Automated Analysis of Time Lapse Microscopy Images

Andrey Kan

Submitted in total fulfilment of the requirements of the degree of Doctor of Philosophy

January, 2013
Abstract

Cells are the building blocks of life, and time lapse microscopy is a powerful way to study cells. Automated video acquisition and analysis of cells opens unprecedented opportunities, ranging from building novel mathematical models supported by rich data to automated drug screening. Unfortunately, accurate and completely automated analysis of cell images is a difficult task. Therefore human intervention is often required, for example, for tuning of segmentation and tracking algorithms or correcting the results of automated analysis. In this thesis, we aim to reduce the amount of manual work required, while preserving the accuracy of analysis.

Two key tasks in automated analysis are cell segmentation and tracking. Segmentation is the process of locating cell outlines in cell images, while tracking refers to establishing cell identities across subsequent video frames. One of the main challenges of automated analysis is the substantial variability in cell appearance and dynamics across different videos and even within a single video. For example, there can be a few rapidly moving cells in the beginning of a video and a large number of cells stuck in a clump by the end of the video. Such variation has resulted in a large variety of cell segmentation and tracking algorithms.

There has been a large body of work on automated cell segmentation and tracking. However, many methods make specific assumptions about cell morphology or dynamics, or involve a number of parameters that a user needs to set manually. This hampers the applicability of such methods across different videos. We first develop portable cell semi-segmentation and segmentation algorithms, where portability is achieved by using a flexible cell descriptor function. We then develop a novel cell tracking algorithm that has only one
parameter, and hence can be easily adopted to different videos. Furthermore, we present a parameter-free variation of the algorithm. Our evaluation on real cell videos demonstrates that our algorithms are capable of achieving accurate results and outperforming other existing methods.

Even the most sophisticated cell tracking algorithms make errors. A user can be required to manually review the tracking results and correct errors. To this end, we propose a semi-automated tracking framework that is capable of identifying video frames that are likely to contain errors. The user can then look only into these frames and not into all video frames. We find that our framework can significantly reduce the amount of manual work required to review and correct tracking results.

Furthermore, in different videos, the most accurate results can be obtained by different methods and different parameter settings. It is often not clear which method should be chosen for a particular video. We address this problem with a novel method for ranking cell tracking systems without manual validation. Our method is capable of ranking cell trackers according to their fitness to a particular video, without the need for manual collection of the ground truth tracks. We simulate practical tracking scenarios and confirm the feasibility of our method.

Finally, as an example of a biological assay, we consider evaluating the locomotion of Plasmodium parasites (that cause malaria) with application to automated anti-malaria drug screening. We track live parasites in a matrigel medium and develop a numerical description of parasite tracks. Our experiments show that this description captures changes in the locomotion in response to treatment with the toxin Cytochalasin D. Therefore our description can form a basis for automated drug screening, where various treatments are applied to different cell populations by a robot, and the resulting tracks are evaluated quantitatively.

In summary, our thesis makes six major contributions highlighted above. These contributions can reduce the amount of manual work in cell image analysis, while achieving highly accurate results.
Declaration

This is to certify that

1. the thesis comprises only my original work towards the degree of Doctor of Philosophy except where indicated in the Preface,

2. due acknowledgment has been made in the text to all other material used,

3. the thesis is fewer than 80,000 words in length, exclusive of tables, maps, bibliographies and appendices.

____________________________________

Andrey Kan
Preface

This thesis could not be possible without the help of collaborators, and, in this section, I would like to acknowledge the researchers and organizations who directly contributed to the projects described in this thesis. I declare that I am the primary author for each of the thesis chapters and that I have contributed > 50% in each chapter.

I would like to thank John Markham, Mark Dowling, Cameron Wellard and Phil Hodgkin of the Hodgkin Lab at the Walter and Eliza Hall Institute (WEHI) for recording and sharing cell videos, and providing their advice regarding the biology of the processes captured in the videos. I am grateful to Daniel Day from the Center for Micro-Photonics at Swinburne University of Technology for supplying the 125 $\mu$m microgrids used for recording some of the cell videos.

Furthermore, I would like to thank Rajib Chakravorty of National ICT Australia, John Markham of WEHI, and James Bailey and Christopher Leckie of the University of Melbourne, for discussions and advice on the design of the methods and evaluation strategies presented in the thesis.

I am also indebted to Khuloud Jaqaman from Harvard Medical School for providing advice regarding the u-track software and Zhaozheng Yin from Carnegie Mellon University for sharing the videos of wound healing assays.

The project described in Chapter 8 has been performed in collaboration with the Baum Lab at WEHI, and I would like to thank Yan Hong Tan of WEHI for conducting the experiments and recording the videos. I am grateful to Kelly Rogers, the Head of the Center for Dynamic Imaging at WEHI, and Jacob Baum, the Head of the Baum Lab at WEHI, for their advice regarding microscopy imaging and the biology of Plasmodium parasites.

The work in this thesis is partially supported by National ICT Australia.
National ICT Australia is funded by the Australian Government’s Backing Australia’s Ability initiative, in part through the Australian Research Council.

Finally, I would like to thank the developers of open source software (particularly in the MATLAB language), as I often used available sub-routines in my algorithm implementations. The code and data that supplement this thesis can be found online at

Acknowledgments

First of all, I would like to sincerely thank my mom and dad, my brother Alexander, my wife Phuong, and my new family members: Ba, Mę, Truc and Hung for being with me, believing in me, and taking care of me on any day, in any month, and any year.

My past few years at the University have been a joyful and fruitful period of my life, sometimes frustrating, sometimes exhausting, but overall extremely rewarding. Meeting great people has constituted a large part of this reward. Here I would like to acknowledge and thank those who supported me during the few past years, and from whom I learnt a lot. These are researchers from different projects in which I have participated. Not every project is reflected in this thesis, but every project had been a great experience that I list here in chronological order.

First of all, I would like to thank my PhD supervisors James Bailey and Christopher Leckie. They have never restricted my freedom in doing research, but they have always been there with good advice. I greatly appreciate their involvement and encouragement. I am also grateful to Jeffrey Chan whom I can call my unofficial supervisor, and with whom we have been having a lot of prolific discussions. James, Christopher and Jeffrey are all from the University of Melbourne.

A large part of my PhD has been spent on the NICTA Cell Tracking project. Many thanks to Rajib Chakravorty of NICTA who introduced me to the project. Furthermore, I am indebted to the collaborating Hodgkin Lab at WEHI. I would like to thank John Markham, Cameron Wellard, Mark Dowling and Phil Hodgkin who shared their cell videos and taught me biology.

I spent three splendid months in Galway, Ireland, doing my research intern-
ship at the Digital Enterprise Research Institute (DERI). Many thanks to Conor Hayes for being such a great research supervisor, and to members of the Unit of Information Mining and Retrieval in DERI for a warm reception. Thank you Jeffrey for pointing out this internship! I am also grateful to Bernie Hogan of the Oxford Internet Institute for insights into sociology.

My job at the Center for Dynamic Imaging in WEHI provided me with a unique opportunity to get my hands on the state-of-the art microscopy equipment. Many thanks to Kelly Rogers for being a liberal boss and for teaching me microscopy.

During my research assistance experience at the University, I have gained one more supervisor and one more colleague. Thank you Rao Kotagiri and Wei Liu for inspiring discussions.

Studying the locomotion of Plasmodium ookinetes have been an amazing project run by the Baum Lab in WEHI. Many thanks to Yan Hong Tan for performing the experiments and exchanging ideas, and to Jacob Baum for leading the project.

At different points in time, I had an honor to have Adrian Pearce as a member of my advisory committee, and to take consultations from Andrew Turpin and Michael Kirley, all of the University of Melbourne.

Finally, I have been heavily involved into an invisible project called “Life in general”. I would like thank my friends from all over the world for great time that we have spent together. Thank you Alexey, Angie, Anirudh, Anton, Andreas, Arif, Christian, Dileban, Giang, Goce, Ho Young, Jens, Jessie, Juan, Katie, Kseniya, Lena, Li, Marina, Masud, Max, Miao, Nikita, Olga, Samantha, Saurav, Sergey, Simone, Sonja, Tien, Vaclav, Yang, Yulia, Yun, and, of course, Xi.
Contents

1 Introduction .................................................. 3
  1.1 Scope of the Thesis .......................................... 4
  1.2 Motivation of our Work ..................................... 5
  1.3 Our Contributions ........................................ 9

2 Background .................................................. 15
  2.1 Optical Microscopy .......................................... 15
    2.1.1 Fluorescence microscopy ................................ 18
    2.1.2 Moving to Digital Video Microscopy ...................... 20
  2.2 Digital Image Processing .................................... 22
    2.2.1 Compensating for Uneven Illumination .................. 23
    2.2.2 Image Smoothing in the Frequency Domain ............... 24
    2.2.3 Image Smoothing in the Spatial Domain .................. 26
    2.2.4 Image Smoothing in a Combined Domain .................. 28
    2.2.5 Stage Shift Removal .................................... 29
  2.3 Cell Segmentation ........................................ 31
    2.3.1 Related Work Overview .................................. 33
    2.3.2 Thresholding ............................................ 35
    2.3.3 Watershed-based Segmentation ............................ 37
    2.3.4 Region Growing and Merging .............................. 39
    2.3.5 Active Contours ......................................... 40
    2.3.6 Summary of Cell Segmentation Methods ................... 43
  2.4 Cell Tracking ............................................. 45
    2.4.1 Measurement Errors ....................................... 46
4.3 Evaluation of SEGEN ........................................... 101
  4.3.1 Segmentation and Detection Accuracy .................. 104
  4.3.2 Detection Accuracy and Portability .................... 106
  4.3.3 Comparison with CLUBS ................................. 108
4.4 Conclusions ............................................... 109

5 NENIA: Nearest Neighbor Based Inter-frame Assignments 111
  5.1 Related Work and Our Method ............................. 112
  5.2 Methods .................................................. 114
    5.2.1 Cell Tracking Preliminaries ......................... 116
    5.2.2 Inter-frame Assignment Problem .................... 121
    5.2.3 The Choice of Gating Distance ...................... 124
    5.2.4 Automated Cell Tracker NENIA ....................... 130
  5.3 Evaluation ............................................... 132
    5.3.1 Cell Videos ........................................ 133
    5.3.2 Evaluation of NENIA ................................ 136
    5.3.3 Evaluation on Synthetic Videos ..................... 141
  5.4 Discussion .............................................. 144
  5.5 Conclusions .............................................. 146

6 SAMTRA: Semi-automated Tracking 149
  6.1 Learning the Probability of Error ....................... 150
  6.2 Evaluation of SAMTRA .................................... 152
    6.2.1 Experimental Setup and Results ..................... 153
    6.2.2 Training SAMTRA .................................... 154
  6.3 Discussion and Conclusion ............................... 155

7 Ranking Cell Trackers 157
  7.1 Related Work ............................................ 159
  7.2 Method Overview, Scope and Preliminaries ............... 160
    7.2.1 Cell Tracking Preliminaries ....................... 162
    7.2.2 Cell Tracking Performance ......................... 164
    7.2.3 CTS Selection Problem .............................. 165
  7.3 Ranking Associations without Ground Truth ............. 165
C  Ranking Cell Tracking Systems  

C.1  Cell Tracking Performance

C.1.1  Measures that require ground truth

C.1.2  Measures that do not require ground truth

C.2  Analysis of the VN-score

C.3  Proof Sketch for Proposition 7.1

C.4  Lengths of Dummy Links

D  Motility of Plasmodium Ookinetes

D.1  Parasites and Mice

D.2  Motility Assay

Bibliography
## List of Figures

1.1  A dividing B lymphocyte cell .............................................. 3
1.2  Time lapse microscopy experiments ..................................... 5
1.3  Challenges of automated cell segmentation .......................... 7

2.1  Overview of the Background chapter ................................. 16
2.2  Bright field and fluorescence microscopy ............................ 17
2.3  Bright field and fluorescence images of B-lymphocytes .......... 18
2.4  Simulated B-lymphocytes in rectangular micro-wells ............ 21
2.5  A bright field image and the compensated image ................ 24
2.6  Image filtering using a rolling ball method ......................... 24
2.7  Image filtering using the Discrete Fourier Transform ........... 26
2.8  Image filtering using a Gaussian kernel ............................ 27
2.9  Cell segmentation .......................................................... 31
2.10 Types of cell segmentation errors ...................................... 32
2.11 Variability in cell appearance across and within videos .......... 33
2.12 Image thresholding ....................................................... 36
2.13 Watershed-based image segmentation ............................... 38
2.14 Watershed-based cell segmentation .................................... 38
2.15 Active contour based image segmentation .......................... 41
2.16 Level-set functions ...................................................... 42
2.17 Cell lineage trees and the corresponding cell events .......... 47
2.18 Measurements correction using a Kalman filter .................. 48
2.19 Tracking via inter-frame assignments ................................. 53
2.20 Sample image analysis pipeline ....................................... 62
2.21 A screen-shot of ImageJ software ..................................... 63
2.22 A screen-shot of Imaris software ........................................... 64

3.1 Sampling and clustering based cell segmentation ......................... 68
3.2 Variability in cell appearance across and within videos ............... 78
3.3 CLUBS detection accuracy as a function of parameters ............... 82
3.4 CLUBS detection accuracy as a function of parameters ............... 83
3.5 CLUBS can be used to group points by wells .......................... 86

4.1 Variability in cell appearance across and within videos ............... 90
4.2 Region enumeration based cell segmentation ............................ 94
4.3 A thresholded image and its distance transform ......................... 95
4.4 Touching cells and a result of SEGEN ................................. 96
4.5 A typical distribution of cell image intensity levels .................. 100
4.6 Performance of SEGEN on individual frames ......................... 107

5.1 Overview of our automated tracker ...................................... 116
5.2 Sample lineage trees that summarize cell events ...................... 117
5.3 A sample assignment between two consecutive frames .............. 122
5.4 Two-round assignment approach ...................................... 123
5.5 Two superimposed video frames ...................................... 125
5.6 Hypothetical curve of the cell tracker performance .................. 126
5.7 Estimation of the number of true links ............................... 128
5.8 Fragments of video frames used in this study ......................... 134
5.9 Sample video frames used in our evaluation .......................... 134
5.10 Performance of NENIA and LJIPDA trackers ...................... 137
5.11 Comparison of method A and method B ............................. 138
5.12 Empirical cell tracking performance curves .......................... 140
5.13 Empirical performance curves on synthetic videos ................. 143

6.1 Overview of our semi-automated tracking ................................ 150

7.1 The roadmap of the chapter ............................................ 161
7.2 An illustration of the concepts of different kinds of links ............ 163
7.3 The length of a false link ........................................... 169
7.4 Equalizing the sizes of tracking results .............................. 170
7.5 An outline of the pairwise comparison procedure .......................... 174
7.6 Representative results ............................................................ 183

8.1 A Plasmodium ookinete in Matrigel ........................................... 194
8.2 Fragment of a visualization of P. berghei ookinete tracks ............... 196
8.3 The right-handed helical primitive ............................................ 200
8.4 Estimation of the radius and pitch of a track ................................ 201
8.5 Reconstructed tracks of 35 ookinetes ....................................... 202
8.6 Reconstructed tracks translated to a common origin ...................... 202
8.7 The distribution of apparent speeds .......................................... 203
8.8 The distribution of the estimated periods ................................... 204
8.9 A sample ookinete track .......................................................... 205
8.10 Apparent speeds and chiralities .............................................. 206
8.11 Effect of 100 nM of Cytochalasin D on the motility ..................... 208
8.12 Apparent speeds and chiralities of control tracks ....................... 209

B.1 Possible configurations of links ............................................... 228
B.2 Two-round assignment approach ............................................ 229
B.3 Median filtering applied to a reconstructed CDF ......................... 235

C.1 Limitations of different tracking performance measures ............... 243
List of Tables

2.1 Comparison of approaches for cell segmentation . . . . . . . . . . 44
3.1 Datasets that we use in our evaluation . . . . . . . . . . . . . . . 78
3.2 Parameters for segmentation methods . . . . . . . . . . . . . . . 79
3.3 Parameter estimates obtained from training frames . . . . . . . . 80
3.4 F-scores achieved by different methods . . . . . . . . . . . . . . . 84
4.1 Comparison of approaches for cell segmentation . . . . . . . . . . 92
4.2 Output of the acceptance function . . . . . . . . . . . . . . . . . 94
4.3 Datasets for used for evaluation . . . . . . . . . . . . . . . . . . . 103
4.4 Results on the U2OS and NIH3T3 datasets . . . . . . . . . . . . . 105
4.5 F-scores achieved by different methods . . . . . . . . . . . . . . . 107
4.6 F-scores achieved by CLUBS and SEGEN methods . . . . . . . . 108
5.1 Common cell tracking challenges . . . . . . . . . . . . . . . . . . . 113
5.2 The key elements of our work . . . . . . . . . . . . . . . . . . . . . 115
5.3 Videos used for tracker evaluation . . . . . . . . . . . . . . . . . . 133
6.1 Effects of using SAMTRA . . . . . . . . . . . . . . . . . . . . . . . 154
6.2 Number of frames used for SAMTRA training . . . . . . . . . . . 155
7.1 Summary of tracking performance measures . . . . . . . . . . . . 172
7.2 Real videos cover a variety of tracking conditions . . . . . . . . . 177
7.3 Real trackers and their parameters . . . . . . . . . . . . . . . . . . 178
7.4 Performance measures that require ground truth . . . . . . . . . . 182
7.5 The results of our real experiments . . . . . . . . . . . . . . . . . . 184
7.6 The results of the G1 experiments . . . . . . . . . . . . . . . . . . . 185
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.7</td>
<td>The results of our evaluation on the G2 videos</td>
<td>187</td>
</tr>
<tr>
<td>7.8</td>
<td>Eleven CTSs used in our evaluation</td>
<td>188</td>
</tr>
<tr>
<td>7.9</td>
<td>The results of our ranking of 11 CTSs</td>
<td>189</td>
</tr>
<tr>
<td>8.1</td>
<td>Features that describe ookinete tracks</td>
<td>207</td>
</tr>
<tr>
<td>B.1</td>
<td>Ratio of correctly resolved cell divisions</td>
<td>239</td>
</tr>
</tbody>
</table>
Glossary

Bright field microscopy – a microscopy technique where light is transmitted through the sample. The sample is illuminated from below and observed from above. The contrast in the resulting image is caused by non-even absorption of the light in different parts of the sample.

Cell tracker – an algorithm that maintains cell identities throughout a cell video. The algorithm takes a set of detected cell locations as input and produces a set of inter-frame associations as output. An association denotes the same cell in subsequent frames.

Cell tracking system (CTS) – a software module that takes an image sequence as input and produces segmented cell outlines in each frame as well as cell trajectories as output.

Chirality – (in the context of this thesis) is a measure that captures the extent to which a given track resembles a right-handed helix.

Cytochalasin D – is a toxic intermediate product of metabolism produced by organisms of the fungus kingdom.

(Cell) Detection algorithm – an algorithm that takes a cell image as input and produces a set of detected cell centroid coordinates as output.

Green fluorescent protein – a protein that exhibits green fluorescence when exposed to light in a certain spectral range (blue to ultraviolet).

False positive error (FP) – generally, an irrelevant result returned by an algorithm. In the case of cell detection, this is a reported cell centroid that does not correspond to any of the true cells in image. In the case of cell segmentation, this is a reported cell segment that does not correspond to any of the true cells in image. In the case of cell tracking, this is a link that does not correspond to a true relationship between cells in subsequent frame.
False negative error (FN) – generally, a missing true result returned by an algorithm. In the case of cell detection, this is a cell centroid that was not detected. In the case of cell segmentation, this is a cell segment that was not segmented. In the case of cell tracking, this is a true relationship between cells that was not reported.

Fluorescence microscopy – a microscopy technique that exploits the phenomenon of fluorescence. In fluorescence microscopy the sample is first illuminated with light (called excitation light). The sample then emits fluorescent light (called emission light).

Half maximal inhibitory concentration ($IC_{50}$) – a quantitative measure that indicates how much of a particular drug or other substance is needed to inhibit a given biological process by half.

High-throughput screening – a scientific experimentation method that involves a large number (e.g., thousands or millions) simultaneous trials (chemical, pharmacological, etc.). This method can be useful for rapid identification of a chemical component with desired properties.

Link – a pair of cell measurements from subsequent video frames. The link represents a hypothesis that the measurements belong to the same cell in subsequent frames.

Link length – denotes a generalized distance between the two measurements in a link. In this thesis, we define link length as the Euclidean distance between the measurement centroids, but other definitions are possible.

Matrigel – is the trade name for a gelatinous protein mixture secreted by certain mouse sarcoma cells and marketed by BD Bioscience. Matrigel is used to imitate complex extracellular environments.

Measurement – a vector of features that represents a detected or segmented cell in a video frame. The measurement can comprise centroid coordinates, area, brightness, etc.

Missing (detection, segmentation or link) – a true result that was not produced by the corresponding algorithm (see false negative error).

Ookinetes – a motile zygote of a large group of protists called apicomplexa. Certain apicomplexan species cause Malaria.

Over-segmentation – representing one cell with multiple segmented objects
by a cell segmentation algorithm.

*Plasmodium* – a genus of protists parasites. Infection by these organisms is known as malaria.

*(Cell) Segmentation algorithm* – an algorithm that takes a cell image as input and produces a set of segmented cell outlines as output.

*Spurious (detection, segmentation or link)* – an irrelevant result produced by the corresponding algorithm (see false positive error).

*Time lapse microscopy* – a microscopy technique in which the same object is photographed at regular intervals during a prolonged period of observation.

*Under-segmentation* – representing more than one cell with a single segmented object by a cell segmentation algorithm.
Notation

$I$: a digital image, i.e., an $M \times N$ matrix
$I(x, y)$: a pixel value (brightness) at position $(x, y)$
$H(u, v), K(i, j)$: two-dimensional kernel functions
$m$: cell measurement, a presumable cell location in video
$P_{\text{links}} = 1 - \frac{\#\text{swap errors}}{\#\text{all measurements}}$: measure of tracking accuracy
$R_g$: gating distance
$D = \{\vec{x}, \vec{y}, \ldots\}$: detection, i.e., a set of all detected locations
$l = \{t, \vec{x}, \vec{y}\}$: link, connects two locations in consecutive frames
$L_{\text{all}} = \{l = \{t, \vec{x}, \vec{y}\} : \vec{x}, \vec{y} \in D\}$: set of all links implied by the detection
$GT$: ground truth, maps each link $l \in L_{\text{all}}$ to the set $\{true, false\}$
$L_{\text{true}}, L_{\text{false}}$: set of true (resp. false) links
$CT$: cell tracker, takes $D$ as input, yields $L_{\text{trk}} \subseteq L_{\text{all}}$ as output
$F_1 = 2 \frac{\text{prec} \cdot \text{recl}}{\text{prec} + \text{recl}}$: F-score
$F_1(CT(D)) = F_1(L_{\text{trk}})$: F-score of a tracker on a given input
$L_i$: set of links that end in frame $i$ according to a cell tracker
$n_i = |L_i|$: number of links made by a tracker that end in frame $i$
$n_{\text{true}}$: number of ground truth links that end in frame $i$
Notation (continued)

VN = \text{Var} \left( \{n_2, \ldots, n_k\} \right): VN-score, \( k \) is the number of frames in video

nSpur; (nMiss): spurious (respectively missed) links

R(l) : \mathbb{R}^k \times \mathbb{R}^k \rightarrow \mathbb{R}: length of the link; in this thesis, \( R(l) = \| \vec{x} - \vec{y} \| \)

E_f(\mathbb{I}_{\text{trk}}): event “a link \( l \) randomly chosen from a set \( \mathbb{I}_{\text{trk}} \) is a false link”

\( N = |\mathbb{I}_{\text{trk}}|: \) number of links in \( \mathbb{I}_{\text{trk}} \)

P_{all} (P_f): PDF of the lengths of all (respectively false) links

\( \mathbb{I}_{\text{trk}}' = \mathbb{I}_{\text{trk}} \cup \mathbb{I}_{\text{pad}}: \) padded output of the cell tracker, \( \mathbb{I}_{\text{pad}}: \) dummy links

T = |\mathbb{I}_{\text{true}} \cap \mathbb{I}_{\text{trk}}|, T^* = |\mathbb{I}_{\text{true}}|: numbers of true links in \( \mathbb{I}_{\text{trk}} \) and \( \mathbb{I}_{all} \)
Chapter 1

Introduction

Cells are the building blocks of life. For example, the human body is composed of multiple types of cells that are responsible for metabolism, immunity and many other vital systems. It is therefore not surprising that there exists a large discipline, cell biology, dedicated to studying cells, including their structure, properties and life cycle.

Information about cells can be obtained in different ways. Among other methods, an important role belongs to optical microscopy, and in particular to time lapse microscopy (Figure 1.1). This thesis is dedicated to automated methods for processing and analyzing the information obtained using time lapse microscopy. We start the thesis with an overview of this technique and the motivation of our work in the next few sections.

Figure 1.1: A dividing B lymphocyte cell observed using time lapse microscopy. Images show the same field of view photographed at regular time intervals. Images courtesy of Hodgkin Lab, WEHI.
1. INTRODUCTION

1.1 Scope of the Thesis

Time lapse microscopy refers to photographing live cells at regular time intervals for a prolonged period of time (e.g., hours or days). This method enables the collection of different kinds of data about cells. Such data include statistics about cell lifetimes, sizes, shapes and trajectories. Time lapse microscopy has been a key technology in a number of studies of important biological systems and processes, including immune response \cite{61,92}, migration of malaria parasites \cite{62}, and pregnancy \cite{59}.

Another important application of time lapse microscopy is automated drug screening \cite{53}. In this application, one can add different chemical compounds to different wells containing the cells under study (e.g., cancer cells), and observe the development of cells under different treatments. The quantitative results of the observations can enable the identification of candidate compounds for drugs.

Note, however, that an output of time lapse microscopy imaging is a sequence of raw (unannotated) images. In order to obtain relevant and meaningful biological information about the cells, one needs to locate the cells in the images and, if required, track cells over time. Therefore the following workflow (Figure 1.2) is commonly used for this purpose.

In this workflow, biologists first design an experiment and prepare cells for recording in vitro (i.e., outside the organism, in a dish, such as in the study of Hawkins et al. \cite{61}), perhaps in the presence of chemical compounds of interest. The living cells are then video recorded using a microscope. Alternatively, the microscope can record cells in vivo (in a whole living organism, such as in the study of Hellmann et al. \cite{62}).

The obtained videos are then processed by a cell tracking system (CTS) which takes an image sequence as input and produces segmented cell outlines in each frame as well as cell trajectories as output. These data then form a basis for a higher level analysis that can involve mathematical modeling of cellular processes \cite{38,84}. The data analysis step is highly specific to a particular biological assay, and in this thesis, we focus on cell tracking systems. The diameter of cells in the videos that we consider in this thesis is in the order of 10 \textmu m.
1.2. MOTIVATION OF OUR WORK

Figure 1.2: Time lapse microscopy experiments can include video recording of living cells placed in wells, image processing and data analysis. Computers can assist with different stages of image analysis, but a considerable amount of manual work is still required. Manual intervention is usually required at all stages of the pipeline: for reviewing the obtained video and selecting the pre-processing method, for selecting and tuning the segmentation and tracking methods, for validating the tracking results and for interpreting the obtained statistics.

The CTS is one of the key components of time lapse experiments. The CTS can be further divided into (1) a cell segmentation module that takes a video as an input and produces numerical descriptions (e.g., boundary, area, brightness, elongation) for each cell found in each frame, and (2) a cell tracking module that establishes inter-frame associations between the cells found at the segmentation step. In other words, the cell tracking module (or the tracker, for brevity), maintains the identities of cells throughout the video. Note that in some CTSs the segmentation and tracking modules are combined into a single algorithm.

We now discuss some open cell tracking challenges that have motivated our work.

1.2 Motivation of our Work

Provided with a sequence of images (a video) from a microscope, one can manually segment and track cells in the video, that is, a human can essentially serve as a CTS. Apart from the subjectivity of such manual results, a major disadvan-
1. INTRODUCTION

The advantage here is that manual segmentation and tracking do not scale well with the number of cells in a video and with the number of videos. This limits the use of manual tracking for high-throughput screening, where a large number (e.g., hundreds) of biological assays can be performed simultaneously [63].

This non-scalability can be illustrated with the following motivating example. A medical research institute records cells placed in hundreds of microwells, and the field of view of the microscope at the required magnification covers only about 20 microwells. The motorized stage with the microwells moves sequentially so that various microwells can be observed. This essentially results in dozens of different videos, one per field of view. Furthermore, the observation of cells lasts for two days (to cover a few rounds of cell divisions), and at the given frame rate this results in each video being about 1500 frames long. Researchers might not need tracking for all the videos, but for a statistically reliable analysis, biologists may need to have at least a hundred lineage trees for each of the experimental conditions. These numbers suggest that it can be very time consuming and tedious to locate and track cells manually just for one experiment. Indeed, in practice, video annotation can take a researcher a few months.

The disadvantages of manual tracking result in a desire to develop an automated CTS. Note that cell segmentation and tracking is a special case of the more fundamental problem of object tracking. Automated object tracking has been a field of intensive research during the past few decades [128], with early techniques, such as Kalman filters, appearing in the 1960s [70]. Furthermore, starting from the late 1990s [76, 85] there has been a growing body of work specifically addressing cell segmentation and tracking [86, 108].

