_j$$

The β and σ parameters can be determined directly from plots of α and RMSD(α) vs. *u*, and depend on the experimental setup through the positional noise and discretization of time (III.2.a). A system-specific set of parameters, *P* and *s*, require indirect methods to determine them (II.5.b and II.5.c).

As a reminder, for the OU process, the secant velocities can be used to calculate P as the characteristic time of the velocity autocorrelation function (equation 3, section II.4.a):

$$\varphi_j^{\exp} = \frac{\sum_{\alpha=1}^{last\ track}\sum_{k=0}^{N_\alpha - j}\vec{u}(t_j + t_k)\cdot\vec{u}(t_k)}{\sum_{\alpha=1}^{last\ track} N_\alpha}$$

By fitting a monoexponential function to the remaining values of φ, the persistence time P can be determined accurately as the characteristic time of the monoexponential fit (II.5.c). I also remind the reader here that the first two points of the autocorrelation velocity function are also affected by more subtle effects (e.g. the fine oscillations in trajectory due to pseudopod action) (Li et al., Phys Biol), which makes this method of determining positional noise subject to some error.

In the following I define the method used to assess the dominant characteristics of the observed cell motility and how to determine if an OU process or a modified version with velocity dependent parameters. Based on the scalar OU process, a general tensor version for anisotropic motility in an unmodified OU process (i.e. P and s independent of v) can be written as

$$\frac{d\vec{v}}{dt}(t) = -\begin{bmatrix} P_x & P_{yx} \\ P_{xy} & P_y \end{bmatrix}^{-1}\vec{v}(t) + \begin{bmatrix} s_x & s_{xy} \\ s_{xy} & s_y \end{bmatrix}\vec{\xi}(t)$$

Equation 1

Using secant quantities, this equation becomes

$$\vec{\alpha}_j = \frac{\Delta\vec{u}_j}{\Delta t} = -\begin{bmatrix} \beta_x & \beta_{xy} \\ \beta_{xy} & \beta_y \end{bmatrix}\vec{u}_j + \begin{bmatrix} \sigma_x & \sigma_{xy} \\ \sigma_{xy} & \sigma_y \end{bmatrix}\vec{\eta}_j$$

Equation 2

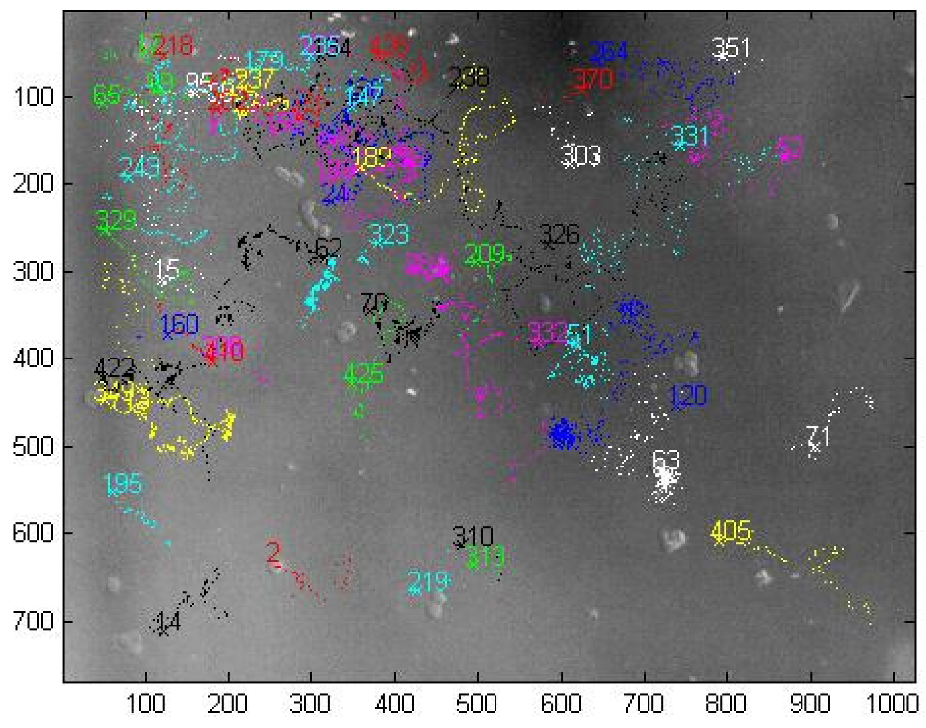
This analysis will study to what extent such a model or modification thereof can be used to describe the data.

b. Experimental data analysis

I have analyzed NIH 3T3 fibroblast motility on topographically anisotropic surfaces. The experimental dataset consists of movies of fibroblasts on PDMS substrates patterned with grooves of 150, 200, and 500 nm respectively (width=depth, aspect ratio 1). A negative control of fibroblast motility on flat PDMS has also been recorded "flat control." The results are summarized in table 1 and the final trajectories following RMSD-filtering are shown in Figures 1-4. Motility on the flat substrate is isotropic, as expected, and the patterned substrates show various degrees of anisotropy. A monotonous relationship exist between the stripe width and χ only if the flat case is regarded as an anisotropic substrate with pitch width of 0 nm.

Sample	χ
500 nm	0.62
200 nm	0.68
150 nm	0.82
Flat	0.94

Table 1: Anisotropy coefficient vs stripe width.



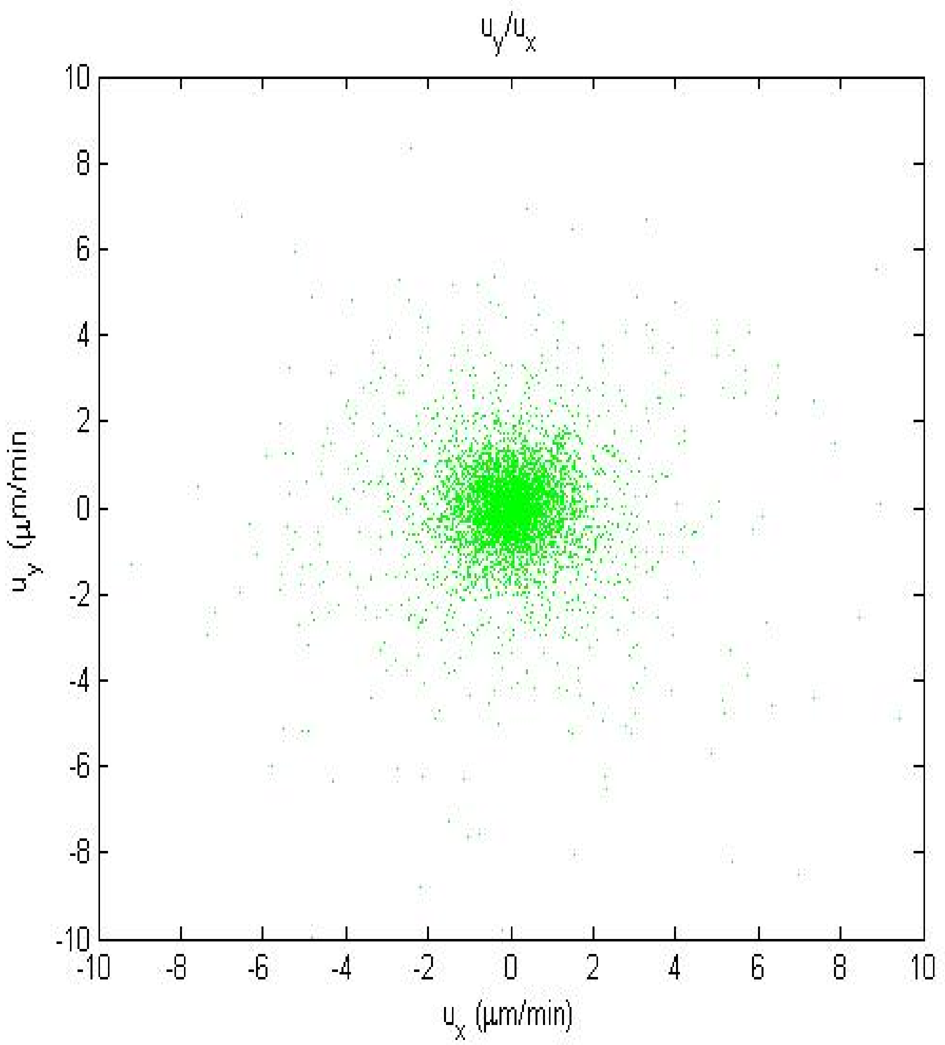
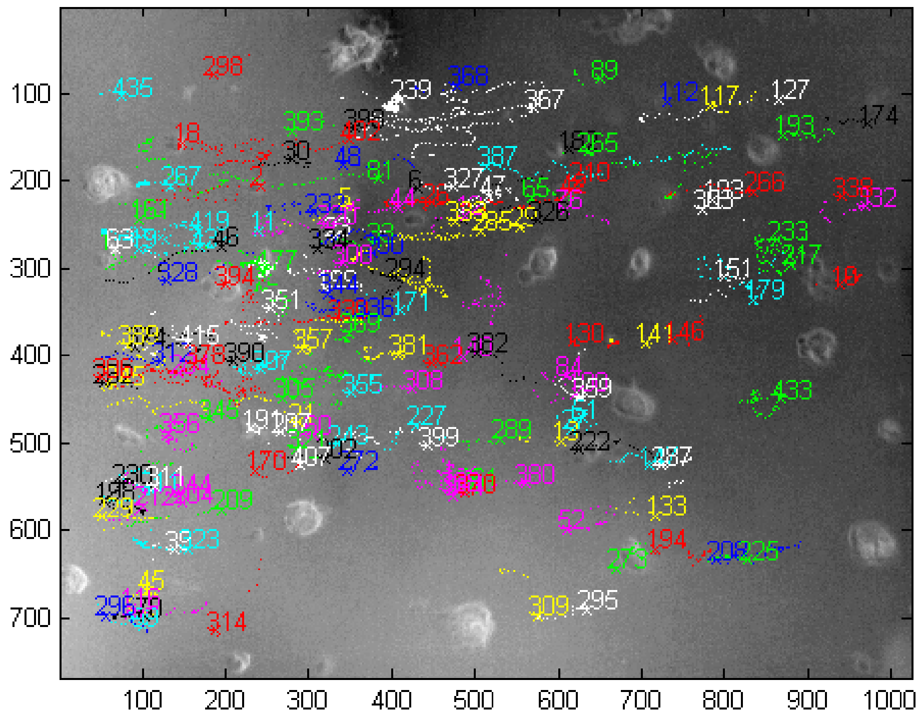


Figure 1. Tracks of 3T3 fibroblasts on *flat* PDMS visualized by light microscopy, tracked by PACT (top). (1 pixel = 0.977 μm; numbers on axes show distance in μm); and plot of x and y components of secant velocities (bottom): $\chi = 0.94$.


100
200
300
400
500
600
700
100
200
300
400
500
600
700
800
900
1000

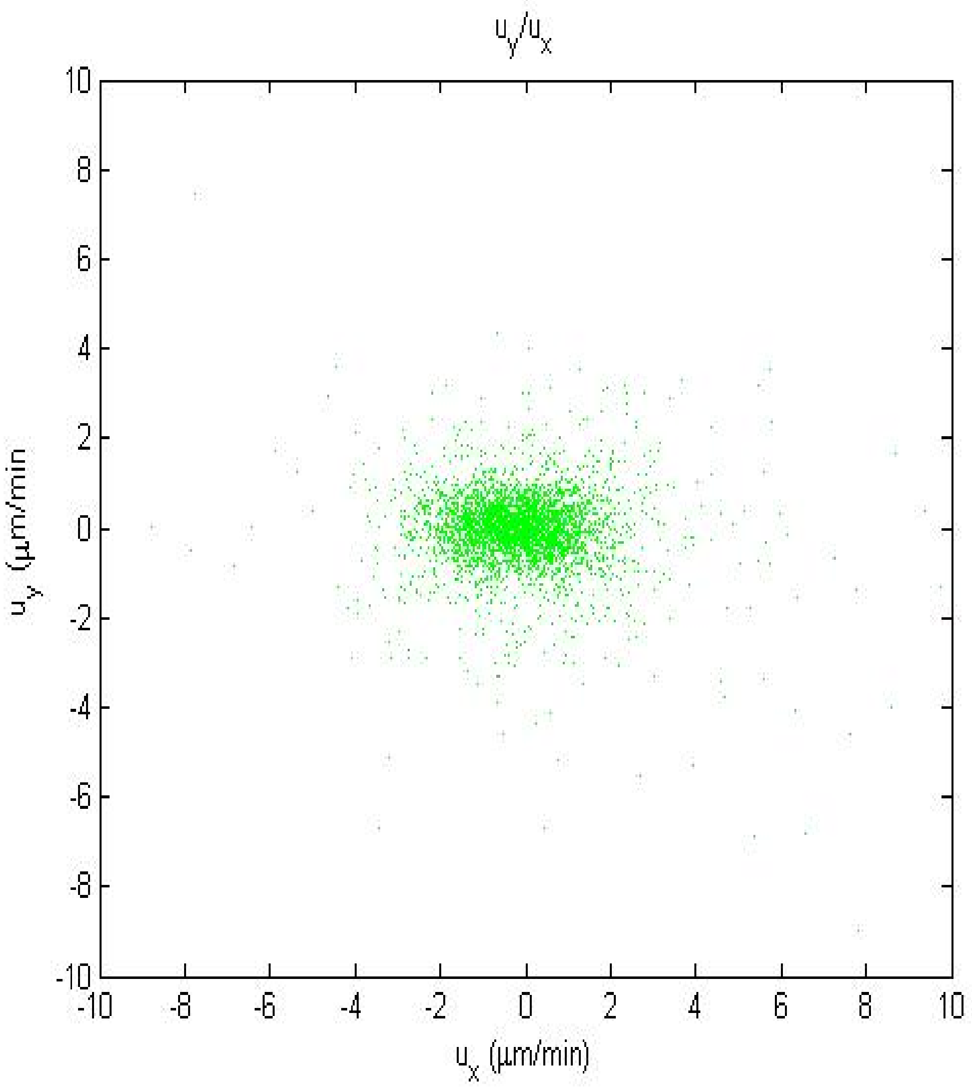
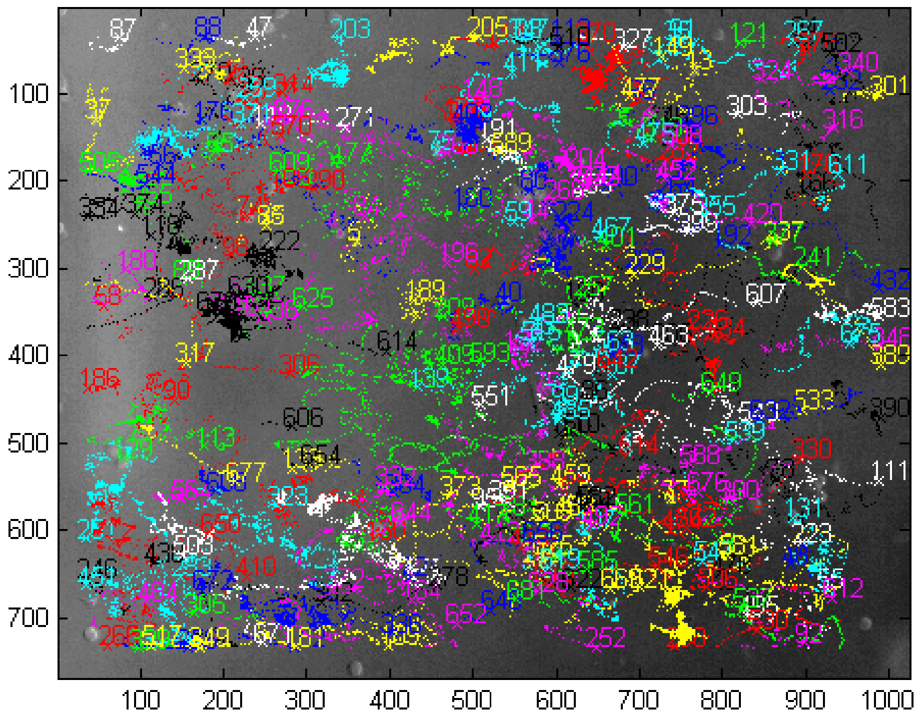


Figure 2. Tracks of 3T3 fibroblasts on *500 nm* patterned substrates visualized by light microscopy, tracked by PACT (top). (1 pixel = 0.977 μm; numbers on axes show distance in μm); and plot of x and y components of secant velocities (bottom) : $\chi = 0.62$.


100
200
300
400
500
600
700
100
200
300
400
500
600
700
800
900
1000

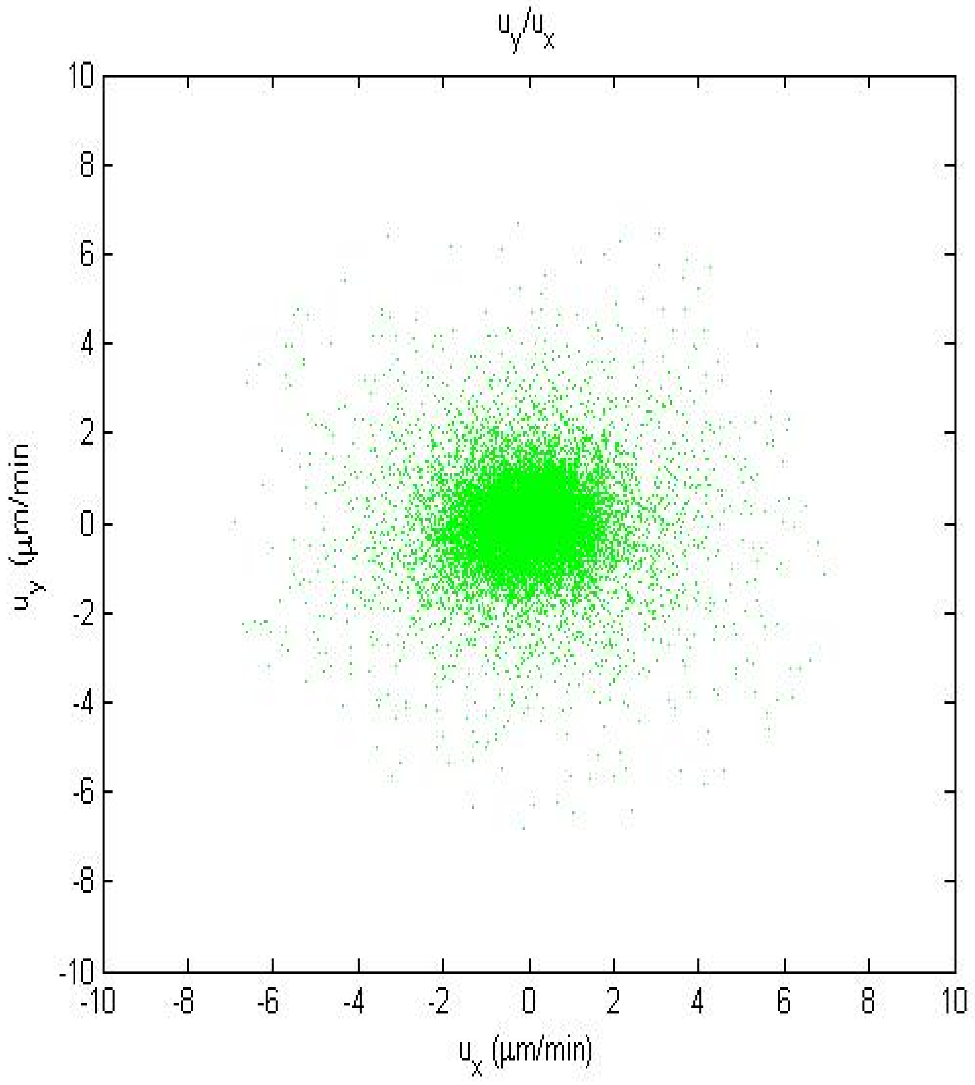
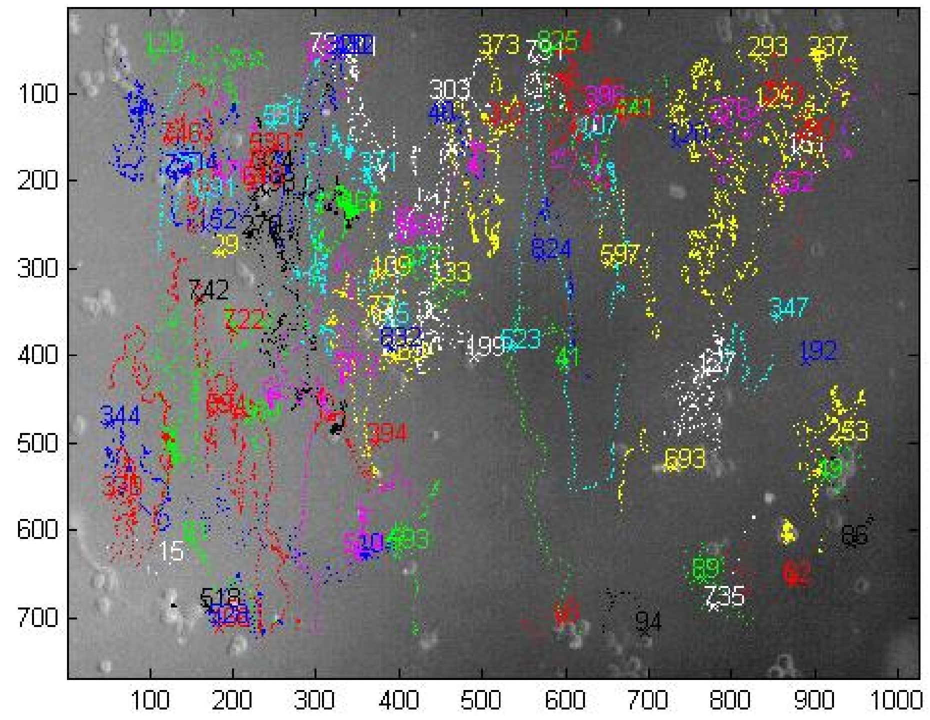


Figure 3. Tracks of 3T3 fibroblasts on *150 nm* patterned substrates visualized by light microscopy, tracked by PACT (top). (1 pixel = 0.977 μm; numbers on axes show distance in μm); and plot of x and y components of secant velocities (bottom) : $\chi = 0.82$.



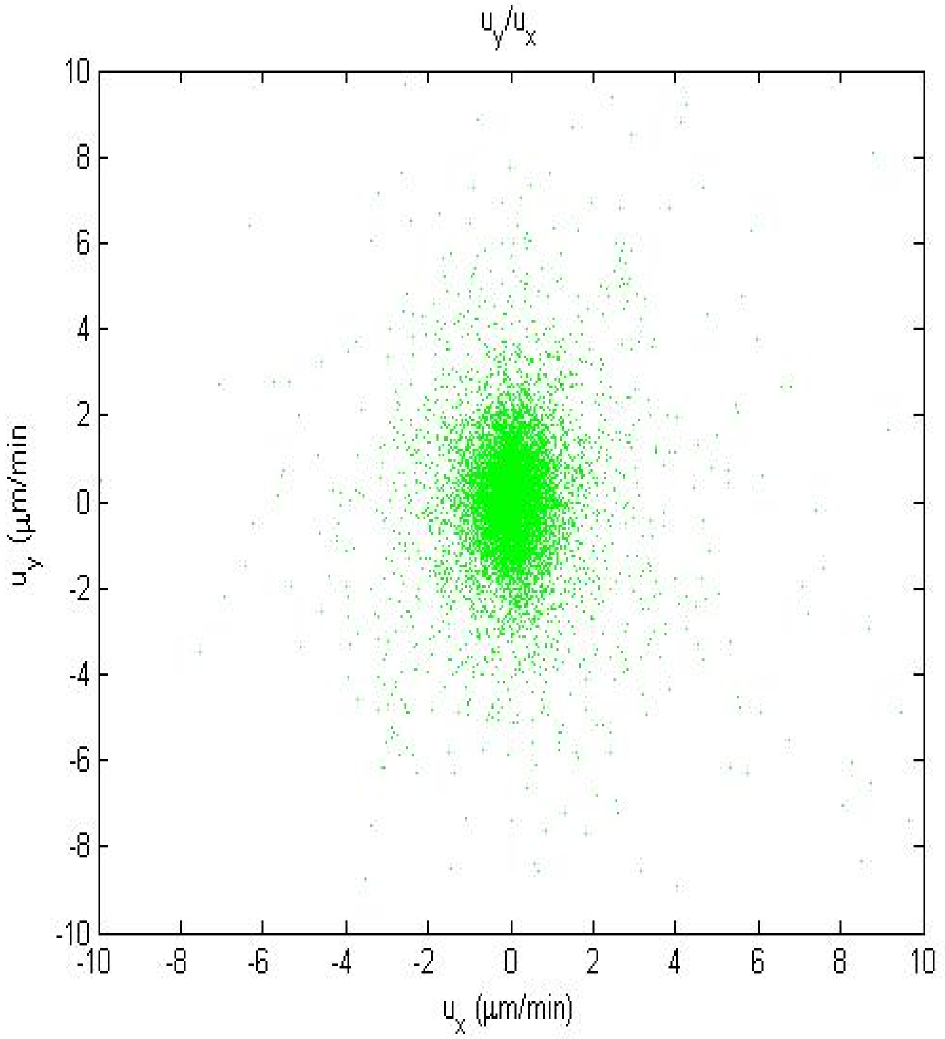


Figure 4. Tracks of 3T3 fibroblasts on 200 nm patterned substrates visualized by light microscopy, tracked by PACT (top). (1 pixel = 0.977 μm; numbers on axes show distance in μm); and plot of x and y components of secant velocities (bottom): $\chi = 0.68$.

c. *Data-driven modelling*

To test the method I have analyzed one of our most anisotropic movies, the 200 nm dataset, as described in section 2. I highlight the fact that, unlike the results presented up until now, the analysis below is preliminary as it is the first of its kind. Hence, although specific results are obtained, more data is necessary in some cases to draw definitive conclusions (such as whether the β_x and β_y parameters or P_x and P_y are indeed equal or just slightly different within their corresponding error bars). Similarly, more data is necessary to quantitatively establish the velocity-dependence of the $\boldsymbol{\sigma}$ tensor.

i. *Accounting for stage drift*

To assess if there is drift or unidirectional motility, I calculated the mean secant velocity (u_x and u_y) and standard error of the mean. In the absence of stage drift, these are expected to be reasonably close to zero within their error bars. The raw tracks and a plot of u_x vs. u_y is shown in Figure 4 above, centered about -0.0008 ± 0.0028 μm/min on the u_x axis and -0.0005 ± 0.0036 μm/min on the u_y axis. The dataset is hence free from drift.

ii. Quantitating anisotropy

To provide a quantitative measure of the level of anisotropy of the motility, I first evaluated the moment of inertia of the secant velocities, **I**, which can be diagonalized to determine the principal axes of inertia.

$$I = \sum_k \begin{bmatrix} u_y^2 & -u_x u_y \\ -u_x u_y & u_x^2 \end{bmatrix}$$

Equation 1

$$RMSD_{axis} = \sqrt{\frac{\frac{1}{N}\sum_{j=1..N}(\vec{u}_j \cdot \vec{v}_{axis})^2 - \left(\frac{1}{N}\sum_{j=1..N}\vec{u}_j \cdot \vec{v}_{axis}\right)^2}{N-1}}$$

Equation 2

where $\vec{v}_{axis}$ is a normalized eigenvector of I.

I then defined the anisotropy quotient:

$$\chi = RMSD_{minor}/RMSD_{major}$$

Equation 3

Although χ can in principle vary between 0 and 1, it will never assume the value 0 for real data due to uncertainty in cell centroid localization (positional noise). Indeed, for purely 1D motility, the theoretically instantaneous velocity v_x and v_y pairs lie on a straight line, but the experimentally available secant velocitie, u_x and u_y, do not. Modeling the positional noise as an uncorrelated, Gaussian distributed term as done previously (II.3.a):

$$\vec{r}_j^{\,exp} = \vec{r}_j + \vec{\xi}_j, \text{ where } \operatorname{cov}(\vec{\xi}_j, \vec{\xi}_k) = \begin{bmatrix} \sigma_{pos}^2 & 0 \\ 0 & \sigma_{pos}^2 \end{bmatrix} \cdot \delta_{j,k}$$

We have

$$\vec{u}_j = \vec{v}_j + \frac{\vec{\xi}_{j+1} - \vec{\xi}_j}{\Delta t}$$

For convenience, I have rotated the (x,y) velocity data (u_x, u_y) such that the major axis now lies along the x-axis: $v_j^y = 0$.

$$u_j^y = \frac{\xi_{j+1}^y - \xi_j^y}{\Delta t}: \quad \operatorname{var}(u_j^y) = \operatorname{var}\left(\frac{\xi_{j+1}^x - \xi_j^x}{\Delta t}\right) = 2\frac{\sigma_{pos}^2}{\Delta t^2}$$

Whereas $\chi^{theor} = 0$ for perfectly linear motility, and noting that $var(v^x) = \langle v_x^2 \rangle - \langle v_x \rangle^2 = \langle v_x^2 \rangle$:

$$\chi^{exp} \overset{def}{=} \frac{RMSD_{minor}}{RMSD_{major}} = \frac{\sqrt{2}\sigma_{pos} / \Delta t}{RMSD_{major}} > 0$$

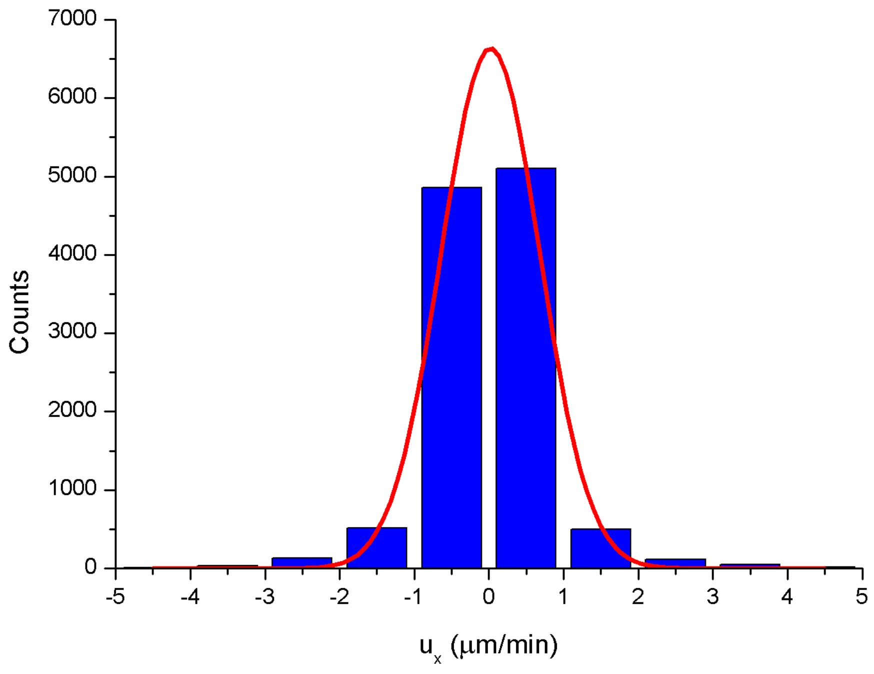
Equation 4

The $RMSD_{minor} = RMSD(u_y)$ was 0.9345 μm/min and $RMSD_{major} = RMSD(u_x)$ 1.3650 μm/min, thus an anisotropy coefficient of 0.68.

iii. Velocity histogram plots

The histogram of secant velocities should be Gaussian in the case of a standard OU process. In the case of the generalized OU process, a diagonally symmetric **β** tensor would ensure a 2D Gaussian shape of the histogram of (u_x,u_y) values because $\vec{\eta}_j$ that defines the noise term of the OU process is white noise. Long tails, if present, are indicative of a velocity dependence of the σ parameter, which can be tested by means of an acceleration-velocity plot (section vi below).

A histogram of the secant velocities is shown in Figure 5, with a Gaussian fit overlaid to them to check for symmetry and skew. The fit parameters were μ, the center of the Gaussian, σ, the standard deviation, and the amplitude. The total number of points was 11328, slightly underestimated by the total area under the fit Gaussian for both the u_x and u_y histograms: 10992 ±140 and 10633 ± 316, respectively. The centers of the Gaussians also do not match perfectly the mean of u_x and u_y quoted in section (i): $\mu_x = 0.019 \pm 0.008$ μm/min and $\mu_y = 0.05 \pm 0.03$ μm/min, but remain reasonably close to zero within error bars. These slight mismatch effects are presumably due to binning. The standard deviations of the velocity histograms were 0.61 ± 0.015 μm/min and 0.92 ± 0.03 μm/min for u_x and u_y, respectively, with a x/y ratio of 0.66, very comparable to the χ of 0.68 determined for this dataset.



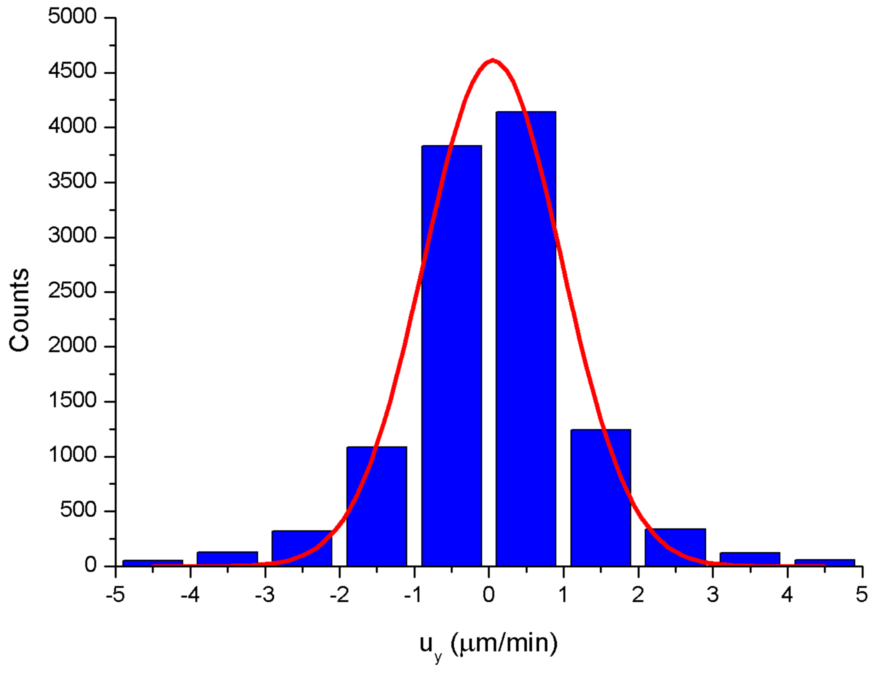


Figure 5. Histogram of secant velocities (u_x and u_y)

As seen from both figures, the histograms do not exhibit perfect Gaussian behavior, especially visible at high speeds, consistent with a slight velocity dependence of the sigma parameters of the OU process which I also will prove using a different method in the following section.

iv. Acceleration-velocity plots

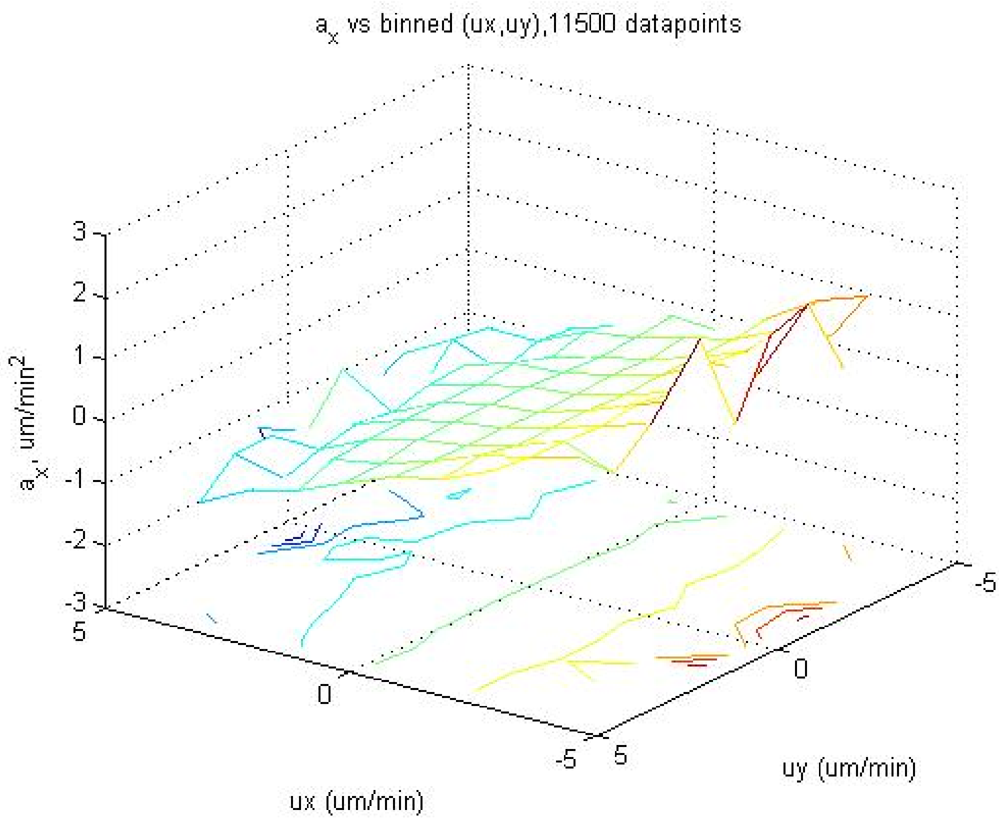
The generalized OU model can be tested by plotting α_x and α_y against (u_x, u_y) to see if a linear relation holds for the expected values of α_x and α_y as functions of (u_x, u_y), as equation 2.a.6 suggests. Such 3D plots should show a scatter plot of triplets e.g. (u_x, u_y, α_x) for the α_x plot centered around a plane passing through the origin, with a slope of $-\beta_x$ along the x axis and a slope of $-\beta_{xy}$ along the y axis. To remove the contribution of the noise term, the data is binned in velocity bins, and the relationship below (following from equation 2, section IV.4.a) is expected to hold true:

$$\langle \alpha_x \rangle_{bin} = -\beta_x \langle u_x \rangle_{bin} - \beta_{xy} \langle \mathrm{u_y} \rangle_{bin} \text{ and } \langle \alpha_y \rangle_{bin} = -\beta_{xy} \langle u_x \rangle_{bin} - \beta_y \langle \mathrm{u_y} \rangle_{bin}$$

Equation 5

Any non-linearity in this relationship is a deviation from the standard OU model (e.g. dependence of the β parameters on secant velocities, III.2.a). This case would require a v-dependent **P** tensor in the first term of the defining equation of the OU process (equation 2, section IV.4.a).

I plotted α_x and α_y respectively as a 3D-function of (u_x, u_y) to test whether the data lie on a plane as expected in the case of the generalized OU process. To remove the noise component I binned the data in (u_x, u_y) bins of 1x1 $\mu m^2/min^2$. The resulting mesh plot is shown below in figures 6 and 7. In the insets I show the original scatter plot viewed along the u_x and u_y axes, respectively, to test whether the slopes of components of α vs. u confirm the model.



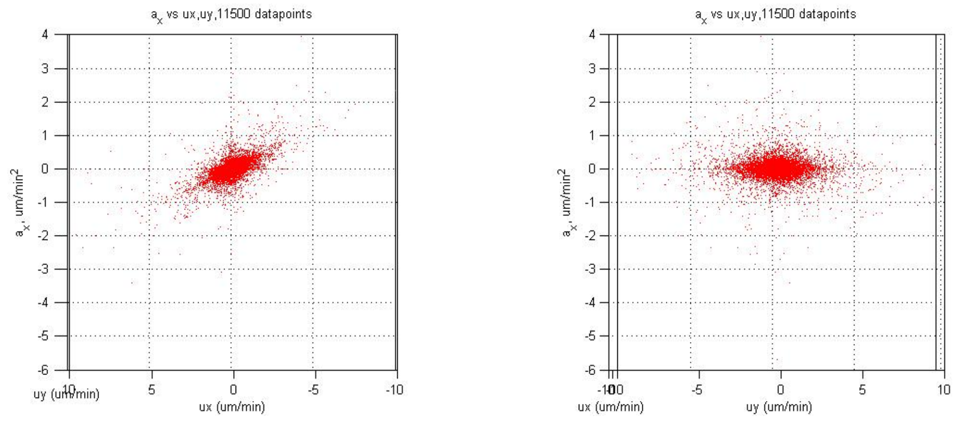
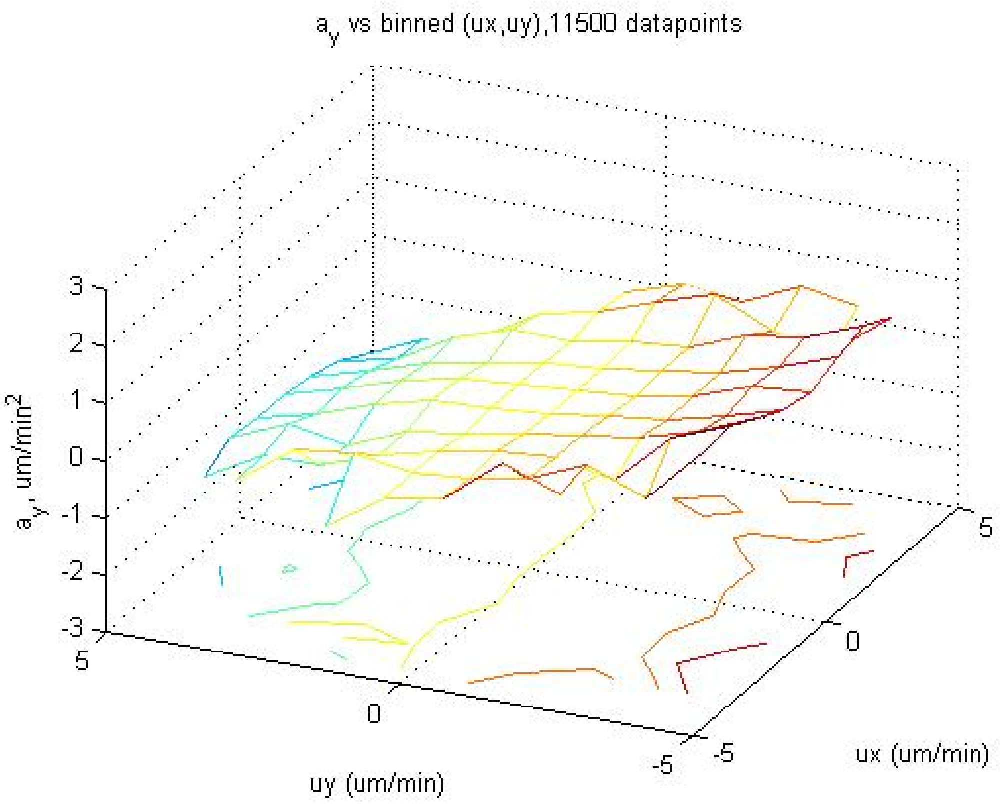


Figure 6. Top: Plot of the average value of a_x, the x component of the secant accelerations, vs. (u_x, u_y), with the latter binned in 1x1 $\mu m^2/min^2$ bins. Bottom: 2D plots showing the original scatter plot viewed along the u_x and u_y axes, respectively.



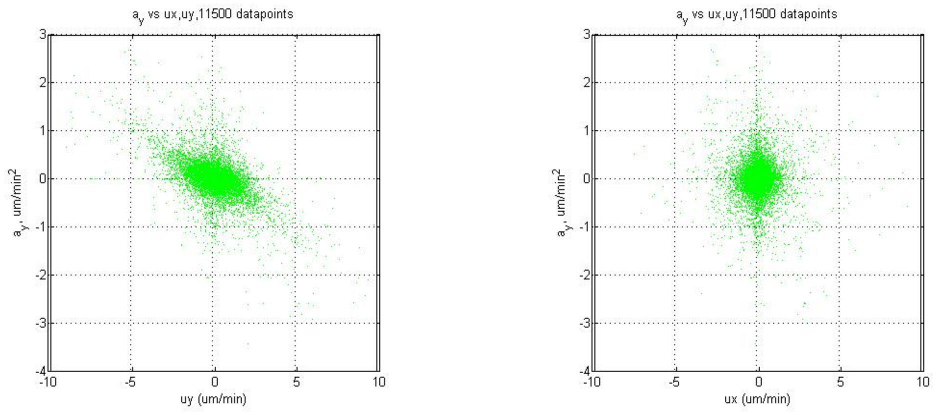


Figure 7. Top: Plot of the average value of a_y, the x component of the secant accelerations, vs. (u_x, u_y), with the latter binned in 1x1 μm^2/min^2 bins. Bottom: 2D plots showing the original scatter plot viewed along the u_x and u_y axes, respectively.

As seen from figures 6 and 7, the plots are planar (also within the error bars which are not shown for visual clarity). This confirms the initial hypothesis that the **β** tensor is u-independent. Having established this, a least-squares fit of a plane (e.g. $\alpha_x = a_0 - \beta_x\langle u_x\rangle - \beta_{xy}\langle u_y\rangle$) to the raw data of ($u_x$, u_y, $\alpha_{x/y}$) triplets shows that this plane passes through the origin ($a_0 = 0$ within error bars). Following this check, the data is fit to a plane forced to pass through the origin (e.g. $\alpha_x = -\beta_x\langle u_x\rangle - \beta_{xy}\langle u_y\rangle$) which results in the following values for the components of the **β** tensor:

$\beta_x = 0.206 \pm 0.002$ min^{-1}
$\beta_{xy} = 0.008 \pm 0.002$ min^{-1}
$\beta_{yx} = 0.011 \pm 0.003$ min^{-1}

$\beta_y = 0.157 \pm 0.002$ min^{-1}

Note that since the variances on the triplets being fit are not necessarily Gaussian distributed, an accurate fit to the raw data requires using maximum likelihood estimation. Alternatively, one can perform a weighted fit to binned data as done below. That has the advantage of ensuring Gaussian distribution of α_{mean} due to bin averaging of non-zero second moment probability distributions of the original data points (central limit theorem), which makes a least squares fit an accurate option (least squares fit is maximum likelihood with Gaussian distributed data). For this reason, I will rely further on the results from the fit to binned data.

$\beta_x = 0.143 \pm 0.006$ min^{-1}
$\beta_{xy} = 0.001 \pm 0.004$ min^{-1}
$\beta_{yx} = 0.003 \pm 0.007$ min^{-1}
$\beta_y = 0.121 \pm 0.006$ min^{-1}

Regardless of the method used, the off-diagonal terms vanish within their respective error bars, indicative of uncorrelated persistence along x and y, respectively. The β_x parameter is consistently slightly larger than β_y, although this could be a statistical fluctuation.

v. Dispersion of accelerations

To assess if there is any influence on velocity in the s-parameters, I plotted the RMSD of the acceleration components in each (u_x, u_y) bin to observe the behavior of the noise term in the generalized OU model (Figures 8 and 9).

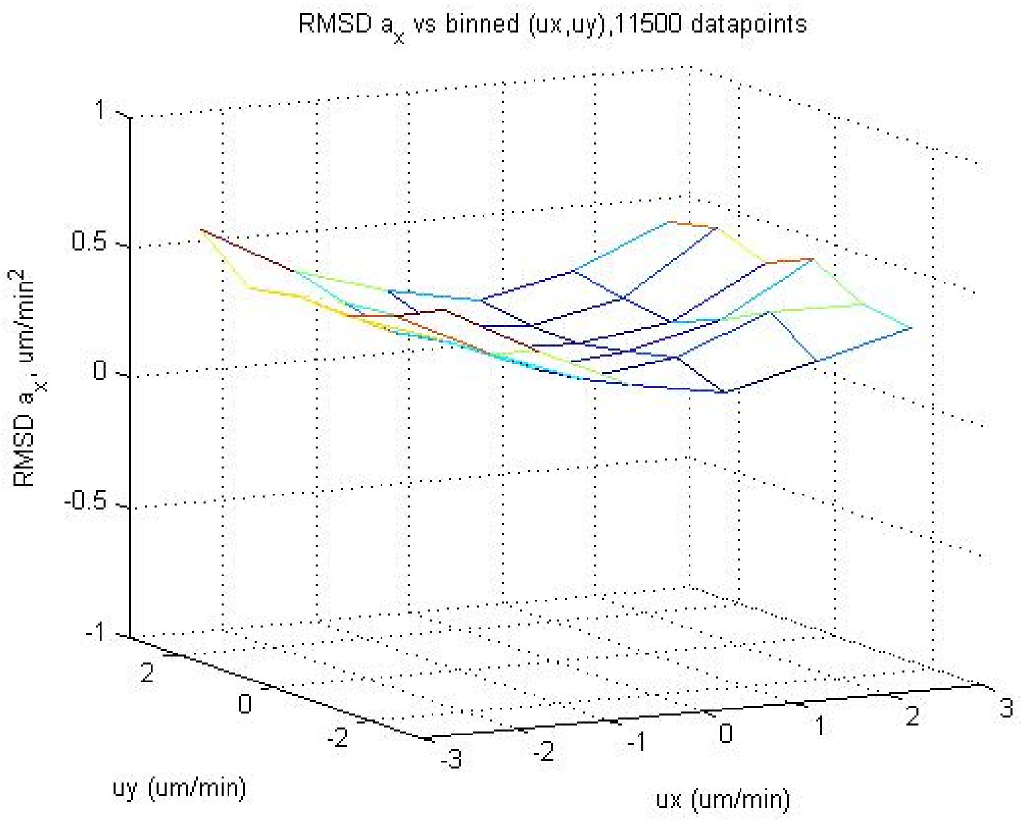


Figure 8. Plot of RMSD of the x component of the secant accelerations vs. 1x1 μm^2/min^2 bins of (u_x, u_y) for 3T3 on 200 nm patterned substrates.

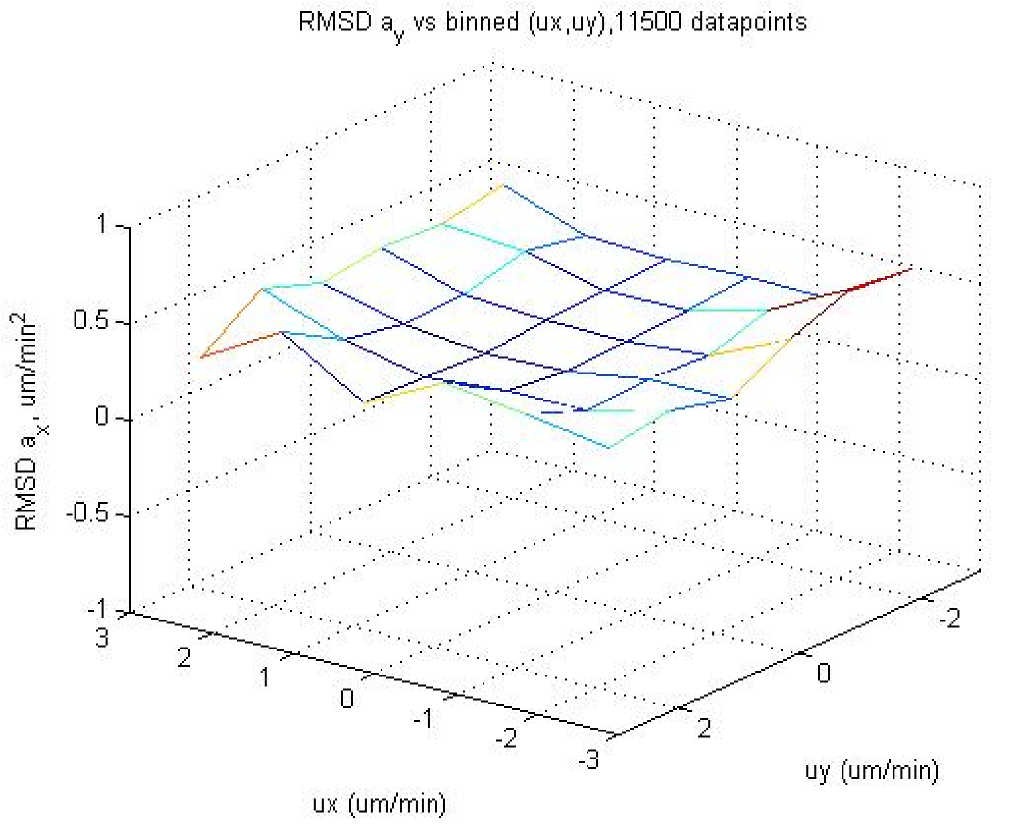


Figure 9. Plot of RMSD of the y component of the secant accelerations vs. 1x1 μm^2/min^2 bins of (u_x, u_y) for 3T3 on 200 nm patterned substrates.