These methods were successful to a certain degree, but frustratingly, a fully automated CTS can rarely be used in practical scenarios. Instead, a common approach is to use a computer-assisted CTS, where some form of manual input is still required. For example, a user can be asked to mark cells in the first frame [59] or to watch a pre-annotated movie for an appearance of specks of color: “the frame at which one could say with near certainty that the cell was fluoresc-

\[1\] The numbers in this example are taken from the author’s work experience at the Walter and Eliza Hall Institute of medical research, and from conversations with biologists.
1.2. MOTIVATION OF OUR WORK

Figure 1.3: Challenges of automated cell segmentation. (Cell image is taken from Coelho et al. [25], annotation was done by the author of the thesis.)

ing was recorded as the moment at which the cell differentiated to plasmablast or isotype switched” [38]. There are two main reasons why it is hard to develop a completely automated CTS.

First, cell segmentation and tracking are difficult problems in general. In cell segmentation, challenges may arise due to touching cells, overlapping cells, variation in cell sizes and intensities, or background noise (Figure 1.3). As a result, there exist real cell videos where humans outperform advanced automated algorithms in segmentation accuracy [25].

Furthermore, the automated tracking problem is also difficult. It is often not possible to increase the frame rate arbitrarily due to the effect of phototoxicity [75]: “difficulties ... arise from the necessity to compromise between maintaining the health of the cell while achieving the appropriate temporal and spatial resolutions required for the study”. Reduced frame rates often result in abrupt cell motions when cells travel large distances between consecutive frames, compared to the distances to neighboring cells. As a result, even the most advanced automated trackers often cannot achieve 100% accuracy of track reconstruction [97]. Note that a single mistake, such as a single swapping of the identities of
1. INTRODUCTION

cells between two frames, can invalidate the trajectories of both cells and their progeny [79].

As discussed above, in many practical scenarios automated systems alone cannot achieve a satisfactory level of performance. We note that for some “easy” videos (e.g., with very slow cells) there can exist an effective automated segmentation or tracking algorithm. However, for both “hard” and “easy” videos another problem remains as follows.

Second, there is a large variability in cell morphology and dynamics, as well as in the experimental conditions. As a result “it is often not clear which particular image ... segmentation, or tracking algorithm is well suited for the given data type. ... It would be helpful if the decision on what type of algorithm should be used, or what particular parameter setting should be used, could be made automatically.” [108]

Consider the following practical scenario. Biologists observe cells under different treatments and produce corresponding videos (e.g., 10 videos are recorded within a single calendar month, where each video is 1500 frames long) [2]. Furthermore, there are a number of available automated CTSs that one can use to process the videos. One approach would be to manually annotate a part of a single video (e.g., the first 30 frames), run the candidate CTSs on this annotated part and select a CTS that produces results that are most consistent with the manual annotation. One can then use this CTS for all videos.

However, since the videos were recorded under different conditions, it is unlikely that the selected CTS remains optimal across different videos. Cell dynamics may change even within a single video. For example, a video can start with 2 cells that then proliferate to 16 cells, and the increased cell density can result in different dynamics of the cells. Therefore, the selected CTS is unlikely to remain optimal even throughout a single video. The user then has to either accept the likely non-optimality of the selected CTS or perform a manual annotation for a part of every video, which is a time consuming operation.

As a consequence of the above two reasons, manual input is often required in the following forms:

---
[2]The numbers in this example are taken from the author’s work experience at the Walter and Eliza Hall Institute of medical research, and from conversations with biologists.
1.3. OUR CONTRIBUTIONS

- manual annotation (segmentation or tracking) of a part of a video, in order to select the optimal automated algorithm or tune its parameters for the given video;
- manual post-processing of results to correct segmentation or tracking errors made by an automated CTS and improve the final accuracy of the results.

In summary, time lapse imaging is often used to study important biological systems, and this technique can also be used for automated drug screening. A CTS is a key step in time lapse imaging experiments. Manual cell tracking is subjective and time consuming, and there is a need for automated CTSs. There have been a number of proposed automated methods, but due to the reasons outlined above, a common practice is to use a computer-assisted CTS with some manual input, rather than a completely automated system.

In this thesis, we develop methods to reduce the amount of manual work while maintaining a satisfactory level of accuracy of cell tracking systems.

1.3 Our Contributions

In order to address the challenges outlined in the previous section, we develop a number of methods for the different stages of a cell tracking pipeline, with a particular emphasis on the portability of these methods. By portability we mean the ease of applying a method across different videos. Cell segmentation is an important first step in image analysis, and we propose two algorithms for detecting and segmenting cells in video frames. We then present a portable method for cell tracking. Furthermore, we develop two methods that are specifically designed to reduce the amount of manual work required to obtain accurate results. These two methods do not perform segmentation or tracking as such, but instead they can be used with any existing cell tracker. Finally, as an example application for cell tracking systems, we consider the problem of automated anti-malaria drug screening. To this end, we develop a method for characterizing the motility of malaria parasites. Our evaluation suggests that the proposed numerical characterization can be used in high-throughput screening.
We summarize our contributions below, along with accompanying publication details.

1. **Algorithm CLUBS for automated semi-segmentation of cells:** CLUBS does not produce cell outlines (and hence it is not a segmentation algorithm), but it can produce features (e.g., centroid location, estimation of area) to describe cells. To the best of our knowledge, this is the only cell semi-segmentation algorithm in the literature. This algorithm is described in Chapter 3. The publication and presentation arising from the work in this chapter are:
   * Kan, A., Chakravorty, R., Bailey, J., Markham, J., and Leckie, C., A Sampling and Clustering Based Semi-Segmentation of Fluorescent Cells, International Conference on Computer Vision and Pattern Recognition, 2013 (in preparation);
   * Kan, A., A Sampling and Clustering Based Semi-Segmentation of Fluorescent Cells, Computational Immunology Conference, Melbourne, April 2012 (oral presentation).

2. **Algorithm SEGEN for cell segmentation:** Our evaluation suggests that SEGEN performs on par or better than existing cell segmentation methods. The advantage of SEGEN is its ability to adapt to differences in the appearances of cells in different videos through the notion of an acceptance function. This algorithm is described in Chapter 4. The publication arising from the work in this chapter is:
   * Kan, A., Markham, J., Bailey, J., Chakravorty, R., and Leckie, C., Fluorescent Cell Segmentation Based on Region Enumeration, IEEE Transactions on Biomedical Engineering (under review).

3. **Algorithm NENIA for cell tracking (points association):** This algorithm has only one parameter, and we also propose a parameter-free version of the algorithm. NENIA is capable of performing on par with existing tracking algorithms, while having fewer parameters, making it more portable. To the best of our knowledge, NENIA is the only cell tracking algorithm in the literature that has a parameter-free version. This algorithm is described in Chapter 5. The publication arising from the work in this chapter is:
1.3. OUR CONTRIBUTIONS

is:


4. Algorithm SAMTRA for semi-automated tracking: This algorithm can be used for manual correction of automated tracking results. Given tracking results of an arbitrary automated cell tracker, SAMTRA identifies video frames that are likely to contain tracking errors. The user can then review only these frames, rather than all frames in a video (which can be very long). This algorithm is described in Chapter 6. The publication arising from the work in this chapter is:


5. A method for ranking CTSs without manual validation: For a given video, this method can be used to recommend a CTS (from a set of candidate CTSs) that is likely to be optimal for the video. This method only requires an estimate of the size of cells in the video, and does not require an annotation of all or part of the video. To the best of our knowledge, this is the only method of its kind for cell tracking. This algorithm is described in Chapter 7. The publication and presentation arising from the work in this chapter are:

* Kan, A., Leckie, C., Bailey, J., Markham, J., and Chakravorty, R, Measures for Ranking Cell Trackers without Manual Validation, Pattern Recognition (under review);


6. A numerical characterization of the motility of Plasmodium ookinetes (motile zygotes of malaria parasites) in vitro: In this chapter, we present a

---

5This is the same publication where we describe NENIA.
specific biological application of cell tracking techniques, namely, to characterize the motility of a particular form of malaria parasites. To the best of our knowledge, our characterization provides the first statistical confirmation of the helical nature of ookinete tracks. Furthermore, our experiments with Cytochalasin D suggest that our method can be used for automated anti-malaria drug screening. We present the method in Chapter 8. The publication arising from the work in this chapter is:


Moreover, the thesis also contributes an original perspective on the literature for cell segmentation algorithms (Section 2.3.1). While our review of image processing and cell tracking methods adheres to a more traditional structure, we present and compare segmentation algorithms with respect to a generalized artificial intelligence algorithm. This perspective allows us to review the strengths and weaknesses of the previous algorithms in a systematic manner. Furthermore, we identify the open research challenge of developing approaches that do not require seeding points or regions for segmentation, while at the same time allowing the user to apply a wide range of fitness functions (Chapter 4).

In this thesis, the technical chapters (Chapters 3 - 8) are self-contained to a large extent. Each chapter contains a brief introduction, motivation and related work survey, specific to the problem addressed in the chapter. Furthermore, a glossary has been provided on page xxiii to enable the reader to look up key terms. As a result, the technical chapters can be read in any order, with the exception of Chapters 3 and 4.

In general, the background chapter (Chapter 2) is not essential for understanding the rest of the thesis. However, readers new to the field of time lapse microscopy and cell imaging are strongly encouraged to read the overview in Sections 2.1 and 2.6 and readers new to digital image processing are advised to read Section 2.2.

Finally, Chapter 9 discusses the novelty and implications of our results, and
1.3. OUR CONTRIBUTIONS

concludes the thesis.
1. INTRODUCTION
Chapter 2

Background

In order to establish the context for the algorithms contributed by this thesis, we start with a review of time lapse microscopy, covering the stages of specimen preparation and image acquisition (Section 2.1). We then switch domains from hardware to software, and proceed with a review of digital image processing methods (Section 2.2). Preliminary image processing is an important step often hidden inside (i.e., performed as a part of) segmentation algorithms. This step aims to improve the quality of the microscopic images so that subsequent segmentation algorithms can work better. Finally, we present a survey and a critical analysis of existing methods for automated cell segmentation (Section 2.3) and cell tracking (Section 2.4). Figure 2.1 illustrates the scope of the review and the associated sections of this chapter.

We now proceed to the review of optical microscopy and cell staining, and we give a number of references for further reading on these topics.

2.1 Optical Microscopy

The optical microscope is a device that uses visible light and a system of lenses to magnify images of objects of interest (e.g., living cells in a well). This device enables a direct observation of cells and internal sub-cellular structures over time, and it is one of the main instruments for cell study. Another technique widely used in studying cells is flow cytometry, and in particular fluorescence-
activated cell sorting (FACS) [43, 106]. The advantage of optical microscopes compared to FACS machines is that the microscope provides the ability to track the motions and fates of individual cells. Such information is not available through FACS experiments, and thus optical microscopy has a potential to reveal different kinds of knowledge about cells.

There are a number of different types of optical microscopes, and in this chapter, we only review two types: bright field and fluorescence microscopy. For more information, including descriptions of other techniques (e.g., phase contrast or differential interference contrast), please refer to a comprehensive textbook such as [90] or to some excellent online microscopy knowledge bases such as [1, 2].

**Bright field microscopy** is the most straightforward technique used to obtain microscopic images. In this technique, the light produced by a light source (commonly a halogen lamp) is passed through the specimen (Figures 2.2 and 2.3). The contrast in the resulting image is caused by the absorption of light by dense parts of the specimen. A **condenser** is a system of lenses that is used to collect scattered light from the light source and focus it onto a small area around the specimen. An **objective** is a system of lenses that produces a magnified image of the specimen. The image can then be magnified further by the eye piece lens.
2.1. OPTICAL MICROSCOPY

Figure 2.2: Schematic representation of bright field microscopy (left) and fluorescence microscopy (right). In bright field microscopy, the light emitted from a lamp passes through the specimen, and the contrast in the resulting image is caused by absorbing light by dense parts of the specimen. A system of lenses is used to concentrate the illumination light around a small area of the specimen (condenser), and to produce a magnified image of the specimen (objective). In fluorescence microscopy, the specimen is first illuminated by light of a specific excitation wavelength, and then emits light with another wavelength. A system of filters is used to separate light paths of excitation and emission light.
2. BACKGROUND

Figure 2.3: Sample bright field (left) and fluorescence (right) images of simulated B-lymphocytes in a hexagonal well. Both images show the same field of view photographed at about the same time. Note that the well boundaries are not visible in the fluorescence image, because they emit very little light compared to cells. Images courtesy of Hodgkin Lab, WEHI.

2.1.1 Fluorescence microscopy

Fluorescence microscopy is based on the phenomenon of fluorescence. Fluorescence is emission of light by the specimen in response to absorption of light of a different wavelength. The specimen is first illuminated by the light source. This light is called excitation light, and it causes electrons in certain molecules in the specimen to be excited to a higher quantum state. These electrons then relax to their ground state by emitting photons of light. This light is called emission light. Usually the emission light has a longer wavelength than the excitation light (Figures 2.2 and 2.3).

A system of filters is used to employ the phenomena of fluorescence for microscopy. Firstly, an excitation filter is used to obtain a nearly monochromatic excitation light from the light source (e.g., a xenon arc lamp or light-emitting diodes). This is needed to ensure that only certain molecules of interest are being affected. Secondly, the excitation light is reflected by a dichroic mirror that reflects the light of the excitation wavelength and passes through the light of the emission wavelength. Finally, an emission filter ensures that only the emission light (and not the excitation) reaches a detector.

A chemical compound capable of fluorescence is called a fluorophore. Plastic
wells containing live cells, as well as the medium around the cells, may contain fluorophores which causes a weak level of light emission, known as background fluorescence. Cells usually do not contain sufficient amount of fluorophores to differentiate them from the background. In order to make use of fluorescence microscopy, one needs to prepare the cells in a specific way, either by staining or by genetic modification. In staining, cells in the specimen are placed for some time into a solution containing a fluorescent dye (e.g., a Hoechst stain or DAPI stain). During this time fluorophores from the dye attach to cell DNA (or other cellular structures, depending on the dye). In contrast, genetic modification can be used to create cells that express fluorescent proteins such as green fluorescent protein (GFP).

After preprocessing, in response to a light of a specific wavelength, cells or cell nuclei emit light of a different wavelength. This emitted light is much stronger than the background fluorescence, and this enables differentiation of objects of interest from the background (Figure 2.3). As a result, it is usually easier to segment cells using automated methods from fluorescence images than from bright field images. This is an important advantage of fluorescence microscopy in the context of cell tracking.

Another big advantage of fluorescence microscopy in general is the selectivity of fluorescence. Different fluorescent dyes can be attached to different sub-cellular structures [90]. Each dye has its own characteristic excitation and emission spectrum, and in a single experiment one can select dyes so that different color channels show specific cellular structures for the same specimen. That is, the specimen is first excited with a wavelength for the first dye, and the emitted light is registered as the first channel. The same specimen is then excited with a wavelength for the second dye and so on. This results in a field of view being covered by multiple color channels.

Furthermore, fluorescence can be made specific to a particular cell-cycle phase [109]. The cell-cycle is a series of events occurring in the cell and leading to its duplication. It is possible to use genetic modification to create cells that emit light of different wavelengths during the different cell-cycle phases.

Three major problems associated with fluorescence imaging are (i) the need for specimen preprocessing, (ii) phototoxicity and (iii) photobleaching [90]. Pre-
viously we mentioned the first two aspects. **Photobleaching** is destruction of fluorophores caused by light. Photobleaching results in a fading of fluorescent images. Over the course of time lapse experiments, the fluorescence level of objects of interest (e.g., cells) decays, and eventually it can reach the level of background fluorescence. Phototoxicity and photobleaching are the main reasons why one cannot arbitrarily shorten the time intervals between snapshots in time lapse experiments.

Despite the above limitations of fluorescence microscopy, this technique is widely used in biology [38, 61, 62]. Hertzberg and Pope remark: “It is now clear that fluorescence-based techniques are likely to be amongst the most important detection approaches used for HTS in the future, given the industry-wide drive to simplify, ... and speed up assays. Fluorescence techniques are well suited to uHTS because they give very high sensitivity ...” [63]. Here HTS means high-throughput screening and uHTS means ultra-HTS, that is methods capable of performing 100,000 assays per day.

### 2.1.2 Moving to Digital Video Microscopy

An optical image formed by a bright field or fluorescence microscopy system (or any other type of optical microscopy) can be directly observed through eye pieces. It is also common to redirect and record this image using a digital camera. Therefore, the results of time lapse experiments are typically stored as sets of digital images. A digital image is a 2 or 3-dimensional matrix, where each element is called a pixel (in the case of 2 dimensions) or voxel (in the case of 3 dimensions). A pixel or voxel represents the intensity of an optical signal registered by the camera at a certain location in space [126].

Due to the construction of lenses, it is hard to achieve uniform illumination of a rectangular frame (Figure 2.4). Uneven illumination often occurs due to the round shape of the lenses or due to the non-perfect alignment of the specimen with respect to the focal plane of the condenser [34]. Recall that the typical size of cells in the images that we consider in this thesis is about 10 \( \mu m \), and it is hard to align the stage holding the specimen at such small scales.

Furthermore, there are different sources of noise registered in the image.
Figure 2.4: A bright field image of simulated B-lymphocytes in rectangular micro-wells. The illumination of the image is not uniform. Furthermore, the image contains noise, and scratches in the wells appear as extraneous objects. Image courtesy of Hodgkin Lab, WEHI.
First, there is thermal noise affecting the sensor in the digital camera. Second, there can be small particles of dust on any of the optical components of the microscope. Third, in the case of fluorescence imaging, there is background fluorescence. Moreover, the plastic wells can have scratches that appear as extraneous objects in the image (Figure 2.4).

In practice, the specimen consists of multiple wells (e.g., hundreds), but the field of view at the given magnification can only cover a few wells. Therefore it is common to use a motorized stage that routinely brings different parts of the grid of wells to the objective. This imposes an additional limitation on the time interval between snapshots of the same field of view, as the stage has to iterate over a number of positions before it returns to the same field of view. Furthermore, the movement of the stage results in shifts between the positions of the same field of view in consecutive time snapshots, because it is hard to return the stage to exactly the same position where it was.

In summary, the physics of the image acquisition heavily affects the resulting images. Scratches are not visible in fluorescence images, but uneven illumination, noise and stage shift can hamper the analysis of digital images. It is difficult to manufacture perfect hardware for image acquisition, and it is therefore common to recover from the above-mentioned artifacts using software methods described in the following sections.

2.2 Digital Image Processing

Digital image processing refers to processing images by means of a digital computer [52]. Although there is no consensus about the boundary between image processing and image analysis [52], in general, image processing refers to algorithms that manipulate individual pixels and can involve only a shallow, if any, understanding of the content of the image. A characteristic feature of such methods is that they are easily transferable across domains, that is, the same methods can be applied to images of cells, cars or stars.

In the context of this thesis, we consider image processing methods as an aid to improve the quality of subsequent steps: segmentation and tracking. Im-
2.2. DIGITAL IMAGE PROCESSING

Digital image processing is commonly used to alleviate the consequences of the artifacts introduced by the image acquisition process as we discussed above [25, 41, 97]. Let $I(x, y)$ be a digital image (i.e., an $M \times N$ matrix, where $x = 0, \ldots, M - 1$, $y = 0, \ldots, N - 1$, and $I(x, y) \in [0, 1]$). We then say that an image processing method is a function that takes an image $I$ as input and produces an image $I'(x, y)$ of the same size as output. In the rest of this thesis, we focus on 2-dimensional gray-scale images for simplicity of narration, but we note that the methods that we contribute can be extended to 3-dimensional color images.

Although, many cell segmentation methods in the literature include image processing methods as sub-routines [25, 41, 97], it is convenient to introduce cell segmentation as a separate step (presented in Section 31). Cell segmentation involves a higher level of understanding of image content. In contrast to a processing method, a segmentation method takes an image as input, but produces a set of segmented objects as output.

In this section, we introduce several image processing methods particularly relevant for cell tracking systems. For a thorough discussion on digital image processing, please refer to the book by Gonzalez and Woods [52].

2.2.1 Compensating for Uneven Illumination

One of the basic methods to compensate for uneven illumination in an image is to subtract an averaged image of the same field of view taken at different time points (a variation of this method is presented by Dorval and Genovesio [34]). Here the assumption is that the background illumination pattern is stationary, whereas the foreground objects move, and do not cover a large part of the background at any time. That is, the averaged image is essentially an estimated background image: $I_{avg}(x, y) = \frac{1}{T} \sum_{t=0}^{T-1} I_t(x, y)$, where $t$ indexes time points. The compensated image is then $I_{comp} = I - I_{avg}$.

Alternatively, the background image can be estimated by smoothing the original image. Smoothing can be done in many different ways, for example, using Gaussian kernels or a median filter (discussed in the next section). It is expected that the smoothed image captures global intensity variation and therefore can be used to compensate for uneven illumination.
2. BACKGROUND

Figure 2.5: An original bright field image (left) and the compensated image (right). The compensated image is obtained by subtracting a blurred version of the image computed via a convolution with a Gaussian kernel. The illumination is more uniform in the compensated image.

Figure 2.6: A rolling ball filter produces an image that comprises the highest intensity levels (red dotted line), achievable by the sphere structuring element as it travels along the surface defined by image intensity levels (black solid line).

Among others, an interesting way to obtain a smoothed background image is to use morphological opening of a gray-scale image with a sphere (also called a rolling ball filter) [36 115]. Here the idea is to produce an image that, at each location, has the maximum intensity achievable by a sphere structuring element as it moves along the surface defined by the original image intensity levels (Figure 2.6). The structuring sphere should not be too small so that it does not fall inside the image peaks (usually cells), and not too large so that it remains sensitive to the variation in the illumination throughout the image.

2.2.2 Image Smoothing in the Frequency Domain

Image noise can be loosely defined as unwanted variation in image intensity [52]. In the context of cell tracking systems, the noise is mostly a signal that does
not originate from the cells. However, certain fine grain intensity variation that originate from cells can also be undesirable. In both cases, image smoothing is applied to reduce the amount of unwanted variation. Here the assumption is that the cells are represented with medium sized image features (contours), whereas noise appear as smaller artifacts. Smoothing refers to reducing the amount of smaller image features while preserving larger shapes.

Different smoothing techniques usually allow a user to control the intensity of blurring (e.g., via specifying the size of a filter, see below). Recall that smoothing can also be applied in order to obtain a background image for compensating for uneven illumination (Section 2.2.1). In this case, smoothing is performed more aggressively, so that the objects (cells) themselves are blurred out in addition to fine grain noise features.

**Filtering** is the process of removing unwanted components from the image and smoothing is usually implemented via a filter that blocks high frequency image components. Filtering can be performed in the frequency domain, in the spatial domain or in a combined domain.

The main filtering method in the frequency domain is based on the Fourier transform. The discrete Fourier transform (DFT) for an image $I(x, y)$ of size $M \times N$ is given by the equation

$$F(u, v) \equiv \frac{1}{M \cdot N} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I(x, y) \cdot \exp\{-i \cdot 2\pi (ux / M + vy / N)\}. \quad (2.1)$$

Here $i$ is an imaginary unit, and $u, v$ must be computed for $u = 0, \ldots, M - 1$, $v = 0, \ldots, N - 1$.

The inverse DFT is given by the equation

$$I(x, y) \equiv \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) \cdot \exp\{i \cdot 2\pi (ux / M + vy / N)\}. \quad (2.2)$$

A useful property of the DFT is that $F(u, v)$ represents a frequency spectrum of the image, that is, a set of weights for different frequency components of the image: global image features will be associated with lower $u$ and $v$, and fine-grain features (such as noise) will be associated with higher values of $u$ and $v$. In
Figure 2.7: An original bright field image (left), a $10 \times 10$ matrix with the lowest Fourier coefficients (middle, amplitudes are shown as gray levels), and a filtered image (right) obtained after suppressing all coefficients except those shown in the middle. The amount of fine grain noise is reduced in the filtered image.

In particular, from Equation 2.1, it is easy to see that the slowest varying frequency component $F(0,0)$ corresponds to an average intensity of the image.

One can now define a filter function $H(u,v)$ which is essentially a matrix of weights. For the purposes of image smoothing, lower weights can be given to higher frequencies. In summary, a basic DFT filter can be summarized as follows [52].

1. Compute a DFT of the image using Equation 2.1.
2. Define a filter function $H(u,v)$.
3. Multiply (element-wise) $F(u,v)$ by the filter function $H(u,v)$.
4. Compute the inverse DFT using Equation 2.2.
5. Obtain the real part of the result (Figure 2.7).

### 2.2.3 Image Smoothing in the Spatial Domain

In the spatial domain, filtering involves processing each image pixel while taking into account a set of neighboring pixels, that is, a sub-image. A common way to define the sub-image is to use a square window with side $(2 \cdot w + 1)$ around the current pixel. For example, in median filtering, the output pixel is the median value of the intensity levels in the window

$$I_{\text{flt}}(x,y) = \text{median}\{I(x+i,y+j); i,j = -w, \ldots, -1, 0, 1, \ldots, w\}. \quad (2.3)$$
Another popular filtering approach is based on convolving the original image with a function \( K(i, j), i, j = -w, \ldots, w \), called a kernel, which is defined on the same window as above (Figure 2.8). In this case, the filtered image is defined by

\[
I_{\text{filt}}(x, y) = \sum_{i=-w}^{w} \sum_{j=-w}^{w} K(i, j) \cdot I(x+i, y+j). \tag{2.4}
\]

It is also possible to use a rectangular filter, but in general, there is no reason to treat the \( X \) and \( Y \) dimensions of the image differently.

In Equation 2.4 a common choice of kernel function is the Gaussian kernel defined by

\[
K(x, y) = \frac{1}{2\pi\sigma^2} \cdot \exp\left\{-(x^2 + y^2)/2\sigma^2\right\}, \tag{2.5}
\]

where \( \sigma \) is a parameter (standard deviation). This kernel gives high weight to pixels near the current position \((x, y)\), and exponentially decaying weights to pixels further away from \((x, y)\).

Gaussian kernels and median filters are often used to reduce noise in cell images \[25\,41\], and there have also been a number of other filtering methods capable of smoothing an image in the spatial domain. We refer an interested reader to a textbook in image processing \[52\], and we proceed to a method that performs filtering in a combined spatial and frequency domain.
2. Background

2.2.4 Image Smoothing in a Combined Domain

As an example of a filtering technique in a combined domain, here we present a method based on the “à trous” (“with holes”) wavelet transform \[97, 114\]. Overall, the idea is to represent the original image as a series of wavelet coefficients, filter out those coefficients that presumably originate from noise, and restore the image from the remaining coefficients. In the “à trous” wavelet transform, the original image \(I_0\) is first represented as a series of approximating images \(I_1, \ldots, I_s\) with different levels of detail. Each \(I_k\) is obtained by taking the previous image \(I_{k-1}\) and convolving it with a discrete filter. However, at each level \(k\), the step between the adjacent pixels participating in the convolution increases (hence the name “with holes”).

Formally

\[
I_k(x, y) = \sum_{i=-w}^{w} \sum_{j=-w}^{w} h(i, j) \cdot I_{k-1}(x + i \cdot 2^{k-1}, y + j \cdot 2^{k-1}),
\]

(2.6)

where \(k = 1, \ldots, s\), \(s\) is the last level of detail, and \(h(i, j)\) is a discrete filter of size \(w\). (Compare this with Equation 2.4) Starck et al. proposed an efficient implementation of the transform with a filter based on \(B_3\)-splines \[114\].

Detail images are then computed as \(W_k(x, y) = I_k(x, y) - I_{k+1}(x, y)\), \(k = 0, \ldots, s - 1\). Wavelet coefficients are elements of a detail image. Finally, the original image \(I_0\) is represented as a set \(\{W_0, \ldots, W_{s-1}, I_s\}\). Note that the original image can be reconstructed from the wavelet representation by taking the sum

\[
I_0(x, y) = I_s + \sum_{k=0}^{s-1} W_k(x, y).
\]

(2.7)

It is then assumed that certain wavelet coefficients are originated from noise, and filtering is performed by setting these coefficients to zero by using a level dependent hard threshold \(\theta_k\). If \(w_{x,y}^{(k)} \in W_k\) is a coefficient, then \(w_{x,y}^{(k)}\) is set to zero if \(w_{x,y}^{(k)} < \theta_k\) \[94\]. The threshold can be selected using a method by Donoho and Johnstone \[33, 94\]. After suppressing noisy coefficients, the smoothed image is obtained using the reconstruction in Equation 2.7.

This is not the only method for performing a wavelet transform. There is
2.2. DIGITAL IMAGE PROCESSING

an extensive literature on wavelets (e.g., [104]). Wavelet transforms resemble Fourier transforms in terms of representing the original image as a series of entities that capture different levels of detail. However, wavelet transforms are performed in a combined spatial and frequency domain. Compared to a spatial convolution, filtering based on a wavelet transform can use different thresholds at different resolution levels, which makes the wavelet-based technique more flexible.

2.2.5 Stage Shift Removal

As we discussed previously in Section 2.1.2, a stage shift might result in consecutive images being photographed at slightly different positions. This is not a problem for cell segmentation algorithms that work independently on each frame (e.g., the methods described in Sections 2.3.2 – 2.3.5). However, this displacement can cause problems in the case of cell tracking or combined segmentation and tracking (Section 2.4). The displacement may cause a loss of tracks or a bias in the statistics about cell motions.

Stage shift removal can be approached on the level of images [57, 97], or on the level of detected cells [24]. In the former case, stage shift removal can be formulated as an image registration problem.

Image registration for two images refers to finding a transformation of the second image that maximizes some matching criterion with the first image. In our case, it often suffices to consider only linear transformations: translation and rotation. Image registration is a large area of research, and we refer interested readers to a survey of registration methods [15, 54]. Here we present a basic method for image registration based on cross-correlation of pixel intensities (e.g., see [9]).

Suppose that we have a larger image $I_{fix}$ of size $(M+L) \times (N+L)$ called a fixed image, and a smaller image $I_{mov}$ of size $M \times N$ called a moving image. For example, the fixed image can be the image taken at the first time point, and the moving image can be a cropped image from the second time point. We crop the second image, because we shift the smaller image within the larger and compute cross-correlation for the overlapped area at each displacement. For a
given displacement \((i, j)\), the **unnormalized cross-correlation** is defined as

\[
R(i, j) \equiv \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I_{\text{fix}}(x + i, y + i) \cdot I_{\text{mov}}(x, y), \quad i, j \in [0, L].
\] (2.8)

One can then aim to find such a displacement \((\hat{i}, \hat{j})\) that maximizes \(R(i, j)\) over \(i, j \in [0, L]\). However, unnormalized cross-correlation is not a reliable metric, because there can be a part of \(I_{\text{fix}}\) that has nothing in common with \(I_{\text{mov}}\), but happens to have high (bright) pixel values. For example, if \(I_{\text{fix}}(x + i', y + i') = \max_{x,y} I_{\text{mov}}(x, y)\) for some \((i', j')\) and \(x = 0, \ldots, M - 1, y = 0, \ldots, N - 1\), then the unnormalized cross-correlation will be maximized at \((i', j')\), which is not necessarily a correct answer.

Therefore, it is more reliable to use **normalized cross-correlation** defined as

\[
R^2_N(i, j) \equiv \frac{R^2(i, j)}{\left[ \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I^2_{\text{fix}}(x + i, y + i) \right] \left[ \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I^2_{\text{mov}}(x, y) \right]}, \quad i, j \in [0, L].
\] (2.9)

A grid search can then be performed in order to find \((i^*, j^*)\) that maximize \(R^2_N(i, j)\) over \(i, j \in [0, L]\).

Alternatively, cell segmentation can be performed independently on each frame without registration. In this case, for each frame a set of cell centroid locations \((x, y)\) can be obtained, and stage shift can be removed by subtracting the mean displacement component over all centroids \[24\]. Let \(S_t\) be a set of centroid locations in frame \(t\). Then the \(x\) component of the mean displacement can be computed as

\[
dx = \frac{1}{|S_{t+1}|} \sum_{x_i \in S_{t+1}} x_i - \frac{1}{|S_t|} \sum_{x_j \in S_t} x_j,
\] (2.10)

and a similar formula applies for the \(y\) component.

Finally, it is worth mentioning that cell tracking itself can be formulated as an image registration problem where one aims to find a mapping between segmented cell areas. However, tracking is a very specific problem and there has
2.3 Cell Segmentation

Basic image processing methods such as those described above are often used as auxiliary steps in cell segmentation methods. A cell segmentation method takes image $I(x, y)$ as input and produces a segmentation mask $S(x, y)$, such that each element of the mask $s_{x,y} = 0, 1, \ldots, C, s_{x,y} \in S$. Here $0$ denotes segmented background area, and other values are the indexes of detected cells (Figure 2.9).

Cell segmentation is an essential image analysis step in cell tracking systems (CTS). Given a segmentation mask, it is straightforward to obtain a range of cell statistics, including cell number, cell sizes and intensity levels, and morphological measurements of cells such as roundness or elongation. Moreover, the results of cell segmentation form a basis of the subsequent cell tracking step, and the accuracy of cell segmentation affects the accuracy of tracking [67, 71].

There are two aspects of the accuracy (or performance) of a cell segmentation algorithm [25, 37, 79]. First, it is the accuracy of cell boundaries estimation with
Figure 2.10: A bright field cell image processed by an automated segmentation algorithm. Outlines made by the algorithm are shown in red. There are four types of detection errors: (i) under-segmentation, i.e., representing more than one cell with a single segmented object; (ii) over-segmentation, i.e., representing one cell with multiple segmented objects; (iii) spurious detection (segmented object); and (iv) missing detection, i.e., a failure to segment a cell. Original image courtesy of Hodgkin Lab, WEHI.

respect to a boundaries drawn by a human. Second, it is the number of cell detection errors. There are four types of detection errors: (i) under-segmentation, i.e., representing more than one cell with a single segmented object; (ii) over-segmentation, i.e., representing one cell with multiple segmented objects; (iii) spurious detection (segmented object); and (iv) missing detection, i.e., a failure to segment a cell (Figure 2.10). The performance of automated segmentation algorithms is usually measured with respect to a ground truth, that is, cell outlines provided by a human. A further discussion on measures for the performance of cell segmentation is presented in Section 2.5.

The performance of manual segmentation can be estimated from the disagreement of ground truth data prepared by different humans for the same set of images. In general, automated cell segmentation is a challenging problem, and there exist sets of real cell images where manual segmentation outperforms automated methods [25].

There are two sources of difficulties for automated segmentation algorithms. First, within a single experiment, there can be a variation in cell sizes and intensity levels, as well as cell touching and occlusion (Figure 1.3). Second, there can be a large variation in cell morphology and imaging conditions across experiments (Figure 2.11).
2.3. CELL SEGMENTATION

Figure 2.11: Variability in cell appearance across and within videos. The figure shows fragments of a fluorescent image of U2OS cells (left, from Peng et al. [102]), a fluorescent image of NIH3T3 cells (middle, from Coelho et al. [25]), and a bright field image of neural progenitor cells (right, from Al-Kofahi et al. [6]).

This variability, as well as the importance and difficulty of the cell segmentation problem, resulted in a wide range of proposed segmentation methods.

Finally, we note other problems relevant to cell segmentation. Cell detection refers to locating cell centroids in images, whereas cell segmentation refers to locating cell boundaries. Every cell segmentation algorithm can be trivially turned into a cell detection algorithm, by extracting the centroids of the resulting segments. However, if only cell detection and not a segmentation is necessary, a simplified segmentation algorithm can be used (e.g., smooth the image and find local intensity maxima). The results of cell segmentation provide information about cells (e.g., cell area or brightness) that is not available from cell detection. On the other hand, cell detection still allows for cell counting, and in certain cases, it can be sufficient for accurate cell tracking [71]. Furthermore, in Section 3, we introduce a novel concept of semi-segmentation.

Traditionally, researchers focused on developing cell segmentation methods as they make it possible to obtain more information. In the next section, we review previous cell segmentation methods.

2.3.1 Related Work Overview

Cell segmentation is a special case of image segmentation, and many cell segmentation methods are aligned with broad classes of image segmentation techniques, such as thresholding or edge detection [99]. In this section, we only review methods that were either specifically designed for cell segmentation...
or tested on cell images (including but not limited to methods described in [25, 41, 58, 97]). As we demonstrate below, our selection of algorithms represents a large variety of basic approaches to image segmentation in general.

There have been a number of reviews of cell segmentation methods [72, 86, 97, 108]. However, in this thesis, we discuss the previous approaches from a fresh perspective. We first outline steps of a meta-algorithm for segmentation, and then discuss existing methods in relation to this meta-algorithm. The advantage of such a perspective is that it allows us to inspect the pros and cons of the existing algorithms in a systematic manner, and identify a gap that we then address with a novel algorithm presented in Chapter 4.

Before presenting the meta-algorithm, we note that the existing segmentation approaches often comprise pipelines with a mixture of techniques. A typical segmentation pipeline can comprise: obtaining a background mask via thresholding, removing noise via blurring, applying a watershed algorithm as a core segmentation step, and filtering out small regions [25]. For each such pipeline step there is a number of available techniques and their variations (e.g., blurring can be achieved using a Gaussian kernel or a median filter presented in Section 2.2.3).

In this section, we categorize methods based on their core segmentation step. The differentiation between the core step and auxiliary steps is not always clear, and, for each method that we cover in this section, we clarify what we mean by the core step. Based on their core steps, we group the previous methods into a few basic classes: thresholding, watershed, active contours, region splitting and merging.

We now note that, similar to many other problems in computer science, on a generalized level, cell segmentation can be approached using the following meta-algorithm.

1. Define a fitness function that scores a candidate image segmentation.

2. Given the input image, find a set of candidate solutions.

3. Choose the solution that maximizes the fitness score.

Here, the fitness function captures prior knowledge about the cells in the given
image. It can be difficult to define a fitness function on the level of the image, but in practice it is often possible to define a fitness function on the level of individual candidate cell segments (connected image regions). For example, one can define a function that takes a connected region as input, and then produces a likelihood of the region representing a cell by looking at morphological properties of the region, such as area, roundness, elongation, number of holes, and so on.

As we discuss in the Introduction, there is a great variety of cell types and imaging conditions. Therefore, desirable properties of a cell segmentation algorithm are (i) the ability to redefine the region fitness function for a particular image, and (ii) to have this function involved in the solution finding process (step 2 of the meta-algorithm). We now look into major classes of segmentation algorithms and discuss to what extent these properties are satisfied in each group of methods.

### 2.3.2 Thresholding

Thresholding is the simplest method for cell segmentation. In this method, each image pixel is classified into a background or foreground class, and then all connected foreground pixels form segmented objects. It is often common to perform a selection on the resulting objects, that is, to discard objects that do not satisfy certain criteria [23]. For example, one can discard small or dark objects, as they are likely to represent noise. We note that the selection step can be applied to any of the segmentation methods, and we do not consider this step in further discussion.

Thresholding methods are often called histogram-based methods, because the threshold is essentially imposed on a histogram of image intensity levels (Figure 2.12). The key decision that one needs to make for thresholding is selection of the threshold value. For example, this value can be tuned directly [37], but it is more common to choose a method for threshold selection. A popular method that is data-dependent and does not require a manual setup of a threshold level is Otsu thresholding [96]. This method assumes that pixel intensity

---

1Please refer to Appendix A.1.1 for a discussion on pixel connectivity.
values come from two classes: background and foreground. The threshold is then selected as a value that minimizes empirical intra-class variance (the same value also maximizes empirical inter-class variance).

Figure 2.12: An original gray-scale image (left), its histogram and the threshold value (middle), and the binarized image produced with the threshold (right). In the middle image the x-axis shows intensity levels and the y-axis shows the number of pixels having that intensity level. The red line denotes the threshold value. This thresholding method was not able to separate touching cells. Original image courtesy of Hodgkin Lab, WEHI.

Another method used for threshold selection for cell segmentation is mean thresholding, that is, setting the threshold to the mean intensity value [25]. Finally, one can assume the presence of 3 to 5 classes for pixel intensities (e.g., background, nuclei and cytoplasm), perform k-means clustering of intensity values, and set the threshold as a boundary for the first class (i.e., background) [32].

The above methods find a single threshold for the entire image (often called a global threshold). It is also possible to apply thresholding adaptively: set different thresholds for different parts of the image [6]. Adaptive thresholds can help in the situations when the illumination of the frame is not uniform. An alternative to adaptive thresholds is using one of the illumination compensation methods presented in Section 2.2.1.

With regards to our meta-algorithm (Section 2.3.1), once a threshold or threshold selection method is fixed, a set of candidate solutions (segmented cells) is fixed as well. At the same time, choosing the best threshold selection method for a given error is a trial-and-error process. Thresholding itself does not have a direct connection with the fitness function that captures cell appearance (meta-algorithm, step 1), and this is why applying a post-processing selection is crucial
2.3. CELL SEGMENTATION

for thresholding.

Despite the above limitations, thresholding methods are popular because of their simplicity. Eventually, there can be images where simple global thresholding performs reasonably well \[37\]. Finally, thresholding is often used as an intermediate step for more sophisticated methods such as watershed algorithms.

2.3.3 Watershed-based Segmentation

Watershed-based algorithms are amongst the most popular methods for cell segmentation \[25, 97, 105\]. In image analysis, watershed algorithms were first applied for contour detection \[12\], later adopted for object segmentation \[118\], and finally for cell segmentation \[105\].

As can be guessed from the algorithm’s name, the idea of a watershed algorithm is borrowed from topography. A gray-scale image is viewed as a topological relief, where gray-scale levels encode elevation levels (Figure 2.13). One can then mentally immerse the relief in water and notice the water levels. The water first fills the lowest valleys and will remain within enclosed regions called catchment basins. As the water level increases, two or more catchment basins can meet, and a watershed line is placed at every meeting point. Once the whole relief goes under water, the set of watershed lines forms contours that can then be used as a basis for image segmentation: each catchment basin becomes an object. The method works because in images, objects are often surrounded by a contrast boundary that serves as a dividing range for a valley that represents an object.

In its simplest form, the “flooding” of valleys starts as soon as the water level reaches the bottom of a valley, in other words, from the local minima of the image (or maxima, depending on the definition of peaks and valleys). This implies that each local minima gives rise to a separate catchment basin. In real images, many local minima may be present due to noise, leading to spurious catchment basins, and, as a result, to an over-segmentation. This problem can be solved either by image smoothing (Section 2.2.3), region merging (Section 2.3.4) or choosing a different domain for building watersheds, as a watershed algorithm can be run on a transformation of the original image \[83, 97, 118\]. For
2. BACKGROUND

Figure 2.13: In a watershed-based segmentation algorithm, an image is viewed as a topological relief where intensity levels encoding elevation. The image is then immersed in water, with its valleys flooding first and forming water catchment basins. As the water level rises (solid blue lines), two or more basins can meet and a dam is built meeting points (dashed red lines). The dams represent detected contours of objects.

Figure 2.14: A background mask obtained as a result of thresholding (left), a distance transform of the mask (middle), and a result of a watershed-based segmentation (right). Colors encode different segmented objects.

example, one can use a gradient transform or distance transform of the original image\(^2\) or even a transformation based on a wavelet representation [97].

In its simple form, a watershed algorithm produces a set of edges represented by watersheds. In other words, it is an edge detection algorithm that, for example, does not separate objects of interest from the background. Therefore, in order to design a cell segmentation algorithm, one first needs to produce a background mask [25, 97] (Figure 2.14).

Choosing a method for locating watershed seeds, as well as the background mask, are two prerequisites of watershed-based methods. In addition, one can decide to run watershed on a transformed image, rather than the original im-

\(^2\)Please refer to Appendix A.1.2 for more information on image transformations.
2.3. CELL SEGMENTATION

While it is convenient to have a certain level of flexibility in designing a segmentation algorithm, the decisions listed above, are, in general, hard to directly relate to the appearance of cells. Consider the situation when, for a given image, it is known that cells have convex ellipsoid shapes, and their sizes fall into a certain range. However, this knowledge does not directly guide decisions such as follows: Use Gaussian blur or median filtering for seed identification? Use Otsu or some other method for image thresholding? Use Sobel kernels or some other method for computing a gradient image?

Furthermore, once all decisions are made, a watershed algorithm uniquely defines a candidate solution, that is, a set of segmented cells. The prior knowledge about cells is not incorporated in the process of generating segmentation hypotheses. This property along with the prerequisites of watershed algorithms are the main limitations of this approach. In order to overcome the former limitation, and partly address the need for prerequisites, researchers proposed more advanced techniques based on watershed algorithms. We present these and related techniques in the next section.

2.3.4 Region Growing and Merging

Region growing and merging methods provide a more straightforward way to incorporate a region fitness function. Recall that the region fitness function $\mathcal{L}(R)$ takes a connected image region (candidate cell segment) as input and produces a likelihood of the region representing a real cell as output. This function provides a way for a user to incorporate prior knowledge about cell morphology, and it is desirable to have this function to guide the process of finding cells.

Generally, the region growing and merging approach consists of two steps: (i) find seeding regions, (ii) grow seeding regions or merge neighboring regions so that the fitness value for the resulting region increases $[80, 123]$. Region growing refers to adding some of the adjacent points to the region, and merging refers to combining two or more regions into one. It is also possible to add a region splitting step $[121]$.

Seeding regions provide a starting point for growing or merging process. This is similar to providing seeds for watershed algorithms, but in the case of
watersheds, seeds are usually understood as points, whereas in the case of region growing and merging, seeds are regions or points. In fact, watershed algorithms can be used to obtain seeding regions \[80, 105, 121\]. Seeding regions can be obtained using some other method, for example, a method based on a Laplacian of Gaussian filter \[7\] (see Appendix A.1.3 for a description of a Laplacian of Gaussian filter).

An advantage of the region growing and merging approach is that the fitness function is involved in the process of constructing the final solution. A drawback of the approach is the need for seeding regions, as the choice of a seeding method is not directly related to the region fitness function.

### 2.3.5 Active Contours

In contrast to the previously described approaches, active contours or snakes based methods start by defining an energy functional (a fitness function in our meta-algorithm), and use partial differential equations to find a solution that minimizes the energy (maximize fitness) \[20, 39, 73, 107\]. An active contour \(C\) is a curve on a 2-dimensional surface that can be defined as a spline or a set of points. The energy of the contour is a functional \(E(C) \rightarrow \mathbb{R}^+\) that takes a contour as input and produces a positive real number. The convention is that having lower energy values is better.

As an analogy, consider a small rubber band and a glass. One can stretch the band and place it around the glass leaving some distance to the glass walls (Figure 2.15), and then release the band. Certain forces will drive the band towards shrinking. At some point in time, the band will hit the glass walls and wrap around the glass. Now, repulsive forces from the walls will prevent the band to go through the glass, and the band will reach the equilibrium. In this analogy, the rubber band is like an active contour, the glass is like an object in an image, and the rules of physics are defined via the energy functional in the case of active contours.

A generalized active contour method works as follows:

1. Contours are placed in their initial positions;
2.3. CELL SEGMENTATION

Figure 2.15: An active contour (dotted red line) at some initial position (left) and in a position that minimizes its energy (right). The blue circle represents some object presented in an image. In this example, the energy of the contour is defined such that smaller internal areas are encouraged. Thus, in the initial position of the contour, there are “forces” (here gradients of the energy) that drag the contour inwards. Another part of the energy functional discourages crossing image edges, and in the resting position the contour minimizes its energy by reaching an equilibrium between different forces.

2. Several iterations are then performed (this simulates the time in real physical systems);

3. At each iteration, partial derivatives of the energy functional might guide the contours to new positions;

4. The algorithm stops when contours are no longer moved;

5. The contours then define the segmented regions.

Within the active contours based approach, one needs to make the following major decisions: (i) how to describe a contour; (ii) how to define the energy functional; and (iii) how to initialize the contours. We briefly comment on each decision below.

(i) A contour can be described as a spline with a set of control points. However, it has become common to represent contours via level sets [20, 10, 95]. Level sets allows us to represent a contour on a fixed grid in a non-parametric manner [95]. Also, with level sets it is easy to follow various contour deformations such as splitting and merging or developing holes (Figure 2.16). A contour
is represented as a set of points where an auxiliary function $\varphi(x, y)$ of two variables intersects the plane $z = 0$. That is, $C = \{(x, y) \mid \varphi(x, y) = 0\}$. The evolution of the contour is implemented via the evolution of the auxiliary function (called the level set function).

(ii) The energy functional is usually defined as $E(C) = E_{\text{int}}(C) + E_{\text{ext}}(C)$, where each term, in turn, consists of a weighted combination of further terms. $E_{\text{int}}(C)$ is called internal energy and it comprises terms defined on a contour itself, such as the length of the curve, or the smoothness $\int_0^1 C(s) \, ds$. In contrast, $E_{\text{ext}}(C)$ is an external energy term, that depends on the image features. This can include, for example, a variance of pixel intensities within the contour $\int_{s \in \Omega} (\mathbb{I}(s) - u(\Omega)) \, ds$, where $\Omega$ is a region bounded by contour $C$ and $u(\Omega)$ is the mean pixel intensity within $\Omega$.

(iii) One approach to initialize active contours in a particular frame is to set them to found cell outlines in the previous frame. (An independent cell
2.3. CELL SEGMENTATION

Segmentation algorithm is required for the first frame.) Another approach is to start with random contours [113]. An arbitrary energy functional does not necessarily lead to a convex optimization problem, and therefore the solution, in general, depends on the initialization step (the solution can be stuck in a local minimum). However, it is possible to define an energy functional that results in a convex optimization problem [10].

In relation to the meta-algorithm (Section 2.3.1), we note that active contour based methods are directly designed to find a solution that maximizes the fitness function, expressed via the energy functional. This is an advantage of this approach. However, an arbitrarily defined energy functional does not necessarily lead to a convex optimization problem. Furthermore, this method requires taking derivatives of the energy functional, and this might limit the use of certain possible functions, especially those involving logical predicates, such as “the contour (cell) is good if its area is more than N pixels, if the cell is not at the image boundary, and if the average brightness of the cell is at least B”. Therefore, it is, in general, not easy to re-define and tailor the energy functional for a particular video, in order to incorporate prior knowledge specific to the video.

The method of Jung and Kim [68], does not use an active contour formulation, but fits in this group, because it seeks for an optimum of a fitness function. However, their fitness function relies on an elliptical cell model, which is not always feasible (e.g., lymphocytes or neural progenitor cells can take irregular shapes).

2.3.6 Summary of Cell Segmentation Methods

We summarize widely used classes of cell segmentation methods in Table 2.1. In the table, the “Searching process” column characterizes to what extent a user-defined fitness function can influence the resulting solution. Note that we omit the post-processing selection step (cell segments returned by any algorithm can be filtered according to user-defined criteria). From the summary, we conclude that there are a lack of methods that (i) have few or no prerequisites, (ii) have their solution searching process guided by the user-defined fitness function, and (iii) provide flexibility in defining and customizing fitness functions. We
address this gap by developing a novel method for cell segmentation in Section 4. In the rest of this section, we mention less common cell segmentation approaches, and conclude our survey of segmentation algorithms.

In a template matching approach, cells can be parametrically described (e.g., circles with radius $R$), and then at each point of the image, the local neighborhood can be matched against the template [69]. As a result, another image is produced, where each pixel has a value proportional to the likelihood of locating the template at this position. Essentially, this method produces what is called a Generalized Hough Transform. In principle, this method allows us to define fitness functions as template descriptions. However, certain fitness functions are difficult to use for template matching (e.g., “a cell is any convex region larger than $N$ pixels”).

In a learning approach, “evident non-cell pixels” of the image are first discarded using a pre-processing step [100]. The remaining pixels are then classified into background and cell pixels. Finally, cell pixels are grouped into objects using another classifier capable of predicting whether two pixels belong to the same object. The two classifiers are trained from a manually labeled image. Prerequisites of this approach are the pre-processing step and the availability of labeled training image(s).

In an edge-based approach, edges are first detected using a Canny edge de-
2.4. CELL TRACKING

tector\textsuperscript{3} Regions enclosed by edges are then filled, and certain regions are discarded based on their size and morphology \cite{79}. Similar to thresholding and watershed-based approaches, edge detectors, in general, are not guided by the fitness function. Instead, one has to rely on the selection step.

In a graphical model approach, pixels are represented with vertices of a Bayesian network, and the neighborhood relation between pixels is represented with edges \cite{21}. Each pixel/vertex can then be labeled as either background or cell. The likelihood of a graph labellings is defined using visual cues from the image (e.g., brightness). Such an approach tends to be computationally slow \cite{21}.

Finally, we note that this thesis focuses on the segmentation from 2D images, whereas in some cases, multiple optical image slices are available. In many cases, cell segmentation algorithms are transparent to image dimensionality (e.g., a 3D watershed algorithm is used in \cite{42}). In other cases, specific algorithms are used to tackle multiple image slices (see a graph matching approach in \cite{82}).

In summary, cell segmentation is a special case of object segmentation and there has been a wide range of proposed methods, including methods specifically designed for cell images. Cell segmentation is an important image analysis stage, and segmentation results form a basis of subsequent cell tracking. We review cell tracking methods in the next section.

2.4 Cell Tracking

We first recall that by a cell tracking system we mean the complete image processing and data analysis pipeline including preprocessing, segmentation and cell association. In contrast, by a cell tracker (or simply tracker) we mean only the last step of the pipeline, that is, cell association.

As a result of cell segmentation, located cells in video frames can be represented with feature vectors called measurements $\mathbf{m}_{f,i} = \{ x_{f,i}^{(1)}, \ldots, x_{f,i}^{(K)} \}$, where

\textsuperscript{3}A Canny edge detector is based on an edge detection operator, such as the Sobel operator described in Appendix A.1.2
2. BACKGROUND

$f$ is the frame index, $i$ is a measurement index in frame $f$, and $x^{(k)}_{f,i}$ is a feature, (e.g., a centroid coordinate or an estimate of cell area). We denote a set of measurements for a video as $\mathcal{M}$. In practice, the measurements can contain errors (e.g., a missed cell or a spurious measurement), and trackers can decide to create artificial measurements that were not originally in the detection. This can result in an extended measurement set $\mathcal{M}^+$ (see further discussion in Section 2.4.1).

Given measurement set $\mathcal{M}$, a cell tracking task is to produce a set of tracks, where each track (or trajectory) is a sequence of locations where locations have consecutive time indices, for example, $T = \{\bar{m}_{3,i}, \bar{m}_{4,j}, \bar{m}_{5,k}\}, \bar{m}_{t,s} \in \mathcal{M}^+$. Each track represents the locations of a single cell throughout the video. The tracks do not necessarily start at the first frame, nor end at the last frame, as new tracks can appear in the middle of the video as a result of cell division or due to entering a field of view. Similarly, a track can end in the middle of the video due to a cell death or a cell leaving the field of view.

In videos, cells can divide one or more times and biologists are often interested in reconstructing cell lineage trees [6, 61]. Therefore, in addition to tracks, it is common to require the cell tracker to produce lineage trees that express relations between the tracks, as well as detected cell events (such as divisions or deaths, Figure 2.17). There is no established common format for lineage trees. One way to encode them is to use an additional record for each track. The record shows whether the track has been terminated due to a detected cell death, end of video or leaving the field of view, or the track has split into two other tracks whose IDs are recorded.

2.4.1 Measurement Errors

One of the major challenges of cell tracking is dealing with an imperfect cell segmentation. As we mention above, in practice, the measurement set $\mathcal{M}$ can contain errors. Previously, we introduced the concept of cell segmentation ac-

---

4In some cases, cell segmentation and tracking is performed in a combined step. In this case, the inputs to the problem are preprocessed video frames rather than a set of measurements. See Section 2.4.3 for examples of such approaches.
Figure 2.17: Sample lineage trees (left) summarize cell events: cell A moves between frames 1, 2, and 3, and divides between frames 3 and 4; cell D moves between frames 1, 2, and 3, and dies between frames 3 and 4. In a sample measurement set resulting from these events (right), $\vec{m}_{t,i}$ are measurements, where $t$ is a frame index and $i$ is a measurement index within a frame. In this example, the correct track for cell A is $\{\vec{m}_{1,1}, \vec{m}_{2,1}, \vec{m}_{3,1}\}$, for cell B it is $\{\vec{m}_{4,1}\}$, for cell C it is $\{\vec{m}_{4,2}\}$, for cell D it is $\{\vec{m}_{1,2}, ?, \vec{m}_{3,3}\}$. There is a false negative error (cell D is not detected) in frame 2, and a false positive error (spurious measurement) in frame 3.

For the purposes of cell tracking, we distinguish between two classes of errors that we call detection errors and noisy measurements. Detection errors comprise false negative and false positive errors (Figure 2.17). A false negative error (or a missed cell) is a situation when a cell in a frame is not represented by any measurement. A false positive error (or a spurious measurement) is a measurement that does not correspond to any true cell in the frame.

Noisy measurements refer to the fact that the values of measurements in $\mathcal{M}$ can deviate from true values for the corresponding cells. For example, one can assume that somehow true centroid coordinates for a cell are defined to be $(x, y)$, whereas the corresponding measurement has values $(x', y')$.

Cell trackers that admit the existence of noisy measurements attempt to smooth tracks by means of stochastic filters (e.g., the Kalman filter described in Appendix A.2.1). As a result, artificial measurements can appear (Figure 2.18) which can result in an extended measurement set $\mathcal{M}^+$ that comprises both original and artificial measurements. We discuss some examples of such trackers in Section 2.4.4.
Figure 2.18: Measurement set \( \mathcal{M} \) comprises measurements \( \vec{m}_{1,1}, \ldots, \vec{m}_{4,1} \) in four frames. However, the tracker decides to smooth the track by means of the Kalman filter. As a result, artificial measurements \( \vec{m}_{1}^{*}, \ldots, \vec{m}_{4}^{*} \) are implied by the tracker. Together measurements \( \vec{m}_{1,1}, \ldots, \vec{m}_{4,1} \) and \( \vec{m}_{1}^{*}, \ldots, \vec{m}_{4}^{*} \) constitute an extended measurement set \( \mathcal{M}^{+} \).

The difference between detection errors and noisy measurements can be subtle, because if a missed detection and a spurious measurement present in a single frame, one can view the spurious measurement as a valid one but with a large deviation from the true cell location. In this situation, the distinction between the two classes of errors is subjective.

In practice, it is often convenient to assume that the noise in measurements is negligible, and that only detection errors need to be accounted for \([6, 71, 72, 97]\).

### 2.4.2 Related Work Overview

Overall, the field of cell tracking resembles that of cell segmentation in several aspects. For example, cell tracking is a specific case of the object tracking problem, and there have been adaptations of general methods as well as methods designed specifically for cell tracking. In our review we focus on methods that were either specifically designed for cell tracking or have been tested in cell tracking applications.