Evaluating a "per-bin" variance of the raw data points, which is equivalent to taking the variance of of the difference equation defining the generalized OU process (equation 2, section IV.4.a):

$$\mathrm{var}(\vec{\alpha}_{\mathrm{bin}}) = -\mathrm{var}\left(\begin{bmatrix}\beta_x & 0\\ 0 & \beta_y\end{bmatrix}\vec{u}_{\mathrm{bin}}\right) + \mathrm{var}\left(\begin{bmatrix}\sigma_x & \sigma_{xy}\\ \sigma_{yx} & \sigma_y\end{bmatrix}\vec{\eta}_{\mathrm{bin},j}\right) = -\begin{bmatrix}\beta_x\ \mathrm{var}(u_{\mathrm{bin},x})\\ \beta_y\ \mathrm{var}(u_{\mathrm{bin},y})\end{bmatrix} + \begin{bmatrix}\sigma_x^2 + \sigma_{xy}^2\\ \sigma_{yx}^2 + \sigma_y^2\end{bmatrix}$$

Equation 1

This result suggests the following quantity should be u-independent, provided the cells follow the generalized OU process:

$$\begin{bmatrix}\sigma_x^2 + \sigma_{xy}^2\\ \sigma_{yx}^2 + \sigma_y^2\end{bmatrix} = \begin{bmatrix}\beta_x\ \mathrm{var}(u_{\mathrm{bin},x})\\ \beta_y\ \mathrm{var}(u_{\mathrm{bin},y})\end{bmatrix} + \mathrm{var}(\vec{\alpha}_{\mathrm{bin}})$$

Equation 2

A plot of the right hand side of this equation is shown in Figures 10 and 11. Clearly, both components of the vector above show strong u dependence, and especially so for $\sigma_x^2 + \sigma_{xy}^2$, which is very dependent on u_x and nearly independent of u_y. The $\sigma_{yx}^2 + \sigma_y^2$ component shows moderate dependence on both u_x and u_y.

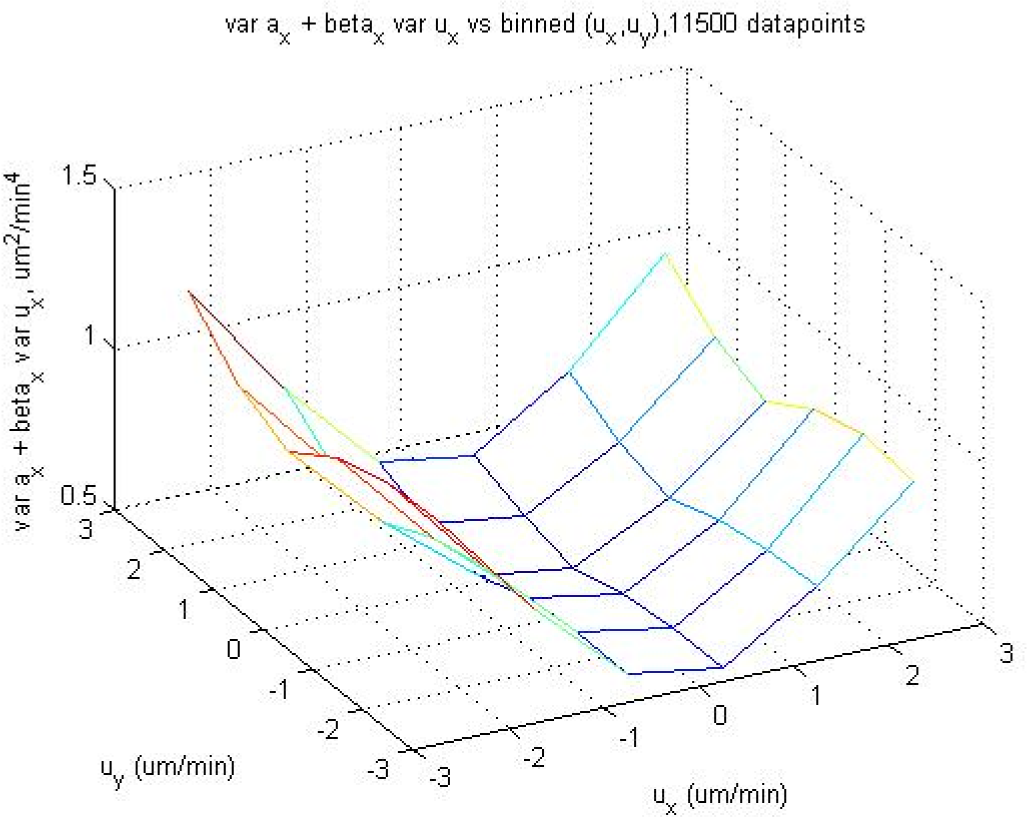


Figure 10. Plot of var(α_x) + $\beta_x$var(u_x) vs. 1x1 μm^2/min^2 bins of (u_x, u_y) for 3T3 on 200 nm patterned substrates. This is expected to be u-independent for a u-independent **σ** tensor characteristic of an unmodified generalized OU process. Clearly, this is not the case here.

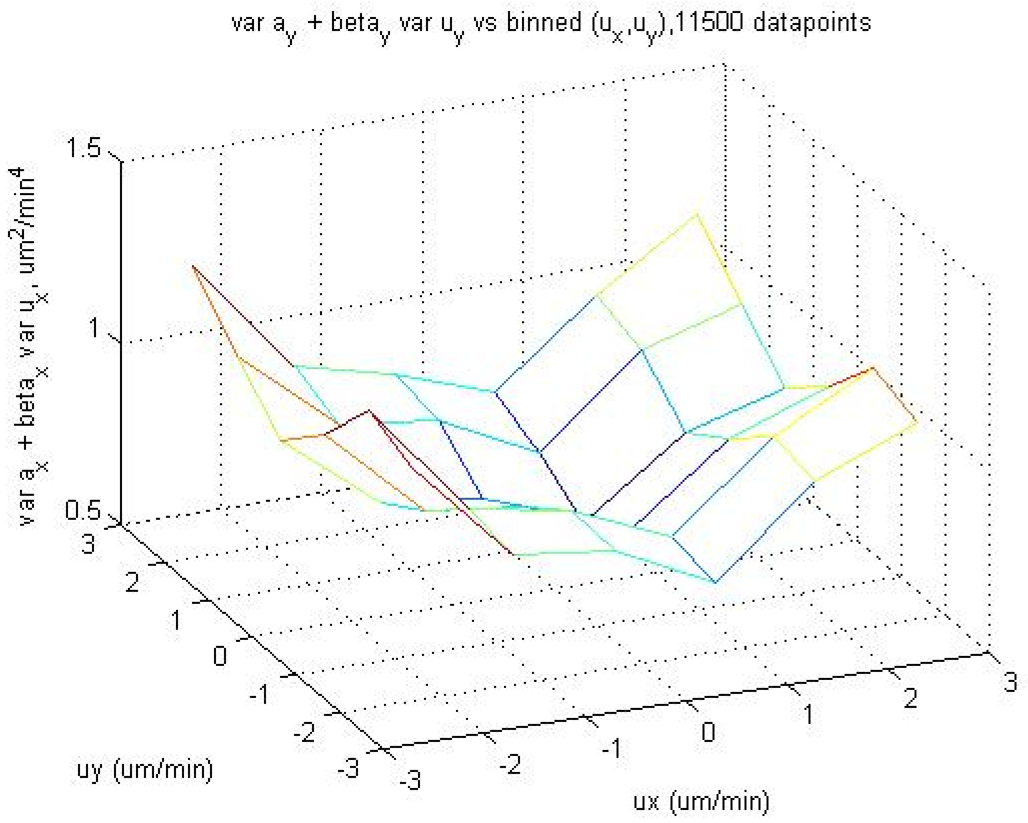


Figure 11. Plot of var(α_y) + $\beta_y$var(u_y) vs. 1x1 μm^2/min^2 bins of (u_x, u_y) for 3T3 on 200 nm patterned substrates. This is expected to be u-independent for a u-independent **σ** tensor characteristic of an unmodified generalized OU process. Clearly, this is not the case here.

The motility model that emerges is thus:

$$\vec{\alpha}_j = -\begin{pmatrix} \beta_x & 0 \\ 0 & \beta_y \end{pmatrix} \cdot \vec{u}_j + \begin{pmatrix} \sigma_x(\vec{u}_j) & \sigma_{xy}(\vec{u}_j) \\ \sigma_{xy}(\vec{u}_j) & \sigma_y(\vec{u}_j) \end{pmatrix} \cdot \vec{\eta}_j$$

Equation 3

While this approach establishes a motility model based on the experimental data, the values of the elements **β** and **σ** tensors are experimental setup-dependent, and it is instead the **P** and **s** tensors that are independent of discretization and positional noise factors. The frame rate Δt = 4 min is relatively small as compared to the P values determined in section vi below (48 and 25 min), although the positional noise determined in section vii below (2.1 and 2.8 μm) is comparable to the square root of the mean squared cell displacement (0.8 μm and 2.7 μm), and the latter factor which contributes to the difference between **s** and **σ** may play a significant role here.

We currently do not have a systematic method for determining the **s** tensor elements yet, but if one should be developed it would involve power spectra analysis (II.5.b). The **P** tensor elements can be determined from the velocity autocovariance function because the off-diagonal elements of the **β** tensor are zero (no cross-correlation).

vi. Velocity autocovariance plots

The velocity autocorrelation function φ is defined in a similar manner to the standard OU process:

$$\varphi_x(t) = \left\langle v_x(t+t') \cdot v_x(t') \right\rangle \text{ and } \varphi_y(t) = \left\langle v_y(t+t') \cdot v_y(t') \right\rangle$$

Equation 6

As before, these are defined in terms of instantaneous velocities. In the general case where the off-diagonal terms of **β** are not zero, the various persistence times that make up the **P** tensor cannot be determined simply by fits to velocity autocorrelation functions as the equations do not separate. Numerical methods may then be employed to determine them instead.

A simpler case arises if the off-diagonal components of the **P** tensor are negligibly small, which is the case with our experimental data. The cross-terms of the velocity autocovariance function $\varphi_{xy}(t) = \left\langle v_x(t+t') \cdot v_y(t') \right\rangle$ and $\varphi_{yx}(t) = \left\langle v_y(t+t') \cdot v_x(t') \right\rangle$ are white noise around zero for a generalized OU process if **P** is diagonal.

I have directly tested whether the off-diagonal elements of the **P** vector are zero (as the work above rests on the assumption that if the **β** tensor is diagonal, that must also be true of **P**. While this has to be the case at low $\Delta t/P$ and σ_{pos}, that statement does not have to be valid in the general case). This was done by plotting the cross-terms of the velocity autocovariance function φ_{xy} and φ_{yx} (figures 12 and 13):

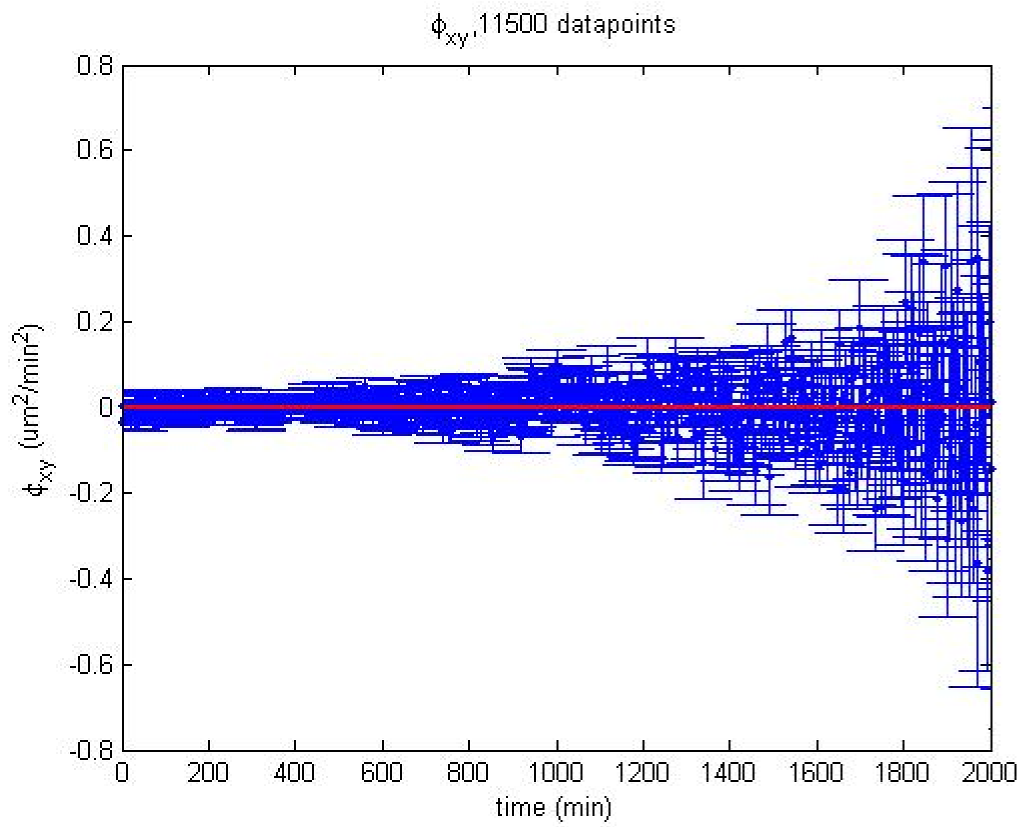


Figure 12. Plot of the xy element of the secant velocities autocorrelation function (blue) and a straight line passing through 0 (red) for 3T3 on 200 nm patterned substrates.

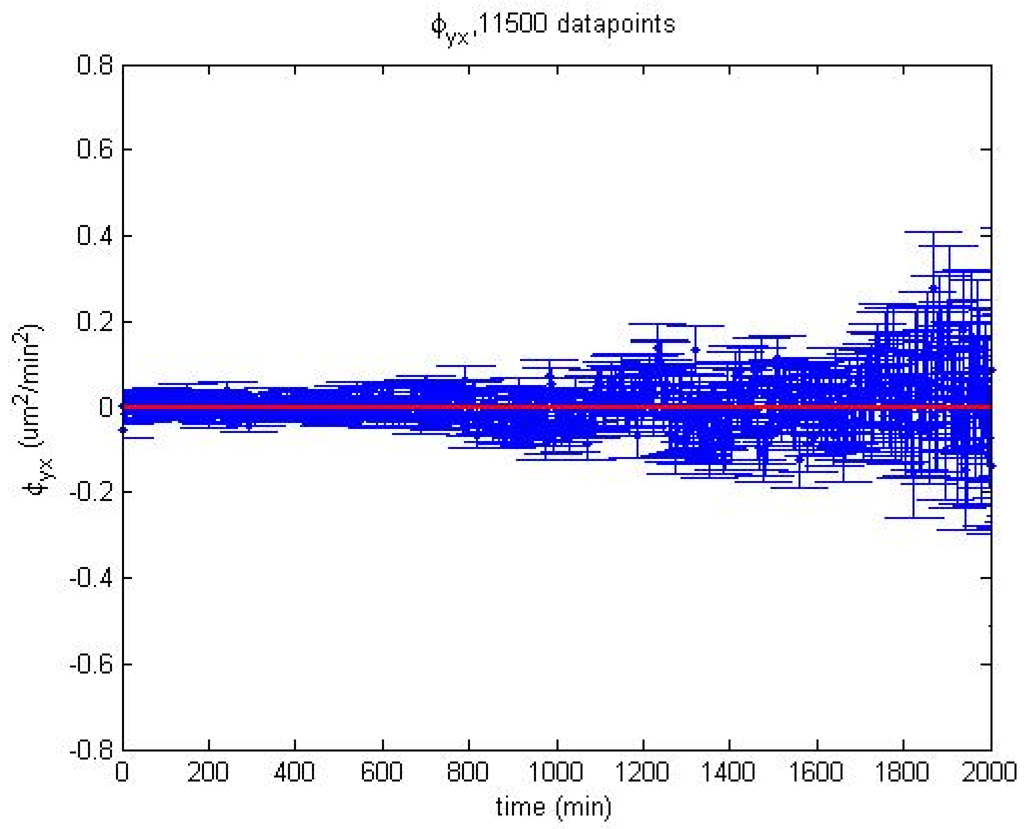


Figure 13. Plot of the yx element of the secant velocities autocorrelation function (blue) and a straight line passing through 0 (red) for 3T3 on 200 nm patterned substrates.

These figures show the cross-terms of the velocity autocovariance function are zero (noise aside), and in order to prove that this requires that the off-diagonal elements of **P** are also zero, I start with equation 2, section IV.4.a:

$$\frac{d}{dt}\begin{pmatrix} v_x \\ v_y \end{pmatrix} = -\begin{bmatrix} P_x & P_{yx} \\ P_{xy} & P_y \end{bmatrix}^{-1} \begin{pmatrix} v_x \\ v_y \end{pmatrix} + \begin{bmatrix} s_x & s_{xy} \\ s_{xy} & s_y \end{bmatrix} \vec{\xi}$$

$$\frac{dv_x}{dt}(t') = -\frac{P_y v_x(t') - P_{yx} v_y(t')}{\det \mathbf{P}} + s_x \eta_x(t') + s_{xy}\eta_y(t')$$

Multiplying by $v_y(t+t')$ and taking expectation values:

$$\frac{d\varphi_{xy}}{dt}(t) = -\frac{P_y \varphi_{yx}(t) - P_{yx}\varphi_y(t)}{\det \mathbf{P}}$$

and similarly

$$\frac{d\varphi_{yx}}{dt}(t) = -\frac{P_x \varphi_{xy}(t) - P_{xy}\varphi_x(t)}{\det \mathbf{P}}$$

Setting $\varphi_{xy} = \varphi_{yx} = 0$,

$$\begin{cases} 0 = \dfrac{P_{yx}\varphi_y(t)}{\det \mathbf{P}} \\ 0 = \dfrac{P_{xy}\varphi_x(t)}{\det \mathbf{P}} \end{cases}$$

with the only possible solution $P_{xy} = P_{yx} = 0$ (since the experimentally measured φ_x and φ_y are non-zero, see below).

When the **P** tensor is diagonal, the velocity autocovariance functions are expected to show simple, monoexponential behavior. Starting with equation 2, IV.4.a:

$$\frac{d\vec{v}}{dt} = -\begin{bmatrix} P_x & 0 \\ 0 & P_y \end{bmatrix}^{-1} \vec{v} + \begin{bmatrix} s_x & s_{xy} \\ s_{xy} & s_y \end{bmatrix} \vec{\xi}$$

and multiplying to the left by $\vec{v}(0)$ and taking the expectation value, this becomes:

$$\varphi_x(t) = -P_x^{-1} v_x(t) \quad \text{and} \quad \varphi_y(t) = -P_y^{-1} v_y(t)$$

Equation 7

As before, the experimentally available quantities are the velocity autocorrelation functions defined in terms of secant velocities. Since in the diagonal **P** case this becomes formally equivalent to two 1D systems, section II.5.c applies fully, and the P_x and P_y parameters, as well as the corresponding $<v_x^2>$ and $<v_y^2>$ can be directly determined from monoexponential fits to $\varphi_x(t_j)$ and $\varphi_y(t_j)$, respectively.

The calculated velocity autocorrelation functions are shown in Figures 14 and 15:

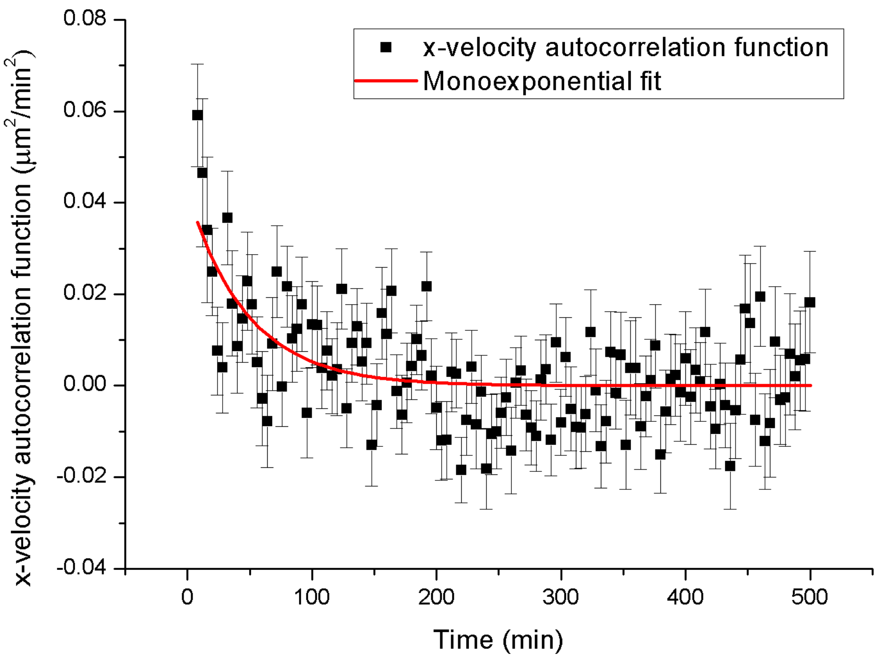


Figure 14. Plot of the autocorrelation function of the x component of the secant velocities (black) and a monoexponential fit to it (red) for 3T3 on 200 nm patterned substrates.

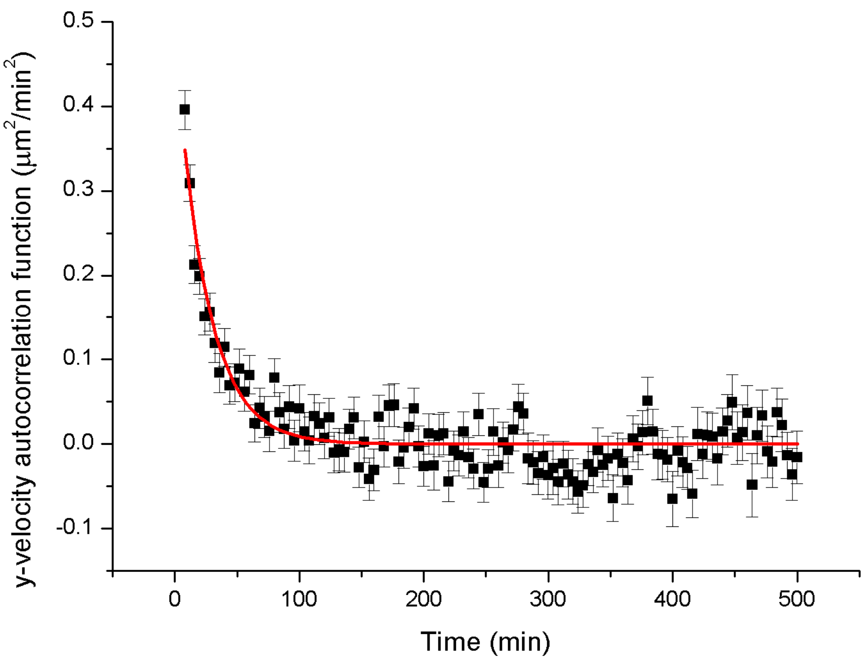


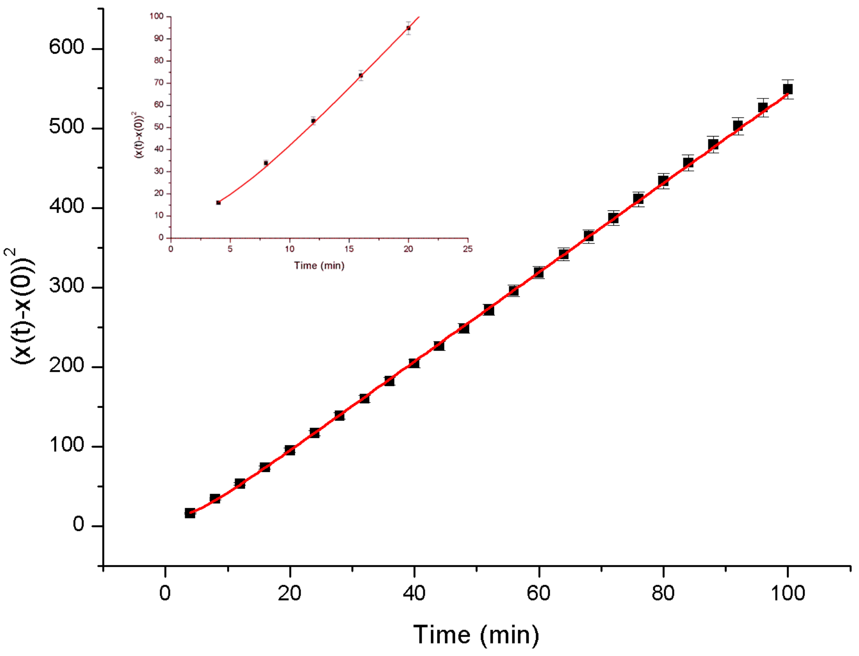
Figure 15. Plot of the autocorrelation function of the y component of the secant velocities (black) and a monoexponential fit to it (red) for 3T3 on 200 nm patterned substrates.

Both velocity autocorrelation functions show clear monoexponential behavior. The persistence times are given by the fits' lifetime: $P_x = 48 \pm 11$ min and $P_y = 25 \pm 2$ min.