Furthermore, there is a large variability in cell dynamics across different videos and even within individual videos. As we discuss in Section 2.6, this constitutes the main challenge of selecting and tuning of both segmentation and tracking algorithms for a specific video.

Finally, similar to segmentation methods, many cell tracking methods consist of a mixture of different techniques \([23, 86]\). For example, the tracker of Li et al. \([79]\) uses active contours and Kalman filters. With such a variety and
2.4. CELL TRACKING

heterogeneity of methods, it is hard to develop an ultimate taxonomy of the algorithms. Instead, in this thesis we categorize the methods into three overlapping groups.

In the first group, we list methods that perform segmentation and tracking in a combined manner. A characteristic feature of these methods is that the input for such trackers is cell images rather than, or in addition to, the measurement set (segmented cells). In the second group, we list methods that use probabilistic techniques for tracking. Finally, in the third group, we list methods that approach tracking from an optimal assignment perspective.

2.4.3 Combined Segmentation and Tracking

This is a diverse group of methods that all share a common property: the methods are capable of simultaneous segmentation and tracking, and a separate cell segmentation step is, in general, not required (although is sometimes still used to improve results). Methods in this group can be further divided into four sub-groups.

First, some methods treat a time series of two-dimensional images as a three-dimensional volume \[65, 98\]. In such a representation, tracks comprise tubes whose boundaries are defined by cell outlines at different time points. A segmentation method is used to identify these tubes, and hence tracks. For example, Padfield et al. \[98\] use active contour based segmentation (Section 2.3.5). A separate appearance-based model is used for connecting tubes of the mother and daughter cells after division. Inglis et al. \[65\] use graph-based segmentation that resembles splitting and merging approaches (Section 2.3.4).

Second, some trackers use active contours in order to segment and track cells simultaneously \[40, 79\]. In these methods, the active contours (Section 2.3.5) are initialized from the previous frame and evolved to fit the appearance in the current frame. A separate segmentation procedure is needed to locate cells in the first frame. As an example of methods from this sub-group, consider the method of Li et al. \[79\]. They define an energy functional with three terms: region, edge and motion. The region term represents the probability of each pixel belonging to the segmented cell region; the second term is used
to drag the contour towards edges in the image; and the motion term is used to adjust the contour evolution according to a predefined cell motion model. In some cases, individual cell tracks can be broken into segments (e.g., at the points where a cell makes an abrupt move). A separate track linking module is used to combine track segments into one, as well as to link mother and daughter tracks during division. It is worth mentioning that the method of Li et al. also uses an independent cell segmentation module and combines the results of the segmentation module and contour evolution module.

Third, some methods use a mean shift algorithm for tracking \cite{5, 29}. The mean shift algorithm is a method for finding the maxima of a probability density function (PDF) from a discrete sample (see Appendix A.2.2 for details). In mean shift based methods, the PDF represents the probability of a cell centroid being located at a particular position. These methods locate the initial density mode estimates at the previously detected cell positions, and obtain new positions by means of the mean shift algorithm. In fact, Debeir et al. \cite{29} use multiple coupled mean shift procedures with different kernels, where each kernel is tailored to track a specific feature of the cell appearance. Adanja et al. \cite{5} apply mean shift in a three-dimensional correlation volume, where each voxel value represents the correlation between the original three-dimensional frame and a cell template placed at that voxel.

Finally, particle filter based methods employ a Bayesian formulation for tracking \cite{18, 31, 93}. These methods operate using cell images as input and do not require a separate cell segmentation step. We describe these methods in more detail in the next section.

2.4.4 Bayesian Approach to Tracking

Object tracking outside of microscopy has a long history or research and there has been a well established Bayesian formulation of the object tracking problem \cite{8, 19, 88, 91}. This formulation is often used by cell tracking methods \cite{18, 31, 49, 50}.

We first start with single cell tracking. We assume that at each frame $t$ the cell is characterized by a state vector $\vec{s}_t$ (e.g., cell coordinates and an estimate
of area). Furthermore, a cell segmentation algorithm produces measurements \( \vec{m}_t \) for each frame. The Bayesian tracking approach consists of a recursive estimation of the posterior distribution \( p(\vec{s}_t|\vec{m}_{1:t}) \), where \( \vec{m}_{1:t} \) is a shorthand for \( \{\vec{m}_1, \ldots, \vec{m}_t\} \). This estimation is obtained by assuming that cell state \( \vec{s}_t \) evolves as a known Markov process \( D(\vec{s}_t|\vec{s}_{t-1}) \). Moreover, it is also assumed that the measurement model is given as a likelihood \( L(\vec{m}_t|\vec{s}_t) \). Under these assumptions, the estimation of the posterior distribution is performed in two steps: prediction and update.

In the prediction step, the estimated posterior densities from the previous frame \( p(\vec{s}_{t-1}|\vec{m}_{1:t-1}) \) are combined to obtain a prior density using the Chapman–Kolmogorov equation

\[
p(\vec{s}_t|\vec{m}_{1:t-1}) = \int D(\vec{s}_t|\vec{s}_{t-1}) \cdot p(\vec{s}_{t-1}|\vec{m}_{1:t-1}) \, d\vec{s}_{t-1}.
\] (2.11)

In the update step, Bayes rule is applied to the prior density in order to obtain the posterior density

\[
p(\vec{s}_t|\vec{m}_{1:t}) \propto L(\vec{m}_t|\vec{s}_t) \cdot p(\vec{s}_t|\vec{m}_{1:t-1}).
\] (2.12)

The above method is not computationally tractable and cannot be directly applied in practice. One solution is to impose additional constraints on the problem. For example, one can require \( D(\vec{s}_t|\vec{s}_{t-1}) \) and \( L(\vec{m}_t|\vec{s}_t) \) to be in a form \( \vec{x}_t = \vec{F}\vec{x}_{t-1} + \vec{w}_t \), where \( \vec{w}_t \) has a multivariate normal distribution. Under these additional assumptions, the Kalman filter (Appendix A.2.1) is a well known solution, which is optimal in a mean square error sense.

Another solution is to derive an approximate solution using a particle filter (Appendix A.2.2). The particle filter does not impose constraints on \( D(\vec{s}_t|\vec{s}_{t-1}) \) and \( L(\vec{m}_t|\vec{s}_t) \), but instead represents an approximation of the posterior \( p(\vec{s}_t|\vec{m}_{1:t}) \) by means of \( N \) random weighted samples \( \{\vec{x}_t^{(i)}, \vec{w}_t^{(i)}\}_{i=1}^N \), \( \sum_{i=1}^N \vec{w}_t^{(i)} = 1 \), as

\[
p(\vec{s}_t|\vec{m}_{1:t}) \approx \sum_{i=1}^N w_t^{(i)} \cdot \delta(\vec{s}_t - \vec{x}_t^{(i)}),
\] (2.13)
where $\delta(\cdot)$ is the Dirac delta function.

Both Kalman and particle filters imply a fixed motion model $D(\tilde{s}_t | \tilde{s}_{t-1})$. This assumption is not always feasible in cell tracking applications as cell dynamics can vary over time. The interacting multiple models (IMM) filter runs different single-model filters in parallel and is capable of supporting a weighted combination of models. The vector of weights evolves over time according to a finite state Markov chain. For example, the method of Li et al. [79] uses an active contour based method to simultaneously segment and track cells. Their energy functional includes a motion term, which is based on motion predictions obtained by an IMM filter.

There have been various adaptations of the Bayesian approach to the multiple cell tracking setup. The method of Carranza et al. [18] models the state of each cell $m$ independently as $\tilde{s}_{t,m}$, and defines a total posterior as a mixture of individual cell posteriors. However for all cells in each frame they only have one measurement which is the input image. The likelihood $L(\tilde{m}_t | \tilde{s}_{t,m})$ is computed for each cell on the same image. A cell segmentation algorithm is essentially embedded in the particle filter.

The method of Delgado-Gonzalo et al. [31] comprises a combination of the active contour and particle filter approaches. Their method does not require a separate cell segmentation step. In their method, particles comprise parametric elliptical active contours called ovuscules, and standard weight propagation is replaced by an energy minimization routine. Multiple cells are tracked independently, and a special biphasic model is used to relate mother and daughter tracks after a cell division.

Alternatively multiple tracking can be viewed as a data assignment problem, where at each frame there is a set of previously established tracks and a set of measurements produced by an independent cell segmentation algorithm. In this formulation, a key step is to decide which measurement to use to update which track. For example, the method of Huth et al. [64] uses Kalman filters for individual cell tracking, and updates tracks with measurements selected based on a nearest neighbor principle. In contrast, the method of Genovesio et al. [49] uses an optimal linear assignment formulation with assignment weights derived using an IMM filter. We review major data assignment approaches in
2.4. CELL TRACKING

Figure 2.19: Given a set of previously established tracks up until frame $t - 1$, the task is to update the set of tracks based on measurements that arrived in frame $t$. As a result of the update, tracks can be extended ($1 - 1$), split ($2 - 2; 2 - 3$), or terminated ($3 - x$). This represents respectively a cell move, cell division, and cell death or loss of track. Additionally, new tracks can be initiated.

2.4.5 Assigning Measurements over Time

A data assignment approach to multi-object tracking comprises a large group of methods [7, 24, 36, 72, 97, 103]. Major techniques include nearest neighbor assignment, optimal linear assignment, multi-frame assignment, and probabilistic data association. Here we first introduce one of the assignment based formulations of the cell tracking problem. We then review a number of representative tracking methods. Different methods can deviate from the basic formulation of the problem.

Given a set of previously established tracks up until frame $t - 1$, the task is to update the set of tracks based on measurements that arrived in frame $t$. As a result of the update, tracks can be extended, split, or terminated (Figure 2.19). This represents respectively a cell move, cell division, and cell death or loss of track. Additionally, new tracks can be initiated.

One of the major challenges is to decide which measurement to use to update which track. An additional challenge is introduced by the fact that measurements can contain detection errors (spurious measurements and missing true measurements). Recall, that measurements can also contain measurement noise. Some methods use stochastic filters (introduced in the previous section) to smooth tracks after updating with a measurement [49, 64]. Other methods...
assume that there is no measurement noise and that resulting tracks comprise sequences of measurements connected by links [6, 97].

One of the simplest ways to assign measurements to tracks is to use nearest neighbor measurements. For example, Huth et al. [64] use a greedy assignment procedure where in each iteration the closest track – measurement pair is matched. However, measurements that are too far from tracks are not considered.

A more systematic approach is to use an optimal assignment formulation, where each track – measurement pair is given a cost, and integer programming is used to find an assignment that minimizes the total cost subject to constraints. In the method of Al-Kofahi et al. [6] costs for track updating and splitting are defined based on a Gaussian of the differences between the last track position and measurement. The further away the measurement is from the track, the higher the cost. Constraints prohibit solutions that are physically impossible under the given formulation. For example, in the method of Al-Kofahi et al., a track cannot split into more than two tracks.

Hypotheses of track termination or initiation can be incorporated into a linear assignment formulation by introducing a dummy track and a dummy measurement [76]. Assigning a track to a dummy measurement represents that a track is either terminated or temporarily lost.

Padfield et al. [97] propose an alternative linear assignment formulation where tracks can merge in addition to splitting. Merging represents cell occlusion in videos. In their method, the assignment problem is formulated as a minimum-cost flow problem in a bipartite graph appended with splitting and merging vertices. The minimum-cost flow problem is solved using linear programming.

Note that the above linear assignment formulation is optimal only between two subsequent frames. A globally optimal solution for the video must take into account measurements from all frames and consider all possible linking combinations. The number of possible track hypotheses grows exponentially and becomes intractable even after a few frames. A tradeoff between locally (inter-frame) and globally (for the video) optimal solutions can be achieved by looking into a multi-frame assignment or using heuristic track management.
2.4. CELL TRACKING

modules that look at measurements across several frames.

For example, Kirubarajan et al. [76] uses iterated multiple assignments to construct all possible assignment hypotheses across several frames, and Al-Kofahi et al. [6] use a separate heuristic track management module. When a track does not receive any measurement at frame $t$, the track is not discarded, but marked as “lost”. Lost tracks can receive measurements in subsequent frames and thus re-appear.

Finally, there is a group of multi-object tracking methods based on probabilistic data association [26, 48]. One of the most advanced methods in this group is Linear Joint Probabilistic Data Association (LJIPDA) [91]. For a detailed description of probabilistic data association methods please refer to [91] and the references therein. Here we briefly outline the LJIPDA algorithm.

This algorithm consists of three steps. In the first step, tracks are processed individually. For each track a priori probabilities that a track exists and that the measurements have originated from the track are computed. Only those measurements that fall within a certain Mahalanobis distance from the track are taken into consideration for the given track. The second step is measurement oriented. For each measurement and each track, probabilities that the measurements are visible to the track are computed. The probability of visibility is the complement to the probability that the measurement originated from some other track. The third step is track oriented. For each track, the assignment probabilities are calculated given the visibility probabilities.

2.4.6 Summary of Cell Tracking Methods

As we have illustrated in the previous few sections, there is a large variety of cell tracking approaches. Furthermore, each approach can employ a combination of different principles. In our review, we put the methods into three overlapping groups.

Methods in the first group perform segmentation and tracking in a combined step, and, in general, do not require a separate cell segmentation algorithm. However, these methods usually rely on smooth cell motions, when cell outlines overlap in subsequent frames, or at least there are no neighbor cells
in close proximity of a moving cell \[97\]. If cell outlines do not overlap or tracks of different cells often cross each other then spatio-temporal clustering methods can break cell tubes into segments. Furthermore, active contours and mean shift trackers initialized from previous cell positions can be confused and attracted to a neighboring cell contour, thus swapping the tracks.

Methods in the second group employ a Bayesian approach to tracking. These methods usually make strong assumptions on cell dynamics. IMM methods allow one to use multiple models, but each such model needs to be specified a priori. In practice, only a limited number of models is considered \[49, 79\]. At the same time, time lapse cell experiments are often performed to obtain new knowledge about cells, and it is not always feasible to require an a priori cell motion model. Furthermore, the classic Bayesian tracking approach presented in Section \[2.4.4\] in itself is not capable of multiple cell tracking. Some form of adaptation is required, and multiple cell tracking is often solved using measurement-to-track assignments.

Methods in the third group focus on establishing measurement-to-track assignments, as well as identifying other cell events such as divisions, deaths or cells leaving the field of view. These methods are often based on an optimal assignment between two frames supplemented with a track management module that takes into account a broader context of multiple frames.

This section completes our survey of cell segmentation and tracking algorithms. We are now ready to discuss performance aspects of CTSs, as well as some practical aspects of cell tracking in the next sections.

### 2.5 Performance of a Cell Tracking System

Ultimately the utility of a CTS depends on the quality of its results with respect to a particular analysis goal. For example, researchers can be interested in the change of the average speed or directionality of cells under different treatments \[59\] or in the distribution of cell lifetimes \[61\]. Given that the segmentation and tracking methods can usually be re-used across different kinds of assays for time lapse microscopy, it is not practical to test, for example, a segmentation
algorithm in every possible assay.

Whatever the goal of the assay is, one would expect the quality of the results to increase with an increased accuracy of cell segmentation and tracking steps (if tracking is needed). Therefore, it is feasible to develop general purpose accuracy measures for segmentation and tracking algorithms, and use these measures for validation of the algorithms. We have already introduced several definitions related to cell segmentation accuracy (Sections 2.3, 2.4.1). In this section, we summarize these definitions and also discuss the accuracy of cell tracking. Furthermore, we review previously proposed measures for cell segmentation and tracking accuracy.

There are three aspects of CTS accuracy (or performance). The first two aspects are related to cell segmentation, and the last aspect is related to cell tracking. We discuss each aspect in a separate section below.

### 2.5.1 Accuracy of Cell Boundaries

Segmentation performance can be understood as the accuracy of segmented cell boundaries. Note that from a segmented cell boundary researchers usually extract further morphological features such as centroid position, area and intensity. If we assume the existence of a true cell boundary then deviations from this boundary are likely to result in noisy measurements of morphological features (recall that measurement noise was discussed in Section 2.4.1).

In practice, the true cell boundary is estimated by human drawn cell outlines. The outlines can be encoded in a binary mask where zero pixels represent background and non-zero pixels represent internal cell areas. We call a manually produced mask a ground truth mask. The performance can then be measured by comparing a produced segmentation mask with a ground truth mask. If different ground truth masks are available from different people, one mask can be randomly chosen as a reference [58]. Furthermore, one can also compare ground truth masks obtained by different people in order to estimate inter-person variance [25]. Alternatively, one can derive an averaged ground truth as a consensus between multiple users [79].
The segmentation and ground truth masks can be compared using F-score\(^5\) defined as \(F = 2 \cdot \frac{TP}{2 \cdot TP + FN + FP}\), where \(TP\) is the number of pixels that have a non-zero value in both the ground truth and segmentation masks, \(FN\) is the number of pixels that are non-zero in the ground truth mask, but zero in the segmentation mask, and \(FP\) is the number of pixels that are non-zero in the segmentation mask, but zero in the ground truth mask \[37\]. Bergeest and Rohr \[10\] measure the accuracy using the Dice coefficient, but in their formulation the Dice coefficient is equivalent to F-score.

The F-score is bounded between 0 and 1, where 1 denotes perfect accuracy (the ground truth and segmentation masks are equivalent). The F-score is often expressed as the harmonic mean of precision \(\text{prec} = \frac{TP}{TP + FP}\) and recall \(\text{recl} = \frac{TP}{TP + FN}\), \(F = 2 \cdot \frac{\text{prec} \cdot \text{recl}}{(\text{prec} + \text{recl})}\). Instead of F-score, Dima et al. \[32\] use another similarity measure defined on precision and recall \(D = \sqrt{(1 - \text{prec})^2 + (1 - \text{recl})^2}\). Alternatively, the accuracy can be measured using the Jaccard index \(J = \frac{TP}{TP + FN + FP}\) \[25\].

There have been other performance measures for the accuracy of cell boundaries. For example, for a given segmented cell and its reference mask, the discrepancy between the boundaries can be measured with the Hausdorff distance \[25\]. We refer the reader to \[25, 58\].

### 2.5.2 Counting Detection Errors

Another aspect of cell segmentation accuracy is the number of detection errors. These errors are counted in terms of detected (or missed) cells or cell parts, rather than pixels. In Section 2.3, we introduced four types of detection errors: (i) under-segmentation or merged errors, i.e., representing more than one cell with a single segmented object; (ii) over-segmentation or split errors, i.e., representing one cell with multiple segmented objects; (iii) spurious errors; and (iv) missing errors, i.e., a failure to segment a cell.

From the perspective of a cell tracker, a simplified categorization often suffices: a false positive is a spurious segmented object and a false negative is a miss-

---

\(^5\)To be precise, we use \(F_1\)-score, a special case of the \(F_\beta\) score. As we only use \(\beta = 1\), we call it F-score for brevity.
ing cell (Section 2.4.1).

While type (iii) and (iv) errors in the four-type classification directly match the definitions of false positives and false negatives, the connection to over-segmentation and under-segmentation is not straightforward. On the other hand, it might be not necessary to define such a relation, as in practice two separate measurement procedures are used.

The first procedure counts errors of the four types [25]. A ground truth mask is first produced where pixels are classified into background and internal cell regions. Then each segmented object is assigned to a ground truth object with which it shares most pixels. The ground truth background is treated as an object. Then the merged error counter is increased when a single segmented object is assigned to more than one ground truth object; the split error counter is increased when a single ground truth object is assigned to more than one segmented object; the spurious error counter is increased when a segmented object is assigned to the background; and the missed error counter is increased when a ground truth object is not assigned to any segmented object.

The second procedure counts the number of false positive and false negative errors [36]. Here only manually located cell centroids are required in the ground truth, not a binary mask. Given ground truth cell centroids and centroids obtained from a segmentation algorithm, an optimal assignment procedure is used to find a set of one-to-one assignments that minimizes the sum of Euclidean distances between matched centroids. The numbers of ground truth and segmented centroids may not be equal. Furthermore, matching is prohibited if the ground truth and segmented centroids are further away from each other than a certain threshold (e.g., an estimate of a cell radius). After the assignment, every unmatched ground truth centroid is counted as a false negative, and every unmatched segmented centroid is counted as a false positive.

The first procedure (with four error types) tends to give more information about the behavior of a cell segmentation algorithm. However, this procedure requires a user to produce manually drawn cell outlines. This is a much more demanding prerequisite compared to the second procedure that only requires locating cell centroids. Furthermore, in the first procedure, for example, an instance of over-segmentation (split error) does not differentiate between the
cases of over-segmenting into two fragments or into ten fragments, whereas the second procedure reflects the number of fragments incorporated into the number of false positive errors.

2.5.3 Accuracy of Cell Tracking

The third aspect of CTS performance is the accuracy of cell tracking. This aspect, in turn, can have different interpretations, which results in a variety of proposed tracking performance measures.

From the perspective of cell state estimation (recall the Bayesian formulation in Section 2.4.4), tracking performance in frame \( t \) can be measured by looking at the discrepancy between the estimated and ground truth locations, respectively \( \vec{s}_t \) and \( \vec{s}^*_t \) at frame \( t \):

\[
\sigma_t = \sqrt{\frac{1}{N} \sum_{i=1}^{N} ||\vec{s}_i - \vec{s}^*_i||^2}.
\]  

(2.14)

Here \( N \) is the number of tracks. The equation is adopted from [76].

However, in such a formulation it is hard to account for track swaps, missing and spurious tracks. Therefore, another suggested measure counts the ratio of the number of valid tracks to the number of all tracks [76, 79]. There have been slightly different definitions of a valid track. Li et al. [79] define a valid track as a track that “follows the same cell through all the frames that the cell appears. Any swapping of identities between two nearby cells will invalidate the trajectories of both cells and their progeny.”

This definitions brings to attention a major challenge of cell tracking. Indeed, consider a tracker that only makes one association mistake in one hundred frames. However, this single cell identity swap can invalidate a measurement of cell life time, as well as the entire lineage tree that stems from that cell.

The proportion of valid tracks is a useful measure, but it can be insensitive to small improvements in tracking algorithms. Therefore, Padfield et al. [97] also count the proportion of correct individual assignments between consecutive frames. A similar statistic is collected by Degerman et al. [30] who count the number of times a track was switched to a different cell (they call it a mix...
Finally, the tracking performance can be measured from the perspective of the end goal, that is, a biological application. For example, Padfield et al. \cite{97} report the proportion of correctly identified cell divisions. Li et al. \cite{79} report the proportion of correctly reconstructed lineage trees. An indirect validation of tracking accuracy can be obtained by comparing the measured effect of certain treatments on cell under study. For example, the toxin Cytochalasin D is known as an inhibitor of cell motility, and the tracking system of Adanja et al. obtains results consistent with this knowledge \cite{5}.

This section concludes our review of the theoretical foundations of automated cell tracking systems. In the last section of this chapter, we reflect on the current state of cell tracking in practice, and identify several open challenges.

## 2.6 Cell Tracking in Practice

As we discuss in the past few sections, there is a wide range of mathematical tools and algorithms used at different stages of a CTS. To some extent this variety is mirrored in the number of available software packages and the number of features provided by the packages.

There is a large variability in the possible appearance of cells and the imaging settings, as well as assay types. As a result, many image analysis problems are unique. They often differ from a generic class of problems in specific details. For example, cell counting is a generic analysis task, which can be implemented via cell segmentation. However, for different types of cells the segmentation routine needs to be adjusted accordingly.

This situation is reflected in the architecture of all software packages that we review in this section. A generic software package allows a user to construct processing pipelines, where each processing module is supported by one or more parametrized algorithms implemented in the package (Figure 2.20). Different pipelines can be used for different assays, such as cell counts, motility analyses or protein co-localization studies.

Two of the most popular open source packages for the analysis of microscopy
2. BACKGROUND

Figure 2.20: Sample image analysis pipeline that can be constructed using a typical software package for microscopy images. Each pipeline module is customizable and is supported by implementations of different algorithms. A different pipeline can be used, for example, for cell tracking or co-localization studies.

images are CellProfiler [17] and ImageJ [4]. Both packages have an open modular architecture, which means that users can extend the functionality of the software by writing additional modules (or plugins). Furthermore, both packages provide some functionality for batch processing, such as performing the same assay for multiple images in a loop. For example, in ImageJ this is implemented via macros that can be written in Java or in the ImageJ macro language. Note that the standard ImageJ interface also supports interactive image processing when the results of processing algorithms are visible straight away (Figure 2.21).

Three of the most popular commercial packages are MetaMorph (by Molecular Devices), Imaris (by Bitplane), and Volocity (by PerkinElmer). All packages support the creation of customizable analysis pipelines (Figure 2.20). In contrast to open-source packages, it is not clear whether the commercial packages can be easily extended by custom user plugins. It appears that Imaris is capable of calling external MATLAB functions, but this functionality is not covered in the official reference manual.
Figure 2.21: A screen-shot of ImageJ software. Standard ImageJ interface also supports interactive image processing when the results of processing algorithms are visible straight away.
2. BACKGROUND

Figure 2.22: A screen-shot of Imaris software with a sample image from Bitplane (provided with the installation). Imaris offers rich 3D visualization functionality.

tated, scaled and sliced at different angles (Figure 2.22). Furthermore, users can create animations.

The state of the art software packages provide a wide range of customizable algorithms. It is certainly helpful to have a rich arsenal of tools available. Paradoxically, this variety of options also brings a major challenge in terms of selection and tuning of algorithms. Algorithms and parameter settings tuned for one cell video are not necessarily optimal for another video (or even another part of the same video). At the same time, the number of combinations of algorithms and parameters chosen for each pipeline step is very large and it is impractical to perform an exhaustive manual tuning of pipelines to each new video. Instead, researchers often use default settings or adjust only a few of the possible parameters, which results in a sub-optimal image analysis. In this thesis, we aim to increase the accuracy of the analysis for the amount of required manual work. To this end, we develop more accurate algorithms (Chapters 3 and 4), algorithms that require less parameter tuning (Chapter 5), and methods for assisting parameter tuning and manual correction of results (Chapters 6 and 7).
In the next chapter, we begin by describing our approach to the problem of accurate and portable cell segmentation.
Chapter 3

CLUBS: Clustering Based Semi-segmentation of Fluorescent Cells

In this chapter, we present our first contribution: a novel method for semi-segmentation of cells in fluorescence microscopy images. This method does not produce cell outlines and it is therefore not a segmentation method. Our method produces a sample of particles that estimate cell locations (Figure 3.1, right). This sample can then be used to infer certain statistics about cells, most importantly cell centroid locations but also estimates of cell size and brightness. We call our method a \textit{semi-segmentation algorithm}, as it can be positioned between detection algorithms that produce only cell centroids, and segmentation algorithms that produce cell outlines. To the best of our knowledge, we are the first to introduce the concept of semi-segmentation of cells.

The high level goal of our thesis is to achieve high accuracy of cell image analysis while minimizing the need for manual intervention. The method we present in this chapter is aligned with this goal because in our evaluation it outperforms previous approaches in terms of cell detection accuracy, thus requiring less manual correction in order to achieve accurate results. Furthermore, we address the challenge of variability in cell videos by introducing an acceptance function that encapsulates any assumptions about cell morphology. This allows the method to adapt to different imaging conditions by replacing the acceptance

\footnote{Recall that finding centroids is called cell detection.}
3. CLUBS: CLUSTERING BASED SEMI-SEGMENTATION OF FLUORESCENT CELLS

Figure 3.1: A typical image of fluorescent cells (left). Sampling from the image (middle). Clustering results where clusters are encoded by colors (right). The images show simulated B lymphocytes in vitro. Original image courtesy of Hodgkin Lab, WEHI.

function across the variety of cell videos, which makes our method portable.

In this chapter (and to some extent in Chapter 4), we focus on fluorescence cell images. However, our other thesis contributions (Chapters 5, 6 and 7) do not depend on a particular type of cell microscopy.

In summary, the contributions of this chapter are as follows: (i) we present a novel methodology for semi-segmentation of fluorescent cells (Section 3.2); (ii) we evaluate our method on real cell videos (Section 3.3); (iii) and we suggest a number of additional applications for our approach (Section 3.4).

We start this chapter by presenting an overview of our semi-segmentation method and relating it to previous work in the next section.

3.1 Semi-segmentation of Fluorescent Cells

Fluorescence cell imaging is used in a number of important biological studies (Section 2.4). In the analysis of cell images (Figure 3.1 left), cell segmentation is an important step, as it allows us to obtain various cell statistics (e.g., cell counts), or it can be used as a basis for cell tracking. However, the range of conditions observed in real videos make automated cell segmentation difficult. Such challenging conditions include touching cells, and varying cell sizes and intensities.

As we discuss in Section 2.3, there have been a number of approaches to
3.1. SEMI-SEGMENTATION OF FLUORESCENT CELLS

automated fluorescent cell segmentation. In brief, popular approaches include thresholding based methods \cite{25, 32}; variations of watershed segmentation \cite{97}; and active contour based methods \cite{10}. Researchers have also tried other algorithms such as the Hough transform and Canny edge detection \cite{16, 32}. Proposed methods often comprise a pipeline of different techniques. For example, Fenistein et al. use Gaussian smoothing, k-means clustering and seeded region growth \cite{41}.