The amplitude parameters were $\varphi_{0,x} = 0.042 \pm 0.008$ µm²/min² and $\varphi_{0,y} = 0.47 \pm 0.03$ µm²/min²; as in the case of 1D motility, these remain approximately equal to $<v_x^2>$ and $<v_y^2>$, respectively (II.5.c). This result shows that cells move significantly faster (~3x) along the y axis (parallel to the stripes) vs. along the x-axis (perpendicular to the stripes). Coupled with the differences in persistence times, this indicates that the cells move an average of $P_x \cdot \sqrt{<v_x^2>} = 9.8$ µm and $P_y \cdot \sqrt{<v_y^2>} = 17$ µm before turning. As the cells have to cross a rugged surface as they move along x (150-500 nm alternating wells and square bumps), this presents a different type of resistance to motility than movement along y where the surface presents itself as a smooth substrate to the cell.

vii. Positional noise

The positional noise can be determined by fitting to Fürth's formula as described before (II.5.a). Fürth's formula plots can also show if there are limitations to cell motion in the direction transverse to the groove axis (assumed parallel to the major axis of motility).



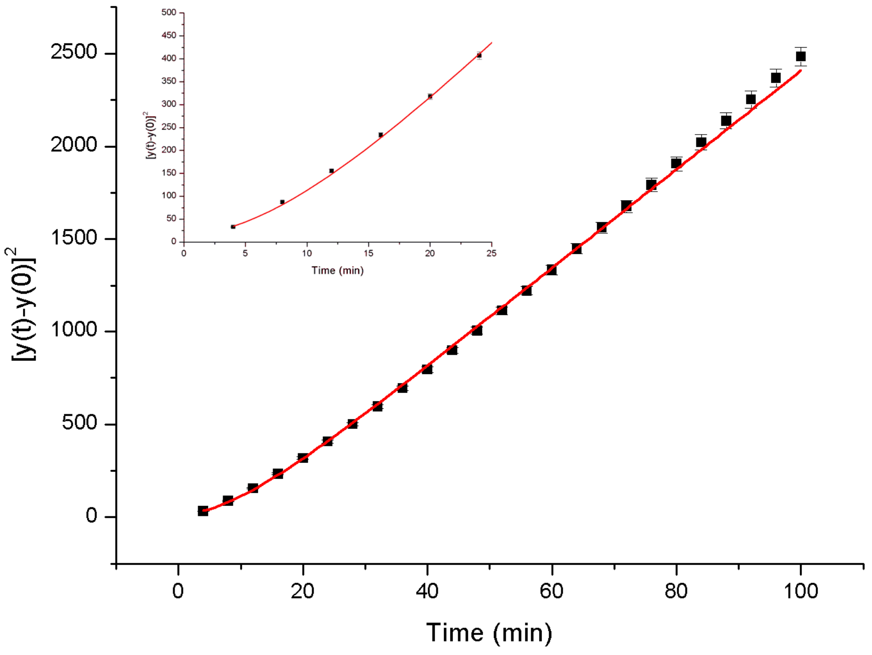


Figure 16. Fit of extended Fürth's formula (red) to the mean squared displacement of the x (left) and y (right) positions of the cell (black). Inset: same plot showing only the first five points.

From the Fürth's formula plots we find the following values for the positional noise: $\sigma_{x,pos} = 2.13 \pm 0.06$ µm and $\sigma_{y,pos} = 2.8 \pm 0.1$ µm, typical if not slightly larger than those previously encountered with PACT-processed data (III.3.b).

3. Conclusions

The 200 nm dataset illustrated strong anisotropic behavior. Based on the analysis above I have found that an modified, generalized OU process can be used to described the data reasonably well, with $P_x = 48 \pm 11$ min and $P_y = 25 \pm 2$ min and $P_{xy}=0$:

$$\frac{d\vec{v}}{dt} = -\begin{bmatrix} P_x & 0 \\ 0 & P_y \end{bmatrix}^{-1} \vec{v} + \begin{bmatrix} s & s_{xy} \\ s_{xy} & s \end{bmatrix} \vec{\xi}$$

Equation 1

where methods for estimating the elements of the **s** tensor are pending investigation.

In the form of a difference equation involving secant quantities:

$$\vec{\alpha}_j = \frac{\Delta\vec{u}_j}{\Delta t} = -\begin{bmatrix} \sigma_x & 0 \\ 0 & \sigma_y \end{bmatrix} \vec{u}_j + \begin{bmatrix} \sigma & \sigma_{xy} \\ \sigma_{xy} & \sigma \end{bmatrix} \vec{\eta}_j$$

Equation 2

where the σ parameters are strongly influenced by setup-dependent discretization effects and have some velocity dependence in the observed dataset. The s parameters are experimental setup-independent, but we do not have a systematic method for determining them yet.

4. References

Ali R, Gooding M, Christlieb M, Brady M. Advanced phase-based segmentation of multiple cells from bright-field microscopy images; 2008 14-17 May 2008. pp. 181-184.

Buettner, H. M., R. N. Pittman, et al. (1994). "A Model of Neurite Extension across Regions of Nonpermissive Substrate: Simulations Based on Experimental Measurement of Growth Cone Motility and Filopodial Dynamics." Developmental Biology **163**(2): 407-422.

Dehmelt, L. and S. Halpain (2004). "Actin and microtubules in neurite initiation: Are MAPs the missing link?" Journal of Neurobiology **58**(1) 18-33.

Dickinson, R. (2000). "A generalized transport model for biased cell migration in an anisotropic environment." J Math Biol **40** 97-135.

Dickinson, R., S. Guido, et al. (1994). "Biased cell migration of fibroblasts exhibiting contact guidance in oriented collagen gels." Annals of Biomedical Engineering **22**(4): 342-356.

Dunn, G. A. and A. F. Brown (1986). "Alignment of fibroblasts on grooved surfaces described by a simple geometric transformation." Journal of Cell Science **83**(1): 313-340.

Heidemann, S., P. Lamoureux et al. (1990). "Growth cone behavior and production of traction force." Journal of Cell Biology **111**(5(1)): 1949-57.

Hu J, Shi J, Zhang F, Lei L, Li X et al. High resolution and hybrid patterning for single cell attachment. Microelectronic Engineering 87(5-8): 726-729.

Ionides, E. L., K. S. Fang, et al. (2004). "Stochastic models for cell motion and taxis." Journal of Mathematical Biology **48**(1): 23-37.

Li, L., E. C. Cox, and H. Flyvbjerg (2011). "Dicty Dynamics: Dictyostelium motility as persistent random motion." Physical Biology **8**, 046006

Matthes, T. and H. Gruler (1988). "Analysis of cell locomotion." European Biophysics Journal **15**(6): 343-357.

Neilson, M. P.; D. M. Veltman et al. (2011). "Chemotaxis: A Feedback-Based Computational Model Robustly Predicts Multiple Aspects of Real Cell Behaviour." PLoS Biology **9**(5) e1000618.

Patlak, C. (1953). "Random walk with persistence and external bias." Bulletin of Mathematical Biology **15**(3): 311-338.

Selmeczi, Mosler, et al. (2005). "Cell motility as persistent random motion: theories from experiments." Biophysics Journal **89**: 912-931.

Richards, G R. (2004). "Quantitative Assays of Chemotaxis and Chemokinesis for Human Neural Cells" ASSAY and Drug Development Technologies. **2**(5) 465-72..

Stokes, C. L., D. A. Lauffenburger, et al. (1991). "Migration of individual microvessel endothelial cells: stochastic model and parameter measurement." Journal of Cell Science **99**(2): 419-430.

Tranquillo, R. T. and D. A. Lauffenburger (1987). "Stochastic model of leukocyte chemosensory movement." Journal of Mathematical Biology **25**(3): 229-262.

Watkins, S. H. a. J. (1995). "A Stochastic Model for the Movement of a White Blood Cell." Advances in Applied Probability **27**(2): 443.

Chapter V: Conclusions

This chapter contains the concluding remarks, which sum up all of the results presented, take a step back to the Objective section of this thesis to show what has been accomplished, and present a number of other research avenues that this work opens.

1. Summary of results against the initial objectives

The objectives of this Ph.D. project have been to develop the theoretical tools needed to characterize cell motility.

The work presented herein developed a theoretical framework for coping in a quantitative manner with challenges frequently encountered in dealing with experimental data, such as positional measurement noise and discrete cell trajectories. They are no longer seen as sources of error, but have been modeled mathematically, which permits treating them as independent parameters that can be extracted from the experimental data. A computer program specifically optimized for tracking cells on the types of surfaces that have been investigated in this project has been written and compared against other similar packages available commercially or on an open-source basis. This led to a thorough study of the potential effect of tracking algorithm choice on motility results. Finally, the question of how to extract motility parameters has been addressed, which resulted in the development of two methods of data analysis and of a computer program that implements them, thereby automating the method of determining motility parameters from a set of cell tracks.

For the purposes of cell motility modeling, various extensions of the standard OU process have been proposed, either to model the motility mouse fibroblasts on flat surfaces similar to that performed in Selmeczi et al., or for cell motility on anisotropic substrates. The former study allowed comparison of fibroblast motility across different species, as Selmeczi et al. modeled the motility of normal human dermal fibroblasts, which is an initial step in characterizing the effect of interspecies differences on cell motility. Finally, several experiments addressing the effect of surface nanotopography on motility found a strong dependence of motility parameters on topography, without any influence on the specific motility model.

2. Outlook

a. Future directions

This work has resulted in specific answers to the initial question of how to best model cell motility and how to interpret the results. This work's contribution to the field raises in turn a few additional questions. Some are natural follow-ups of the research already presented, such as – on the mathematical side – what is the optimal way of determining the cell motility parameters σ and especially the experimental setup-independent cell motility coefficient, s. An accurate and precise way of determining them may shed light on the interplay of cell type and surface topology on the motility models and parameters, thereby clarifying the biological meaning of the persistence time and of the cell motility coefficients. Further, modeling anisotropic motility in the way described in chapter IV is preliminary and, to my knowledge, pioneering, and better data resolution, as well as inclusion of more surface types/anisotropic elements may lead to better characterization of cell motility in anisotropic environments.

On the biological side, the preliminary observation that fibroblasts from different organisms and tissue types follow slightly different motility models, and significantly different motility parameters may provide a way to discriminate between such cells in much the same way that a biological marker does.

Appendices

1. Proofs and formulas

a. Estimation of D by unweighted LSQ fitting:

The unweighted least-squares fitting criterion for determining D requires minimization of the following quantity:

$$\sum_{n=1}^{n_{\max}} \left(\delta_n - 4nD\,\delta t\right)^2 = \min$$

thus its first derivative with respect to D must be zero:

$$\frac{d}{dD}\sum_{n=1}^{n_{\max}} \left(\delta_n - 4nD\,\delta t\right)^2 \Bigg|_{est(D)_{n_{\max}}} = 0$$

$$8\delta t \cdot est(D)_{n_{\max}} \sum_{n=1}^{n_{\max}} \left(n\delta_n - 4n^2\delta t \cdot est(D)_{n_{\max}}\right) = 0$$

$$\sum_{n=1}^{n_{\max}} n\delta_n = 4\delta t \cdot est(D)_{n_{\max}} \sum_{n=1}^{n_{\max}} n^2$$

thus:

$$\mathrm{est}(D)_{n_{\max}} = \frac{3\sum_{n=1}^{n_{\max}} n\,\delta_n}{n_{\max}\left(n_{\max}+1\right)\left(2n_{\max}+1\right)2\delta t}$$

As a measure of this estimator's precision, taking its expectation value shows that it is unbiased

$$\left\langle \operatorname{est}(D)_{n_{\max}} \right\rangle = \frac{3\sum_{n=1}^{n_{\max}} n\left\langle \delta_n \right\rangle}{n_{\max}\left(n_{\max}+1\right)\left(2n_{\max}+1\right)2\delta t}$$

Reverting to the definition of δ_n (equation 2.a.1):

$$\left\langle \operatorname{est}(D)_{n_{\max}} \right\rangle = \frac{3\sum_{n=1}^{n_{\max}} \frac{n}{N-n-1} \sum_{j=1}^{N-n-1} \left\langle \left(\vec{r}(t_j+n\delta t)-\vec{r}(t_j)\right)^2 \right\rangle}{n_{\max}\left(n_{\max}+1\right)\left(2n_{\max}+1\right)2\delta t}$$

And given equation 1.b.3:

$$\left\langle \operatorname{est}(D)_{n_{\max}} \right\rangle = \frac{3\sum_{n=1}^{n_{\max}} \frac{n}{N-n-1} \sum_{j=1}^{N-n-1} 4nD\,\delta t}{n_{\max}\left(n_{\max}+1\right)\left(2n_{\max}+1\right)2\delta t}$$

which simplifies to

$$\left\langle \operatorname{est}(D)_{n_{\max}} \right\rangle = D$$

$$\operatorname{var}\operatorname{est}(D)_{n_{\max}} = \left\langle \left(\operatorname{est}(D)_{n_{\max}}\right)^2 \right\rangle - \left\langle \operatorname{est}(D)_{n_{\max}} \right\rangle^2$$

$$\operatorname{var}\operatorname{est}(D)_{n_{\max}} = \left(\frac{3}{n_{\max}\left(n_{\max}+1\right)\left(2n_{\max}+1\right)2\delta t}\right)^2 \left[\left\langle \left(\sum_{n=1}^{n_{\max}} n\,\delta_n\right)^2 \right\rangle - \left\langle \left(\sum_{n=1}^{n_{\max}} n\,\delta_n\right) \right\rangle^2\right]$$

which can be rewritten as:

$$\left\langle \left(\sum_{n=1}^{n_{\max}} n\,\delta_n\right)^2 \right\rangle - \left\langle \left(\sum_{n=1}^{n_{\max}} n\,\delta_n\right) \right\rangle^2 = \left\langle \left(\sum_{n=1}^{n_{\max}} n\,\delta_n\right)\left(\sum_{m=1}^{n_{\max}} m\,\delta_m\right) \right\rangle - \left\langle \sum_{n=1}^{n_{\max}} n\,\delta_n \right\rangle \left\langle \sum_{m=1}^{n_{\max}} m\,\delta_m \right\rangle$$

$$\left\langle \left(\sum_{n=1}^{n_{\max}} n\,\delta_n\right)^2 \right\rangle - \left\langle \left(\sum_{n=1}^{n_{\max}} n\,\delta_n\right) \right\rangle^2 = \left\langle \sum_{n=1}^{n_{\max}}\sum_{m=1}^{n_{\max}} nm\delta_n\delta_m \right\rangle - \sum_{n=1}^{n_{\max}} n\left\langle \delta_n \right\rangle \sum_{m=1}^{n_{\max}} m\left\langle \delta_m \right\rangle$$

$$\left\langle \left(\sum_{n=1}^{n_{\max}} n\,\delta_n\right)^2 \right\rangle - \left\langle \left(\sum_{n=1}^{n_{\max}} n\,\delta_n\right) \right\rangle^2 = \sum_{n=1}^{n_{\max}}\sum_{m=1}^{n_{\max}} nm\left\langle \delta_n\delta_m \right\rangle - \sum_{n=1}^{n_{\max}}\sum_{m=1}^{n_{\max}} nm\left\langle \delta_n \right\rangle\left\langle \delta_m \right\rangle$$

and finally:

$$\left\langle \left(\sum_{n=1}^{n_{\max}} n\,\delta_n\right)^2 \right\rangle - \left\langle \left(\sum_{n=1}^{n_{\max}} n\,\delta_n\right) \right\rangle^2 = \sum_{n=1}^{n_{\max}}\sum_{m=1}^{n_{\max}} nm\operatorname{cov}\delta_{m,n}$$

thus

$$\operatorname{var}\operatorname{est}(D)_{n_{\max}} = \left(\frac{3}{n_{\max}(n_{\max}+1)(2n_{\max}+1)2\delta t}\right)^2 \cdot \sum_{m,n=1}^{n_{\max}} mn\cdot\operatorname{cov}\delta_{n,m}$$

where

$$\operatorname{cov}\delta_{n,m} = \langle\delta_n\delta_m\rangle - \langle\delta_n\rangle\langle\delta_m\rangle$$

As seen before (equation 1.b.3):

$$\langle\delta_n\rangle = 4nD\,\delta t$$

$$\delta_n = \frac{1}{N-n-1}\sum_{j=1}^{N-n-1}\left(\vec r(t_j+n\delta t)-\vec r(t_j)\right)^2$$

and

$$\langle\delta_n\delta_m\rangle = \left\langle \sum_{i=1}^{n_{\max}-n-1}\left(\vec r(t_i+n\delta t)-\vec r(t_i)\right)^2 \sum_{j=1}^{n_{\max}-m-1}\left(\vec r(t_j+m\delta t)-\vec r(t_j)\right)^2\right\rangle$$

$$\langle\delta_n\delta_m\rangle = \sum_{i=1}^{n_{\max}-n-1}\sum_{j=1}^{n_{\max}-m-1}\left\langle\left(\vec r(t_j+m\delta t)-\vec r(t_j)\right)^2\left(\vec r(t_i+n\delta t)-\vec r(t_i)\right)^2\right\rangle$$

Collecting the terms appropriately,

$$\operatorname{cov}\delta_{n,m} = \frac{16D^2\delta t^2}{N-m}\cdot\begin{cases} \frac{1}{6}\cdot(N-n)^3-\frac{2}{3}m(N-n)^2+\left(m^2-\frac{1}{6}\right)\cdot(N-n) \\ +m\left(nm+\frac{2}{3}-m^2\right) \quad \text{for}\, N\le m+n \\ m\left(-\frac{m^2}{3}+\frac{1}{3}+nm\right)-\frac{m^2}{6}\left(m^2-1\right)\cdot\frac{1}{(N-n)} \qquad \text{otherwise}\end{cases}$$

b. Estimation of D by weighted LSQ fitting:

The weighted least-squares fitting criterion differs from the unweighted fitting by the addition of weights inversely proportional to the variance of δ_n:

$$\sum_{n=1}^{n_{\max}}\frac{1}{\operatorname{var}\delta_n}\left(\delta_n-4nD\,\delta t\right)^2 = \min$$

As before,

$$\frac{d}{dD}\sum_{n=1}^{n_{\max}}\frac{1}{\operatorname{var}\delta_n}\left(\delta_n - 4nD\,\delta t\right)^2\Bigg|_{est(D)_{n_{\max}}} = 0$$

thus

$$\underset{n_{\max}}{\operatorname{est}_{\mathrm{wLSQ}}}(D) = \frac{\sum_{n=1}^{n_{\max}}\frac{n\,\delta_n}{\operatorname{var}\delta_n}}{4\delta t\sum_{n=1}^{n_{\max}}\frac{n^2}{\operatorname{var}\delta_n}}$$

This is also an unbiased estimator:

$$\left\langle \underset{n_{\max}}{\operatorname{est}_{\mathrm{wLSQ}}}(D)\right\rangle = \frac{\sum_{n=1}^{n_{\max}}\frac{n\langle\delta_n\rangle}{\operatorname{var}\delta_n}}{4\delta t\sum_{n=1}^{n_{\max}}\frac{n^2}{\operatorname{var}\delta_n}} = \frac{\sum_{n=1}^{n_{\max}}\frac{n\left\langle\left(\vec{r}(t_j+n\delta t)-\vec{r}(t_j)\right)^2\right\rangle}{\operatorname{var}\delta_n}}{4\delta t\sum_{n=1}^{n_{\max}}\frac{n^2}{\operatorname{var}\delta_n}} = \frac{\sum_{n=1}^{n_{\max}}\frac{4n^2D\,\delta t}{\operatorname{var}\delta_n}}{4\delta t\sum_{n=1}^{n_{\max}}\frac{n^2}{\operatorname{var}\delta_n}}$$

which simplifies to

$$\left\langle \operatorname{est}(D)_{n_{\max}}\right\rangle = D$$

Similarly,

$$\operatorname{var}\underset{n_{\max}}{\operatorname{est}_{\mathrm{wLSQ}}}(D) = \left\langle\left(\underset{n_{\max}}{\operatorname{est}_{\mathrm{wLSQ}}}(D)\right)^2\right\rangle - \left\langle\underset{n_{\max}}{\operatorname{est}_{\mathrm{wLSQ}}}(D)\right\rangle^2$$

$$\operatorname{var}\underset{n_{\max}}{\operatorname{est}_{\mathrm{wLSQ}}}(D) = \frac{1}{\left(4\delta t\sum_{n=1}^{n_{\max}}\frac{n^2}{\operatorname{var}\delta_n}\right)^2}\left[\left\langle\left(\sum_{n=1}^{n_{\max}}\frac{n\,\delta_n}{\operatorname{var}\delta_n}\right)^2\right\rangle - \left\langle\left(\sum_{n=1}^{n_{\max}}\frac{n\,\delta_n}{\operatorname{var}\delta_n}\right)\right\rangle^2\right]$$

$$\operatorname{var}\underset{n_{\max}}{\operatorname{est}_{\mathrm{wLSQ}}}(D) = \frac{1}{\left(4\delta t\sum_{n=1}^{n_{\max}}\frac{n^2}{\operatorname{var}\delta_n}\right)^2}\cdot\sum_{m,n=1}^{n_{\max}}\frac{mn}{\operatorname{var}\delta_n\cdot\operatorname{var}\delta_m}\cdot\operatorname{cov}\delta_{n,m}$$

c. Estimation of D by generalized LSQ fitting:

The quantity of interest is var $\delta_{est} = \langle\, (\delta_{est} - \langle\, \delta_{est}\rangle)^2\, \rangle = \langle\, \delta_{est}{}^2\, \rangle - \langle\, \delta_{est}\rangle^2$:

$$\operatorname{var}\left(\delta_{est}\right) = \left(\mathbf{N}^T\cdot\mathbf{\Sigma}^{-1}\cdot\mathbf{N}\right)^{-1}\cdot\mathbf{N}^T\cdot\mathbf{\Sigma}^{-1}\left\langle\left(\Delta-\langle\Delta\rangle\right)\left(\Delta-\langle\Delta\rangle\right)^T\right\rangle\mathbf{\Sigma}^{-1^T}\cdot\mathbf{N}\cdot\left(\mathbf{N}^T\cdot\mathbf{\Sigma}^{-1}\cdot\mathbf{N}\right)^{-1}$$

But since $\mathbf{\Sigma} = \langle (\mathbf{\Delta} - \langle \mathbf{\Delta} \rangle)(\mathbf{\Delta} - \langle \mathbf{\Delta} \rangle)^{\mathrm{T}} \rangle$:

$$\mathrm{var}(\delta_{est}) = \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N}\right)^{-1} \cdot \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1^T} \cdot \mathbf{N}\right) \cdot \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N}\right)^{-1}$$

Note that the middle factor is a scalar, thus equal to its transpose:

$$\mathrm{var}(\delta_{est}) = \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N}\right)^{-1} \cdot \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N}\right) \cdot \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N}\right)^{-1} = \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N}\right)^{-1}$$

Inspecting numerically the covariance matrix $\mathbf{\Sigma}$ and the $\mathbf{A} = \mathbf{\Sigma}^{-1}\mathbf{N}$ vector for a variety of N and n_{max} values, it can be seen that the first element of $\mathbf{A}$ is equal to $1/\mathrm{var}(\delta_{est})$ while all the others are zero. To prove this, consider:

$$\mathbf{N} = \mathbf{\Sigma} \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N} = \mathbf{\Sigma} \cdot \mathbf{A} \Rightarrow \sum_{n=1}^{n_{\max}} \mathrm{cov}\,\delta_{m,n}\, a_n = n_m = m \quad \forall m = 1..n_{\max}$$

and since from the Equation 1.a.5, $\mathrm{cov}\ \delta_{m,1} = 16\ mD^2\delta t^2/(N\text{-}1)$,

$$\frac{16mD^2\delta t^2}{N-1} a_1 + \sum_{n=2}^{n_{\max}} \mathrm{cov}\,\delta_{m,n}\, a_n = m \quad \forall m = 1..n_{\max} \Rightarrow \sum_{n=2}^{n_{\max}} \mathrm{cov}\,\delta_{m,n}\, a_n = m\left(1 - \frac{16D^2\delta t^2}{N-1} a_1\right)$$

Since $\mathbf{\Sigma}$ is a covariance matrix, its determinant is non-zero, thus all columns are linearly independent. Specifically, no linear combination of columns (other than the first one which as seen above is proportional to $\mathbf{N}$) can yield $\mathbf{N}$. However, this is what the identity above implies, and these two statements can only be reconciled if $a_1 = (\mathrm{N}\text{-}1)/16D^2\delta t^2$ and $a_\mathrm{n} = 0$ for $n>1$. Consequently,

$$\mathrm{var}(\delta_{est}) = \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N}\right)^{-1} = \left(\mathbf{N}^T \cdot \mathbf{A}\right)^{-1} = a_1^{-1} = \frac{16D^2\delta t^2}{N-1}$$

or, reverting to the definition of δ_{est}:

$$\underset{n_{\max}}{\mathrm{est}_{\mathrm{gLSQ}}}(D) = \frac{4D^2\delta t}{N-1} \cdot \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{\Delta}\right)$$

Furthermore, since $\mathbf{\Sigma}^{-1}\mathbf{N}$ is zero except for the first element, the same holds true of its transpose $\mathbf{N}^{\mathrm{T}}\ \mathbf{\Sigma}^{-1}$, and since $\delta_{est} \sim \mathbf{N}^{\mathrm{T}}\ \mathbf{\Sigma}^{-1}\ \mathbf{\Delta}$, it follows that effectively only the first element of $\mathbf{\Delta}$, namely δ_1, contributes to the estimate of D:

$$\underset{n_{\max}}{\mathrm{est}_{\mathrm{gLSQ}}}(D) = \frac{4D^2\delta t}{N-1} \cdot \frac{N-1}{16D^2\delta t^2}\delta_1 = \frac{\delta_1}{4\delta t}$$

d. Estimation of D in the presence of positional measurement noise:

The proofs in this section make extensive use of the covariance matrix cov($\delta_{n,m}$) for data affected by positional noise. It is derived below:

Since (equation 3.a.2):

$$\delta_n \vec{r}_i = \delta_n \vec{r}_i^{\,th} + \vec{\xi}_{i+n} - \vec{\xi}_i$$

and

$$\mathrm{cov}(\delta_{n,m}) = \langle \delta_n \delta_m \rangle - \langle \delta_n \rangle \langle \delta_m \rangle$$

it follows that:

$$\mathrm{cov}(\delta_{n,m}) = \left\langle \overline{\delta_n \vec{r}_i^{\,th} + \vec{\xi}_{i+n} - \vec{\xi}_i} \cdot \overline{\delta_m \vec{r}_j^{\,th} + \vec{\xi}_{j+m} - \vec{\xi}_i} \right\rangle - \left\langle \overline{\delta_n \vec{r}_i^{\,th} + \vec{\xi}_{i+n} - \vec{\xi}_i} \right\rangle \left\langle \overline{\delta_m \vec{r}_j^{\,th} + \vec{\xi}_{j+m} - \vec{\xi}_j} \right\rangle$$

Since

$$\mathrm{cov}(\delta_{n,m}^{th}) = \left\langle \overline{\delta_n \vec{r}_i^{\,th}}_{\,i} \cdot \overline{\delta_m \vec{r}_j^{\,th}} \right\rangle - \left\langle \overline{\delta_m \vec{r}_j^{\,th} + \vec{\xi}_{j+m} - \vec{\xi}_j} \right\rangle$$

$$\langle \delta_n \delta_m \rangle = \sum_{\mathrm{i}=1}^{n_{\max}-n-1} \sum_{j=1}^{n_{\max}-m-1} \left\langle \left(\vec{r}(t_j + m\delta t) - \vec{r}(t_j)\right)^2 \left(\vec{r}(t_i + n\delta t) - \vec{r}(t_i)\right)^2 \right\rangle$$

$$\mathrm{cov}(\delta_{n,m}) = \begin{cases} \mathrm{cov}(\delta_{n,m}^{th}) + \dfrac{4\sigma_{\exp}^4}{N-m}\left(4 + 2\dfrac{\delta_{m,n} - m}{N-n}\right) + \dfrac{32mD\delta t\sigma_{\exp}^2}{N-m} & \text{if } N > m+n \\ \mathrm{cov}(\delta_{n,m}^{th}) + \dfrac{8\sigma_{\exp}^4 + 32mD\delta t\sigma_{\exp}^2}{N-m} + \dfrac{8\sigma_{\exp}^4 \delta_{m,n}}{(N-m)(N-n)} & \text{otherwise} \end{cases}$$

To determine the unweighted LSQ estimator for *D* and σ_{exp}^2, the quantity below needs to be minimized:

$$\sum_{n=1}^{n_{\max}} \left(\overline{\delta_n r^2} - 4nD_{est}\,\delta t - 2\sigma_{\exp}^2\right)^2 = \min$$

thus:

$$\begin{cases} \sum_{n=1}^{n_{\max}} \left[\overline{\delta_n r^2} - 4nD_{est}\,\delta t - 2\sigma_{\exp}^2\right] \frac{\partial}{\partial(\sigma_{\exp}^2)} \left[\overline{\delta_n r^2} - 4nD\delta t - 2\sigma_{\exp}^2\right] = 0 \\ \sum_{n=1}^{n_{\max}} \left[\overline{\delta_n r^2} - 4nD_{est}\,\delta t - 2\sigma_{\exp}^2\right] \frac{\partial}{\partial D} \left[\overline{\delta_n r^2} - 4nD\delta t - 2\sigma_{\exp}^2\right] = 0 \end{cases}$$

$$\begin{cases} \sum_{n=1}^{n_{\max}} \left[\overline{\delta_n r^2} - 4nD_{est}\,\delta t - 2\sigma_{\exp}^2\right] = 0 \\ \sum_{n=1}^{n_{\max}} n \cdot \left[\overline{\delta_n r^2} - 4nD_{est}\,\delta t - 2\sigma_{\exp}^2\right] = 0 \end{cases}$$

$$\begin{cases} \sum_{n=1}^{n_{\max}} \overline{\delta_n r^2} = 2(n_{\max}+1)n_{\max} D_{est}\,\delta t + 2n_{\max}\sigma_{\exp}^2 \\ \sum_{n=1}^{n_{\max}} n\overline{\delta_n r^2} = \frac{2}{3} n_{\max}(n_{\max}+1)(2n_{\max}+1) D_{est}\,\delta t + (n_{\max}+1)n_{\max}\sigma_{\exp}^2 \end{cases}$$

and finally:

$$\begin{cases} \underset{n_{\max}}{est_{uwLSQ}}(D) = 3\frac{\sum_{n=1}^{n_{\max}} n\overline{\delta_n r^2} - \frac{n_{\max}+1}{2}\sum_{n=1}^{n_{\max}} \overline{\delta_n r^2}}{(n_{\max}-1)n_{\max}(n_{\max}+1)\delta t} \\ \underset{n_{\max}}{est_{uwLSQ}}(\sigma_{\exp}^2) = \frac{(2n_{\max}+1)\sum_{n=1}^{n_{\max}} \overline{\delta_n r^2} - 3\sum_{n=1}^{n_{\max}} n\overline{\delta_n r^2}}{(n_{\max}-1)n_{\max}} \end{cases}$$

The weighted LSQ estimators are derived similarly by minimizing the following quantity:

$$\sum_{n=1}^{n_{\max}} \frac{1}{\operatorname{var}\delta_n} \left(\delta_n - 4nD_{est}\,\delta t - 2\sigma_{\exp}^2\right)^2 = \min$$

thus:

$$\begin{cases} \sum_{n=1}^{n_{\max}} \frac{1}{\operatorname{var}\delta_n} \left[\overline{\delta_n r^2} - 4nD_{est}\,\delta t - 2\sigma_{\exp}^2\right] \frac{\partial}{\partial(\sigma_{\exp}^2)} \left[\overline{\delta_n r^2} - 4nD\delta t - 2\sigma_{\exp}^2\right] = 0 \\ \sum_{n=1}^{n_{\max}} \frac{1}{\operatorname{var}\delta_n} \left[\overline{\delta_n r^2} - 4nD_{est}\,\delta t - 2\sigma_{\exp}^2\right] \frac{\partial}{\partial D} \left[\overline{\delta_n r^2} - 4nD\delta t - 2\sigma_{\exp}^2\right] = 0 \end{cases}$$

$$\begin{cases} \sum_{n=1}^{n_{\max}} \frac{1}{\operatorname{var}\delta_n} \left[\overline{\delta_n r^2} - 4nD_{est}\,\delta t - 2\sigma_{\exp}^2\right] = 0 \\ \sum_{n=1}^{n_{\max}} \frac{n}{\operatorname{var}\delta_n} \cdot \left[\overline{\delta_n r^2} - 4nD_{est}\,\delta t - 2\sigma_{\exp}^2\right] = 0 \end{cases}$$

$$\begin{cases} \sum_{n=1}^{n_{\max}} \dfrac{\overline{\delta_n r^2}}{\operatorname{var}\delta_n} = 4D_{est}\delta t \sum_{n=1}^{n_{\max}} \dfrac{n}{\operatorname{var}\delta_n} + 2\sigma_{\exp}^2 \sum_{n=1}^{n_{\max}} \dfrac{1}{\operatorname{var}\delta_n} \\ \sum_{n=1}^{n_{\max}} \dfrac{n\overline{\delta_n r^2}}{\operatorname{var}\delta_n} = 4D_{est}\delta t \sum_{n=1}^{n_{\max}} \dfrac{n^2}{\operatorname{var}\delta_n} + 2\sigma_{\exp}^2 \sum_{n=1}^{n_{\max}} \dfrac{n}{\operatorname{var}\delta_n} \end{cases}$$

and finally:

$$\begin{cases} \underset{n_{\max}}{est_{uwLSQ}}(D) = \dfrac{\sum_{m,n=1}^{n_{\max}} n \dfrac{\overline{\delta_n r^2} - \overline{\delta_m r^2}}{\operatorname{var}\left(\overline{\delta_n r^2}\right)\operatorname{var}\left(\overline{\delta_m r^2}\right)}}{4\delta t \sum_{m,n=1}^{n_{\max}} n \dfrac{n-m}{\operatorname{var}\left(\overline{\delta_n r^2}\right)\operatorname{var}\left(\overline{\delta_m r^2}\right)}} \\ \underset{n_{\max}}{est_{uwLSQ}}(\sigma_{\exp}^2) = \dfrac{\sum_{m,n=1}^{n_{\max}} n \dfrac{\overline{\delta_n r^2} + \overline{\delta_m r^2}}{\operatorname{var}\left(\overline{\delta_n r^2}\right)\operatorname{var}\left(\overline{\delta_m r^2}\right)}}{2 \sum_{m,n=1}^{n_{\max}} n \dfrac{n+m}{\operatorname{var}\left(\overline{\delta_n r^2}\right)\operatorname{var}\left(\overline{\delta_m r^2}\right)}} \end{cases}$$

Finally, the generalized LSQ estimators are:

$$\mathbf{P}_{est} = \begin{pmatrix} 2\sigma_{\exp}^2 \\ \delta_{est} \end{pmatrix} \Rightarrow \delta_{est} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}^T \begin{pmatrix} 2\sigma_{\exp}^2 \\ \delta_{est} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}^T \mathbf{P}_{est} =$$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}^T \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N}\right)^{-1} \cdot \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{\Delta}\right)$$

The quantity of interest is var $\delta_{est} = \langle (\delta_{est} - \langle \delta_{est} \rangle)^2 \rangle$:

$$\operatorname{var}\left(\delta_{est}\right) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}^T \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N}\right)^{-1} \cdot \mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \left\langle \left(\Delta - \langle\Delta\rangle\right)\left(\Delta - \langle\Delta\rangle\right)^T \right\rangle \mathbf{\Sigma}^{-1^T}$$

$$\cdot \mathbf{N} \cdot \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1^T} \cdot \mathbf{N}\right)^{-1} \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

But since $\mathbf{\Sigma} = \langle (\mathbf{\Delta} - \langle \mathbf{\Delta} \rangle)(\mathbf{\Delta} - \langle \mathbf{\Delta} \rangle)^{\mathrm{T}} \rangle$:

$$\operatorname{var}\left(\mathbf{P}_{est}\right) = \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N}\right)^{-1} \cdot \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1^T} \cdot \mathbf{N}\right) \cdot \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N}\right)^{-1}$$

Note that the middle factor is a two-by-two symmetric matrix (due to the symmetry of **Σ)**, thus equal to its transpose:

$$\operatorname{var}\left(\mathbf{P}_{est}\right) = \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N}\right)^{-1} \cdot \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N}\right) \cdot \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N}\right)^{-1} = \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N}\right)^{-1}$$

and since

$$\mathbf{P}_{est} = \begin{pmatrix} 2\sigma^2_{\exp} \\ \delta_{est} \end{pmatrix} \Rightarrow \ \mathrm{var}(\delta_{est}) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}^T \mathrm{var}\begin{pmatrix} 2\sigma^2_{\exp} \\ \delta_{est} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}^T \mathrm{var}\left(\mathbf{P}_{est}\right) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}^T \left(\mathbf{N}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{N}\right)^{-1}$$

2. *Power spectra analysis*

An additional complication arises from the fact that trajectories are finite and of unequal length. This is due to various factors – cell tracking software tracking gaps, cells moving outside of the view field of the microscope, partially unusable trajectories due to e.g. cell division, clustering, lysis etc.

The finite length of the trajectory introduces a slight error due to end-effects in the construction of the power spectrum, which increases in magnitude as the length of the trajectory decreases and thus as the sampling frequency 1/Δt approaches the persistence time of the process, P. Starting from equation 8 that recursively defines u_j and applying the following operator on both sides:

$$\Delta t \sum_{j=1}^{N-1} e^{ij2\pi k/N}$$

then

$$\Delta t \sum_{j=1}^{N-1} e^{ij2\pi k/N} u_{j+1} = c\Delta t \sum_{j=1}^{N-1} e^{ij2\pi k/N} u_{j+1} + \Delta t \sum_{j=1}^{N-1} e^{ij2\pi k/N} \Delta u_j$$

Following a change in variables from j+1 to j on the left side and identifying $\hat{u}_k$ on the right hand side,

$$e^{-i2\pi k/N}\hat{u}_k = c\hat{u}_k - cu_N + u_1 - A\eta_1^a - B\eta_1^b + \sum_{j=1}^{N-1}(e^{ij2\pi k/N}A + a)\eta_j^a + (e^{ij2\pi k/N}B + b)\eta_j^b$$

thus

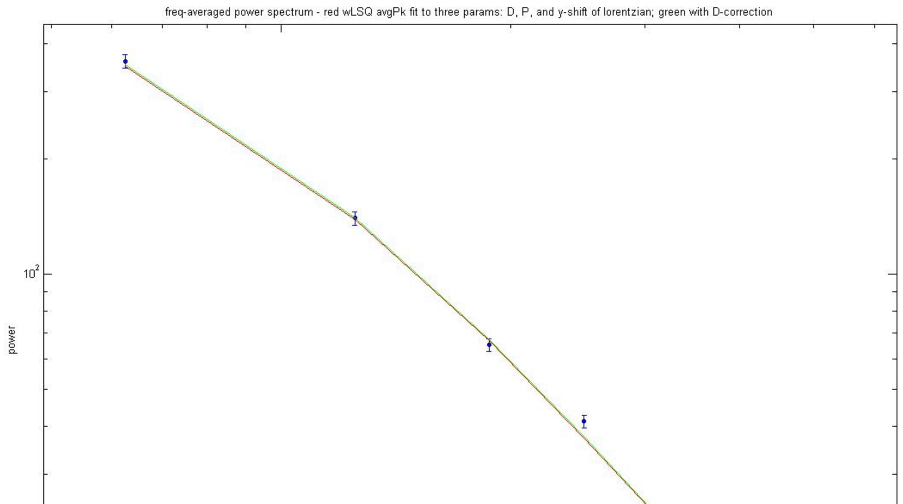
$$P_k^{u,th,endeff} = \frac{\left\langle \hat{u}_k^* \hat{u}_k \right\rangle}{N\Delta t} = P_k^{u,th} + \frac{\lambda_k}{N\Delta t}$$

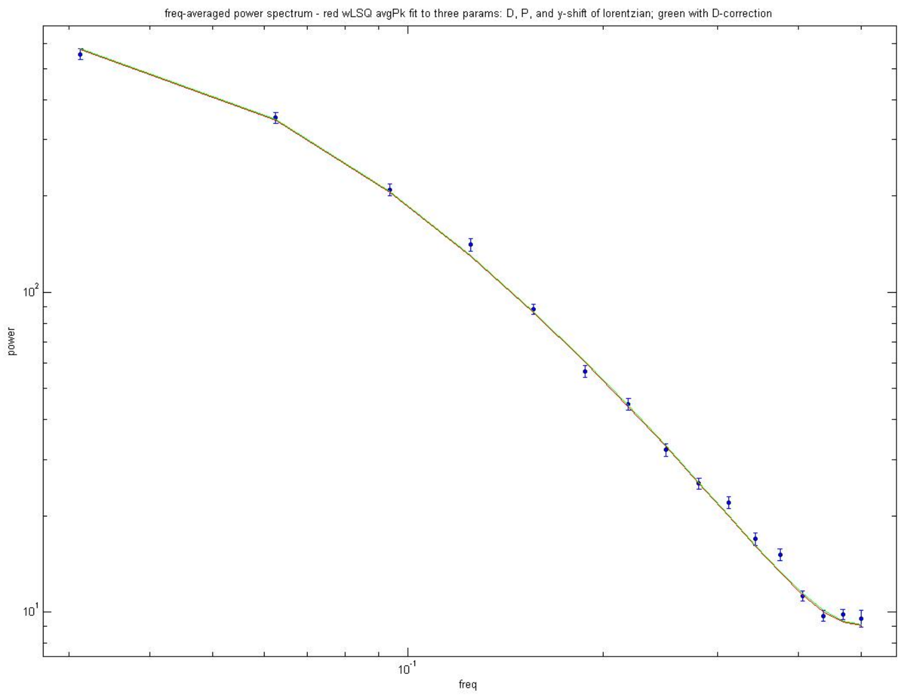
where

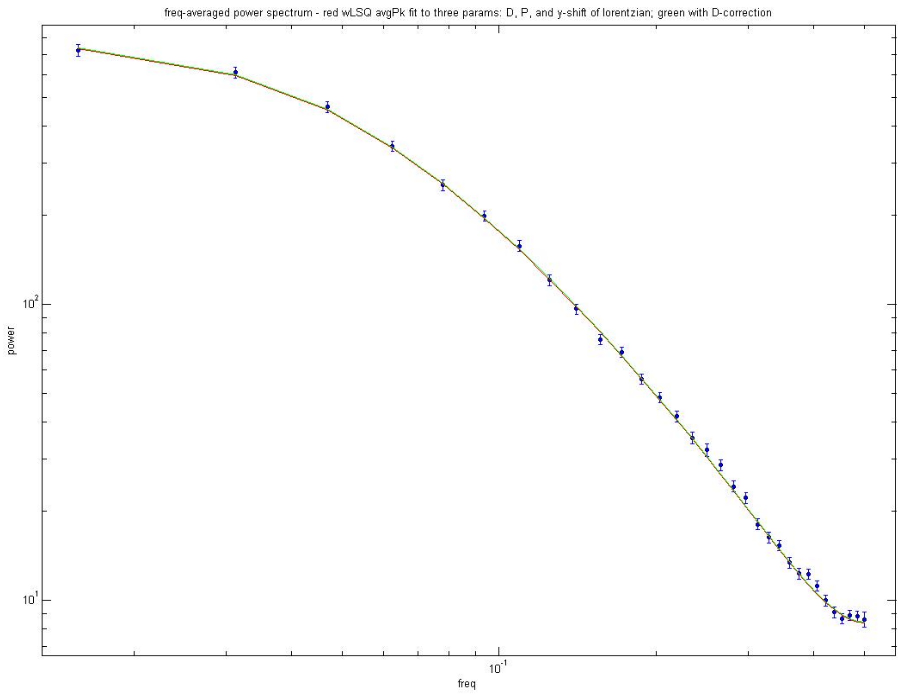
$$\lambda_k = (u_1 - cu_N)^2 + A^2 + B^2 - 2\left(\frac{a^2c + aAc^2 + b^2c + bBc^2}{1-c} + \frac{\left(caA + A^2c^2\right)\left(\cos(2\pi k/N) - c\right) +}{c^2 + 1 - 2\cos(2\pi k/N)} \right.$$
$$\frac{+ 2A^2\cos^2(2\pi k/N) - A^2 - \left(A^2c + a\right)\cos(2\pi k/N) - ac + \left(cbB + B^2c^2\right)\left(\cos(2\pi k/N) - c\right)}{}$$
$$\left. \frac{- B^2 - \left(B^2c + b\right)\cos(2\pi k/N) - bc}{} \right)$$

Indeed, a correction which varies as 1/N needs to be added to the theoretical power spectrum in the cases where N is not a large number. From a practical point of view, this is easiest done by fitting the experimental power spectrum, averaged over multiple independent trajectories, to $P_k^{inst,th}$-K as in the case of $N \to \infty$ and then iteratively calculating the correction term from the values of the parameters D and P and repeating the process. This is, however, only necessary when the trajectory lengths are short enough that the 1/N correction becomes significant.

For the purpose of assessing end-effects in power spectra fitting of velocities of motile cells on surfaces, a simulation of the OU process was used to generate 300 distinct trajectories of varying length. To that end, 17, 33, 65, and resp. 129 cell spatial coordinates were collected from the *in silico* data and used to calculate sets of 16, 32, 64, and resp. 128 discrete velocities, which were then used to calculate a velocity power spectrum. The 300 datapoints corresponding to each frequency in the power spectrum were averaged and the resulting averages with their respective standard deviations are shown in Fig 1A-D below. Since the power spectrum values are exponentially distributed, the averages are distributed according to the Γ_{300} distribution, which can be approximated with sufficient accuracy to a Gaussian distribution. Thus, the weighted least-squared can be employed to fit these averages to the expected Lorentzian curve. The parameters of the fit were D, P, and another constant (dependent in a complex way on D and P) which effectively shifts the Lorentzian downwards on the y axis. The resulting fit is shown in red in Fig 1.




freq-averaged power spectrum - red wLSQ avgPk fit to three params: D, P, and y-shift of lorentzian; green with D-correction
power
10^2
10^1
10^{-1}
freq


freq-averaged power spectrum - red wLSQ avgPk fit to three params: D, P, and y-shift of lorentzian; green with D-correction
power
10^2
10^1
10^{-1}
freq

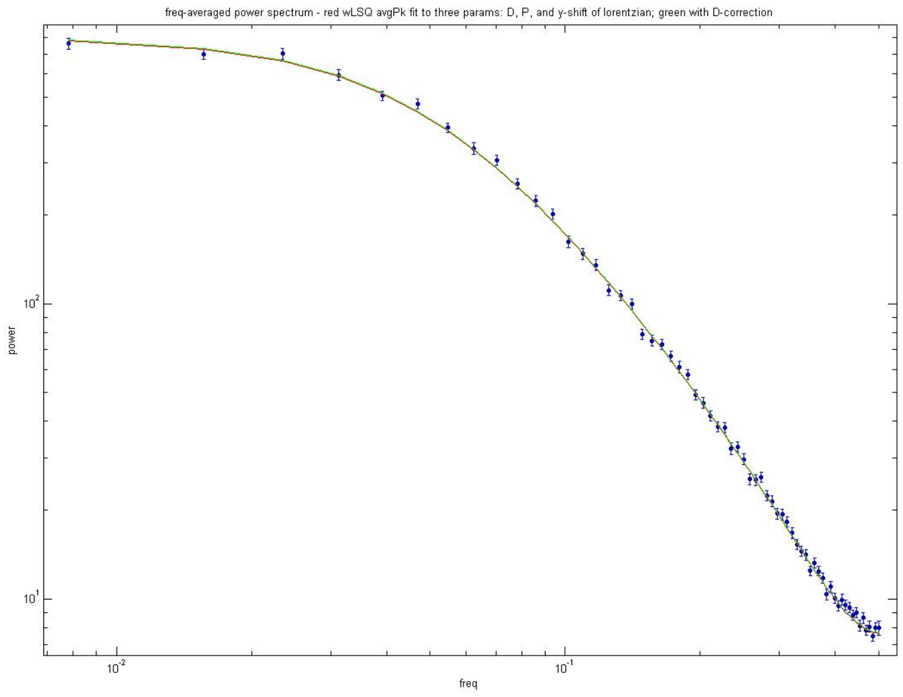


Fig 1. Frequency-averaged power spectrum of the discrete velocities of 300 individual motile cells *in silico* (blue) and the corresponding wLSQ fit to a theoretical expression (Lorentzian+constant), red. Trajectories have been recorded for 17, 33, 65, and resp. 129 frames. (A-D) Green trace shows the theoretical curve plotted with the wLSQ fit parameters corrected for bias (D'=D*N/(N-2)).

3. *Fitting experimental data*

a. *Fits to Experimental Data*

I provide below the defining equation of the velocity autocovariance function (<…> denotes ensemble averaging):

$$\varphi(t) = \left\langle \vec{v}(t) \cdot \vec{v}(0) \right\rangle$$

and the formula that has been used to calculate the experimental velocity autocovariance function from the motility data:

$$\varphi_j^{\exp} = \frac{\sum_{\alpha=1}^{last\,track} \sum_{k=0}^{N_\alpha - j} \vec{u}(t_j + t_k) \cdot \vec{u}(t_k)}{\sum_{\alpha=1}^{last\,track} N_\alpha}$$

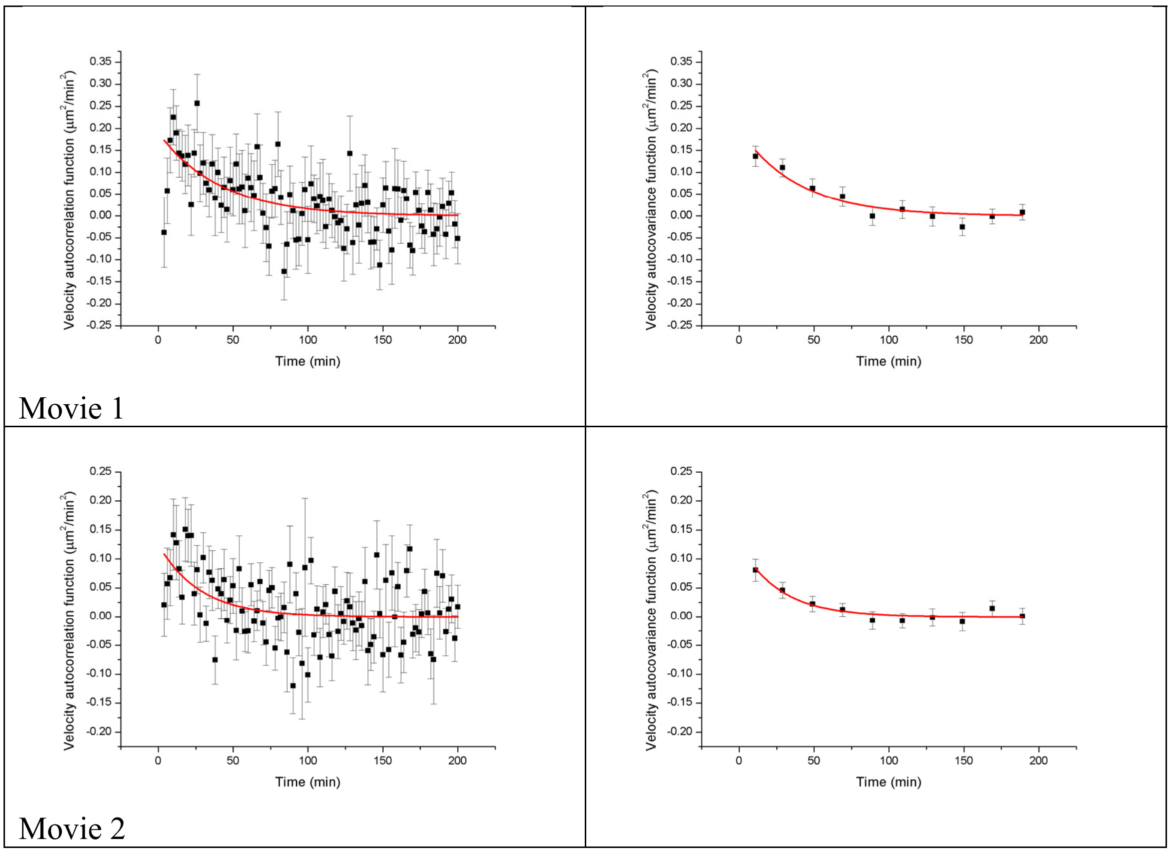
Equation 1

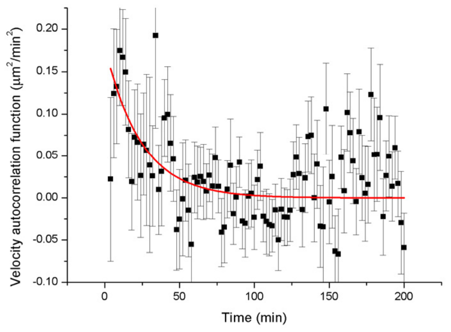
Here N_α denotes the number of points in track α, $t_j = j\Delta t$ and $t_k = k\Delta t$ as all of our movies were recorded at a fixed frame rate, and u(t) are secant velocities (Li et al. 2011), different from the instantaneous velocity v(t) due to discretization and positional noise.

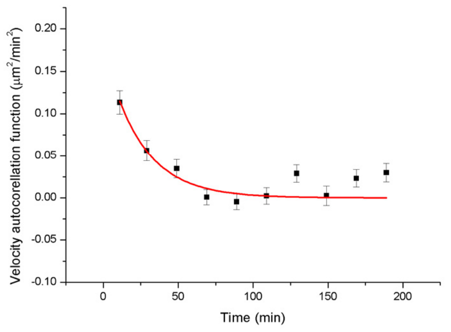
Table 1 - Motility parameters for 3T3/glass (9 tracks remaining after post-processing from movie 1, 12 tracks from movie 2, 15 from movie 3, 8 from movie 4, and 14 from movie 5; the combined datasets includes the data from all these movies: 58 tracks).

Movie number	φ_0 / baseline ($\mu m^2/min^2$)	P (min)
Combined	0.18 ± 0.02 / 0.003 ± 0.004	36 ± 4
1	0.19 ± 0.03 / -0.02 ± 0.02	41 ± 9
2	0.12 ± 0.04 / -0.002 ± 0.008	28 ± 9
3	0.16 ± 0.03 / 0.009 ± 0.004	26 ± 5
4	0.14 ± 0.03 / -0.001 ± 0.009	39 ± 8
5	0.25 ± 0.03 / 0.014 ± 0.009	45 ± 6

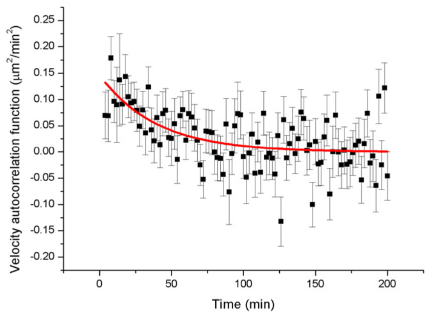
Figure 2 - Left: Monoexponential fits (red) to the velocity autocovariance function of NIH 3T3 cells, shown as black squares in the graphs below (fit results shown in figure 4 and Table S1). Right: Red trace showing fits to 10-point bins of the velocity autocovariance function data points, shown in black. The motility parameters measured by fit to the binned autocovariance function match extremely well the ones shown in Table 1.

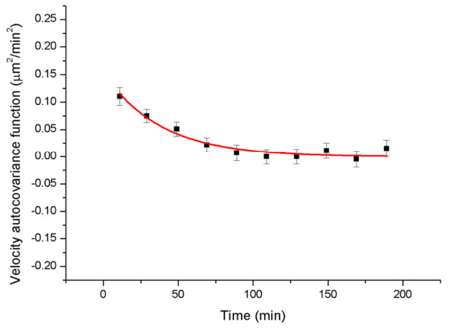




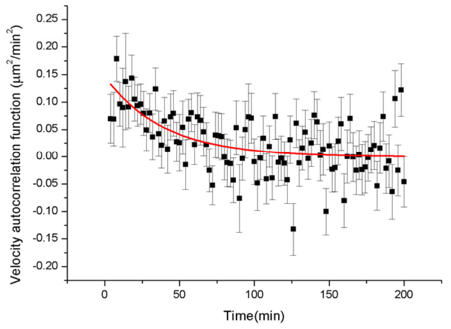


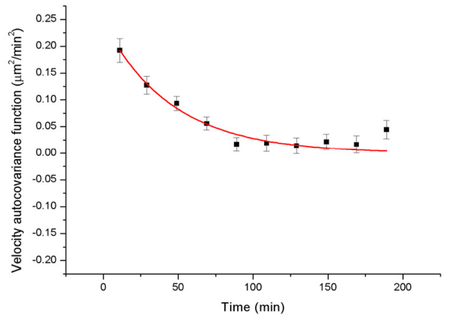
Movie 3



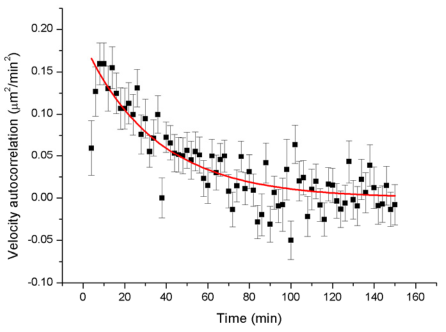


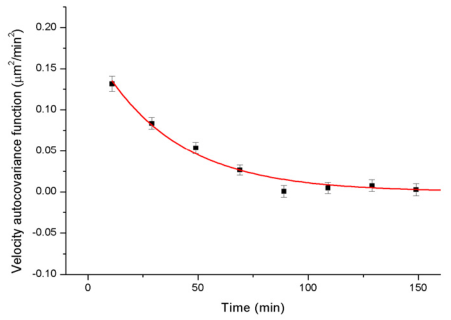
Movie 4





Movie 5





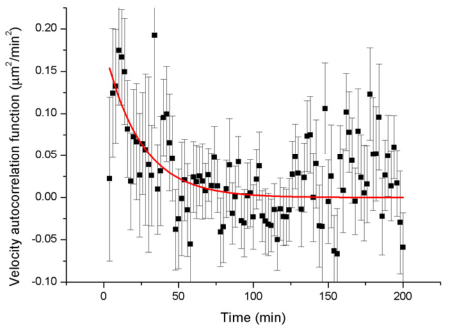
Combined dataset

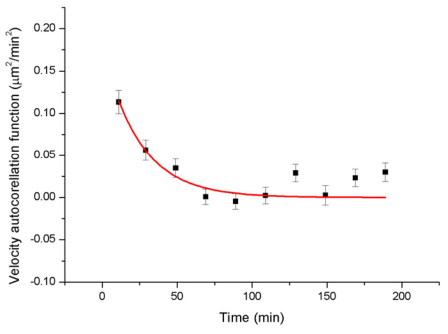
Table 3 - Comparison of PACT, Autozell, and TLA with regards to data analysis

Program	φ_0 ($\mu m^2/min^2$)	P (min)
PACT	0.16 ± 0.03	26 ± 5
Autozell	0.09 ± 0.02	34 ± 8

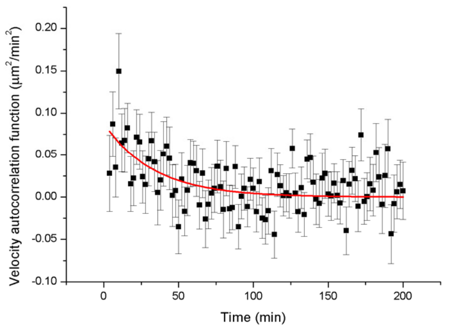
TLA/Kalman filter/MA smoothing	0.09 ± 0.01	59 ± 6
TLA/no filter/no smoothing	0.10 ± 0.01	58 ± 4
PACT/common	0.18 ± 0.04	24 ± 6
Autozell/common	0.18 ± 0.02	37 ± 4
TLA/common/no Kalman filter	0.16 ± 0.04	30 ± 4
TLA/common/Kalman filter	0.10 ± 0.01	58 ± 7
PACT/common/truncated	0.18 ± 0.04	25 ± 6

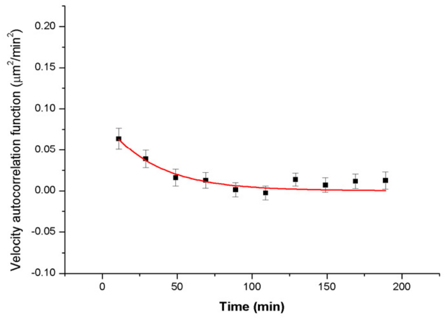
Figure 4 - Left: Monoexponential fits (red) to the velocity autocovariance function of NIH 3T3 cells, shown as black squares in the graphs below (fit results shown in figure 5 and Table 3). Right: Right: Red trace showing fits to 10-point bins of the velocity autocovariance function data points, shown in black.



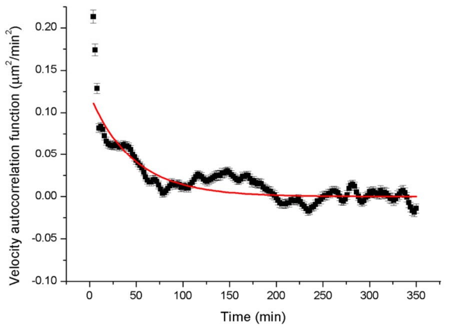


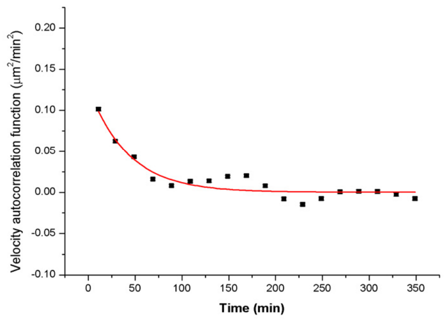
PACT/Original dataset



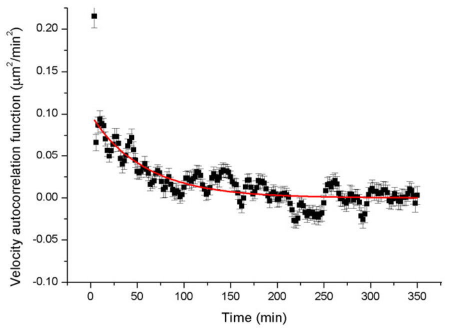


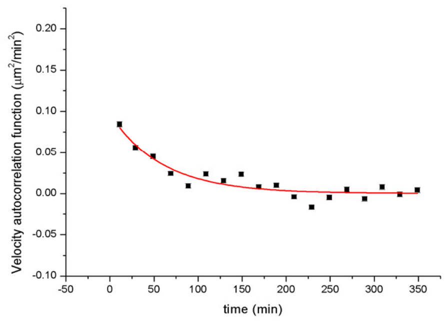
Autozell/Original dataset



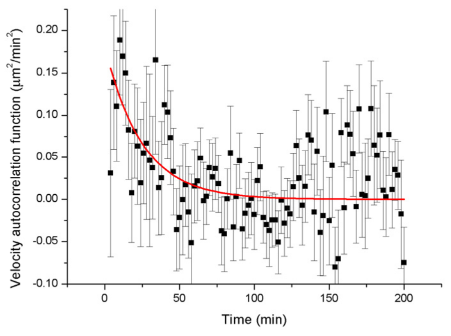


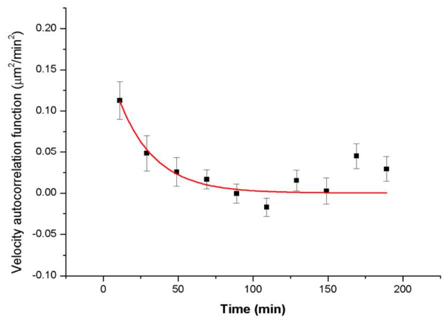
TLA/Original dataset



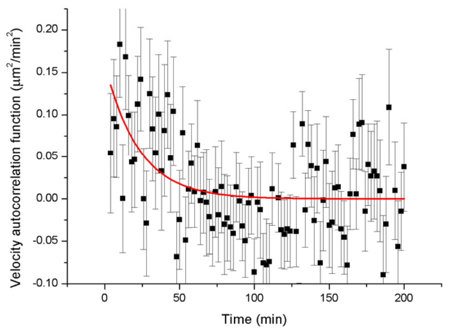


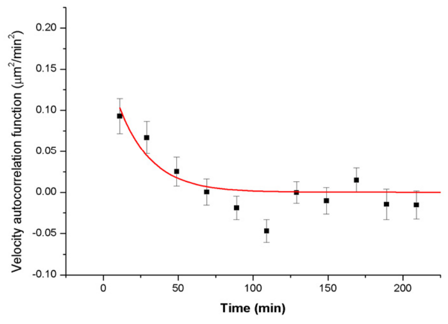
TLA:no filter-smoothing/Original dataset



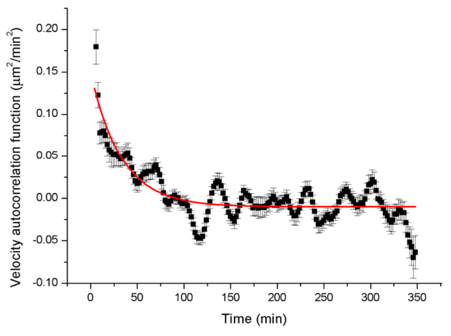


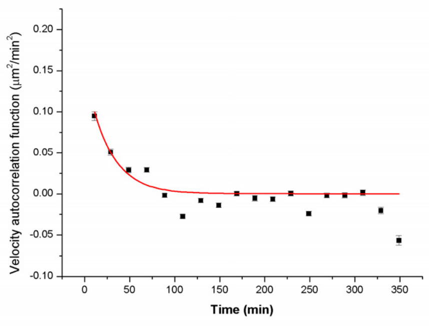
PACT/subset





Autozell/subset





TLA/subset

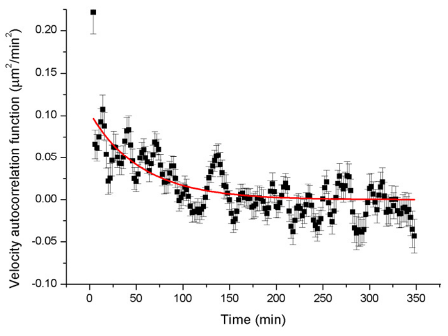
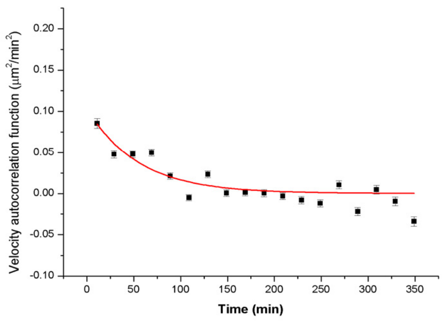
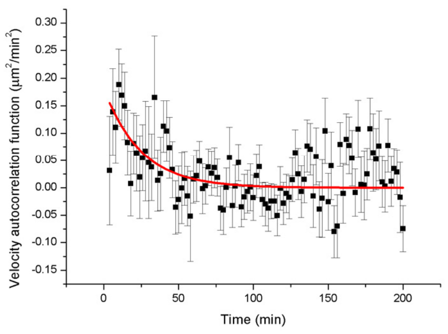
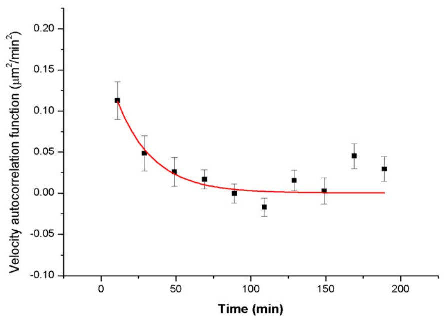
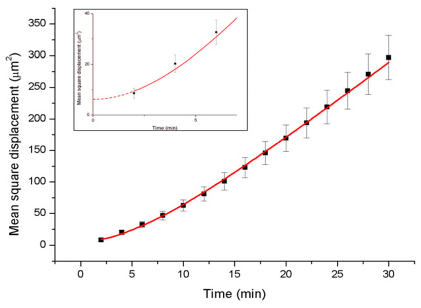
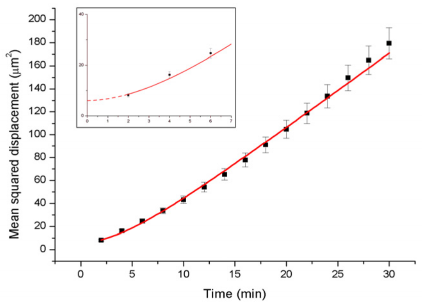
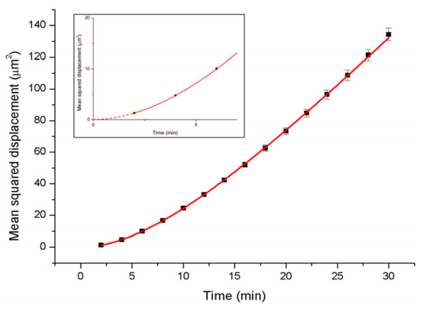
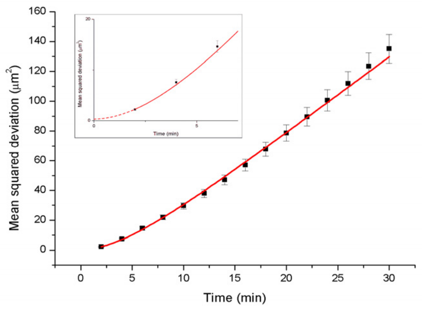
 TLA/subset/no filter-smoothing	
 PACT/subset/nearest digit-truncated	

Table 6 - Modified Fürth's formula fits (red) to mean square displacement of NIH 3T3 cells, shown as black squares in the graphs below (fit results shown in section (viii) of this paper). The insets show the first five points of the mean squared displacements and the corresponding fit, extrapolated to zero (dotted line). The y intercept is used to determine the positional noise.

 PACT	 Autozell
 TLA	 TLA/no filter

b. Data over-dispersion correction

I have initially determined the weighted mean and standard deviation of the mean for the motility parameters P and φ_0 from the 5 NIH 3T3/glass movies (Table S1, Figure S2, Supplementary information 1) using the standard formulas (N is the number of movies, in this case 5; equation shown for P and similar for φ_0):

$$\bar{P} = \sum_{i=1}^{N} \frac{P_i}{\sigma_i^2} \bigg/ \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \quad \text{and} \quad \sigma_{\bar{P}}^2 = 1 \bigg/ \sum_{i=1}^{N} \frac{1}{\sigma_i^2}$$

Equation S2

A correction is due to account for data overdispersion in the individual P_i values. This method assumes that the data is drawn from distributions of the same mean P value (which is sensible given the data in figure 4 is from the same sample), but with different σ_i values (also a reasonable assumption given the differing number of tracks and track lengths included in each of the 5 movies in figure 4). The χ^2 (quantity being minimized as part of the least-squares fit) is expected to be 1 for data dispersed in accordance to the provided σ_i's, but that is not the case for this dataset: χ^2 (P) = 1.8 and χ^2 (φ_0) = 2.5. Thus, we correct the error bar of the mean P value above:

$$\sigma_{\bar{P}}^{corr} = \sqrt{\chi^2(P)} \cdot \sigma_{\bar{P}}, \quad \text{with} \quad \chi^2(P) = \frac{1}{N-1} \sum_{i=1}^{N} \frac{\left(P_i - \bar{P}\right)^2}{\sigma_i^2}$$

Equation S3

In the absence of additional information, we have assumed equal overdispersion for all 5 movies and thus estimated the real dispersion of the motility parameters for an individual movie:

$$\sigma_{P_i}^{corr} = \sqrt{\chi^2(P)} \cdot \sigma_{P_i}$$

Equation S4

4. MATLAB code employed for simulations

a. Brownian motion

```
function []=t(D, dt, NN, Nens)
ensavg_var_dnr2mean=zeros(1,NN);
est_var_dnr2mean=zeros(1,NN);
ensavg_dnr2mean=zeros(1,NN);
expval_dnr2mean=zeros(1,NN);
n=1;
for N=n+1:NN
    N
for i_ens=1:Nens

    % create a trajectory vector r(t):
    r=zeros(2,N);
    % starting at (x0,y0)=(0,0)

    % generate a Gaussian-distributed velocity from N(0,sqrt(2D))
    dx=random('norm', 0, sqrt(2*D*dt), [1, N-1]);
    dy=random('norm', 0, sqrt(2*D*dt), [1, N-1]);
    dr=[dx; dy];

    % generate trajectories
    for i=1:N-1
        r(:,i+1)=r(:,i)+dr(:,i);
    end

    % Part 2: data analysis

    dnr2mean=0;

        dnr2=zeros(1,N-n);
        for i=1:N-n
            dnr2(i)=(r(1,i+n)-r(1,i))^2+(r(2,i+n)-r(2,i))^2;
        end
        dnr2mean=mean(dnr2);
        ensavg_var_dnr2mean(N)=ensavg_var_dnr2mean(N)+(dnr2mean-4*n*D*dt)^2;
        ensavg_dnr2mean(N)=ensavg_dnr2mean(N)+dnr2mean;
end

% get actual dnr2mean and var_dnr2mean: note no bias due to using actual miu above.

    ensavg_var_dnr2mean(N)=ensavg_var_dnr2mean(N)/Nens;
    expval_dnr2mean(N)=4*n*D*dt;
    ensavg_dnr2mean(N)=ensavg_dnr2mean(N)/Nens;

% since we are dealing with correlated data in the dnr2(i) set
% we need a different way of estimating the error bars on dr2mean(n)
% see notebook 1 for derivation, Sep 11 2008

        est_var_dnr2mean(N)=(4/(N-1))*(2*D*dt)^2;

end

% plot the points with their respective ensemble and estimated error bars.
figure;
plot ([n+1:NN], [ensavg_dnr2mean(n+1:NN)], '-r', [n+1:NN], [expval_dnr2mean(n+1:NN)-
sqrt(ensavg_var_dnr2mean(n+1:NN))], '-r', [n+1:NN],
[expval_dnr2mean(n+1:NN)+sqrt(ensavg_var_dnr2mean(n+1:NN))], '-r', [n+1:NN],
[expval_dnr2mean(n+1:NN)], '-b', [n+1:NN], [expval_dnr2mean(n+1:NN)-
sqrt(est_var_dnr2mean(n+1:NN))], '-b', [n+1:NN],
[expval_dnr2mean(n+1:NN)+sqrt(est_var_dnr2mean(n+1:NN))], '-b');
title(['d_nr^2mean(1..N-1) +/- error bars (red ensavg, blue expval)']);
xlabel('N');
ylabel('d_nr^2(n)+/-var');

% plot the fractional residuals for ensavg_dnr2mean vs expval_dnr2mean
figure
plot ([n+1:NN], [ensavg_dnr2mean(n+1:NN)./expval_dnr2mean(n+1:NN)-1], '-g' );
title(['residuals EnsAvg vs ExpVal:d_nr^2mean(1..N-1)']);
xlabel('N');
ylabel('resid d_nr^2mean(n)');

% plot the fractional residuals for estSigma_dnr2mean vs ensavg_var_dnr2mean
figure
plot ([n+1:NN], [sqrt(est_var_dnr2mean(n+1:NN)./ensavg_var_dnr2mean(n+1:NN))-1], '-m' );
title(['residuals EstVar vs EnsAvgVar:d_nr^2mean(1..N-1)']);
xlabel('N');
ylabel('resid var d_nr^2(n)');

end
```

b. The standard OU process

```
function []=OUphi()
% Monte Carlo simulation of Persistent 2D motion of cell as defined by the OU Process
% positions and velocities simulated simultaneously as per Norrelykke et
% al. 2009

% This program generates (x,y) pairs [with positional noise sig_pos] which
% behave as experimental-like data, as well as the (x,y, vx, vy) quads for
% purely theoretical considerations.
```

```
clear all
close all
clc

% use power of 2 for optimal fft:
N=128;

% use NN=50 for ensemble averaging (number of cells/independent trajectories)
% that's what you'd typically get from some 5-6 movies.
NN=50;

%%%% PARAMETERS %%%%

D=7.3; % um^2/min, some typical value as in FK's 3T3/glass.
P=40; % 20 min, some typical value
sig_pos=1; % um, some reasonable value (std of 1 um in centroid localization)4

% Conversion parameters
dt=.25; % Amount of time between successive frames, min
um2pix=0.977; % size of a pixel in um.

% program starts below

c=exp(-dt/P);
Aplus=sqrt(D*P/2*(1-c^2));
Aminus=sqrt(D*dt);
alpha=sqrt((2*P/dt)*((1-c)/(1+c)));


'***************************************************************************'
'************** NEW RUN ********************** NEW RUN ********************'
'***************************************************************************'


% Generate data
v=zeros (N,2,NN);
r=zeros (N,2,NN);

% generate a Gaussian-distributed noise from N(0,1)
eta_a=random('norm', 0, 1, [N-1,2, NN]);
eta_b=random('norm', 0, 1, [N-1,2, NN]);

% generate trajectories & velocities
v(1,:,:)=sqrt(D/P)*random ('norm', 0, 1, [1, 2, NN]);

% for simplicity it is ok to have intersecting trajectories on this
% 750x1000 pixel-image, as well as cells exiting and entering the
% viewfield. Then all trajectories will have length N.
r(1,1,:)=random('unif', 0, 1024, [1,1, NN]);
r(1,2,:)=random('unif', 0, 768, [1,1, NN]);

for j=1:N-1
    r(j+1,:,:)=r(j,:,:)+P*(1-c)*v(j,:,:)+eta_a(j,:,:)*(Aminus-Aplus)*sqrt(1+alpha)-
eta_b(j,:,:)*(Aminus+Aplus)*sqrt(1-alpha);
    v(j+1,:,:)=c*v(j,:,:)+eta_a(j,:,:)*Aplus*sqrt(1+alpha)/P+eta_b(j,:,:)*Aplus*sqrt(1-
alpha)/P;
end

% now add Gaussian-distributed positional noise with std sig_pos

r=r+random ('norm', 0, 1, [N,2,NN])*sig_pos;

% generate dataset in the format accepted by data_analysis.

tr=zeros(NN*N,6);
index=1;
for i=1:NN
    for j=1:N
            tr(index,1)=r(j,1,i)/um2pix; % units of pixels
            tr(index,2)=r(j,2,i)/um2pix; % units of pixels
            tr(index,3)=j;
            tr(index,4)=i;
            tr(index,5)=v(j,1,i)*dt/um2pix; % units of pix/dt
            tr(index,6)=v(j,2,i)*dt/um2pix; % units of pix/dt
            index=index+1;
    end
```

```
end

tr=tr(1:index-1,:);
% save data to ascii file: x y time index vx vy
save -ascii 'trajs-and-instant-vels.txt' tr;

tr1=tr(:,1:4);
% save data to ascii file: x y time index
save -ascii 'trajectories-picked.txt' tr1;

% the latter uses data_analysis_exp

 data_analysis_exp()

end
```