Due to the challenging segmentation conditions, in some cases, manual segmentation still outperforms automated methods \cite{25}, thus leaving an opportunity for improvement. Here we significantly depart from previous approaches, and present a novel method capable of accurate semi-segmentation of fluorescent cells. Our method CLUBS (clustering based semi-segmentation) does not produce cell outlines, but is capable of producing features (e.g., estimations of size or fluorescence) to describe cells.

Here we introduce \textit{semi-segmentation} as a problem of identifying the number of cells \( k^* \), and \( k^* \) clusters of locations \( C_1, \ldots, C_{k^*} \), \( C_i = \{(x_{i}^{(t)}, y_{i}^{(t)}), t = 1, \ldots, |C_i|\} \), such that cluster centroids correspond to true cell centroid locations in a given cell image \( I(x, y) \). The difference with the conventional cell segmentation is that cluster \( C_i \) needs not contain all of cells pixels, and can contain duplicate locations. We find that such a formulation allows CLUBS to locate cell centroids efficiently.

CLUBS is based on sampling from the image and clustering the sampled points. We address most of the segmentation challenges by posing the segmentation problem as a clustering task, and employing the advantages (e.g., the ability to separate touching clusters) of a general purpose clustering algorithm. A clustering approach for segmentation is an old idea (e.g., Otsu thresholding). The novelty of our method is that we do not cluster pixels, but rather points sampled from an image-based distribution. This resembles importance sampling in particle filters. Here, we treat the image itself as the distribution from which to sample. We encode the brightness of the image at a particular point by the local density of our samples around that point.

We emphasize that CLUBS is capable of separating touching cells by using a clustering algorithm that can separate touching clusters. Another advantage of
CLUBS is that it does not require image denoising. This is due to our choice of
the clustering algorithm that is capable of identifying (and filtering out) clusters
that originated from noise.

CLUBS can be turned into a cell detection method by reporting the centroids
of the produced clusters. Our evaluation on a variety of real cell videos shows
that when CLUBS is treated as a detection method it can outperform existing
cell detection methods. Furthermore, we propose a number of additional ap-
plications for CLUBS such as grouping cells by microwells, locating empty mi-
crowells, and tracking groups of cells.

3.2 Sampling and Clustering Based
Semi-segmentation

Our key observation is that gray-scale fluorescent images resemble plots of two-
dimensional probability density functions (PDF). This resemblance arises be-
cause during a cell experiment, wells with the cells contain fluorescent molecules,
and the microscope essentially captures a smoothed density distribution of these
molecules. We suggest to reverse this process by sampling points from the gray-
scale images, where brighter pixels denote a higher probability of a point to be
sampled at the corresponding location (Figure 3.1).

After the sampling, the problem of cell segmentation becomes a clustering
problem, where a cluster of points corresponds to a cell. This is an important
observation, because data clustering is a well studied problem with over a thou-
sand proposed algorithms [66], and potentially many clustering algorithms can
be applied to cell segmentation.

In this chapter, we investigate semi-segmentation of cells using agglomer-
ative hierarchical clustering (see Appendix A.3.1). This algorithm was chosen
as one of the most popular and relatively simple techniques. One can also try
a more modern technique (e.g., affinity propagation [45]). However, the choice
of a particular clustering algorithm is not essential for demonstrating the main
idea of this chapter.

Another avenue that we do not pursue in this chapter is the use of a non-
3.2. SAMPLING AND CLUSTERING BASED SEMI-SEGMENTATION

parametric clustering technique (e.g., [74]). Parameter free clustering has an emphasis on exploratory data mining, whereas in the cell segmentation application, it can be preferable to exploit our knowledge about cells, for example, in order to filter out noisy clusters that are likely to arise from gunge in the field of view. With a parameter free algorithm, such a filtering could have been applied post clustering. However, we find that with hierarchical clustering we are able to intertwine the filtering step with clustering process, thus enabling a dynamic pruning of infeasible hypotheses.

In our hierarchical clustering, the Euclidean distance is a natural choice for the cluster dissimilarity metric, as it captures the distance between cells in the input image. Furthermore, we use centroid linkage as the linkage criterion. Single linkage has a known problem of chaining, and complete linkage is not reliable when clusters have different diameters. In our sampling, each point has equal weight, and therefore we use unweighted centroid linkage. When calculating the distances between samples, we only consider the locations of the samples, and not their brightness. Recall that we encode the brightness of the image via the local density of samples.

Hierarchical clustering itself is not capable of deciding on the number of clusters. However, the resulting hierarchy represents a set of possible clustering hypotheses. We suggest to enumerate the hypotheses, evaluate each of them using an acceptance function and report the simplest accepted hypothesis as a result (i.e., the hypothesis with the smallest number of clusters). Specifically, we start with \( k = 1 \) cluster. For a given number \( k \), we cut the hierarchical tree so that it breaks into \( x \) sub-trees (clusters). We evaluate the resulting clusters using the acceptance function, and if this clustering is accepted, we stop, otherwise, we try \( k + 1 \) clusters. Our method is summarized in Algorithm 3.1, and the individual steps are explained below.

3.2.1 Image Preprocessing

We first apply gray-scale morphological opening with a disk structuring element (also known as top-hat filtering) [115] to compensate for uneven image illumination. The radius of the filter is set to an upper bound estimate of cell
Algorithm 3.1 CLUBS: sampling and clustering based semi-segmentation of fluorescent cells.

**Input:** gray-scale image $I$, 4 parameters (see text);

**Output:** a set of clusters, where each cluster is a set of points belonging to the same cell;

1. $I' = \text{preprocessing}(I)$;
2. $S = \text{sampling}(I')$;
3. $hTree = \text{hClusering}(S)$;
4. for $k = 1$ to $|S|$;
5.   $C = \text{cut}(hTree, x)$; // $C$ is a set of $x$ clusters
6.   if $\text{accepted}(C)$
7.     $C' = \text{removeNoise}(C)$;
8.     return $C'$;
9.   end if
10. end for
11. return $\emptyset$; // no cells found
3.2. SAMPLING AND CLUSTERING BASED SEMI-SEGMENTATION

radius $MaxR$, provided by the user. We then emphasize the difference between the dark and bright pixels by a power transformation of the image (gamma transform). For a given image $I_{rb}(x, y)$ after the top-hat filtering, the transformed image is $I'(x, y) = (I_{rb}(x, y) - \min_{x,y}(I_{rb}))^\alpha$. Here $\alpha$ is selected empirically and set to the same value $\alpha = 2$ for all our datasets. The higher is $\alpha$ the better is the separation between the foreground and background pixels. However, a higher $\alpha$ also introduces a bias towards brighter cells, and $\alpha = 2$ is found to be a good tradeoff.

One can apply more sophisticated techniques for image pre-processing, for example, filtering the image with anisotropic diffusion. We deliberately keep the pre-processing step simple in order to demonstrate that a simple step suffices for our algorithm to work. Requiring a simple pre-processing step is an advantage of our approach, as more complicated methods usually require more parameters to be specified.

3.2.2 Sampling from Images

In principle, a sample can comprise all image pixels, where each pixel is repeated a number of times in proportion to its brightness. That is, sampling as such is not necessary. However, reducing the number of samples allows us to significantly speed up the algorithm. Therefore, we produce a sample of points by placing random points in different image locations with the probability of the location being proportional to its brightness $(x, y) \sim I'(x, y) / \sum_{x,y} I'(x, y)$. Note that we allow more than one point to be placed in the same location, that is, we use sampling with replacement.

Here the user needs to specify the number of points in the sample. This can be estimated from the expected number of cells and the desired number of points per cell. In our evaluation, we find that varying this number does not affect our results significantly, as long as there are at least a few (e.g., 5) samples per cell.
3.2.3 Acceptance Function

In our algorithm, we suggest to formalize the description of a cell in terms of an acceptance function. The function $\text{accepted}(C)$ applies classification rules to, and obtains a label for each cluster in $C$. If at least one cluster is labeled as “too large” then the candidate clustering $C$ is rejected.

There are three possible labels for a given candidate cluster: “too large”, “OK”, and “noise”. If the label is “too large” then the current candidate set of clusters cannot be accepted, but it is worth trying a sub-clustering with a larger number of clusters. “OK” means that the cluster can be accepted as a valid cell. Finally, “noise” means that the cluster is not a valid cell and cannot become a valid cluster after any further division. For example, if a cluster already contains too few points or the points are too dark then a subsequent clustering will not result in valid clusters (function $\text{removeNoise}()$ filters out “noise” clusters).

$$
\text{label}(C_i) = \begin{cases} 
\text{"too large"}, & (\text{rad}(C_i) > \text{MaxR}) \\
\text{"OK"}, & (\text{rad}(C_i) \leq \text{MaxR}) \text{ and } (\text{intens}(C_i) \geq \text{MinB}) \\
\text{"noise"}, & (\text{rad}(C_i) \leq \text{MaxR}) \text{ and } (\text{intens}(C_i) < \text{MinB}). 
\end{cases}
$$

(3.1)

Here, $\text{rad}(C_i)$ is a half of the maximum Euclidean distance between the points in cluster $C_i$ and $\text{MaxR}$ is the upper bound estimate of the cell radius. The function $\text{intens}(C_i)$ is the median intensity for the pixels inside the smallest convex hull containing all points of cluster $C_i$, and $\text{MinB}$ is the lower bound estimate of cell intensity. We discard clusters with less than 3 unique samples\(^2\). The thresholds $\text{MaxR}$ and $\text{MinB}$ are provided by the user. Note that the same $\text{MaxR}$ is used in preprocessing.

In general, one can adjust the classification rules in $\text{label}(C_i)$ according to the expected cell appearance, for example, by adding extra conditions. In our evaluation, we find that two simple criteria ($\text{MaxR}$ and $\text{MinB}$) can be sufficient for a

\(^2\)Although the value 3 here can be considered a parameter, in our algorithm this number is fixed based on the fact that 1 or 2 distinct points do not define a polygon. As such it is unclear whether, for example, the intensity of such clusters can be estimated in a manner consistent with that for other clusters.
3.2. SAMPLING AND CLUSTERING BASED SEMI-SEGMENTATION

reliable semi-segmentation of cells across a variety of videos. The rules allow us to formalize the intuitive process of making a decision regarding the candidate solution. For example, in our implementation, reasoning “cell candidate is OK if it is not too large, and not too dark” is formalized with a set of predicates. Furthermore, one can use automated methods to learn classification rules from a training set (not used in our present work). Examples of other possible criteria include a measure of the convexity of candidate clusters and the uniformity of sample locations within the clusters.

3.2.4 Algorithm Complexity

We conclude the description of our method with the following lemma regarding the computational complexity of CLUBS.

**Lemma 3.1.** Given an image of size $M \times N$, the worst case computational complexity of Algorithm 3.1 with the acceptance function defined in Section 3.2.3 is bounded by $O(M \times N + S^3)$, where $S$ is the number of samples (assuming $S > \log(M \times N)$).

**Proof.** At the preprocessing step, the computational complexity of the top-hat filter is $O(M \times N)$, because the flat structuring element in the filter (i.e., the disk) must iterate over every position in the image. At any given position, one needs to find the minimum intensity among the pixels covered by the disk, but this calculation can be implemented iteratively as the disk moves along the image.

The sampling step can be implemented by (1) linearizing the image into vector of pixels $\mathbf{I}_{\text{lin}} = p_1, \ldots, p_{M \times N}$, (2) computing a vector of cumulative sums $s_1, \ldots, s_{M \times N}$, where $s_i = \sum_{j=1}^{i} p_j$, (3) uniformly sampling particles $z_1, \ldots, z_S$ from the interval $1, \ldots, s_{M \times N}$, and (4) for each particle $k$, finding the index (i.e., pixel location) $i$ such that $s_{i-1} < z_k \leq s_i$. Assuming that sampling of an individual particle can be done in constant time (e.g., the complexity of generating a pseudo-random number is $O(1)$), and using binary search at step (4), the entire sampling step requires $O(S \times \log(M \times N))$ time.

The agglomerative hierarchical clustering in Algorithm 3.1 has a complexity of $O(S^3)$. Furthermore, after the hierarchical tree has been constructed, at
3. CLUBS: CLUSTERING BASED SEMI-SEGMENTATION OF FLUORESCENT CELLS

a given tree cut $k$, resulting clusters $C_1^{(k)}, \ldots, C_k^{(k)}$ must be validated using the acceptance function. The acceptance function involves testing the pairwise distances between the samples in each cluster (Equation 3.1), but these distances can be obtained while constructing the hierarchical tree. Once an appropriate level $k^*$ is found, the convex hull for each cluster and the median intensities within each convex hull need to be calculated. Note that this is required to be done only once.

Given a cluster with $c_i = |C_i|$ samples, the convex hull can be found in $O(c_i \times \log c_i)$ time [55], and the total number of pixels in the convex hulls for all clusters is limited by $M \times N$. Identification of the convex hulls for all clusters requires $O\left(\sum_{i=1}^{k} (c_i \times \log c_i)\right)$. By noting that $\sum_{i=1}^{k} (c_i \times \log c_i) \leq \sum_{i=1}^{k} c_i^2 \leq S^2$, where $S = \sum_{i=1}^{k} c_i$, we estimate the upper bound on complexity of the convex hull construction step $O(S^2)$.

From the above arguments, we conclude that the upper bound estimate on the total complexity is $O(M \times N + S \times \log(M \times N) + S^3 + M \times N + S^2)$. Assuming $S < \log(M \times N)$, this can be simplified to $O(M \times N + S^3)$.

3.3 Evaluation of CLUPS

The aim of our evaluation has been to verify if the CLUPS algorithm is capable of producing an accurate semi-segmentation across a variety of videos. To this end we use four sets of real cell images that we present in Section 3.3.1. We also use a range of popular previous segmentation algorithms as our baseline. All the methods tested and their parameters are described in Section 3.3.2. Finally, we discuss the obtained results in Section 3.3.4.

Before we proceed to the next sections, we explain the method of measuring the quality of results. In Sections 2.5.1 and 2.5.2 we mention different performance measures for cell segmentation and detection accuracy. CLUPS does not produce cell outlines and therefore traditional measures for segmentation accuracy (e.g., Jaccard index or Hausdorff metric [25]) are not directly applicable to semi-segmentation results. Fortunately, the quality of cell detection can still be
measured numerically.

There are at least two ways of measuring cell detection accuracy (Section 2.5.2): (1) count the number of split, merged, spurious and missing errors using the procedure outlined in [25]; or (2) count unassigned detections using the method described in [36]. The first method can be slightly less sensitive to the detection quality, because if a cell, for example, is over-segmented into several parts, this still counts as one error. Furthermore, we cannot directly apply the first procedure to our results, because the procedure is based on matching cell regions [25]. Therefore we use the second method which is also widely used for cell detection quality.

Consider a set of detected cell centroids and a set of ground truth centroids. In the second method, one first performs an optimal assignment\(^3\) between the sets, while prohibiting the assignment of detections that are further away from each other than a certain distance (see AvgR in Table 3.3). The unassigned centroids from the detected and ground truth sets comprise spurious and missed cell errors respectively. One can then compute the precision \( \text{prec} = \frac{TP}{TP + FP} \), recall \( \text{recl} = \frac{TP}{TP + FN} \), and F-score \( F = 2 \cdot \frac{\text{prec} \cdot \text{recl}}{\text{prec} + \text{recl}} \), where \( TP \) is the number of assigned centroids, \( FP \) is the number of spurious errors and \( FN \) is the number of missed cell errors.

Furthermore, we manually review our results for a random selection of frames and confirm that overall the resulting clustering matches an intuitive expectation (such as in Figure 3.1 right). In rare cases, CLUBS assigns sample points located near, but outside, a cell boundary to a cell. This is not necessarily a critical issue, as our algorithm essentially returns a sample that approximates the PDF of cell pixel locations.

### 3.3.1 Datasets

We use four real fluorescence cell videos that show a considerable variation in cell appearance both across and within the videos (Figure 3.2). More precisely, we use four sets of images, where each set is a random selection of frames from

---

\(^3\)See Appendix A.3.2 for details of the Hungarian algorithm that forms the basis of our optimal assignment.
3. CLUBS: CLUSTERING BASED SEMI-SEGMENTATION OF FLUORESCENT CELLS

Figure 3.2: Variability in cell appearance across and within videos. Two of the figures (left, middle) show fragments of frames from the same dataset (fluorescent imaging of NIH3T3 cells \[25\]), and the third image (right) shows simulated B-lymphocytes. Image contrast was enhanced for a better visibility.

Table 3.1: Four datasets that we use in our evaluation. HEX and SQUARE show simulated B-lymphocytes in hexagonal and square microwells under different magnification. “#Total cells” is the total number of cells in all tested frames.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Frame size</th>
<th>#Tested frames</th>
<th>#Total cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIH3T3</td>
<td>1344 × 1024</td>
<td>44</td>
<td>2013</td>
</tr>
<tr>
<td>U2OS</td>
<td>1349 × 1030</td>
<td>43</td>
<td>1631</td>
</tr>
<tr>
<td>HEX</td>
<td>1388 × 1040</td>
<td>5</td>
<td>629</td>
</tr>
<tr>
<td>SQUARE</td>
<td>1388 × 1040</td>
<td>5</td>
<td>870</td>
</tr>
</tbody>
</table>

Two of the datasets show U2OS and NIH3T3 cells \[25\] (U2OS was first reported by Peng et al. \[102\]), and the other two datasets show simulated B-lymphocytes in hexagonal (HEX) and square (SQUARE) microwells \[61\]. For each of the datasets, we use 5 frames for training (parameters estimation), and the remaining frames for testing the algorithms. We have ground truth segmentation masks for the NIH3T3 and U2OS datasets. This ground truth was published by Coelho et al. \[25\]. We only have a ground truth cell detection (marked centroids) for testing frames in the HEX and SQUARE datasets. A researcher (other than the thesis author) was asked to manually mark the centroids in these frames.

\[4\]The U2OS and NIH3T3 datasets are available on-line from the Murphy Lab website at \texttt{http://murphylab.cbi.cmu.edu/data/2009_ISBI_Nuclei.html}.
Table 3.2: Segmentation methods and CLUBS, and their parameters that we use in our evaluation. The same values are re-used across the methods, but the values may differ across the datasets. Note that $\alpha$ and $N$ used in CLUBS are transferable (set to the same values) across the datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>Required parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>threshOtsu</td>
<td>MaxR, MinArea</td>
</tr>
<tr>
<td>threshGlob</td>
<td>MaxR, MinArea, ThGlob</td>
</tr>
<tr>
<td>wsGauss</td>
<td>MaxR, MinArea</td>
</tr>
<tr>
<td>activCont</td>
<td>7 parameters, see [113]</td>
</tr>
<tr>
<td>CLUBS</td>
<td>MaxR, MinB, $\alpha$, $N$</td>
</tr>
</tbody>
</table>

3.3.2 Baseline Methods

We use several popular segmentation approaches as baselines for our method. We note that thresholding and watershed based segmentation is included in most software packages (such as CellProfiler, MetaMorph and Imaris, see Section 2.6), and such methods can be considered de facto industrial standards. In our evaluation, we compare the performance of CLUBS with two thresholding methods, a watershed method, and an active contours based method (Table 3.3.2). These baseline methods are segmentation methods, which return segmented cell regions. We produce centroids from the cell regions by finding weighted centroids of the regions, where the weights correspond to intensity levels in the original input images.

A naive implementation of any of these approaches almost never gives sensible results. Therefore, we apply additional steps as follows. For all methods, except the active contours based method, we first apply the top-hat filter to compensate for uneven image illumination. The radius of the filter is set to MaxR, because the structuring element in the filter must be comparable in size but slightly larger than a cell. For all methods, except CLUBS and the active contours based method, we apply post-filtering, where we discard any obtained cell regions that are smaller than MinArea.

In the first baseline method threshOtsu, we use the Otsu method to find a threshold independently for each frame. In the second method threshGlob, we first find the minimum and maximum intensity levels across all frames in the
Table 3.3: Parameter estimates obtained from training frames. \( \text{AvgR} \) (resp. \( \text{MaxR} \)) is an estimate of an average (resp. maximum) cell radius in pixels. Recall that \( \text{AvgR} \) is used in the evaluation (optimal assignment). \( \text{MinArea} \) is the minimum allowed size of a cell region in pixels. \( \text{ThGlob} \) and \( \text{MinB} \) are intensity based thresholds. See text for details.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>AvgR</th>
<th>MaxR</th>
<th>MinArea</th>
<th>ThGlob</th>
<th>MinB</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIH3T3</td>
<td>25</td>
<td>75</td>
<td>1500</td>
<td>0.04</td>
<td>7</td>
</tr>
<tr>
<td>U2OS</td>
<td>25</td>
<td>95</td>
<td>2500</td>
<td>0.08</td>
<td>13</td>
</tr>
<tr>
<td>HEX</td>
<td>10</td>
<td>13</td>
<td>70</td>
<td>0.07</td>
<td>15</td>
</tr>
<tr>
<td>SQUARE</td>
<td>7</td>
<td>8</td>
<td>40</td>
<td>0.10</td>
<td>30</td>
</tr>
</tbody>
</table>

dataset. We then normalize the image such that the minimum level corresponds to the new 0 level and the maximum corresponds to the new 1 level (other levels are now between 0 and 1). Finally, we use a user-specified global threshold \( \text{ThGlob} \in [0, 1] \) for all images, to binarize the image.

In the watershed based method \( \text{wsGauss} \) we first apply Gaussian blur with the standard deviation \( \text{MaxR}/3 \) (i.e., we assume that 3 standard deviations of the filter approximate \( \text{MaxR} \); we also try \( \text{MaxR}/2 \) and use it if it gives better results). We then find local maxima of the blurred image as watershed seeds and apply a watershed algorithm on the blurred image. We discard background pixels based on binarization using the mean pixel value in each image, as suggested in [25].

As a representative active contours based method we use the implementation of the algorithm by Srinivasa et al. [113]. We use their implementation without any additional steps, as the authors did not indicate the need for any such steps in their evaluation. We follow the guidelines in [113], and use our training frames in order to estimate parameters for \( \text{activCont} \).

### 3.3.3 Parameters

We empirically estimate parameter values (separately for different datasets), from the training images (Table 3.3). We estimate the size related parameters \( \text{MaxR} \) and \( \text{MinArea} \) using direct measurements from the images. We estimate intensity related parameters \( \text{ThGlob} \) and \( \text{MinB} \) using image histograms.
For the CLUBS algorithm, as described in Sections 3.2.1 and 3.2.2, we set $\alpha = 2$ and $S = 4000$ (number of samples) for all datasets, where $S$ denotes the number of samples per frame. In every dataset we have at least 20 samples per cell.

We study the effect of parameters $\alpha$ and $S$ on randomly selected sets of frames from our datasets. Representative results obtained using the U2OS and HEX datasets are shown in Figures 3.3 and 3.4 (the other two datasets exhibit similar patterns). In the figures, the numbers show detection accuracy measured with the F-score. The detection accuracy tends to increase with the number of samples, but at certain point it saturates. We choose $S = 4000$ because at this point the performance plots (at various fixed $\alpha$) saturate. We also find that small variations in $S$ (e.g., $4000 \pm 500$, not shown) do not significantly affect the accuracy.

Recall that $\alpha$ is the power used in gamma correction (Section 3.2.1). Original images are used when $\alpha = 1$. Higher values of $\alpha$ result in a better separation of background and cell intensity levels (i.e., $\mu_{cell} - \mu_{back}$, where $\mu$ is the mean intensity of the corresponding group of pixels). However, there is also a variation in mean intensity for different cells which is especially prominent for the U2OS and NIH3T3 cells (Figure 3.2). Therefore, at certain point when $\alpha$ increases it starts to separate bright cells from dark cells as well. Sampling is designed to draw samples from the brightest image regions, and therefore, dim cells can become underrepresented by samples. We use $\alpha = 2$ as a tradeoff value observed across different datasets. We also find that small variations in $\alpha$ (e.g., $2 \pm 0.5$, not shown) do not significantly affect the accuracy.

Note that we do not try to find a globally optimal (in the sense of maximizing the F-score) parameter combination for each of the methods on each particular data. Exhaustive tuning of more than one parameter even for a single method is difficult as the convexity of such an optimization is not guaranteed. In practice, users tend to try a few reasonable guesses and choose the best trial. This almost always results in suboptimal parameter settings, and we call the resulting performance a practical performance. In our evaluation, we follow this practical scenario, and for each parameter we tried only a few different values deemed reasonable from the training frames. Whenever methods share a common pa-
Figure 3.3: Detection accuracy (F-score) averaged for 5 random images from the U2OS dataset at different values of $S$ and $\alpha$. In general, values $\alpha \in [2,3]$ and $S > 4000$ define a relatively stable region without significant deviations in the F-score.
Figure 3.4: Detection accuracy (F-score) obtained on the testing frames of the HEX dataset at different values of $S$ and $\alpha$. In general, values $\alpha \in [2, 3]$ and $S > 4000$ define a relatively stable region without significant deviations in the F-score.
Table 3.4: F-scores achieved by different methods on different datasets. Overall, CLUBS is the most accurate method across all datasets. For CLUBS, the table shows the 95% bootstrap confidence interval of the mean performance from 20 runs. “x” denotes cases where we could not obtain any sensible results. For each dataset the difference between the accuracy of CLUBS and any other method is statistically significant ($p < 0.001$, two-sided signed rank test). The best result for each dataset is highlighted.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>threshOtsu</th>
<th>threshGlob</th>
<th>wsGauss</th>
<th>activCont</th>
<th>CLUBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIH3T3</td>
<td>0.563</td>
<td>0.789</td>
<td>0.807</td>
<td>0.548</td>
<td>0.836 – 0.843</td>
</tr>
<tr>
<td>U2OS</td>
<td>0.867</td>
<td>0.816</td>
<td>0.846</td>
<td>0.572</td>
<td>0.873 – 0.878</td>
</tr>
<tr>
<td>HEX</td>
<td>0.944</td>
<td>0.946</td>
<td>0.523</td>
<td>x</td>
<td>0.962 – 0.967</td>
</tr>
<tr>
<td>SQUARE</td>
<td>x</td>
<td>0.939</td>
<td>0.927</td>
<td>x</td>
<td>0.967 – 0.969</td>
</tr>
</tbody>
</table>

In the tables, we arrange the datasets in the order of increasing performance achieved by CLUBS, that is from “hard” to “easy”. Interestingly, other methods exhibit roughly the same trend of increased performance. This suggests that there is an intrinsic “difficulty” attributed to each of the datasets. Indeed, there is a large variability in cell shapes and brightness in the NIH3T3 frames, whereas in SQUARE most of the cells have a round shape and are prominent on a dark background.

The difference betweenthreshOtsu and threshGlob is that the former method selects the threshold automatically, and in the latter method the optimal threshold is provided by the user. However threshOtsu sometimes outperforms threshGlob because the Otsu method is applied independently to each frame, and is therefore adaptive, whereas the global threshold is set once for all frames. At the
3.4. ADDITIONAL APPLICATIONS

same time threshOtsu is less reliable. For example, on SQUARE dataset the automated threshold fails completely (not a single correct detection).

The activCont implementation from [113] yields an “Out of Memory Error” for two of our datasets. We cropped the frames in these datasets to obtain a smaller frame size. On the cropped set HEX (which covers only parts of frames) we achieved the score of 0.9, which is still below the score of CLUBS. In general, activCont tends to have lower practical performance. This can be attributed to suboptimal parameter settings. Note that this algorithm has 7 parameters. While there is a guideline for parameter selection [113], overall we found that these parameters are less intuitive and harder to set than the parameters of other methods.

The running time for CLUBS on datasets NIH3T3, U2OS, HEX and SQUARE is, respectively, 3.4, 4.2, 17.2, and 16 seconds per frame. We tested a MATLAB implementation on Intel Core Quad CPU @ 2.66 GHz, 3.25 GB of RAM in Windows XP, Service Pack 3. Note that while CLUBS is in general computationally slower than the other methods, its running time is still practical for use in real-time cell imaging, when the time between frames is usually in the order of minutes.

3.4 Additional Applications

In this section, we propose a number of additional applications for our approach, and present preliminary results for some of the applications. We first note that CLUBS not only produces a clustering of points, but also a hierarchy of the clusters. In certain cases, we can utilize this property.

In biological experiments cells can be constrained in microwells [28]. This addresses the problem of cells migrating out of the field of view and prevents groups of cells from mixing. Therefore, the microwells can help in cell tracking. However, to take advantage of microwells, one needs to identify the borders of the wells. This is not always trivial, as the wells can have different shapes and the plates with wells cannot always be placed precisely at the same positions. CLUBS can address this problem by fixing the number of clusters to the known
3. CLUBS: CLUSTERING BASED SEMI-SEGMENTATION OF FLUORESCENT CELLS

Figure 3.5: CLUBS can be used to group points by wells. The groups of cells are coded with colors. The image shows 3 hexagonal wells. Approximate locations of well borders are highlighted in red. Original images courtesy of Hodgkin Lab, WEHI.

number of wells in the frame. As a result, all points (and subsequently located cells) can be grouped by wells (Figure 3.5). Note that wells in the grids are separated by walls with a certain thickness [28] (not seen in fluorescence images). This prevents cells from different wells from moving close to each other. Furthermore, in certain types of wells, the well bottom is concave and the cells tend stay somewhere around the center of their wells.

Another application of CLUBS is locating groups of cells, for example, in wound healing assays, where two groups of cells move towards the center of the frame to cover the wound [72]. In this scenario, rather than tracking individual cells, one can use CLUBS to locate two groups of cells (by setting the number of clusters to 2), and then track the centroids of the two groups.