5. PACT: MATLAB code

```
function track_movie = track_movie_ver_1_0()

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%%%%%%%%%%%%%%%%%%%%%% VERSION 1.0: CELL TRACKING
%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%
% This program will process images of sparse motile cells on a surface
and
% output the cell centroid coordinates.
%
% The frames must be located in same folder as the .m files and a
% "processed" folder is created where the processed data is saved.
% If it already exists, a warning message will be output.
%
% The image filename template must be set inside the .m file under the
% section "WORKING PARAMETERS." That section also needs to be changed if
% the image is inverted (i.e. light cells on dark background).
%
%%%%%%%%%%%%%%%%%%%%%%%%% HOW THE PROGRAM WORKS
%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% The image may be color or bw; the program will remove the color
component
% if present. 8-bit depth is sufficient if the contrast is good, else
16-,
% 24-, etc. may be needed.
%
% A bandpass filter will be first applied to remove pixel noise and
% non-uniform background illumination effects. These high- and
% low-frequency limits are set up in the WORKING PARAMETERS section.
%
% Next the peaks brighter than a user-defined peak exclusion threshold
are
% selected; they are the initial guess for where the cells are and will
be
% shown as red crosses on the image. The cell contour is defined as the
% contour line laying at a user-defined fraction of the height of its
% corresponding peak and is shown as a blue contour line around the
peak.
%
```

```
% Further selection includes restrictions on minimal and maximal cell
area
% and minimal distance between cells. The default parameters set up in
the
% WORKING PARAMETERS section are generally useable, but may be changed
for
% special circumstances.
%
% Once these parameters defined, you are ready to run the program; you
will
% be guided further as it runs. All output from the program may be found
in
% logfile.txt once it has finished.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% WORKING PARAMETERS %%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Generic name for frame files [e.g. image_file*.jpg]
filename_template=('*.jpg');

% Image should be dark cells on light background, else set invert to
false
% [true or false]
invert=true;

% Bandpass filter limits:
%
% Lower limit removes high-freq noise due to fluctuations from
% pixel-to-pixel. This thus should be set to 1-3, unless features on the
% surface (such as nanostructures visible under the microscope) are
% present.
%
% Upper limit removes low-freq noise due to fluctuations in
illumination,
% but should not remove cell-sized objects, thus set to the typical cell
% diameter in pixels.

BP_low=3;
BP_high=30;

% Minimum distance between peaks: set by default to twice the diameter
of a
% typical cell.  This excludes peaks located closer than this limit, to
% avoid finding two peaks on the same cell.
min_dist=2*BP_high;

% Minimum cell area [pixels]:
% by default no smaller than the area of a circle whose diameter is 1/3
% than that of the "average cell diameter" defined above as BP_high
min_area=round(pi*(BP_high/2/3)^2);

% Maximum cell area [pixels]
% by default no larger than the area of a circle whose diameter is
triple
% that of the "average cell diameter" defined above as BP_high.
max_area=round(pi*(3/2*BP_high)^2);
```

```
% Program starts below:

close all
clc

logfile=fopen('logfile.txt','w');

fprintf('**** TRACK_MOVIE program *****\n');
fprintf('Type help track_movie_ver_1_0 for information on using this
software.\n');
fprintf(logfile,'**** TRACK_MOVIE program *****\n');
fprintf(logfile,'Type help track_movie_ver_1_0 for information on using
this software.\n');



% read image
files=dir(filename_template);
[filenumber nc]=size(files);

if filenumber==0
    fprintf('ERROR: NO IMAGE FILES MATCHING THE FILENAME TEMPLATE FOUND.
PLEASE CONSULT THE HELP FILE.\n\n\n', filenumber);
    fprintf(logfile,'ERROR: NO IMAGE FILES MATCHING THE FILENAME
TEMPLATE FOUND. PLEASE CONSULT THE HELP FILE.\n\n\n', filenumber);
    return
else
    fprintf('Found %d frames in the current directory.\n\n\n',
filenumber);
    fprintf(logfile,'Found %d frames in the current directory.\n\n\n',
filenumber);
end
fprintf('The program will now process the first and last image in the
sequence.\n');
fprintf('You will be asked to choose a peak exclusion threshold and
contour cutoff for each by visual inspection.\n');
fprintf('In most cases they will be the same; however, for long movies,
they may differ substantially.\n');
fprintf('In that case a linear function will be used to model the
changing threshold, and the average cutoff will be used.\n\n');
fprintf(logfile,'The program will now process the first and last image
in the sequence.\n');
fprintf(logfile,'You will be asked to choose a peak exclusion threshold
and contour cutoff for each by visual inspection.\n');
fprintf(logfile,'In most cases they will be the same; however, for long
movies, they may differ substantially.\n');
fprintf(logfile,'In that case a linear function will be used to model
the changing threshold, and the average cutoff will be used.\n\n');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%INITIAL ANALYSIS ON FIRST FRAME%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

a = uint8(imread(files(1).name));

% remove color component if present
file_dim=size(size(a));
if file_dim(2)==3
    a=mean(a,3);
end;
```

```
if invert
    a=255-a;
end

b = bpass(a,BP_low,BP_high);
b=uint8(b.*255/max(max(b)));

% find peaks as an initial guess as to where the cells are

figure
peak_thresh=60;
contour_cutoff=60;
fprintf('****Processing first frame now****\n\n');
fprintf(logfile,'****Processing first frame now****\n\n');

cont_thr=true;
cont_cutoff=true;
while cont_thr | cont_cutoff
    pk = peak_find(b,peak_thresh,min_dist);
    [nr,nc]=size(b);
    bw=zeros(nr,nc);
    colormap('gray'), imagesc(a);
    hold on
    [no_peaks nr]=size(pk);
    fprintf('Setting threshold to %d/255 and cutoff to %d%%...\n',
peak_thresh, contour_cutoff);
    fprintf(logfile,'Setting threshold to %d/255 and cutoff to
%d%%...\n', peak_thresh, contour_cutoff);
    %plot these as red cross-hairs on top of the original image
    plot (pk(:,1),pk(:,2),'xr');

    % find boundaries within contour_cutoff% of peak.
    [bw_x,bw_y]=find(b(:)>peak_thresh*contour_cutoff/100);
    bw([bw_x,bw_y])=1;

    % remove all objects containing fewer than min_area pixels
    bw = bwareaopen(bw,min_area);

    % remove all objects containing more than max_area pixels
    bw = bwareaopen_max(bw,max_area);

    % remove all objects that do not contain one of the peaks in pk
    bw=bwselect(bw,pk(:,1),pk(:,2));

    % find boundaries
    [B,L] = bwboundaries(bw,'noholes');

    fprintf('Found %d peaks shown as red crosses and %d cells, with
contours shown in blue.\n\n', no_peaks, length(B));
    fprintf(logfile,'Found %d peaks shown as red crosses and %d cells,
with contours shown in blue.\n\n', no_peaks, length(B));

    for k = 1:length(B)
      boundary = B{k};
      plot(boundary(:,2), boundary(:,1), 'b', 'LineWidth', 1)
    end
    title (strcat('First frame thresholded at: ', num2str(peak_thresh),
'/255, contour cutoff=', num2str(contour_cutoff), '%'));
    readout=input('Select new threshold or enter to continue (decrease
for more peaks, increase for fewer): ');
    fprintf(logfile,'Select new threshold or enter to continue (decrease
for more peaks, increase for fewer): %d\n', readout);
if (readout~=0)
        peak_thresh=readout;
    else
        cont_thr=false;
    end
    readout=input('Select new contour cutoff or enter to continue
(increase for smaller area, decrease for larger): ');
    fprintf(logfile,'Select new contour cutoff or enter to continue
(increase for smaller area, decrease for larger): %d\n', readout);
    if (readout~=0)
        contour_cutoff=readout;
    else
        cont_cutoff=false;
    end
    hold off
end

fprintf('\nFinal parameters accepted for first frame: peak threshold
%d/255, contour cutoff %d%%.\n\n\n', peak_thresh, contour_cutoff);
fprintf(logfile,'\nFinal parameters accepted for first frame: peak
threshold %d/255, contour cutoff %d%%.\n\n\n', peak_thresh,
contour_cutoff);

peak_thresh_start=peak_thresh;
contour_cutoff_start=contour_cutoff;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%INITIAL ANALYSIS ON LAST FRAME%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

a = uint8(imread(files(filenumber).name));

% remove color component if present
file_dim=size(size(a));
if file_dim(2)==3
    a=mean(a,3);
end;

if invert
    a=255-a;
end

b = bpass(a,BP_low,BP_high);
b=uint8(b.*255/max(max(b)));

% find peaks as an initial guess as to where the cells are

figure

% use old params from frame %1 to start with

fprintf('****Processing last frame now****\n\n');
fprintf(logfile,'****Processing last frame now****\n\n');
```

```
cont_thr=true;
cont_cutoff=true;
while (cont_thr | cont_cutoff)
    pk = peak_find(b,peak_thresh,min_dist);
    [nr,nc]=size(b);
    bw=zeros(nr,nc);
    colormap('gray'), imagesc(a);
    hold on
    [no_peaks nr]=size(pk);
    fprintf('Setting threshold to %d/255 and cutoff to %d%%...\n',
peak_thresh, contour_cutoff);
    fprintf(logfile,'Setting threshold to %d/255 and cutoff to
%d%%...\n', peak_thresh, contour_cutoff);
    %plot these as red cross-hairs on top of the original image
    plot (pk(:,1),pk(:,2),'xr');

    % find boundaries within contour_cutoff% of peak.
    [bw_x,bw_y]=find(b(:)>peak_thresh*contour_cutoff/100);
    bw([bw_x,bw_y])=1;

    % remove all objects containing fewer than min_area pixels
    bw = bwareaopen(bw,min_area);

    % remove all objects containing more than max_area pixels
    bw = bwareaopen_max(bw,max_area);

    % remove all objects that do not contain one of the peaks in pk
    bw=bwselect(bw,pk(:,1),pk(:,2));

    % find boundaries
    [B,L] = bwboundaries(bw,'noholes');

    fprintf('Found %d peaks shown as red crosses and %d cells, with
contours shown in blue.\n\n', no_peaks, length(B));
    fprintf(logfile,'Found %d peaks shown as red crosses and %d cells,
with contours shown in blue.\n\n', no_peaks, length(B));

    for k = 1:length(B)
      boundary = B{k};
      plot(boundary(:,2), boundary(:,1), 'b', 'LineWidth', 1)
    end

    title (strcat('Last frame thresholded at: ', num2str(peak_thresh),
'/255, contour cutoff=', num2str(contour_cutoff), '%'));
    readout=input('Select new threshold or enter to continue (decrease
for more peaks, increase for fewer): ');
    fprintf(logfile,'Select new threshold or enter to continue (decrease
for more peaks, increase for fewer): %d\n', readout);
    if (readout~=0)
        peak_thresh=readout;
    else
        cont_thr=false;
    end
    readout=input('Select new contour cutoff or enter to continue
(increase for smaller area, decrease for larger): ');
    fprintf(logfile,'Select new contour cutoff or enter to continue
(increase for smaller area, decrease for larger): %d\n', readout);
    if (readout~=0)
```

```
        contour_cutoff=readout;
    else
        cont_cutoff=false;
    end
    hold off
end

fprintf('\nFinal parameters accepted for last frame: peak threshold
%d/255, contour cutoff %d%%.\n\n\n', peak_thresh, contour_cutoff);
fprintf(logfile,'\nFinal parameters accepted for last frame: peak
threshold %d/255, contour cutoff %d%%.\n\n\n', peak_thresh,
contour_cutoff);

peak_thresh_end=peak_thresh;
contour_cutoff_end=contour_cutoff;

%Set up peak_thresh linear function

thr_slope=(peak_thresh_end-peak_thresh_start)/(filenumber-1);
thr_intercept=peak_thresh_start-thr_slope;

if peak_thresh_start==peak_thresh_end
    fprintf('Equal thresholds chosen for first and last frames, will use
thresh=%d/255 for all frames.\n', peak_thresh_start);
    fprintf(logfile,'Equal thresholds chosen for first and last frames,
will use thresh=%d/255 for all frames.\n', peak_thresh_start);
else
    fprintf('Different thresholds chosen for first and last frames, will
use thresh=%3.1f+(frame #)*%3.3f to process data.\n', thr_intercept,
thr_slope);
    fprintf(logfile,'Different thresholds chosen for first and last
frames, will use thresh=%3.1f+(frame #)*%3.3f to process data.\n',
thr_intercept, thr_slope);
end

peak_thresh=zeros(filenumber,1);
peak_thresh(:)=thr_intercept+(1:filenumber)*thr_slope;

contour_cutoff=(contour_cutoff_start+contour_cutoff_end)/2;

close all;
figure

[est_no_cells,dim]=size(pk);
pos_list=zeros(3,filenumber(1)*est_no_cells*2);
pos_list_index=0;
if exist('./processed')==7
    fprintf('\nWARNING: directory ./processed already exists.\n');
    fprintf(logfile,'\nWARNING: directory ./processed already
exists.\n');
    readout=input('Press Enter to continue and overwrite, CTRL-C to stop
here: ');
    fprintf(logfile,'Press Enter to continue and overwrite, CTRL-C to
stop here: %d\n', readout);
else mkdir('.\processed');
end
for fileindex=1:filenumber(1)
%for fileindex=1:10
```

```
    fprintf('Currently processing frame #%d with peak_thresh=%3.1f and
contour_cutoff=%d%% : %s\n', fileindex, peak_thresh(fileindex),
contour_cutoff, files(fileindex).name);
    fprintf(logfile,'Currently processing frame #%d with
peak_thresh=%3.1f and contour_cutoff=%d%% : %s\n', fileindex,
peak_thresh(fileindex), contour_cutoff, files(fileindex).name);
    a = double(imread(files(fileindex).name));

file_dim=size(size(a));
if file_dim(2)==3
    a=mean(a,3);
end;

if invert
    a=255-a;
end

b = bpass(a,BP_low,BP_high);
b=uint8(b.*255/max(max(b)));

pk = peak_find(b,peak_thresh(fileindex),min_dist);

[nr,nc]=size(b);
bw=zeros(nr,nc);

% find boundaries within contour_cutoff% of peak.
[bw_x,bw_y]=find(b(:)>peak_thresh(fileindex)*contour_cutoff/100);
bw([bw_x,bw_y])=1;

% remove all objects containing fewer than min_area pixels
bw = bwareaopen(bw,min_area);

% remove all objects containing more than max_area pixels
bw = bwareaopen_max(bw,max_area);

% remove all objects that do not contain one of the peaks in pk
bw=bwselect(bw,pk(:,1),pk(:,2));

% find boundaries
[B,L] = bwboundaries(bw,'noholes');

colormap('gray'), imagesc(a);
hold on

stats = regionprops(L,'Area','Centroid');

for k = 1:length(B)
  boundary = B{k};
  plot(boundary(:,2), boundary(:,1), 'b', 'LineWidth', 1)
  area = stats(k).Area;
  centroid = stats(k).Centroid;
  pos_list_index=pos_list_index+1;
  plot(centroid(1),centroid(2),'xr');
  pos_list(1,pos_list_index)=centroid(1);
  pos_list(2,pos_list_index)=centroid(2);
  pos_list(3,pos_list_index)=fileindex; % time, really.

end
```

```
title (strcat(files(fileindex).name, ' - frame # ',
num2str(fileindex)));
saveas(gcf, strcat('.\processed\',files(fileindex).name,'-processed -
frame # ', num2str(fileindex),'.jpg'), 'jpeg')
pause(0.01);
hold off

end

% create & plot trajectories
pos_list=pos_list(:,1:pos_list_index)';
tr = track_m(pos_list, min_dist);

figure
a = double(imread(files(1).name));
file_dim=size(size(a));
if file_dim(2)==3
    a=mean(a,3);
end;

if invert
    a=255-a;
end

colormap('gray'), imagesc(a);
hold on
color_traj='bgrcmykw';

q=1;
[nr nc]=size(tr);


for q=1:nr
    plot (tr(q,1), tr(q,2), strcat('.',color_traj(mod(tr(q,4),8)+1)) ,
'markersize', 4);
    if q~=1
        if tr(q-1,4)<tr(q,4)
            text(tr(q,1)-10, tr(q,2)-10, num2str(tr(q,4)),
'color',rgb(strcat(color_traj(mod(tr(q,4),8)+1))));
            plot (tr(q,1), tr(q,2),
strcat('x',color_traj(mod(tr(q,4),8)+1)));

        end
    else
        text(tr(1,1)-15, tr(1,2)-15, num2str(tr(1,4)),
'color',rgb(strcat(color_traj(mod(tr(q,4),8)+1))));
    end
end

hold off

saveas(gcf, 'final-trajs-dots.fig', 'fig');

fprintf('\nFound a total of %d datapoints in %d frames.\n',nr,
filenumber);
fprintf('One last exclusion criterion entails removing the non-moving
cells.\n');
```

```
fprintf(logfile,'\nFound a total of %d datapoints in %d frames.\n',nr,
filenumber);
fprintf(logfile,'One last exclusion criterion entails removing the non-
moving cells.\n');
readout=input(strcat('Exclude trajectories with rmsd < BP_high/2=',
num2str(BP_high/2),' or enter new exclusion threshold (0 if no rmsd-
filtering desired): '));
fprintf(logfile,'Exclude trajectories with rmsd < BP_high/2=%d or enter
new exclusion threshold: %d\n', BP_high/2, readout);
if mean(size(readout))==0
    readout=BP_high/2;
end
% weed out those trajectories with less than the set rmsd.
q=1;
[nr nc]=size(tr);

while q<nr
    q_init=q;
    while tr(q,4)==tr(q+1,4) % still in the same trajectory
        q=q+1;
    end
    q_fin=q;
    % calculate rmsd(r)=sqrt(rmsd(x)^2+rmsd(y)^2) for each trajectory.
    rmsd=sqrt(std(tr(q_init:q_fin,1))^2+std(tr(q_init:q_fin,1))^2);
    if rmsd < readout
        tr(q_init:nr+q_init-q_fin-1,:)=tr(q_fin+1:nr,:);
        nr=nr+q_init-q_fin-1;
        q=q_init-1;
    end;
q=q+1;
end

tr=tr(1:nr,:);

fprintf('\nFollowing this, %d datapoints in %d frames, all output in
./trajectories-preprocessed.txt\n', nr, filenumber);
fprintf('File format: (x) (y) (frame #) (trajectory index)\n');
fprintf('You are now ready for the post-processing step.\n');
fprintf(logfile,'\nFollowing this, %d datapoints in %d frames, all
output in ./trajectories-preprocessed.txt\n', nr, filenumber);
fprintf(logfile,'File format: (x) (y) (frame #) (trajectory index)\n');
fprintf(logfile,'You are now ready for the post-processing step.\n');
tr=[tr; [0 0 0 0]];

% and plot after weeding

figure
a = double(imread(files(1).name));
file_dim=size(size(a));
if file_dim(2)==3
    a=mean(a,3);
end;

if invert
    a=255-a;
end

% all-dot format
colormap('gray'), imagesc(a);
hold on
```

```
color_traj='bgrcmykw';

for q=1:nr
    plot (tr(q,1), tr(q,2), strcat('.',color_traj(mod(tr(q,4),8)+1)) ,
'markersize', 4);
    if q~=1
        if tr(q-1,4)<tr(q,4)
            text(tr(q,1)-10, tr(q,2)-10, num2str(tr(q,4)),
'color',rgb(strcat(color_traj(mod(tr(q,4),8)+1))));
            plot (tr(q,1), tr(q,2),
strcat('x',color_traj(mod(tr(q,4),8)+1)));

        end
    else
        text(tr(1,1)-15, tr(1,2)-15, num2str(tr(1,4)),
'color',rgb(strcat(color_traj(mod(tr(q,4),8)+1))));
    end
end

hold off

saveas(gcf, 'final-trajs-dots-rmsd-excluded.fig', 'fig');


% save data to ascii file
save -ascii 'trajectories-preprocessed.txt' tr;
fclose(logfile);

end
```