Our sampling approach (without clustering) can also be used for background estimation. It is often useful to obtain an empty background patch, to learn the distribution of the background pixels which can help to segment cells. In order to verify whether a given image patch contains cells, we suggest to sample points from the patch and then test for the uniformity of the sample. If the patch is a pure background, we expect the sample to be uniformly spatially distributed across the patch. We test the uniformity of samples using Pearson’s chi-square test. In our SQUARE video, in every frame, we choose 4 patches: 2 of them cover empty wells and the other 2 cover wells with cells. For each sample from an empty well, the chi-square test fails to reject the null hypothesis of uniformity at the 5% significance level, whereas for each sample from a well
with cells, the chi-square test rejects the uniformity hypothesis. This suggests that our sampling approach can be used for background estimation.

Finally, CLUBS can be used for estimating the initial state distribution in particle filter based cell trackers [18]. One important requirement of particle filters is the initial estimation of the state density. We suggest to use CLUBS to obtain such an estimation, with points representing particle sets for different cells, and the intensity of the points used as weights.

3.5 Conclusions

In this chapter, we present a novel method for semi-segmentation of fluorescent cells. By semi-segmentation we mean locating cells and producing a vector of features (e.g., estimations of the size or intensity) for each cell, but not producing outlines of cells in the image. Our CLUBS algorithm samples points from a gray-scale image and clusters the points, such that each cluster represents a cell.

Our evaluation shows that CLUBS is capable of outperforming existing segmentation algorithms on a variety of real cell videos. Moreover, we suggest a number of additional applications for our method, such as grouping by microwells, locating groups of cells, estimating background patches and initializing particle filters.

One of the main ideas behind CLUBS is sampling from images in order to reconstruct the distribution of fluorescent molecules. Therefore, a limitation of CLUBS is that it is, in general, restricted to fluorescent cells. Moreover, outlines of cells are not produced, while they may be desirable in certain applications (e.g., to test how many cells are actually touching each other). Therefore, in next chapter we further develop our concept of the acceptance function (which is another idea behind CLUBS) in a novel cell segmentation algorithm. Our segmentation algorithm produces cell outlines and is more widely applicable than CLUBS, while still providing a high detection accuracy.
Recall that a segmentation algorithm partitions an image into background and cell regions, where different segmented cell regions must correspond to different cells. Previously, we explained that cell segmentation in general is a difficult task (Sections 1.2 and 2.3). Challenges arise due to various image conditions such as touching cells or background noise. Our semi-segmentation algorithm CLUBS addresses these challenges, but is not capable of producing cell outlines. In this chapter, we present a novel cell segmentation algorithm SEGEN. This algorithm produces cell outlines based on enumerating candidate cell regions. SEGEN inherits one of the ideas behind CLUBS, namely the concept of an acceptance function.

The acceptance function is introduced in order to overcome another cell segmentation challenge: variability in cell appearance across videos (Figure 4.1). While the main focus of SEGEN is on fluorescence cell images, our evaluation shows that the acceptance function enables us to apply SEGEN to other types of cell images.

The acceptance function enhances the portability of SEGEN, and the portability of algorithms is one of the properties that we are aiming for in our thesis. Furthermore, we evaluate SEGEN on a range of real fluorescent and bright field cell videos. SEGEN is capable of achieving nearly optimal segmentation.
4. SEGEN: CELL SEGMENTATION BASED ON ENUMERATION

Figure 4.1: Variability in cell appearance across and within videos. The figure shows fragments of a fluorescent image of U2OS cells (left, from Peng et al. [102]), a fluorescent image of NIH3T3 cells (middle, from Coelho et al. [25]), and a bright field image of neural progenitor cells (right, from Al-Kofahi et al. [6]).

and detection accuracy (e.g., an accuracy comparable to that of humans), while requiring fewer parameters and a shorter running time than some previous algorithms. Therefore SEGEN provides a useful tradeoff between the portability of the algorithm and the accuracy of the results.

In summary, this chapter contributes a novel method for fluorescent cell segmentation, and an evaluation of the method on a range of real cell videos. We also contribute a complexity analysis for the SEGEN method, while such an analysis is often omitted in previous work [11, 79, 105]. We present the discussion of previous methods in Section 4.1, and we describe our method in Section 4.2. We evaluate our method in Section 4.3 and we conclude the chapter in Section 4.4.

4.1 Previous Work and Our Approach

A thorough review of cell segmentation methods is given in Section 2.3 of this thesis. Here we present only a brief high level overview of related methods, and comment on the connection of our new method with previous approaches.

In Section 2.3 we noted that there have been a number of proposed approaches for cell segmentation [41, 97], and these approaches can be analyzed from the perspective of a meta-algorithm: (1) define a fitness function that scores a candidate image segmentation; (2) given the input image, find a set of candidate solutions; and (3) choose the solution that maximizes the fitness score. In this section, we take this perspective in order to motivate our method.
and contrast it with previous work.

Recall that we broadly divide the previous work according to the core segmentation step that they use. In brief, the groups are as follows.

The *thresholding* approach requires a user to define a method for threshold selection (e.g., Otsu, mean, or adaptive [6, 25]). Such a method is usually chosen via trial and error for a particular dataset, without a direct connection with the fitness function. Furthermore, once the thresholding method is fixed, there is essentially only one candidate solution (connected regions obtained after thresholding). That is, the fitness function does not guide the choice of candidate solutions. Instead, researchers usually apply a filtering step (e.g., discarding small regions), and this is where the expected appearance of cells can be brought into the algorithm.

The *watershed* approach [118] has more prerequisites. It requires (i) identification of seeds, (ii) identification of a working space (original image, gradient transform, or distance transform), and (iii) a background mask. The latter two prerequisites are usually satisfied using a trial and error method, and the first prerequisite is often satisfied by finding local minima of a smoothed image. Once the prerequisites are satisfied, the watershed algorithm essentially produces only one candidate solution. The fitness function does not directly guide the algorithm.

In contrast, the *active contours* approach explicitly seeks a solution that maximizes the fitness function (in this case, usually called the energy functional) [11, 107]. The energy functional is defined analytically, such that it is possible to derive a theoretical solution for finding a global maximum [11]. However, a change in the fitness function results in re-defining (re-developing) of the search strategy. In some cases, it can be hard to define a solution for an arbitrary fitness function, particularly one with logical predicates (e.g., consider a function: “a solution is good if there are no regions smaller than $S$ that are not at the image edges”). The method of Jung and Kim [68] does not use an active contour formulation, but fits in this group, because it seeks for an optimum of a fitness function. However, their fitness function relies on an elliptical cell model, which may not always be feasible (e.g., lymphocytes or neural progenitor cells can take irregular shapes).
### Table 4.1: Comparison of approaches for cell segmentation

<table>
<thead>
<tr>
<th>Method</th>
<th>Prerequisites</th>
<th>Searching process</th>
<th>Customized fitness function</th>
</tr>
</thead>
<tbody>
<tr>
<td>thresholding</td>
<td>threshold selection</td>
<td>fixed for a given threshold</td>
<td>n/a</td>
</tr>
<tr>
<td>watershed</td>
<td>seeds, input space, background mask</td>
<td>fixed for given prerequisites</td>
<td>n/a</td>
</tr>
<tr>
<td>growing/merging</td>
<td>seeding regions</td>
<td>guided by the fitness function</td>
<td>arbitrary</td>
</tr>
<tr>
<td>active contours</td>
<td>proper initialization or choosing a functional that leads to a convex optimization problem</td>
<td>guided by the fitness function</td>
<td>certain class of functions</td>
</tr>
<tr>
<td>our method (SEGEN)</td>
<td>none</td>
<td>guided by the fitness function</td>
<td>arbitrary</td>
</tr>
</tbody>
</table>

The region growing and merging approach [80, 85] is between the two extrema. On one hand, it allows the knowledge about cell appearance to be incorporated in the process of searching for the solution (e.g., as a stopping criterion for growing the regions, which is a form of the fitness function). On the other hand, the searching approach (i.e., the growing and merging procedure) in general is independent of the fitness function, and allows the function to be customized easily. However, the region growing and merging approach requires prior specification of region seeds. For example, the watershed approaches are often used to obtain seeding regions [80, 105].

The previous approaches are summarized in Table 4.1. We note that, among others, some desirable properties of a segmentation algorithm are (i) the ability to customize the fitness function, and (ii) the ability of the fitness function to guide the search for a candidate segmentation. This is due to the fact that there is a large variability in the appearances of cells across different videos. Region growing and merging methods satisfy these properties, but require the user to specify the seeds.

Interestingly, the seeds for the region growing approach can be obtained from the CLUBS algorithm presented in the previous chapter. However, such
an initialization would imply having an additional analysis step (i.e., CLUBS plus region growing), which would result in an increased computational time, and is likely to result in an increased number of required parameters. Therefore, we leave this investigation for future work, which is out of the scope of this chapter.

Next we propose a novel segmentation algorithm SEGEN that also satisfies these properties, but has no additional prerequisites. SEGEN has similarities to the watershed algorithm in the way it relies on intensity levels. However, SEGEN does not grow regions from seeding points. Our evaluation shows that our algorithm is able to outperform several previous baseline methods. We describe SEGEN in the next section.

4.2 Segmentation Based on Region Enumeration

We start with a description of the problem and an overview of our method. We then present details of our algorithm (Section 4.2.1), and proposed cell description functions (Section 4.2.3).

In this work, we consider 2-dimensional gray scale images, where each pixel of an image takes values from some finite set \( L_{\min}, \ldots, L_{\max} \) (e.g., gray levels from 0 to 255), and higher pixel values correspond to brighter levels. The segmentation problem is then to assign each pixel an integer value 0, 1, \ldots, where 0 indicates a background pixel, and the other integers label different regions corresponding to cells.

Let a region be a connected subset of image pixels\(^1\), and let an \( L \)-platform be a region where each pixel has a level brighter than or equal to \( L \), and every pixel immediately outside the platform border has a level lower than \( L \). Note that an \( L \)-platform can contain smaller \( M \)-platforms, where \( M > L \). A split of an \( L \)-platform is a non-empty set of \( M \)-platforms, \( M > L \). For example, in Figure 4.2, a split of platform (b) comprises platforms (c) and (d). Note that a platform can contain gaps inside.

---

\(^1\)We use 4-connectivity as the most conservative form of connectivity. For example, 8-connectivity is more likely to connect touching cells. See also Appendix A.1.1.
Figure 4.2: Evaluating platforms as candidate cell regions with respect to some acceptance function. In this example, platform (a) is too small and is discarded. Platform (b) is accepted temporarily, and is split further. Platforms (c) and (d) are small and are discarded, and (b) is accepted as a final result. Platform (e) is too large and is not accepted, but is split further. Platform (g) is too small and is discarded, and platform (f) is accepted. The final segmentation comprises contours of platforms (b) and (f).

Our method iteratively constructs candidate platforms and evaluates them using an acceptance function (Figure 4.2). Note that the acceptance function (Table 4.2) is applied to a platform (i.e., candidate cell region), not to the whole image.

One of the main difficulties in cell segmentation is how to separate touching cells. In some cases, threshold based methods are capable of separation, but choosing an appropriate threshold is essential for doing this. In practice, auto-

<table>
<thead>
<tr>
<th>Output</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>rejected, small</td>
<td>This platform is too small and cannot be accepted. There is no point in splitting the platform further.</td>
</tr>
<tr>
<td>rejected</td>
<td>The platform cannot represent a cell, but it might contain sub-platforms that represent cells.</td>
</tr>
<tr>
<td>accepted (and scored)</td>
<td>The output can represent a cell. The function returns a real-valued acceptance score (larger is better).</td>
</tr>
</tbody>
</table>
4.2. SEGMENTATION BASED ON REGION ENUMERATION

Various methods (Otsu or mean value) do not guarantee a good threshold [25]. Furthermore, the watershed based approach needs seeding points from which to grow the regions. Taking the local intensity maxima requires smoothing in order to cope with noise in the intensity levels. The smoothing, in turn, requires specifying an appropriate parameter value. Alternatively, the watershed seeds can be taken as the maxima of the distance transform. However, non-convex cell outlines can result in spurious maxima, thus leading to over-segmentation (Figure 4.3).

Figure 4.3: A thresholded image (left) and the distance transform of the thresholded image with marked local maxima (right). The non-convex cell outline results in spurious maxima.

In SEGEN, we employ an observation that there is usually a dim space between bright platforms representing the touching cells. This happens because the cells tend to be thinner near the boundaries (if viewed from above) and thus contain less fluorescent molecules. Therefore, in practice, SEGEN is capable of segmenting many touching cases properly (Figure 4.4). Note that platform outlines are not always smooth, but this is arguably not a disadvantage as the platform outlines are defined by image intensity levels. Such an approach preserves objective low level information. Humans tend to produce smooth outlines when creating a ground truth, thus introducing a bias at the level of segmentation. At the same time, a higher level analysis can be a better strategy to cope with non-smooth outlines.

Note that a more elaborate platform splitting strategy (e.g., based on the shape) can be included in SEGEN, because in our algorithm the splitting step is encapsulated in a single function $\text{Split}$ (next section). However, we find that using simple intensity based splits can be sufficient for achieving a nearly optimal segmentation (Section 4.3). At the same time, involving more elaborate
strategies tends to increase the number of algorithm parameters and hamper the portability and running time.

In our work, we consider images where brighter objects of interest appear on a dark background. Therefore, we present our algorithm as a fluorescence cell segmentation algorithm, although we find that it can be applied to bright field images as well (Section 4.3). Also note that the extension of SEGEN to 3-dimensional images is straightforward. We are now ready to present our algorithm in detail.

### 4.2.1 Enumeration Algorithm

Algorithms 4.1 and 4.2 describe SEGEN in detail. At each iteration there is a set of unprocessed platforms (at the first iteration, there is only one platform, which is the entire image). The algorithm then takes one of the unprocessed platforms and acts depending on the result of the acceptance function. Here, $\text{Split}(P, L)$ returns a set of $L$-platforms contained within $P$, or an empty set if such sub-platforms do not exist.

Algorithm 4.2 iteratively splits a platform at different levels (line 8). If at some level there appear more than one acceptable platform then the current platform is discarded and the new split is taken as part of the current candidate segmentation (line 9). In other words, it is considered more beneficial to segment two objects, each of which can be a cell (cannot be rejected), rather than choose the segmentation with only one object. Furthermore, if at some splitting level there is a platform with a higher acceptance score (line 9), then
Algorithm 4.1 Segmentation algorithm SEGEN

**Input:** image $\mathbb{I}$, acceptance function $\mathcal{A}(\mathbb{P})$, where $\mathbb{P}$ is a platform;

**Output:** outlines of cells;

1: $S_{conf} = \emptyset$; // set of confirmed platforms
2: $S_{pend} = \mathbb{I}$; // set of pending platforms
3: while $S_{pend}$ is not empty
4: $\mathbb{P} =$ any platform from $S_{pend}$;
5: remove $\mathbb{P}$ from $S_{pend}$;
6: if ($\mathcal{A}(\mathbb{P}) =$ “rejected”)
7: $L_{\text{min}} =$ minimum level in $\mathbb{P}$;
8: add $\text{Split}(\mathbb{P}, L_{\text{min}} + 1)$ to $S_{pend}$;
9: else if ($\mathcal{A}(\mathbb{P}) =$ “accepted”)
10: $S_{conf}, S_{pend} \leftarrow$
11: $\text{Process}(\mathbb{P}, S_{conf}, S_{pend}, \mathcal{A}(\mathbb{P}))$;
12: end
13: end
14: return outlines of platforms in $S_{conf}$;
this platform becomes the current cell candidate. Finally, if the current splitting level leads to a split where each platform is too small (line 15), then there is no need in further exploration of the current platform, and the current platform is confirmed.

Note that at each iteration of Algorithm 4.1 a platform is removed from the set of pending platforms. At the same time a split of a platform can be added to the pending set. However, note that the sum of the areas of the platforms in the split is always less than the area of the original platform (by definition, also see Figure 4.2). Therefore the sum of the areas of the platforms in the pending set decreases at each iteration, and hence Algorithm 4.1 always terminates.

4.2.2 Algorithm Complexity

The complexity of SEGEN depends on the complexity of the acceptance function $A(\mathcal{P})$. In what follows, we assume that the acceptance function is linear in the size of the input (in the next section, we present examples of such linear functions used in practice). Furthermore, we note that the split operation involves finding connected components, which can be done using a two pass algorithm, and hence $\text{Split}(\cdot)$ is a linear function.

The complexity of SEGEN can now be estimated by noting that at the beginning of Algorithm 4.1 each image pixel is assigned to a platform with a certain intensity level. Then at the end of each iteration (lines 3–13), each pixel is either discarded from further consideration, left unchanged or assigned to a new platform with a higher intensity level (promoted). Also note that at least one pixel is discarded or promoted at each iteration, hence the total number of iterations is limited by $N \times M \times L$, where $N \times M$ are the pixel dimensions of the image and $L$ is the total number of intensity levels in the image. There is an inner loop in Algorithm 4.2 (line 3), but each pixel is only evaluated in this loop once during the entire SEGEN run, and hence the complexity of SEGEN is $O(N \times M \times L)$.

The thresholding based cell segmentation has only $O(N \times M)$ complexity, but its detection accuracy is often inferior to that of more advanced methods [25, 37]. The basic watershed algorithm also runs in $O(N \times M)$ time [118]. However, in practice, the watershed approach forms the basis of a more sophisticated
4.2. SEGMENTATION BASED ON REGION ENUMERATION

Algorithm 4.2 Processing an accepted platform

Input: platform $\mathbb{P}$, acceptance function $\mathcal{A}(\mathbb{P})$, sets of confirmed and pending platforms $\mathbb{S}_{\text{conf}}, \mathbb{S}_{\text{pend}}$

Output: updated $\mathbb{S}_{\text{conf}}$ and $\mathbb{S}_{\text{pend}}$

1: $L_{\text{min}}, L_{\text{max}} =$ lowest, highest level in $\mathbb{P}$
2: $\text{acc}_{\text{max}} = \mathcal{A}(\mathbb{P})$
3: for $L = (L_{\text{min}} + 1)$ to $L_{\text{max}}$
4:   $\{R_1, \ldots, R_k\} = \text{Split}(\mathbb{P}, L)$;
5:   test each $R_i$ with $\mathcal{A}(R_i)$;
6:   if (more than one $R_i$ is accepted)
7:       add $\{R_1, \ldots, R_k\}$ to $\mathbb{S}_{\text{pend}}$;
8:       return $\mathbb{S}_{\text{conf}}$ and $\mathbb{S}_{\text{pend}}$;
9:   else if (only one $R_i$ is accepted)
10:      acc$_{\text{cur}} = \mathcal{A}(R_i)$;
11:      if (acc$_{\text{cur}} > \text{acc}_{\text{max}}$)
12:         acc$_{\text{max}} = \text{acc}_{\text{cur}}$;
13:         $\mathbb{P} = R_i$;
14:      end
15:      else if (each $R_i$ is “rejected, small”)
16:         break;
17:      end
18:    end // for
19: add $\mathbb{P}$ to $\mathbb{S}_{\text{conf}}$;
20: return $\mathbb{S}_{\text{conf}}$ and $\mathbb{S}_{\text{pend}}$;
4. SEGEN: CELL SEGMENTATION BASED ON ENUMERATION

Figure 4.5: A typical distribution of cell image intensity levels. In a typical distribution, the majority of pixels belongs to a dark background, and the number of pixels decreases rapidly with the increase of the intensity level.

region merging and splitting procedure, and the complexity of the resulting procedure increases. The complexity analysis for such methods is often omitted, perhaps, because it is hard to estimate such a complexity accurately [80, 105, 122].

In the contour evolution approach, the energy functional often includes the mean intensity of the segmented regions and the background region [11, 79]. Therefore such methods run at least in $O(N \times M)$. However, multiple iterations that are involved in the contour evolution, increase the complexity further. Again, an estimate of the number of iterations and the total complexity are often not provided for contour evolution based methods [11, 79, 107].

In contrast, for our method, we are able to give an upper bound complexity estimate which is $O(N \times M \times L)$. Moreover, we note that a more precise estimate is $O(S)$, where $S = \sum_{l=0}^{L-1} S_l$, where $S_l$ is the number of pixels with an intensity level higher than or equal to $l$. In practice, one can expect $S \ll N \times M \times L$, because of a typical distribution of cell image intensity levels (Figure 4.5). In a typical distribution, the majority of pixels belong to a dark background, and the number of pixels decreases rapidly with the increase of the intensity level. Moreover, no two $L$-platforms have any pixels in common, which suggests that SEGEN can be effectively parallelized even on the level of a single image.
4.2.3 Acceptance Function

SEGEN allows us to plug in arbitrary acceptance functions that match the interface defined in Table 4.2. This way SEGEN addresses the challenge of variability in cell morphology across videos. Surprisingly, we find that relatively simple acceptance functions can be sufficient for producing segmentations with an accuracy comparable to that of the state-of-the-art methods.

The Acceptance Function 1 (default) that we use in our implementation is a simple formalization of a logical predicate: a platform is accepted if it is neither too small, nor too large, nor too dark. That is, given size and brightness thresholds $A_{\text{min}}, A_{\text{max}},$ and $L_{\text{min}},$ the function returns “rejected, small” if $\text{PixelArea}(P) < A_{\text{min}},$ the function returns “rejected” if $\text{PixelArea}(P) > A_{\text{max}}$ and $\text{MeanIntensity}(P) < L_{\text{min}},$ and the function returns “accepted” otherwise. If the platform is accepted, the acceptance score is computed as $\text{PixelArea}(P) / \text{Perimeter}(P),$ meaning that this acceptance function encourages “roundness” of cell regions.

Moreover, we use the Acceptance Function 2 (customized) for one of the datasets. While our method was developed primarily for fluorescence images, we also tested the algorithm on images acquired using an inverted bright field microscope [6]. A characteristic property of the cells in these images is a high contrast at the cell boundaries due to a bright halo around the cells. We employ this property in a customized acceptance function presented in Algorithm 4.3. In these images cells do not appear to be round, and therefore our acceptance score merely encourages larger sizes of segmented objects, rather than “roundness”.

Finally we note that Algorithms 4.1 and 4.2 (the main SEGEN routines) do not include any parameters. All parameters are concentrated in the acceptance functions. The default acceptance function has 3 parameters, and the customized function has 4 parameters.

4.3 Evaluation of SEGEN

The aims of our evaluation have been to compare the performance of our method with other baseline methods (Section 4.3.1), and to validate the ease of cus-
Algorithm 4.3 Customized acceptance function

**Input:** platform $P$;

**Output:** status $S$, score $A$;

**Parameters:** $\text{MinSize}, \text{MaxSize}, \text{Thick}, \text{Contr}$

1. $A = \text{PixelArea}(P)$;
2. If $(A < \text{MinSize})$ // platform is too small
   3. return $S = \text{"rejected, small";}$
   4. end
3. If $(A > \text{MaxSize})$ // platform is too large
   5. return $S = \text{"rejected";}$
   6. end
8. $O = P$; // $O$ will contain platform outline
9. For $i = 1$ to $\text{Thick}$
   10. $O_1 = \text{pixel-wide external outline of } O$;
   11. $O = O \cup O_1$;
12. end
13. $O = O \setminus P$;
14. val$_1 = \text{median level in } P$;
15. val$_2 = \text{median level in } O$;
16. If $(\text{val}_1/\text{val}_2 \leq \text{Contr})$
   17. return $S = \text{"rejected";}$
   18. end
19. return $\{A, S = \text{"accepted";}\}$;
Table 4.3: Datasets for evaluation of segmentation and detection accuracy [6, 25, 61, 102], and parameter values fitted for training frames for SEGEN and baseline algorithms.  

<table>
<thead>
<tr>
<th>Name</th>
<th>Frame size</th>
<th>Image type</th>
<th>#cell</th>
<th>#frm</th>
<th>$A_{\text{min}}$, $A_{\text{max}}$</th>
<th>$L_{\text{min}}$, $T$</th>
<th>$R$, $\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>U2OS</td>
<td>$1349 \times 1030$</td>
<td>fluor. nuclei</td>
<td>1831</td>
<td>5/43</td>
<td>2500, 17000</td>
<td>0.05, 0.08</td>
<td>95, 31.7</td>
</tr>
<tr>
<td>NIH3T3</td>
<td>$1344 \times 1024$</td>
<td>fluor. nuclei</td>
<td>2178</td>
<td>5/44</td>
<td>1500, 15000</td>
<td>0.02, 0.04</td>
<td>75, 25</td>
</tr>
<tr>
<td>HEX</td>
<td>$1388 \times 1040$</td>
<td>fluor. cells</td>
<td>629</td>
<td>5/5</td>
<td>70, 491</td>
<td>0.09, 0.07</td>
<td>13, 6.5</td>
</tr>
<tr>
<td>SQUARE</td>
<td>$1388 \times 1040$</td>
<td>fluor. cells</td>
<td>870</td>
<td>5/5</td>
<td>40, 177</td>
<td>0.07, 0.1</td>
<td>8, 2.7</td>
</tr>
<tr>
<td>AK</td>
<td>$170 \times 165$</td>
<td>bright field</td>
<td>73</td>
<td>5/10</td>
<td>100, 800</td>
<td>0.3, 0.09</td>
<td>7, 2.3</td>
</tr>
</tbody>
</table>

In order to remove timestamps and file names overlaid on frames in the AK images, we cropped the images which resulted in a $170 \times 165$ frame size. We evaluated SEGEN on 5 sets of real cell images (Table 4.3). The U2OS and NIH3T3 datasets show fluorescent nuclei [25, 102]. The HEX and SQUARE datasets show simulated B lymphocytes in hexagonal and square microwells [61]. Recall that these four datasets also appeared in Chapter 3 of this thesis. Finally, the AK dataset comprises several randomly selected frames from the work of Al-Kofahi et al. [6], which is an example of bright field inverted microscopic images.
also inverted the frames in the AK dataset, $\mathbb{I}(x, y) = \max_{x,y} \mathbb{I}(x, y) - \mathbb{I}(x, y)$, because our algorithm expects bright objects on a dark background. Finally, we normalized pixel values for each image in all our datasets:

$$\mathbb{I}_{\text{norm}}(x, y) = \left[ \mathbb{I}(x, y) - \min_{x,y} \mathbb{I}(x, y) \right] / \left[ \max_{x,y} \mathbb{I}(x, y) - \min_{x,y} \mathbb{I}(x, y) \right].$$

We use several popular segmentation approaches as baselines for our method. These are the same baseline methods as in Section 3.3 where we evaluated the CLUBS algorithm, and we briefly outline the methods here.

In our baseline methods, we first apply the top-hat filter with radius $R$. In the first variation of the thresholding based algorithm ($\text{threshOtsu}$) we use the Otsu method in each image, and in the second variation of the thresholding method ($\text{threshGlob}$), we first normalize the images with respect to the maximum and minimum intensity across the corresponding dataset, and then set a hard threshold $T$. In the baseline watershed based method ($\text{wsGauss}$), we apply Gaussian blur with standard deviation $\sigma$ and use the maxima of the blurred image as watershed seeds. (The distance transform gave worse results.) Finally, in all baseline methods, we discard connected regions smaller than $A_{\text{min}}$.

We also used an active contour based method by Srinivasa et al. [113], but obtained poor results (see Section 3.3). This can be attributed to the difficulty of tuning the 7 parameters of their method. We omit the results of this algorithm.

### 4.3.1 Segmentation and Detection Accuracy

Earlier in the thesis, we discussed cell segmentation performance in Sections 2.5.1 and 2.5.2. In brief, there are two aspects of performance: the accuracy of segmented cell regions and the accuracy of cell detection. There have been a number of proposed measures for the accuracy of segmented cell outlines [11, 25], but we note that these measures often correlate. Therefore, we only use the Dice coefficient defined as $D = \frac{2|C_{\text{gt}} \cap C_{\text{alg}}|}{|C_{\text{gt}}| + |C_{\text{alg}}|}$, where $C_{\text{gt}}$ and $C_{\text{alg}}$ are sets of segmented cell pixels according to a ground truth segmentation and some algorithm, respectively [11, 37].

We measure the detection accuracy in the same way as in the work of Bergeest and Rohr [11], which is a variant of procedure described by Coelho et al. [25]. In the work of Bergeest and Rohr, segmented cell regions are matched with the
4.3. EVALUATION OF SEGEN

Table 4.4: Results on the U2OS and NIH3T3 datasets. Best automated results are highlighted. Dice shows the Dice coefficient, and S/M shows the number of spurious and missed cells (see text).

<table>
<thead>
<tr>
<th>Method</th>
<th>U2OS Dice</th>
<th>U2OS S/M</th>
<th>NIH3T3 Dice</th>
<th>NIH3T3 S/M</th>
</tr>
</thead>
<tbody>
<tr>
<td>threshOtsu</td>
<td>0.89</td>
<td>0.2/4.7</td>
<td>0.52</td>
<td>1.3/23.7</td>
</tr>
<tr>
<td>threshGlob</td>
<td>0.86</td>
<td>0.3/7.1</td>
<td>0.83</td>
<td>1.2/2.5</td>
</tr>
<tr>
<td>wsGauss</td>
<td>0.86</td>
<td>0.4/6.0</td>
<td>0.79</td>
<td>1.9/4.4</td>
</tr>
<tr>
<td>SEGEN</td>
<td>0.93</td>
<td>1.7/2.7</td>
<td>0.85</td>
<td>3.9/2.9</td>
</tr>
<tr>
<td>act. cont.</td>
<td><strong>0.94</strong></td>
<td>0.5/3.8</td>
<td><strong>0.85</strong></td>
<td>2.8/6.1</td>
</tr>
<tr>
<td>manual</td>
<td>0.93</td>
<td>0.6/2.2</td>
<td>0.87</td>
<td>0.0/3.2</td>
</tr>
</tbody>
</table>

ground truth cell regions. One can then count unassigned segmented regions as spurious cells and unassigned true regions as missing cells. The results are presented in Table 4.4. A ground truth segmentation is only available for two datasets. The other datasets are used for evaluating detection (see below).

The active contours based method of Bergeest and Rohr is known to outperform many previous approaches on the U2OS and NIH3T3 datasets [11]. Unfortunately, the implementation of the method was not available at the time of preparation, but we present the results from [11]. This comparison is reasonable because in their work they use the same data and the same performance metrics.

Furthermore in previous work [11, 25], researchers estimated the accuracy of a human segmentation, that is, the amount of discrepancy between two humans segmenting the same images. We find that SEGEN performs on par with the manual segmentation, meaning that our algorithm achieves an optimal or nearly optimal accuracy for the given datasets. Note that SEGEN uses only 3 parameters, whereas the method of Bergeest and Rohr requires 4 parameters ($\nu, \lambda, \kappa_0, \kappa_1$) and a convergence criterion.

The running time for the U2OS and NIH3T3 datasets was 20 and 13 seconds per frame respectively (MATLAB 7.11.0 without using parallel commands, Intel Core Quad CPU @ 2.66 GHz, 3.25 GB of RAM, Windows XP, Service Pack 3), which is significantly faster than the time required for active contours to stabilize (e.g., 100 sec for $E^c_3$ functional in [11] on a better machine with 46 GB of
4. SEGEN: CELL SEGMENTATION BASED ON ENUMERATION

RAM).

In summary, SEGEN performs on par with the method of Bergeest and Rohr, which is the best known method for these datasets. However, SEGEN requires fewer parameters and runs significantly faster than its competitor. Furthermore, SEGEN has the advantage of the flexibility of the acceptance function. This aspect is discussed in the next section.

4.3.2 Detection Accuracy and Portability

For the HEX, SQUARE and AK datasets, we only have ground truth detection (manually identified centroid locations), and we evaluate the detection accuracy. Note that the detection quality is an important aspect of any segmentation algorithm, as the detected locations can provide a basis for cell counting and subsequent cell tracking [71].

Without a ground truth segmentation we cannot repeat the procedure of Coelho et al. [25], but we can use the F-score as described in Section 3.3 (evaluation of the CLUBS method). Recall that we match detected centroid locations produced by the algorithm against the ground truth centroid locations. If two locations are closer than \( d \) to each other, we count them as matched (\( TP \)). Any unmatched centroids produced by the algorithm are counted as false positives (\( FP \)), and any unmatched true centroids are counted as false negatives (\( FN \)). The quality of the detection is then measured with the F-score defined as \( F_1 = 2 \cdot TP / (2 \cdot TP + FN + FP) \). Results are presented in Table 4.5, and performance per frame for two of the datasets is shown in Figure 4.6.

We conclude that SEGEN reliably outperforms the baseline methods on all datasets, except AK. The threshold-based methods are very sensitive to specific conditions within frames (e.g., background intensity compared to cell intensity).

We then use the acceptance function 2 for SEGEN (Algorithm 4.3). By defining a customized acceptance function we are able to utilize our observation about halos surrounding cells in the AK images, and achieve a significant increase in the detection accuracy without using extra steps such as filtering (Table 4.5).
Table 4.5: F-scores achieved by different methods on different datasets. Best results are highlighted. For the AK dataset, the results before and after using a customized acceptance function are shown. SEGEN outperforms the baseline methods (Wilcoxon rank-sum test, $p < 0.001$).

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
 & U2OS & NIH3T3 & HEX & SQUARE & AK \\
\hline
d (see text) & 25 & 25 & 10 & 7 & 15 \\
threshOtsu & 0.87 & 0.57 & 0.94 & 0.00 & 0.90 \\
threshGlob & 0.82 & 0.79 & 0.95 & 0.94 & 0.86 \\
wsGauss & 0.85 & 0.81 & 0.52 & 0.93 & 0.80 \\
SEGEN & 0.91 & 0.88 & 0.98 & 0.96 & 0.83/0.99 \\
\hline
\end{array}
\]

Figure 4.6: Performance of SEGEN and the baseline algorithms per frame on the U2OS and NIH3T3 datasets (respectively, left and right panels). The x-axis shows frame indices, and the y-axis shows F-score. The panels show only every fourth frame for clarity. SEGEN consistently outperforms the baselines in most of the individual images. SEGEN outperforms the baseline methods (Wilcoxon rank-sum test, $p < 0.001$).
Table 4.6: F-scores for detection achieved by CLUBS and SEGEN on different datasets. For CLUBS (a stochastic algorithm), the table shows the 95% bootstrap confidence interval of the mean performance from 20 runs. Overall SEGEN performs similar to CLUBS, but SEGEN is slower. The last two columns show the estimated running times in seconds per frame.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CLUBS mean</th>
<th>SEGEN</th>
<th>CLUBS, sec</th>
<th>SEGEN, sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIH3T3</td>
<td>0.836 – 0.843</td>
<td>0.840</td>
<td>3.4</td>
<td>9.8</td>
</tr>
<tr>
<td>U2OS</td>
<td>0.873 – 0.878</td>
<td>0.911</td>
<td>4.2</td>
<td>27.3</td>
</tr>
<tr>
<td>HEX</td>
<td>0.962 – 0.967</td>
<td>0.964</td>
<td>17.2</td>
<td>26.2</td>
</tr>
<tr>
<td>SQUARE</td>
<td>0.967 – 0.969</td>
<td>0.959</td>
<td>16.0</td>
<td>68.4</td>
</tr>
</tbody>
</table>

### 4.3.3 Comparison with CLUBS

Finally, we compare the detection accuracy of SEGEN with the accuracy of the CLUBS method that we introduced in the previous chapter. We cannot directly run CLUBS on the AK dataset (because the sampling stage in CLUBS requires fluorescent images), and therefore we compare the accuracy of these two algorithms on the remaining four datasets.

Note that CLUBS does not produce cell outlines, and therefore we only compare running time and detection accuracy for both algorithms. Recall that the detection accuracy refers to the ability to locate cell centroids.

The results are presented in Table 4.6. This table also shows the algorithms’ running times measured using MATLAB implementations on an Intel Core Quad CPU @ 2.66 GHz, 3.25 GB of RAM in Windows XP, Service Pack 3.

Recall that our analytical bound on the worst case complexity of CLUBS is $O(M \times N + S^3)$, where $S$ is the number of samples used in CLUBS (Section 3.2.4). This is asymptotically larger than $O(M \times N \times L)$, when $S$ is comparable to $M$ or $N$. The $S^3$ term in the complexity of CLUBS appears due to hierarchical clustering of samples. However, note that CLUBS uses sampling with replacement (Section 3.2.2), and the number of samples at distinct positions $\tilde{S}$ is smaller than the original number of samples $S$ (up to 10% smaller in our evaluation). Furthermore, in practice less than $S^3$ operations can be required for hierarchical clustering (e.g., computing pairwise distances takes $\tilde{S} \times (\tilde{S} - 1)/2$ operations and the number of distances needed to be computed decreases at
4.4. CONCLUSIONS

each iteration). As a consequence, in practical settings, CLUBS can run faster than SEGEN (in our evaluation both algorithms were implemented in MATLAB and run on the same machine).

We conclude that CLUBS and SEGEN have a similar performance in terms of detection accuracy, but CLUBS can run considerably faster in practical scenarios. On the other hand, SEGEN produces cell outlines and does not have a stochastic part. Furthermore, SEGEN is portable across different image modalities, as we illustrated in the previous section.

4.4 Conclusions

We present a novel cell segmentation algorithm SEGEN that enumerates image regions and validates them against an acceptance function. The acceptance function serves as a formal description of cell shapes. Our evaluation shows that SEGEN represents a practical trade off between portability and performance. On one hand, SEGEN can be used with a relatively small number of parameters, on the other hand, SEGEN is capable of achieving a nearly optimal accuracy on the range of videos tested.

While we develop SEGEN with fluorescence imaging in mind, we find that a customization of the acceptance function allows us to achieve an accurate cell detection on bright field images. A possible future work direction can be developing specialized acceptance functions and evaluating SEGEN in the context of a multichannel segmentation.

We compare SEGEN with our previous contribution CLUBS and find that they achieve a similar detection accuracy. CLUBS runs faster and has a number of additional applications (Section 3.4). On the other hand, SEGEN is a segmentation algorithm (capable of producing cell outlines), and does not have a stochastic part.

SEGEN further develops the idea of the acceptance function first introduced in CLUBS (Chapter 3). The design of both algorithms is oriented towards portability, in order to achieve the main goal of the thesis: maximize the accuracy of automated analysis, while minimizing the required manual effort (of tuning
the algorithms to different videos). As the next step, we address the portability challenge in the design of a cell tracking (i.e., data association) algorithm.

Cell segmentation inherently depends on the image being analyzed, and it is hard if not impossible to develop a generic parameter-free segmentation routine because cell appearance changes across videos. Although the dynamics of cells also varies, it appears that the variability in cell dynamics is less crucial for cell tracking algorithms. Interestingly, we find it possible to develop a one-parameter and a parameter-free versions of a cell tracking routine. We present our cell tracking algorithm in the next chapter.
Chapter 5

NENIA: Nearest Neighbor Based Inter-frame Assignments

A cell tracking system comprises different analysis stages, including cell segmentation and cell tracking (i.e., data association). Previously, we developed algorithms for semi-segmentation and segmentation with improved portability and performance. In this chapter, we focus on the cell tracking stage.

The cell tracking stage takes detected cell locations (and other features such as area and shape) as input, and establishes associations between the locations throughout a cell video. Cell tracking does not directly deal with cell images, thus it is not bound to a specific imaging type (e.g., fluorescence or phase contrast). However, variability in cell density and dynamics still presents a major challenge for automated cell tracking. Previous methods tend to handle this challenge either by introducing a number of parameters, or by making assumptions on cell dynamics. This hampers the portability of previous methods across different cell videos, because the cell motion model and the optimal parameter values are often not known a priori. As a result, a manual learning stage is required for each new video.

In our work, we aim to reduce the amount of manual work in cell data analysis. Therefore, we present a portable cell tracking algorithm with only one parameter. We show that the assumptions underlying our algorithm are quite general and practical for a variety of videos (Section 5.2.1). Furthermore, we pro-
pose a parameter-free variation of our method. To the best of our knowledge, we are the first to develop a parameter-free approach to cell tracking.

Note that we do not aim to develop the most accurate cell tracker. However, we find that our method is capable of achieving high performance in many cases. In other cases, a specialized fine-tuned tracker can outperform our method. At the same time, the parameter tuning stage for such methods usually requires a ground truth, at least for a part of the video. Our method can help in fine-tuning specialized algorithms through a bootstrapping-like method as follows.

Given a new video a user can use our method for automated tracking. The automated tracking is not perfect, and the user might still be required to correct the automated results manually. However, the vast majority of inter-frame associations should be correct at this stage. Therefore, the manual correction stage can take significantly less time compared to a completely manual annotation performed from scratch. This scenario provides an additional motivation for our method.

To summarize, in this chapter, we contribute two versions of a novel cell tracking algorithm. One version has only one parameter, and the other version is parameter-free. Removing all parameters is achieved at the cost of a slightly degraded performance. We start this chapter with a brief overview of related work in Section 5.1. We then present our algorithm in detail in Section 5.2. We evaluate our method in Section 5.3 and discuss our results in Section 5.4. Finally, we conclude the chapter in Section 5.5.

5.1 Related Work and Our Method

There has been substantial work in the field of cell tracking. In Section 2.4 of this thesis, we group cell tracking methods into several overlapping categories: (i) combined segmentation and tracking; (ii) Bayesian approaches; (iii) inter-frame assignments. Methods from the first group perform segmentation and tracking simultaneously, for example, using an active contour or a mean shift based procedure [5, 29]. Methods from the second group employ stochastic
5.1. RELATED WORK AND OUR METHOD

Table 5.1: Common challenges that hamper applying tracking methods across different videos

<table>
<thead>
<tr>
<th>Group of methods</th>
<th>Unaddressed Challenge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Combined segmentation and tracking</td>
<td>abrupt cell motions</td>
</tr>
<tr>
<td>Bayesian approaches</td>
<td>require prior knowledge</td>
</tr>
<tr>
<td>Inter-frame assignments</td>
<td>cost function for a variety of conditions</td>
</tr>
</tbody>
</table>

filters, such as Kalman filters [18, 31, 49]. Finally, methods from the third group focus on assigning detected cell locations across video frames [7, 24, 36, 72, 97].

While there are many advantages of the proposed methods, we note that in each case there is an unaddressed challenge that affects the portability of each particular method across different videos. We summarize the challenges in Table 5.1 and we present several examples of the previous methods below.

Dzyubachyk et al. [40] initialize cell contours from segmented cell positions in the previous frame, and adapt the contours to the current frame. However, abrupt cell motions and crossing cell trajectories can confuse the contour fitting process. Thus some form of inter-frame assignment can be necessary.

There have been a number of assignment based methods. For example, Al-Kofahi et al. [6] use an integer programming framework to find an optimal assignment of detected cell locations in subsequent frames. Padfield et al. [97] formulate the assignment problem as a minimum cost flow problem.

Al-Kofahi et al. and Padfield et al. propose different cost functions for links corresponding to different events (e.g., cell moves or cell divisions), and give explicit forms for these functions (Gaussian functions and absolute differences). However, it is not obvious why these particular forms should be used and what is the set of tracking conditions for which these functions are effective. In particular, the choice of the cost function for division events is non-trivial. In contrast, in this chapter, we provide a systematic argument for using only one cost function. We also state the assumptions that determine the possible choices of the cost function.

Bayesian filters can be used in conjunction with other methods. For example, Li et al. [79] use an active contour based approach supplemented with Kalman
filters, and Genovesio et al. [49] use stochastic filters with optimal linear assignment. However, Bayesian methods require an appropriate cell motion model in order to be effective. This requirement can be partly addressed by using Interacting Multiple Modules (IMM), but IMM require prior probabilities of model switching.

Based on the previous approaches, Meijering et al. conclude that “the consensus arising from the literature seems to be that any specific tracking task requires dedicated (combinations of) algorithms to obtain optimal results” [86]. Our aim has been to minimize the amount of manual work needed to enable cell tracking across a variety of different videos. To this end, we propose a novel automated tracking method called NENIA (nearest neighbor based inter-frame assignments).

In the design of our automated tracker we addressed the factors that affect the portability of cell trackers. Specifically, we propose a parameter-free version of the tracker. By parameter-free, we mean that our algorithm (in one of the versions) requires only the problem input (cell detection), and does not require any additional information from the user. Internally, our algorithm uses a parameter (gating distance), but this parameter is estimated automatically from the input data.

In this work, we found that using only cell centroid locations can be sufficient for achieving reasonably high tracking performance. However, as we argue in Section 5.4, other features, such as cell size or fluorescence, can be seamlessly added to our tracker, while keeping the algorithm parameter-free.

5.2 Methods

Informally, our goal has been to develop an automated cell tracker that is (i) portable, i.e., easily transferable across a variety of videos, (ii) capable of handling cell divisions and deaths, and (iii) able to perform reasonably well, i.e., can correctly resolve the majority of cell moves between frames.

In order to meet the portability requirement, we address the previously discussed challenges (Table 5.1). In Table 5.2, we repeat these challenges and com-
Table 5.2: The key elements of our work that are designed to address the portability challenges

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Solution</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>abrupt cell motions</td>
<td>using assignment based method</td>
<td>Sections 5.2.1 and 5.2.2</td>
</tr>
<tr>
<td>cost function for a variety of conditions</td>
<td>mathematical framework that shows that one can use a single cost function, and which function to use</td>
<td>Section 5.2.3</td>
</tr>
<tr>
<td>require prior knowledge</td>
<td>method for automated estimation of the optimal gating distance</td>
<td>Sections 5.2.1 and 5.2.2</td>
</tr>
</tbody>
</table>

As discussed in the previous section, tracking methods based on evolving models and clustering of spatio-temporal volumes tend to rely on smooth cell motions. Therefore we use an assignment based method.

A novel aspect of our work is a mathematical framework that allows us to (a) demonstrate that using a single cost function for both cell movement and division events is sufficient, (b) show which cost function can be used, and (c) treat gating distance as an optimization criterion. Preliminaries and formal definitions are given in Section 5.2.1 and the framework is presented in Section 5.2.2.

Another novel aspect is a method for automated estimation of the optimal gating distance presented in Section 5.2.3. The gating distance is a key variable of our tracker, and the automated estimation of its value makes it easier to transfer the algorithm between videos.

An overview of our cell tracker is presented in Figure 5.1. In addition to an inter-frame assignment method, our automated tracker employs a track management module that uses the assignment results to maintain (e.g., update or split) the set of cell tracks.

Previously we introduced a generic cell tracking (i.e., data association) problem in Section 2.4. We mentioned that the details of the problem formulation can differ depending on the approach. In the next section, we present our problem
5. NEENIA: NEAREST NEIGHBOR BASED INTER-FRAME ASSIGNMENTS

Figure 5.1: Overview of our automated tracker. Details are given in the corresponding sections.

formulation of the cell tracking problem.

5.2.1 Cell Tracking Preliminaries

Consider a video of living cells. We characterize a cell by its location, that is, its centroid position, in every video frame (other cell features, such as size, can be added to this definition). A cell occurrence is the cell location in a certain frame. The video is first processed by a detection algorithm, which identifies the likely cell locations in each frame. We refer to each given estimate of a cell location as a measurement $\vec{m}_{f,i}$, where $f$ is the frame number (when unambiguous, we omit this subscript), and $i$ is the measurement index within a frame. The cell occurrence may or may not have a corresponding measurement.

Between any two consecutive frames $f$ and $f+1$, a cell can either move, divide or die (Figure 5.2). In the case of a move, the cell persists in both frames (remaining still is characterized as a “move”), and $P(i \rightarrow j)$ denotes the probability that a cell moves from measurement $\vec{m}_{f,i}$ to $\vec{m}_{f+1,j}$. A cell offset is a vector pointing from the previous cell position to the next position, and a cell displacement is the magnitude of the cell offset.

In the cases of a division or a death, the cell ceases to exist in all frames starting from $f+1$. Also in the case of a division, the cell that ceases to exist is called the mother cell, and two new cells, called daughter cells, appear in frame $f+1$.

For a given video, across all frames we define the following probabilities of
5.2. METHODS

Figure 5.2: Sample lineage trees (left) that summarize cell events: cell A moves between frames 1, 2, and 3, and divides between frames 3 and 4; cell D moves between frames 1, 2, and 3, and dies between frames 3 and 4. In a sample detection resulting from these events (right), $\vec{m}_{i,j}$ are measurements, where $i$ is a frame index and $j$ is a measurement index within a frame. In this example, the correct track for cell A is \{\vec{m}_{1,1}, \vec{m}_{2,1}, \vec{m}_{3,1}\}, for cell B it is \{\vec{m}_{4,1}\}, for cell C it is \{\vec{m}_{4,2}\}, for cell D it is \{\vec{m}_{1,2}, ?, \vec{m}_{3,3}\}. There is a false negative error (cell D is not detected) in frame 2, and a false positive error (spurious measurement) in frame 3.

A cell move $P_m$, cell division (split) $P_s$, and cell death $P_d$ as

$$P_m = \frac{\text{#occurrences when cell moves}}{\text{#occurrences}},$$  
(5.1)

$$P_s = \frac{\text{#occurrences when cell divides}}{\text{#occurrences}},$$  
(5.2)

$$P_d = \frac{\text{#occurrences when cell dies}}{\text{#occurrences}}.$$  
(5.3)

Here # occurrences is the total number of occurrences of all cells in the video (except the last frame), and # occurrences when cell moves (divides, dies) is the number of occurrences corresponding to cell moves (divisions, deaths). Note that $P_m + P_s + P_d = 1$.

The probabilities $P_m$, $P_s$, $P_d$, and $P(i \rightarrow j)$ characterize the cells. In general, these probabilities are unknown prior to tracking. Moreover, tracking is often used to derive these probabilities [30][61].

In our tracker, we do not directly address cells entering and leaving the field of view. We observe that using small wells or microgrids [28] is becoming increasingly common to address the problem of cells migrating into or out of the field of view. Further, in our tracker, entering the view is registered as a division
(if a new cell enters somewhere close to another cell) or as a false positive (otherwise), and leaving the view is registered as a death. Therefore, if handling of entering and leaving cells is required, our tracker can be supplemented with an extra step to distinguish between cell divisions and enterings, and cell deaths and leavings. This step can be based on the proximity to the frame border.

The cell detection algorithm extracts a set of measurements from all frames of the video. We refer to this set of measurements as a detection. Recall that in practice, detections can contain errors: spurious measurements (false positives) and missed cell occurrences (false negatives). The quality of the cell detection can be measured by standard precision and recall metrics, but in our framework it is convenient to introduce complementary measures: the probability of false positives ($P_{fp}$) and the probability of false negatives ($P_{fn}$), defined as

$$P_{fp} = \frac{FP}{TP + FP} = 1 - Precision,$$

$$P_{fn} = \frac{FN}{TP + FN} = 1 - Recall.$$

Here $FP$ and $FN$ are the numbers of false positive and false negative errors, and $TP$ is the number of cell occurrences that have the corresponding measurements in the detection.

The detection forms the input of the tracking problem. The output of the tracking problem is a set of tracks, where a track is a list of measurements from the subsequent frames. The track may contain dummy measurements that indicate that at a particular frame the track was not detected (there was a false negative error). The quality of the output is measured as explained below.

### Cell Tracking Performance

We discussed cell tracking performance measures in Section 2.5.3. In this chapter, we use a tracking accuracy measure that is based on the ratio of the correctly reconstructed links between consecutive frames [30]. This measure is more sensitive to improvements in tracking than the ratio of correctly reconstructed tracks [79].
A cell tracker produces a set of tracks where each track (not necessarily correct) has a unique non-zero identification number (ID). Then each measurement can be labeled by the ID of the containing track, or with 0, if the measurement is not included in any track (considered as a false positive). A swap error is a situation when the labels of two consecutive measurements in a correct track change. Let \( \text{#all measurements} \) be the number of measurements in all correct tracks. We then propose to measure the tracking performance by the ratio of correctly reconstructed links defined as

\[
P_{\text{links}} = 1 - \frac{\text{#swap errors}}{\text{#all measurements}}. \tag{5.6}
\]

Resolving cell divisions is critical for reconstructing lineage trees. A correctly resolved division is one for which the frame number when the division occurs, and both mother-daughter relations are established correctly. We measure the ability of NENIA to resolve cell divisions by using the ratio of correctly resolved divisions to the total number of divisions in a video. At the same time we emphasize that resolving cell moves is also critical for reconstructing lineage trees. For example, a swap error can result in a loss of two cell tracks, and further in an inability to construct lineage trees for these cells. Moreover, note that in the videos, the number of cell moves tends to be much larger (often by several orders of magnitude) than the number of divisions \([61, 79, 97]\). Therefore \( P_{\text{links}} \) alone can give an estimation of how useful NENIA can be in practice, although we also report the ratio of resolved divisions for NENIA.

**Assumptions**

We conclude our cell tracking preliminaries with a list of assumptions. While the assumptions determine the scope of our work, we also provide comments explaining why our work is applicable to a wide range of real world situations.

**Assumption 5.1.** Probabilities of cell and detection events: We assume that the cell video has good temporal resolution, that is, the time between successive frames is small in comparison to a cell lifetime. Specifically, we assume that \( P_s + P_d \leq 0.1 \) (here \( P_s \) and \( P_d \) are defined by Equations \(5.2\) and \(5.3\)). This assumption means that there are on average at least 10 frames per cell lifetime. The videos reported in the literature...
satisfy this property \[6, 61\]. We also assume a reasonable cell detection quality, specifically \(P_{fp} + P_{fn} \leq 0.1\). In practice, researchers have reported detections with \(P_{fp}, P_{fn} < 0.03\) \[79, 97\].

If the above assumption does not hold, it is hard to expect an accurate cell data analysis regardless of the quality of tracking. Poor temporal resolution implies a coarse estimation of cell lifetimes and speeds.

Regarding the assumption about cell detection quality, we note that poor detection quality leads to poor tracking quality (see \[67\]; or our evaluation in Section 5.3.2). Given a poor detection quality, any tracking algorithm can be expected to produce inaccurate results that are not useful in practice. Therefore in our tracking algorithm, we focus on scenarios with a reasonable detection quality.

**Assumption 5.2.** Independence: *In any given frame, the decisions whether to move, divide or die are made independently for different cells. In the case of a move, the direction and the cell displacement occur independently for different cells.*

Although there is likely to be a correlation in the lifetimes and fates of cells, under Assumption 5.1 the time between frames is much shorter than a cell lifetime. On this short time scale, the behavior of cells is more stochastic, and the independence assumption is reasonable.

**Assumption 5.3.** Cell dynamics: (i) The probability \(P(i \rightarrow j)\) of a cell move between locations \(\vec{x}_i\) and \(\vec{x}_j\) decreases according to the squared Euclidean distance \(\|\vec{x}_i - \vec{x}_j\|^2\) between the locations. (ii) The spatial distribution of cells is similar across consecutive frames.

The first part of this assumption is applicable to a number of distributions that are likely to describe the cell dynamics. In particular, normal-like distributions are prominent in nature. Gomez et al. \[51\] introduce the multivariate power exponential distribution as a generalization of the normal distribution. This generalized normal distribution satisfies our assumption (see Appendix B.1 for mathematical details).

Notice that Assumption 5.3 allows us to develop a parameter-free tracker, without an explicit specification of the distribution of cell offsets. Moreover,
our experiments (Section 5.3.2) show that the assumption does not affect the applicability of NENIA across real cell videos with different distributions of cell displacements. Our cell tracker performs well on each of these videos.

The second part of Assumption 5.3 is related to cell density. If the directions of cell moves are random, we can expect cell density to remain approximately the same between consecutive frames. If there is a prevalent direction, we can expect the whole group of cells to shift in one direction, again preserving the density. Finally, if there are sub-populations of cells with different prevalent directions, we still expect them to keep the spatial distribution between consecutive frames, because under Assumption 5.1 the time between frames is relatively short.

The key step of NENIA is inter-frame assignments. This step is described in the next section.

5.2.2 Inter-frame Assignment Problem

In this section, we describe our novel approach to the assignment problem in the context of cell tracking. In this context, the assignment problem has already been described in the literature [67, 76]. We first re-introduce some of the related definitions, and then provide the details of our new approach.

Suppose that at frame \( f \) we have a set of previously established tracks \( T_f = \{t_i\} \), where a track \( t_i \) has the location \( \vec{m}_{f,i} \). Further, suppose that at frame \( f + 1 \) we have a set of measurements \( \vec{m}_{f+1} = \{\vec{m}_{f+1,j}\} \).

Consider assigning measurements in \( \vec{m}_{f+1} \) to tracks in \( T_f \) such that a measurement \( \vec{m}_i \) can be assigned to 0 or 1 tracks, and 0, 1, or 2 measurements can be assigned to a track \( t \). Figure 5.3 explains the choice of this assignment setup with an example. We refer to an individual assignment of measurement \( j \) to track \( i \) as a link \( i \rightarrow j \). The link \( i \rightarrow j \) represents a potential cell offset from the location \( \vec{m}_{f,i} \) to the location \( \vec{m}_{f+1,j} \). The length of a link \( i \rightarrow j \) is the Euclidean distance \( r_{ij} = \|\vec{m}_i - \vec{m}_j\| \) between the locations of \( \vec{m}_i \) and \( \vec{m}_j \). Where unambiguous, we use the term link to denote the length of a link.

We refer to a set of tracks, measurements, and established links between them as an assignment. Our task is to find the assignment that corresponds to
5. NENIA: NEAREST NEIGHBOR BASED INTER-FRAME ASSIGNMENTS

Figure 5.3: A sample assignment between two consecutive frames. Measurements are assigned to previously established tracks. In the assignment, there are four possible combinations that represent different cell events, such as moves, divisions, deaths and noise in cell detections.

the true set of cell events that occur between frames $f$ and $(f + 1)$. A popular approach for solving this kind of task is based on Bayesian inference. Here the likelihood of an assignment is expressed using the probabilities of individual events implied by the assignment $\{76, 88\}$.

Under the independence assumption $5.2$, we can express the likelihood of assignment $A$ as $L(A) = \prod_{e \in E(A)} P(e)$. Here $E(A)$ is the set of all events implied by assignment $A$. Event $e \in E(A)$ can be one of the following (Figure 5.3): cell move, cell division, cell disappearance due to cell death or a false negative error, and appearance of a new measurement due to a false positive error. The probabilities of the events in $E(A)$ can be expressed with the parameters $P_{s}$, $P_{d}$, $P_{fp}$, and $P_{fn}$ (see Appendix B.2). Now the problem is to find the assignment $A' = \arg\max_{A} L(A)$.