6. *data_analysis_exp.m: MATLAB code*

```
function data_analysis_exp = data_analysis_exp()

%this function accepts *experimental and experimental-like* data (i.e.
%found in a file called trajectories-picked.txt and with the following
%format: (x) (y) (frame #) (trajectory #)

%The data could come from either real data (gotten from track_movie.m) or
%from simulated data gotten from OUx.m or OUphi.m etc.

% The program will then calculate u(t) values from that and process them
% accordingly.


close all
clc

% Conversion parameters
dt=2; % Amount of time between successive frames,min
um2pix=0.977; % size of a pixel in um.

vmax=10; % um/min, may need to change this
v_bin_size=0.5; % um/min, may need to change this

rsd_filter=15; %um, as per David's paper. That's about 1/2 cell diameter.

% Determine N from the trajectory file as the length of the longest
% trajectory.

logfile=fopen('logfile.txt','w');


fprintf('***********************************************************************\n');
fprintf('********************** EXPERIMENTAL DATA ANALYSIS **********************\n');
fprintf('***************************************************************************\n\n\n'
);

fprintf(logfile,'**************************************************************************
*\n');
fprintf(logfile,'*********************** EXPERIMENTAL DATA ANALYSIS
***********************\n');
fprintf(logfile,'**************************************************************************
*\n\n\n');

fprintf('DATE: %s\n\n', datestr(now));
fprintf(logfile,'DATE: %s\n\n', datestr(now));

filename=input('Enter the name of the file containing the selected trajectories: ', 's');
fprintf(logfile,'Enter the name of the file containing the selected trajectories: %s\n',
filename);

tr=load(filename);
N=0;
NN=1;
[trajs_len nc]=size(tr(:,1));
for i=1:trajs_len-1
    % velocities
    if (tr(i,4)~=tr(i+1,4)) % change in index number
        NN=NN+1;
       if tr(i,3)>N
           N=tr(i,3);
       end
    end
end
if tr(trajs_len,3)>N
    N=tr(i,3);
end

fprintf('Found %d trajectories and %d data points. Longest trajectory spans %d
frames\n\n', NN, trajs_len, N);
fprintf(logfile,'Found %d trajectories and %d data points. Longest trajectory spans %d
frames\n\n', NN, trajs_len, N);

avg_drift=zeros(N-1,5);
avg_xy_drift=zeros(N+1,2);

if exist('./trajectories-fixed.txt')==2
    trf=load('trajectories-fixed.txt');
    % avg_drift(23,1) is x coord of 23rd entry; (23,2) is y; tr(23,3) # of entries in the
average.
    %4th column is delta x^2, 5th is delta y^2

figure
hold on
color_traj='bgrcmykw';
for q=1:size(trf(:,1))
    trf(q,1)=trf(q,1)-avg_xy_drift(trf(q,3),1);
    tr(q,2)=trf(q,2)-avg_xy_drift(trf(q,3),2);
    plot (trf(q,1), trf(q,2), strcat('.',color_traj(mod(trf(q,4),8)+1)) , 'markersize',
4);
    if q~=1
        if trf(q-1,4)<trf(q,4)
            text(trf(q,1)-15, trf(q,2)-15,
num2str(trf(q,4)),'color',rgb(strcat(color_traj(mod(trf(q,4),8)+1))));
        end
    else
        text(trf(1,1)-15, trf(1,2)-15,
num2str(trf(1,4)),'color',rgb(strcat(color_traj(mod(trf(q,4),8)+1))));
    end
end
title('Trajectories-fixed that will be used to assess stage drift.');

% Convert pixels into um and frames into minutes.

trf(:,1:2)=trf(:,1:2)*um2pix; % x,y in um

    for i=2:size(trf(:,1))
        if (trf(i,4)==trf(i-1,4))  % still same trajectory
            if trf(i,3)==trf(i-1,3)+1  % and only 1 time step away (i.e. continuous)
                avg_drift(trf(i,3)-1,1)=(trf(i,1)-trf(i-1,1))+avg_drift(trf(i,3)-1,1);
                avg_drift(trf(i,3)-1,2)=(trf(i,2)-trf(i-1,2))+avg_drift(trf(i,3)-1,2);
                avg_drift(trf(i,3)-1,4)=((trf(i,1)-trf(i-1,1)))^2+avg_drift(trf(i,3)-1,4);
                avg_drift(trf(i,3)-1,5)=((trf(i,2)-trf(i-1,2)))^2+avg_drift(trf(i,3)-1,5);
                avg_drift(trf(i,3)-1,3)=avg_drift(trf(i,3)-1,3)+1;
            end
        end
    end

    % now the actual average velocities
    avg_drift(:,1)=avg_drift(:,1)./avg_drift(:,3);
    avg_drift(:,2)=avg_drift(:,2)./avg_drift(:,3);
    avg_drift(:,5)=avg_drift(:,5)./avg_drift(:,3);
    avg_drift(:,4)=avg_drift(:,4)./avg_drift(:,3);

    figure
    plot(1:N-1, avg_drift(:,1), 'o-');
    title ('stage x drift correction');
    figure
    plot(1:N-1, avg_drift(:,2), 'o-');
    title ('stage  y drift correction');

    % and replace columns 4 and 5 with weights = 1/std^2 now (N-1)/N/(x^2mean - xmean^2)
    avg_drift(:,5)=(((avg_drift(:,3)-1)./avg_drift(:,3))./(avg_drift(:,5)-
avg_drift(:,2).^2));
    avg_drift(:,4)=(((avg_drift(:,3)-1)./avg_drift(:,3))./(avg_drift(:,4)-
avg_drift(:,1).^2));

    % tr format x y t traj#
    % subtract strage drift correction from data and plot trajs on the original
    % frame #1

    % note that if there is only one trj in the average, weight is zero

    % binned 5/bin: use weights: xmean = sum(wi*xi)/sum(wi)
    bin_size=5;

    bin_avg=zeros(floor((N-1)/bin_size),4);
    for i = 1:floor((N-1)/bin_size)
        bin_avg(i,1)=(avg_drift(i*bin_size-(bin_size-
1):i*bin_size,1))'*(avg_drift(i*bin_size-(bin_size-
1):i*bin_size,4))/sum(avg_drift(i*bin_size-(bin_size-1):i*bin_size,4));
        bin_avg(i,2)=(avg_drift(i*bin_size-(bin_size-
1):i*bin_size,2))'*(avg_drift(i*bin_size-(bin_size-
1):i*bin_size,5))/sum(avg_drift(i*bin_size-(bin_size-1):i*bin_size,5));
        % and corresponding weighted mean std: 1/sqrt(sum(wi))
        bin_avg(i,3)=sqrt(1./sum(avg_drift(i*bin_size-(bin_size-1):i*bin_size,4)));
        bin_avg(i,4)=sqrt(1./sum(avg_drift(i*bin_size-(bin_size-1):i*bin_size,5)));
    end


    % and now fit to a smooth function to account for actual stage drift.
    % namely Ax - Bx exp(-dt/T) for x_drift, similar for y_drift.

    % an alternative is to fit 40-84 to a flat line (Ax and Ay) and then
    % fix those:
    Ax=(avg_drift(floor(N/2):N-1,1))'*(avg_drift(floor(N/2):N-
1,4))/sum(avg_drift(floor(N/2):N-1,4));
    Ay=(avg_drift(floor(N/2):N-1,2))'*(avg_drift(floor(N/2):N-
1,5))/sum(avg_drift(floor(N/2):N-1,5));

    parameterx = lsqnonlin(@mycurvex,[Ax 1 10],[Ax-0.01 -Inf -Inf],[Ax+0.01 Inf
Inf],[],0:floor(N/3),avg_drift(1:floor(N/3)+1,1)');
        parameterx
    parametery = lsqnonlin(@mycurvey,[Ay 1 10],[Ay-0.01 -Inf -Inf],[Ay+0.01 Inf
Inf],[],0:floor(N/3),avg_drift(1:floor(N/3)+1,2)');
        parametery

    figure
    plot(floor(bin_size/2):bin_size:floor((N-1)/bin_size)*bin_size-(floor(bin_size/2)),
bin_avg(:,1), 'o-', floor(bin_size/2):bin_size:floor((N-1)/bin_size)*bin_size-
floor(bin_size/2), bin_avg(:,1)+bin_avg(:,3), '-r', floor(bin_size/2):bin_size:floor((N-
1)/bin_size)*bin_size-floor(bin_size/2), bin_avg(:,1)-bin_avg(:,3), '-r', 1:N-1,
parameterx(1)-parameterx(2).*exp(-(1:N-1) / parameterx(3)), '-g');
    figure
    plot(floor(bin_size/2):bin_size:floor((N-1)/bin_size)*bin_size-(floor(bin_size/2)),
bin_avg(:,2), 'o-', floor(bin_size/2):bin_size:floor((N-1)/bin_size)*bin_size-
floor(bin_size/2), bin_avg(:,2)+bin_avg(:,4), '-r', floor(bin_size/2):bin_size:floor((N-
1)/bin_size)*bin_size-floor(bin_size/2), bin_avg(:,2)-bin_avg(:,4), '-r', 1:N-1,
parametery(1)-parametery(2).*exp(-(1:N-1) / parametery(3)), '-g');
    title ('stage y drift correction binned');


    % Here's the integral:
    for i=1:N
        avg_xy_drift(i,1)=(parameterx(1)*(i-1)-
parameterx(3)*parameterx(2)+parameterx(3)*exp(-(i-1) / parameterx(3))*parameterx(2));
        avg_xy_drift(i,2)=(parametery(1)*(i-1)-
parametery(3)*parametery(2)+parametery(3)*exp(-(i-1) / parametery(3))*parametery(2));
    end

    avg_xy_drift_noise=zeros(85,2);
    for i=2:N
        avg_xy_drift_noise(i,1)=avg_xy_drift_noise(i-1,1)+avg_drift(i-1,1);
        avg_xy_drift_noise(i,2)=avg_xy_drift_noise(i-1,2)+avg_drift(i-1,2);
    end

    figure
    plot(avg_xy_drift(:,1),avg_xy_drift(:,2), '-
g',avg_xy_drift_noise(:,1),avg_xy_drift_noise(:,2), 'or');
    title ('stage drift correction - fitted');

avg_xy_drift=zeros(N,2);
for i=1:N-1
    avg_xy_drift(i+1)=avg_xy_drift(i)+avg_drift(i);
end

end

% back to our dataset.

figure
hold on
color_traj='bgrcmykw';
for q=1:size(tr(:,1))
    tr(q,1)=tr(q,1)-avg_xy_drift(tr(q,3),1);
    tr(q,2)=tr(q,2)-avg_xy_drift(tr(q,3),2);
    plot (tr(q,1), tr(q,2), strcat('.',color_traj(mod(tr(q,4),8)+1)) , 'markersize', 4);
    if q~=1
        if tr(q-1,4)<tr(q,4)
            text(tr(q,1)-15, tr(q,2)-15,
num2str(tr(q,4)),'color',rgb(strcat(color_traj(mod(tr(q,4),8)+1))));
        end
    else
        text(tr(1,1)-15, tr(1,2)-15,
num2str(tr(1,4)),'color',rgb(strcat(color_traj(mod(tr(q,4),8)+1))));
    end
end
plothandle=gca;
set(plothandle, 'Color', [0,0,0] );
title('Trajectories-picked after stage drift correction, where available');
xlabel('x axis - pixels');
ylabel('y axis - pixels');

% root-square displacememt-based filtering where necessary.
% criterion keep only trajs with sqrt(max((r_j-r_0)^2))>rsd_filter
% "excluding cells with rms displacement that never exceeded 20 um for
% NHDF"

% first calculate the rsd from 1st point for each trajectory, then update
% tr so as to reflect that change.

while q<trajs_len
    q_init=q;
    while tr(q,4)==tr(q+1,4) % still in the same trajectory
        q=q+1;
    end
    q_fin=q;
    % determine largest rsd for each trajectory.
    max_rsd=0
    for j=q_init:q_fin
        if max_rsd<sqrt((tr(j,1)-tr(q_init,1))^2+(tr(j,2)-tr(q_init,2))^2)
            max_rsd=sqrt((tr(j,1)-tr(q_init,1))^2+(tr(j,2)-tr(q_init,2))^2);
        end
    end
    if max_rsd < rsd_filter
        tr(q_init:trajs_len+q_init-q_fin-1,:)=tr(q_fin+1:trajs_len,:);
        trajs_len=trajs_len+q_init-q_fin-1;
        q=q_init-1;
    end;
q=q+1;
end

tr=tr(1:trajs_len,:);

N=0;
NN=1;
for i=1:trajs_len-1
    % velocities
    if (tr(i,4)~=tr(i+1,4)) % change in index number
        NN=NN+1;
       if tr(i,3)>N
           N=tr(i,3);
       end
    end
end
if tr(trajs_len,3)>N
    N=tr(i,3);
end

fprintf('After rsd filter of %d um, found %d trajectories and %d data points. Longest
trajectory spans %d frames\n\n', rsd_filter,NN, trajs_len, N);
fprintf(logfile, 'After rsd filter of %d um, found %d trajectories and %d data points.
Longest trajectory spans %d frames\n\n', rsd_filter,NN, trajs_len, N);

fprintf('This analysis uses secant velocities and accelerations.\n\n');
fprintf(logfile, 'This analysis uses secant velocities and accelerations.\n\n');


figure
hold on
color_traj='bgrcmykw';

for q=1:size(tr(:,1))
    plot (tr(q,1), tr(q,2), strcat('.',color_traj(mod(tr(q,4),8)+1)) , 'markersize', 4);
    if q~=1
        if tr(q-1,4)<tr(q,4)
            text(tr(q,1)-15, tr(q,2)-15,
num2str(tr(q,4)),'color',rgb(strcat(color_traj(mod(tr(q,4),8)+1))));
        end
    else
        text(tr(1,1)-15, tr(1,2)-15,
num2str(tr(1,4)),'color',rgb(strcat(color_traj(mod(tr(q,4),8)+1))));
    end
end
plothandle=gca;
set(plothandle, 'Color', [0,0,0] );
title('Trajectories-picked after stage drift correction and rsd filtering, where
available');
xlabel('x axis - um');
ylabel('y axis - um');

% and then since we're at this anyway, plot Fuerth's formula as per our
% manuscript for a guess on std_pos mostly.

fuerth=zeros(N,4);
for i=1:N
    fuerth(i,1)=(i-1)*dt;
end

% now pass through the entire tr variable and collect data
for i=1:trajs_len-1
        j=i;
        while ((j<trajs_len) && (tr(j,4)==tr(j+1,4))) % go up to the end of this
contiguous section
            % simply leave out the very last entry in tr - it's absent anyway in
            % real data, and won't make much of a difference in the
            % simulated one
            fuerth(j-i+1,2)=fuerth(j-i+1,2)+(tr(j,1)-tr(i,1))^2+(tr(j,2)-tr(i,2))^2;
            fuerth(j-i+1,3)=fuerth(j-i+1,3)+((tr(j,1)-tr(i,1))^2+(tr(j,2)-tr(i,2))^2)^2;
            fuerth(j-i+1,4)=fuerth(j-i+1,4)+1;
            j=j+1;
        end
end

fuerth(:,2)=fuerth(:,2)./fuerth(:,4);
fuerth(:,3)=sqrt((fuerth(:,3)-fuerth(:,2).^2)./(fuerth(:,4).*(fuerth(:,4)-1)));

fuerth=fuerth(2:floor(N/2),:);

figure
errorbar(fuerth(:,1), (fuerth(:,2)), fuerth(:,3), '.b');
title('<(r(t)-r(0))^2> vs Fuerth''s formula');

save -ascii 'fuerth.txt' fuerth;


% plot a_perp and a_parallel vs speed. Only successive steps are considered
% for v and a calculations.

[trajs_len nc]=size(tr(:,1));

tr(:,1:2)=tr(:,1:2)*um2pix; % x,y in um

newton=zeros(trajs_len,7);
% format: index/time/vx/vy/ax/ay/validity (0 is nothing in there, 1 if only
% v's, 2 if both v's and a's).
for i=1:trajs_len-1
    % velocities u; note I call it u(i) but it really is u(i+1/2)
    if (tr(i,4)==tr(i+1,4)) && (tr(i,3)==tr(i+1,3)-1) % still same cell and only 1 time
step away (i.e. continuous)
       newton(i,3)=(tr(i+1,1)-tr(i,1))/dt;
       newton(i,4)=(tr(i+1,2)-tr(i,2))/dt;
       newton(i,7)=1;
    end
    % accelerations a(i)=(u(i+1/2)-u(i-1/2))/dt
      if (i>1) && (tr(i,4)==tr(i+1,4)) && (tr(i,4)==tr(i-1,4)) && (tr(i,3)==tr(i+1,3)-1)
&& (tr(i,3)==tr(i-1,3)+1)
       newton(i-1,5)=(newton(i,3)-newton(i-1,3))/dt;
       newton(i-1,6)=(newton(i,4)-newton(i-1,4))/dt;
       newton(i-1,7)=2;
    end
    newton(i,1)=tr(i,4);
    newton(i,2)=tr(i,3)*dt;
end



% average of velocities for drift considerations
uxmean=zeros(N-1,1);
uymean=zeros(N-1,1);
counts=zeros(N-1,1);
std_uxmean=zeros(N-1,1);
std_uymean=zeros(N-1,1);
for i=1:N-1
    [counts nr]=size(newton(find(tr(:,3)==i),3));
    if counts == 0
        std_uxmean(i)=Inf;
        std_uymean(i)=Inf;
        uxmean(i)=0;
        uymean(i)=0;
    else
        std_uxmean(i)=std(newton(find(tr(:,3)==i),3))./counts;
        std_uymean(i)=std(newton(find(tr(:,3)==i),4))./counts;
        uxmean(i)=mean(newton(find(tr(:,3)==i),3));
        uymean(i)=mean(newton(find(tr(:,3)==i),4));
    end
end

figure
plot([1:N-1]*dt, uxmean, 'or', 'markersize', 5);
hold on
errorbar([1:N-1]*dt, uxmean, std_uxmean);
title ('average u_x of all cells in a frame');
xlabel('Time (min)');
ylabel ('u_x mean, um/min');

figure
plot([1:N-1]*dt, uymean, 'or', 'markersize', 5);
```

```
hold on
errorbar([1:N-1]*dt, uymean, std_uymean);
title ('average u_y of all cells in a frame');
xlabel('Time (min)');
ylabel ('u_y mean, um/min');

% Now averaging all frames

avg_ux=0;
avg_uy=0;
    for j=1:N-1
        avg_ux=uxmean(j)/(std_uxmean(j)^2)+avg_ux;
        avg_uy=uymean(j)/(std_uymean(j)^2)+avg_uy;
    end
    avg_ux=avg_ux/sum(1./(std_uxmean.^2));
    avg_uy=avg_uy/sum(1./(std_uymean.^2));
    wx=sum(1./(std_uxmean.^2));
    wx2=sum(1./(std_uxmean.^4));
    wy=sum(1./(std_uxmean.^2));
    wy2=sum(1./(std_uymean.^4));
    avg_std_ux=sqrt(wx/wx2);
    avg_std_uy=sqrt(wy/wy2);

fprintf('Time-averaged ux_mean = %7.4f ± %6.4f um/min\n', avg_ux,avg_std_ux);
fprintf('Time-averaged uy_mean = %7.4f ± %6.4f um/min\n', avg_uy,avg_std_uy);
fprintf('\nThe numbers above should be zero within their respective errorbars as that
indicates lack of stage drift.\n\n');

fprintf(logfile,'Time-averaged ux_mean = %7.4f ± %6.4f um/min\n', avg_ux,avg_std_ux);
fprintf(logfile,'Time-averaged uy_mean = %7.4f ± %6.4f um/min\n', avg_uy,avg_std_uy);
fprintf(logfile,'\nThe numbers above should be zero within their respective errorbars as
that indicates lack of stage drift.\n\n');

% Actual data analysis starts here


valid_newton_indices=find(newton(:,7)==2);
%format newtwon_index/speed/a_perp/a_parallel
a2v=zeros(length(valid_newton_indices),4);
a2v(:,1)=valid_newton_indices;
a2v(:,2)=sqrt(newton(a2v(:,1),3).^2+newton(a2v(:,1),4).^2);
% a parallel = a dot v / |v|
a2v(:,4)=(newton(a2v(:,1),3).*newton(a2v(:,1),5)+newton(a2v(:,1),4).*newton(a2v(:,1),6))./
a2v(:,2);
% a perp = proj/z  of a x v / |v|
a2v(:,3)=(newton(a2v(:,1),4).*newton(a2v(:,1),5)-
newton(a2v(:,1),6).*newton(a2v(:,1),3))./a2v(:,2);


%%%%%%%%%%%%%%%%%%%%%% PLOT a_par and a_perp vs. u: scatterplot %%%%%

a_par=-mean(a2v(:,4)./a2v(:,2));

[size_v nc]=size(a2v(:,4));
da_par=std(a2v(:,4)./a2v(:,2))/sqrt(size_v-1);

a_perp=-mean(a2v(:,3)./a2v(:,2));

[size_v nc]=size(a2v(:,3));
da_perp=std(a2v(:,3)./a2v(:,2))/sqrt(size_v-1);


% a perp versus speed
figure
plot(a2v(:,2), a2v(:,3), '.', 'markersize', 3);
title (strcat('a_p_e_r_p vs speed, ', num2str(length(a2v(:,1))), ' datapoints'));
xlabel('speed (um/min)');
ylabel ('a_p_e_r_p, um/min^2');


% a parallel versus speed
figure
plot(a2v(:,2), a2v(:,4), '.', 'markersize', 3);
title (strcat('a_p_a_r_a_l_l_e_l vs speed, ', num2str(length(a2v(:,1))), ' datapoints'));
xlabel('speed (um/min)');
ylabel ('a_p_a_r_a_l_l_e_l, um/min^2');
```

```
% bin a_perp and a_par vs. speed
% make velocity bins of v_bin_size um/min, from 0 to vmax
% format:
% speed/mean_a_perp/sig_mean_a_perp/mean_a_par/sig_mean_a_par

vmax=min(vmax,max(a2v(:,2)));
mean((a2v(:,2)))


binned_a2v=zeros(floor(vmax/v_bin_size),5);
for i=1:floor(vmax/v_bin_size)
    binned_a2v(i,1)=(i-1)*v_bin_size+v_bin_size/2; % middle of bin
    % find all with speed in the i-1,i range
    vec=find(a2v(:,2).^2-(2*i-1)*v_bin_size.*a2v(:,2)+(i^2-i)*v_bin_size^2<0);
    [nr nc]=size(vec)
    binned_a2v(i,2)=mean(a2v(vec,3));
    binned_a2v(i,3)=std(a2v(vec,3))/sqrt(nr);
    binned_a2v(i,4)=mean(a2v(vec,4));
    binned_a2v(i,5)=std(a2v(vec,4))/sqrt(nr);
    binned_a2v(i,6)=nr;
end


% a versus speed

figure
errorbar(binned_a2v(:,1), binned_a2v(:,2), binned_a2v(:,3), '.g');
hold on
errorbar(binned_a2v(:,1), binned_a2v(:,4), binned_a2v(:,5), '.r');
plot([0:vmax], a_perp.*[0:vmax], '-g');
plot([0:vmax], (a_perp-da_perp).*[0:vmax], '-g');
plot([0:vmax], (a_perp+da_perp).*[0:vmax], '-g');
plot([0:vmax], -a_par.*[0:vmax], '-r');
plot([0:vmax], -(a_par-da_par).*[0:vmax], '-r');
plot([0:vmax], -(a_par+da_par).*[0:vmax], '-r');
title('bin a_p_e_r_p (green) and a_p_a_r_a_l_l_e_l (red) vs u')
xlabel('u (um/min)');
ylabel ('bin a, um/min^2');

% mean std/std of mean std
% speed/mean_sig_a_perp/sig_mean_sig_a_perp/mean_sig_a_par/sig_mean_sig_a_par
binned_std_a2v=zeros(floor(vmax/v_bin_size),5);
for i=1:floor(vmax/v_bin_size)
    binned_std_a2v(i,1)=(i-1)*v_bin_size+v_bin_size/2; % middle of bin
    % find all with speed in the i-1,i range
    vec=find(a2v(:,2).^2-(2*i-1)*v_bin_size.*a2v(:,2)+(i^2-i)*v_bin_size^2<0);
    [nr nc]=size(vec);
    rmsd_a2v=a2v(vec,:);
    rmsd_a2v(:,3)=abs(rmsd_a2v(:,3)-mean(rmsd_a2v(:,3)));
    rmsd_a2v(:,4)=abs(rmsd_a2v(:,4)-mean(rmsd_a2v(:,4)));
    binned_std_a2v(i,2)=std(a2v(vec,3));
    binned_std_a2v(i,3)=sqrt(sum((rmsd_a2v(:,3)-binned_std_a2v(i,2)).^2)/((nr-1)*nr));
    binned_std_a2v(i,4)=std(a2v(vec,4));
    binned_std_a2v(i,5)=sqrt(sum((rmsd_a2v(:,4)-binned_std_a2v(i,4)).^2)/((nr-1)*nr));
end

% velocity autocovariance function: take all nth order "distances" into
% account
% reminder newton format:index/time/vx/vy/ax/ay/validity (0 is nothing in there, 1 if only
% v's, 2 if both v's and a's).

% phi format:
% delta_time /phi(delta_t)/std_phi/# datapoints

phi=zeros(N,4);
for i=1:N
    phi(i,1)=(i-1)*dt;
end

% now pass through the entire newton variable and collect data into phi
for i=1:trajs_len-1
    if newton(i,7)>0 % i.e. there's good velocities
        j=i;
        % calculate drift for data decorrelation purposes
        while ((j<trajs_len) && (newton(j,1)==newton(j+1,1))) && (newton(j,7)>0) % go up
to the end of this contiguous section
```

```
            % simply leave out the very last entry in tr - it's absent anyway in
            % real data, and won't make much of a difference in the
            % simulated one
            phi(j-i+1,2)=phi(j-i+1,2)+newton(i,3)*newton(j,3)+newton(i,4)*newton(j,4);
            phi(j-i+1,3)=phi(j-i+1,3)+(newton(i,3)*newton(j,3)+newton(i,4)*newton(j,4))^2;
            phi(j-i+1,4)=phi(j-i+1,4)+1;
            j=j+1;
        end
    end
end

phi(:,2)=phi(:,2)./phi(:,4);
phi(:,3)=sqrt((phi(:,3)-phi(:,2).^2)./(phi(:,4).*(phi(:,4)-1)));


figure
errorbar(phi(:,1), phi(:,2), phi(:,3), '.b');
hold on

ph=phi(3:floor(N/2),1:3);
save -ascii 'phi_Origin.txt' ph;

%%% bin phi (exclude phi(0) and phi(1) b/c those are off due to pos noise)
%%% bin in sets of 10.
% weighted binning.

wbinned_phi=zeros(floor(N/10),4);
wght=1./phi(:,3).^2;

wbinned_phi(1,1)=mean(phi(3:10,1));
wbinned_phi(1,2)=sum(wght(3:10).*phi(3:10,2))/sum(wght(3:10));
wbinned_phi(1,3)=sqrt(1./sum(wght(3:10)));

for i=2:floor(N/10)
    wbinned_phi(i,1)=mean(phi(10*i-9:10*i,1));
    wbinned_phi(i,2)=sum(wght(10*i-9:10*i).*phi(10*i-9:10*i,2))/sum(wght(10*i-9:10*i));
    wbinned_phi(i,3)=sqrt(1./sum(wght(10*i-9:10*i)));
end

fprintf('Weight-binned phi_u_posn(t): (time [dt]) (phi [pix^2/dt]) (std_phi [pix^2/dt]:')
wbinned_phi(:,1:3)

figure
plot(wbinned_phi(:,1), wbinned_phi(:,2), 'o-r', 'markersize', 3);
hold on
plot(wbinned_phi(:,1), wbinned_phi(:,2)+wbinned_phi(:,3), '-r');
plot(wbinned_phi(:,1), wbinned_phi(:,2)-wbinned_phi(:,3), '-r');
title('wght-binned velocity autocovariance function and error bars');

save -ascii 'phi_Origin_binned.txt' wbinned_phi;

% uwLSQ fit of a+bt to ln phi_u as a rough guess for
% P and phi1.

phi_mean=zeros(N,1);
for i=7:N
    phi_mean(i-2)=mean(phi(3:i,2));
end


figure
errorbar(phi(:,1), log(phi(:,2)), (log(phi(:,2)+phi(:,3))-log(phi(:,2)-phi(:,3)))/2,
'.b');
hold on
plot(phi(:,1), log(exp(beta(1))*exp(-phi(:,1)/(-1/beta(2)))), '-r', 'markersize', 3);
plot([(N_Pest-1)*dt (N_Pest-1)*dt], [log(phi(N_Pest,2)) log(phi(3,2))], '-r');
title('loglin experimental phi\_u\_posn(t) with error bars and uwLSQ fit to phi');
xlabel('Time/min');
ylabel('Ln velocity autocovariance)');
xlim([-dt/2 N*dt]);


% also do the speed histogram and speed^2.
% format vx/vy

speeds=newton(find(newton(:,7)>0), 3:4);
figure
```

```
counts=histc(sqrt(speeds(:,1).^2+speeds(:,2).^2), ((0:30)-.5));
% this counts the number of observations in each bin
% each bin contains one integer
bar((0:30), counts, .3, 'b', 'EdgeColor', 'b')
title('speed histogram');
xlabel('speed um/min');
ylabel('Counts');



figure
counts=histc((speeds(:,1).^2+speeds(:,2).^2), ((0:100)-.5));
% this counts the number of observations in each bin
% each bin contains one integer
bar((0:100), counts, .3, 'b', 'EdgeColor', 'b')
title('speed^2 histogram');
xlabel('speed^2 um^2/min^2');
ylabel('Counts');

fclose(logfile);


    function err = mycurvex(parameter,real_x, real_y)
        fit = parameter(1)-parameter(2).*exp(-real_x / parameter(3));
        err = fit - real_y;

        % weight the error according to the |WEIGHT| vector
        err_weighted = err.*avg_drift(1:size(real_x),4)';
        err = err_weighted;
    end
    function err = mycurvey(parameter,real_x, real_y)
        fit = parameter(1)-parameter(2).*exp(-real_x / parameter(3));
        err = fit - real_y;

        % weight the error according to the |WEIGHT| vector
        err_weighted = err.*avg_drift(1:size(real_x),5)';
        err = err_weighted;
    end
    function nc = nocomplex(x)
        nc=x
        for i=1:size(x)
            if imag(x(i))~=0
            nc(i)=Inf
            end
        end
    end
end
```