The search space of all possible assignments is usually reduced by introducing gating. Gating involves filtering out assignments with invalid links. An invalid link is a link with length $r_{ij} \geq R_{g}$, where $R_{g}$ is called the gating distance (or gating threshold). The exact definition of an invalid link varies in the literature $\{67, 76\}$, but the intuition is that the links that are deemed to be unlikely are not considered. In the case of our method, gating plays an important role. As we show later in this chapter, there is a dependency between the gating distance $R_{g}$ and the tracking performance.

So far we introduced an assignment problem formulation that closely fol-
5.2. METHODS

Figure 5.4: Two-round assignment approach. In a single round two-to-one links are not permitted, hence division events are eliminated from a single round. However, a division event can occur when there are measurements assigned to the same track in both rounds.

follows previous work [67, 76]. In what follows we present one of our contributions, a novel approach to the assignment problem.

Two Round Assignment Approach

Note that a straightforward solution of the problem (e.g., using integer programming) requires values for parameters $P_s$, $P_d$, $P_f^p$, and $P_f^n$. We propose to replace the four parameters by a single parameter $R_g$, and then formulate the problem as a maximization of performance with respect to $R_g$.

We claim that an approximate solution to the assignment problem can be found using the two round assignment procedure outlined below (Figure 5.4). The formal argument for our claim is presented in Appendix B.2.

In both assignment rounds we impose the following constraints: (1: no splits) at most one measurement can be assigned to one track, (2: gating) assigning measurement $j$ to track $i$ is only allowed if $r_{ij} < R_g$, and (3: maximum) we consider only the assignments with the maximum possible number of links.

Constraint 1 implies that in a single assignment round we eliminate the possibility of splitting a track. Instead, we capture a cell division event when in both rounds a measurement is assigned to the same track. Constraint 2 introduces the gating, and constraint 3 is explained as follows.

Consider, for example, assigning 2 measurements to 2 tracks, and suppose that no links are filtered out by the gating. There are a number of possible assignments: (a) link 1 — 1, and unassigned measurement 2, (b) links 1 — 1 and
2 — 2, (c) links 1 — 2 and 2 — 1, and so on. Constraint 3 states that assignment (a) cannot be considered because it consists of one link, whereas making two links is possible.

Recall that our task is to assign the measurements $\vec{m}_{f+1}$ in the next frame to tracks $T_f$ in the current frame. In the first assignment round, we find an assignment of measurements $\vec{m}_{f+1}$ to tracks $T_f$ that minimizes the cost $C = \sum_{[i-j] \in \text{links}} r_{ij}^2$, subject to constraints 1 — 3. Let $\vec{m}_{un} \in \vec{m}_{f+1}$ be the set of measurements (possibly empty) that were not assigned in the first round. In the second round, we find the assignment of $\vec{m}_{un}$ to $T_f$ that minimizes the same cost $C = \sum_{[i-j] \in \text{links}} r_{ij}^2$ subject to the same constraints.

Cases when two measurements (one from each round) are assigned to the same track represent cell divisions. All other links represent cell moves. The measurements that were not unassigned in both rounds represent false positives, and the tracks that received no measurements after the both rounds represent disappearance due to cell death or a false negative error.

Our two round assignment approach allows us to find an approximate solution for an assignment problem (see Appendix B.2). At the same time, our approach addresses the possibility of cell divisions and deaths, yet it uses a single and simple expression for the cost $C$. Essentially we have reduced our inter-frame association problem to the global nearest neighbor assignment task in each round. This problem can be solved using the polynomial time Hungarian algorithm (see [89] or Appendix A.3.2).

Moreover, our approach does not require values of $P_s$, $P_d$, $P_{fp}$, and $P_{fn}$. We replaced these parameters by a single parameter $R_g$. In the next section, we present a method for deriving the value of this remaining parameter.

### 5.2.3 The Choice of Gating Distance

In previous work, the choice of gating distance has not been considered from the perspective of tracking performance. Indeed, it is an open challenge to derive a principled procedure for determining the gating distance. For example, Jaqaman et al. [67] choose the distance “beyond which a link between two particles
Figure 5.5: A sample frame $f$ superimposed on frame $f+1$. Measurements in frame $f$ are labeled with numbers (e.g., 1). Measurements in frame $f+1$ are labeled with numbers with prime signs (e.g., 1'). Measurements with the same number correspond to the same cell, so 3' is a spurious measurement. Solid lines represent true links, and dotted lines represent false links.

was deemed impossible". Further, deriving the value for the gating distance has required prior knowledge about cell offsets, that is, it required training [50, 76]. In contrast, we provide an analytical framework for the choice of gating distance that maximizes tracking performance for a given video. Moreover, we propose an automated method for estimating the value of the optimal gating distance.

The Optimal Gating Distance

Previously, we approached the cell tracking problem as a likelihood maximization task. Now we consider the same tracking problem from a different perspective. We analyze how the tracking performance depends on the choice of the gating distance $R_g$. The performance is measured by the ratio of correctly reconstructed links $P_{\text{links}}$ (Equation 5.6).

Let us divide all possible links between measurements $\vec{m}_i$ and $\vec{m}_j$ into two groups. If the two measurements correspond to occurrences of the same cell, the link is a true link, otherwise the link is a false link (Figure 5.5). For a given distance $R$, let the number of true [false] links shorter than $R$ be $n_t(R) \ [n_f(R)]$. For brevity we also denote these values as $n_t$ and $n_f$. The tracking performance can then be expressed as a function $P_{\text{links}}(R) = P_{\text{links}}(n_t, n_f)$. Further, let the first derivative of $P_{\text{links}}$ with respect to $n_t$ ($n_f$) be denoted as $dP_{\text{link}}/dn_t$ ($dP_{\text{link}}/dn_f$).

Gating implies filtering out links that are longer than some gating distance $R_g$. In turn, filtering out a true link implies a swap error. Moreover, includ-
5. NENIA: NEAREST NEIGHBOR BASED INTER-FRAME ASSIGNMENTS

Figure 5.6: Hypothetical curve of the cell tracker performance as a function of gating distance. \( R_{t_{\text{max}}} \) (\( R_{f_{\text{max}}} \)) is the maximum length of a true (false) link. Our hypothesis is that the portion of allowed true links is the dominant factor that affects the performance (\( 0 \leq R_g < R_{t_{\text{max}}} \)). This factor is stronger than the portion of allowed false links (\( R_{t_{\text{max}}} \leq R_g < R_{f_{\text{max}}} \)). The choice of \( R_g \geq R_{f_{\text{max}}} \) corresponds to a tracking without gating. The tracking performance is the same for all \( R_g \geq R_{f_{\text{max}}} \).

If \( R_{t_{\text{max}}} \leq R_g < R_{f_{\text{max}}} \) then we have all true links involved in the tracking process, i.e., \( n_t \) is fixed. Now the only factor that changes is the portion of false links.

Consider an example that illustrates the dependency between the gating distance and the tracking performance (Figure 5.6). Here \( R_{t_{\text{max}}} \) (\( R_{f_{\text{max}}} \)) is the maximal length of a true (false) link. If \( 0 \leq R_g < R_{t_{\text{max}}} \) then both \( n_t \) and \( n_f \) are increasing according to \( R_g \). Under our hypothesis (Equation 5.7), the increasing proportion of true links is the dominating factor, hence the performance rises.

If \( R_{t_{\text{max}}} \leq R_g < R_{f_{\text{max}}} \) then we have all true links involved in the tracking process, i.e., \( n_t \) is fixed. Now the only factor that changes is the portion of false links.
links that are passed in tracking, and we have that \( dP_{\text{link}}/dn_f < 0 \), hence the performance decreases. Under our hypothesis, the part of the curve between 0 and \( R_{tmax} \) is steeper than the curve between \( R_{tmax} \) and \( R_{fmax} \). After the point \( R_{fmax} \), both \( n_t \) and \( n_f \) are fixed, thus the choice of \( R_g \) has no effect on the performance.

The optimal gating distance is the distance when NENIA achieves the best performance. Under our hypothesis, the optimal gating distance is the distance \( R_g = R_{tmax} \) (this suggestion is validated by our experiments). It is natural to assume \( R_{fmax} > R_{tmax} \). However, the optimal gating distance is the same \( R_g = R_{tmax} \) in the case when \( R_{fmax} < R_{tmax} \). In the case when \( R_{fmax} < R_{tmax} \) (diagram not shown), in the interval \( 0 \leq R_g < R_{fmax} \), the proportion of true links is increasing. Under our hypothesis, the proportion of true links is a dominating factor, hence the performance is increasing. In the interval \( R_{fmax} < R_g < R_{tmax} \), \( n_f \) is fixed and we also have \( dP_{\text{link}}/dn_t > 0 \). Thus the maximum performance is achieved at \( R_g = R_{tmax} \).

In order to find the point \( R_{tmax} \), we reconstruct the cumulative density function (CDF) for the true links (i.e., cell displacements) as described in the next section.

**Distribution of Cell Displacements**

Let \( P_t(r \leq R) \), \( P_f(r \leq R) \), and \( P_{all}(r \leq R) \) be the probabilities that the length of a true, false, and any link is less than a certain distance \( R \), and let \( CDF_t \), \( CDF_f \), \( CDF_{all} \) be the cumulative distribution functions for true, false and all links respectively.

We can empirically estimate \( CDF_{all} \) by considering all possible links between any two measurements in consecutive frames in the video.

Now consider false links. These links connect two unrelated measurements in consecutive frames. Under Assumption 5.3, the spatial distribution of cells within a frame does not change greatly, hence connecting two unrelated measurements in consecutive frames is approximately the same as connecting unrelated measurements within a frame. This can be illustrated using two superimposed frames (Figure 5.5). We therefore propose to approximate \( CDF_f \) with...
the distribution of links between measurements within a frame.

Now let \(N_t, N_f\) and \(N_{all}\) be the numbers of true, false and all links in a video. Here \(N_t + N_f = N_{all}\). We can calculate \(N_{all}\) by counting the number of all potential links in the video. At the moment, assume that we know \(N_t\).

At a given \(R\) let \(n_t, n_f\) and \(n_{all}\) be the numbers of true, false and all links that are less than \(R\). Note that \(n_t + n_f = n_{all}\). Then \(P_t(r \leq R) = \frac{n_t}{N_t}, P_f(r \leq R) = \frac{n_f}{N_f}\) and \(P_{all}(r \leq R) = \frac{n_{all}}{N_{all}}\), and for all \(R\), the CDFs can be reconstructed using the following equation

\[
P_t(r \leq R) = \frac{N_{all}P_{all}(r \leq R) - (N_{all} - N_t)P_f(r \leq R)}{N_t}. \tag{5.8}
\]

Finally, we show how to find \(N_t\). Consider a frame \(i\) with \(n_i\) measurements (Figure 5.7). Among these measurements there are on average \(n_iP_{fp}\) false positives, hence we expect on average \(n_i(1 - P_{fp})\) cells. However, on average \(n_i(1 - P_{fp})P_{fn}\) of these cells are not detected in the previous frame \(i - 1\). Therefore, there are on average \(n_i(1 - P_{fp})(1 - P_{fn})\) true links between frames \(i - 1\) and \(i\), and we can approximate \(N_t\) as

\[
N_t \approx (1 - P_{fp})(1 - P_{fn}) \sum_{i=2}^{N_f} n_i, \tag{5.9}
\]

where \(N_f\) is the number of frames in the video.

Let \(\hat{N}_t \in (0, N_{all})\) be an approximation of \(N_t\). We then set the gating distance
\[ R_g = \hat{R}_{tmax}, \text{ where } \hat{R}_{tmax} \text{ is the first point such that} \]
\[ \frac{N_{all} P_{all}(r \leq R) - (N_{all} - \hat{N}_t) P_f(r \leq R)}{\hat{N}_t} = 1. \]  
\[ (5.10) \]

We refer to the method of estimating \( N_t \) using Equation 5.9 as method A. Here we require two parameters \( P_{fp} \) and \( P_{fn} \) for the estimation of \( N_t \). Estimation of \( P_{fp} \) and \( P_{fn} \) generally involves manual verification of cell detection results. Therefore, in the rest of this section, we present method B — an approximation of \( N_t \) without these parameters.

Let us denote \( Q = (1 - P_{fp})(1 - P_{fn}) \). Notice that, under Assumption 5.1 about cell detection quality, \( Q \in [0.9; 1] \). Further, note the following property.

**Lemma 5.4.** If \( \hat{N}_t \geq N_t \) then \( \hat{R}_{tmax} \geq R_{tmax} \).

Recall that our hypothesis was that the part of the error curve (Figure 5.6) that is on the left of \( R_{tmax} \) is steeper than the part on the right. Further, on the right hand side, there is an intercept value \( R_{f_{max}} \), beyond which we do not have performance degradation. Thus, we conclude that it is “safer” to overestimate \( R_{tmax} \) rather than underestimate. We therefore suggest to use
\[ N_t \approx \hat{N}_t = \sum_{i=2}^{N_t} n_i \]  
\[ (5.11) \]
as an approximation for \( N_t \).

Algorithm 5.1 summarizes our method for deriving the gating distance value. Method A and method B differ only in the way they set the value \( N_t \).
Algorithm 5.1 Estimation of Gating Distance

1: function EstimateGatingDistance($M, P_{fp}, P_{fn}$)
2:      // $M$ is the set of measurements in the entire video;
3:      // $P_{fp}, P_{fn}$ are not required in method B
4:      method A: use $P_{fp}, P_{fn}$ and Equation 5.9 to estimate the number of true links $N_t$;
5:      (or method B: use Equation 5.11 to estimate the number of true links $N_t$);
6:      use Equation 5.10 to estimate the distribution of true links;
7:      find the minimum $R_g$ such that $P_t(r \leq R_g) = 1$;
8:      return $R_g$

We have now described the key parts of our tracking algorithm. In the following section, we summarize the algorithm.

5.2.4 Automated Cell Tracker NENIA

Previously, we formulated the cell tracking task as a problem of iterative assignment between two frames. However, in each iteration, this method does not take into consideration the information available outside the two frames. Iterative inter-frame assignment represents a greedy approach to cell tracking.

Cell tracking can be formulated as a global assignment problem, but then the solution becomes computationally infeasible even for relatively short videos. It is, therefore, common to use heuristic methods that represent trade-offs between the greedy and global approaches [76,111].

Our cell tracking algorithm (see Algorithm 5.2) uses inter-frame assignments supplemented with a track management module. The track management module uses the information available outside the two frames to improve the inter-frame assignments.

The algorithm takes as input sets of measurements in each frame of a video and produces a set of established tracks with lineage relations between tracks.
5.2. METHODS

Algorithm 5.2 Parameter-free tracker NENIA

**Input:** measurements $\vec{m}_1, ..., \vec{m}_F$; detection quality $P_{fp}, P_{fn}$ (these two are not required in method B);

**Output:** set of tracks $T$;

1. $T \leftarrow \text{InitNewTracks}(\vec{m}_1)$;
2. $R_g \leftarrow \text{EstimateGatingDistance}(\vec{m}_1, ..., \vec{m}_F, P_{fp}, P_{fn})$;
3. for each frame $f = 2...F$
   4. $\text{mapping}_1 \leftarrow \text{FirstAssignment}(T, \vec{m}_f, R_g)$;
   5. $T \leftarrow \text{UpdateTracks}(T, \text{mapping}_1)$;
   6. $T \leftarrow \text{ManageLostTracks}(T, \text{mapping}_1)$;
   7. $T \leftarrow \text{SplitTracks}(T, \text{mapping}_1)$;
   8. $\vec{m}_{\text{un1}} \leftarrow \text{UnassignedMeasurements}(\text{mapping}_1)$;
   9. $\text{mapping}_2 \leftarrow \text{SecondAssignment}(T, \vec{m}_{\text{un1}}, R_g)$;
10. $T \leftarrow \text{MarkDivCandidates}(T, \text{mapping}_2)$;
11. end
12. return $T$;

At each iteration, an inter-frame assignment is performed in two rounds (lines 4 and 9). Cell move events are processed by appending a new measurement to the corresponding track (line 5).

NENIA uses an extra frame to supplement inter-frame assignments. The extra frame is implicitly used in functions *ManageLostTracks* (line 6) and *SplitTracks* (line 7). At iteration $i$, these functions use the result of the assignment between frames $i - 1$ and $i$ to complete processing of tracks based on the assignment between frames $i - 2$ and $i - 1$. Specifically, the disappearance events in the first assignment round between frames $i - 2$ and $i - 1$ result in marking the corresponding tracks as lost. A track is then terminated only if it is lost in two consecutive frames (line 6).

Similarly, in the second round of assignment between frames $i - 2$ and $i - 1$, there can be division events, but then the mother track is only marked as a division candidate (line 10). The division is processed only if both daughter
tracks persist in two consecutive frames (line 7).

In summary, lines 4, 8, and 9 implement our two round assignment approach; line 5 updates tracks; line 6 records lost tracks; and lines 7 and 10 implement track splitting.

Lemma 5.5. The probability of encountering a sequence of two or more false positive errors linked together by NENIA is less than 0.05. Similarly the probability of encountering a sequence of two or more false negative errors missing from the same track is less than 0.05.

Lemma 5.5 gives an upper bound estimation for the probabilities. In practice, we expect sequences of related errors to be even more rare. Based on this lemma we decided to use only one look-ahead frame. We find that such a strategy can be efficient in practical scenarios as we demonstrate in the following section.

5.3 Evaluation

The goal of our experiments has been to evaluate the proposed automated and semi-automated cell tracking methods across a variety of tracking scenarios. To this end, we used five videos that differ in cell density and motility. The videos are described in Section 5.3.1.

We implemented our cell tracking system in MATLAB, version R2009a. In our system, we used one of the assignment algorithm implementations freely available from the Web. This implementation is based on the Hungarian algorithm, and is capable of processing rectangular cost matrices with infinite costs representing prohibited links.

In the gating distance estimation, the CDF of true links was reconstructed as described in Section 5.2.3 (with both methods A and B). We also applied median filtering to reduce noise in the reconstructed CDF curve. See Appendix B.4 for details of the filtering.

5.3. EVALUATION

Table 5.3: Videos used for tracker evaluation. These videos differ in cell density, and motility. The parameter $\beta_n$ reflects the “difficulty of video” for cell tracking (defined in Section 5.3.3).

<table>
<thead>
<tr>
<th>Name</th>
<th>Frames</th>
<th>Cells</th>
<th>$\beta_n \times 10^2$</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ak$</td>
<td>300</td>
<td>18</td>
<td>0.25</td>
<td>dense clutter of cells, low motility, 7 divisions</td>
</tr>
<tr>
<td>$hex.6$</td>
<td>100</td>
<td>6</td>
<td>5.94</td>
<td>starts with 2 cells, 2 divisions</td>
</tr>
<tr>
<td>$hex.16$</td>
<td>100</td>
<td>16</td>
<td>3.45</td>
<td>dense clutter of cells, no events, except cell moves</td>
</tr>
<tr>
<td>$hex.22$</td>
<td>100</td>
<td>22</td>
<td>3.18</td>
<td>dense clutter of cells, 4 divisions, 2 deaths</td>
</tr>
<tr>
<td>$square$</td>
<td>50</td>
<td>35</td>
<td>0.14</td>
<td>cells are uniformly distributed with low density, 2 divisions</td>
</tr>
</tbody>
</table>

We evaluate the effect of the gating distance on the tracker’s performance in Section 5.3.2. In the same section, we compare the performance of NENIA with the performance of an advanced probabilistic cell tracker. Finally, we evaluate NENIA on synthetic videos in Section 5.3.3.

5.3.1 Cell Videos

The videos used in our evaluations are summarized in Table 5.3. These video sequences are extracted from the same cell videos that we used in previous chapters for evaluation of semi-segmentation and segmentation algorithms. However, these are not the same frames that we used previously. For evaluating cell tracking, we need a ground truth for a large number of subsequent frames. Therefore, we cropped a region from the original videos, but followed the region for a large number of frames. This is different from our previous evaluation (Chapters 3 and 4), where we did not crop the original images, but looked only at a few frames.

Video $ak$ shows the development of neural progenitor cells. This video sequence consists of the first 300 frames (enough to observe 3 division rounds) of supplementary movie one2 from the work of Al-Kofahi et al. [6].

---

2Used with permission. Available on-line from the Cell Cycle journal website at http:
Figure 5.8: Fragments of video frames used in this study. The images show B lymphocytes at different scales (left, middle), and neural progenitor cells (right). There is a variation in cell dynamics: videos with B lymphocytes contain abrupt cell motions, while neural progenitor cells tend to have smooth motions. There is also a variation in cell density. B lymphocytes images courtesy of Hodgkin Lab, WEHI. The neural progenitor cells image is from Al-Kofahi et al. [6].

Figure 5.9: Frames from videos (left to right) hex.6, hex.16, and hex.22. These videos have similar appearance, however we found differences in cell dynamics across the videos. Hence the same detection algorithm can be used, but the variability challenge still needs to be addressed by the tracker. Images courtesy of Hodgkin Lab, WEHI.

The remaining four videos show the development of B lymphocytes. Primary B lymphocytes were purified from the spleen of C57/BL6 mice modified to ubiquitously express GFP, prepared as described in the work of Hawkins et al. [61] These were placed in 250 µm hexagonal microwells (videos hex*) and 125 µm square microgrids (video square), and photographed once every 2 minutes using an Axiovert 200m microscope with a 20× objective at different scales for hexagonal and square microwells.

Figure 5.8 shows the fragments of frames from videos hex.16 (left), square (middle), and ak (right), and Figure 5.9 shows sample frames from videos hex.6 and hex.22.

We find that our videos represent a variety of tracking scenarios, such as

different cell density and dynamics (see details in Appendix B.5). This is despite four of the videos showing the same type of cells. Further, three videos give an example of variation in cell density and dynamics, while having similar cell appearance. In this case, the same detection algorithm can be used, but the variability challenge still needs to be addressed by the tracker.

Some of the previously proposed methods made an assumption of the Brownian motion of cells [13, 67]. However, in our videos, we found that at least in four of them cells are unlikely to follow Brownian motion (see details in Appendix B.5).

In order to evaluate our tracker on detections with different quality, we first manually produce a cell detection ground truth for all datasets. Then, for each video, we generate detections with different qualities specified by \( \{P_f^p, P_f^n\} \). We refer to a combination \( \{\text{video}, P_f^p, P_f^n\} \) as a case. For each case, we generate 5 random versions (see details in Appendix B.5) and report the average performance across 5 versions. Cases with \( P_f^p + P_f^n \leq 0.06 \) correspond to high detection quality.

We do not report the sizes of videos in pixels, because the sizes are not directly related to the tracking performance. Instead, we report a dimensionless parameter \( \beta_n \) which is the normalized object density. This parameter is formally defined in Section 5.3.3.

The number of frames in our videos is not very large. In practice, videos can comprise thousands of frames. However, our videos include several instances of cell division and death events. Having short videos does not affect our evaluation, since NENIA is based on independent frame by frame assignments. Further, we compare our tracker with a competitor’s tracking algorithm called LJIPDA (introduced in the next section). Having short videos does not affect the evaluation of LJIPDA, because LJIPDA is also assignment based (however, LJIPDA uses a different approach for the assignment).

Despite the variety in cell density and dynamics, NENIA is capable of achieving high tracking accuracy on different detection qualities, as illustrated in the next section.
5.3.2 Evaluation of NENIA

In this section, we first analyze the performance of NENIA on different detections and compare it with the performance of a probabilistic tracker. We then compare the tracking performance for estimation methods A and B. For each video, all video frames are used for automated derivation of the gating distance and evaluation of performance. The derivation of the gating distance parameter is one of the crucial steps in NENIA. Therefore, we conclude with the measurement of the effect of gating distance on the performance of our cell tracker (measured with $P_{\text{links}}$).

Figure 5.10 shows the performance of NENIA (with method A for selection of the gating distance) compared with the performance of the Linear Joint Integrated Probabilistic Data Association (LJIPDA) tracker. We choose LJIPDA because it represents a different approach to tracking, and it is one of the most advanced probabilistic trackers [91].

When the detection quality is high (cases when $P_{fp} / P_{fn} = 0/0, 1/0, 0/1, 3/3$, expressed in %), NENIA outperforms LJIPDA and the difference is statistically significant. On average across all cases, there is no statistically significant difference in performance (the statistical significance was measured with the Wilcoxon matched-pairs signed-rank test at the 5% significance level). At the same time NENIA does not require setting parameter values manually, whereas for LJIPDA we have to use the manually produced ground truth to tune the parameter values for LJIPDA.

We now compare the performance corresponding to methods A and B in Figure 5.11. On average across all cases, method A is better and the difference is statistically significant as measured with the Wilcoxon matched-pairs signed-rank test at the 5% significance level. On the other hand, in all observed cases, method B recovers more than 80% of cell moves correctly, thus it still performs well enough to be useful. The two methods provide a trade-off: if the detection quality can be estimated easily, then method A gives better tracking accuracy; if it is hard to estimate the detection quality, then method B can still be useful.

Finally, Figure 5.12 shows empirical curves for the tracking performance as a function of gating distance. It can be observed that in all cases, empirical data
Figure 5.10: Performance of NENIA compared with the performance of LJIPDA on different detection qualities. Over all cases, both algorithms have similar performance. NENIA outperforms LJIPDA on high detection qualities. The lines connecting points have no interpretation and are given for perceptual convenience. Note the different scales on the y-axes.
Figure 5.11: Tracking performance of NENIA with method A compared with method B. On average, method A outperforms method B. Both methods recover more than 80% of cell moves correctly in all cases. The lines connecting points have no interpretation and are given for perceptual convenience. Note the different scales on the y-axes.
5.3. EVALUATION

supports our hypothetical curve (compare with Figure 5.6). When the gating
distance is too small, filtering out a part of the true links has a dramatic effect
on the performance. When the gating distance lets all the true links in, the
performance depends on the proportion of the true and false links. However,
when the gating distance is too large and allows all true and false links, the
performance does not depend on the gating.

Vertical strokes in Figure 5.12 show the gating distances, estimated by method
A. Both methods A and B estimate suboptimal gating distances (strokes are not
shown for method B). The optimal performance, on average, is only 1.6% higher
than the performance of method A (data not shown).

When the cell detection is perfect ($P_{fp} = P_{fn} = 0\%$), NENIA is able to recover
all division events correctly. We further investigate the ability of NENIA to
resolve cell divisions in our synthetic experiments in the next section.

We summarize our evaluation results as follows.

- For high cell detection quality, NENIA resolves almost 100% of moves cor-
rectly and can be used as a main tracking algorithm. In all other cases, NE-
NIA (with method A or B) resolves more than 80% of cell moves correctly
and could be used for collecting the ground truth for training a more spe-
ialized tracking algorithm. The ground truth can first be automatically
generated by NENIA. Then a human operator is required to label at most
20% of links.

- For high cell detection quality, NENIA (with method A) outperforms LJIPDA,
despite NENIA being a simpler parameter-free algorithm.

- Method A for determining the gating distance outperforms method B.
However, in the case when the estimation of $P_{fp}$ and $P_{fn}$ is hard, method
B still gives reasonable performance, more than 80% in all cases.

- The running time for our cell tracker was on average 65 frames per second
(i.e., in total, 10 seconds for all 5 videos, which includes the estimation of
gating distance) run on a desktop PC (Intel Core 2 Duo CPU E8400 @
3.00 GHz and 3.4 GB RAM, Ubuntu 9.10). The scalability of NENIA for
larger numbers of cells is evaluated in the next section.
Figure 5.12: Empirical cell tracking performance curve as a function of gating distance for detections with different qualities. Empirical performance curves support our hypothesis about the dependency of performance on the gating distance (compare with Figure 5.6). Vertical strokes correspond to the gating distances estimated by a method A. Note the different scales in the top two diagrams (y-axes), and all diagrams (x-axes).
5.3. EVALUATION

We emphasize that, in method B, rather than directly estimating the optimal gating distance from data, we (a) impose a set of assumptions on characteristics $P_s$, $P_d$, $P_{fp}$ and $P_{fn}$ (and justify these assumptions, see Section 5.2.1), and (b) use analytical properties of the gating distance (see Section 5.2.3) to set an estimate that minimizes the expected performance loss. We have also used our assumptions in order to set reasonable prior values for LJIPDA parameters. However, LJIPDA has over 10 parameters, some of which are less intuitive (e.g., process and sensor covariance matrices or track confirmation probability). At the same time, wrong parameter settings can lead to a significant LJIPDA performance degradation (data not shown). Hence, requiring a user to provide only one or none parameters is an important advantage of NENIA.

5.3.3 Evaluation on Synthetic Videos

In addition to our real videos, we use synthetic videos for evaluating different aspects of NENIA: scalability in terms of the tracking performance and running time, validity of our hypothesis about tracking performance as a function of gating distance, and the ability of NENIA to resolve cell divisions. In this section, we outline the main findings on our synthetic videos. Below we outline our synthetic videos, and we give additional details in Appendix B.6.

The first type of synthetic video that we use consists of pairs of frames. The cells are randomly placed with a uniform distribution in the first frame, and randomly moved with a bivariate normal distribution. In the synthetic videos we vary a number of parameters, such as the number of cells $N$, the variance of a random offset $\sigma$, and the side of a square video frame $L$. We also add random false positives and false negative errors with error rates $P_{fp}$, $P_{fn}$.

Mori et al. [88] define a normalized object density as a dimensionless value $\beta_n = (N \cdot \sigma^2 \cdot \pi / L^2)$ and show that this density determines the “difficulty” of the tracking problem (a similar observation has been made by [67]). An intuitive interpretation is that, given a fixed frame size, increasing the number of cells or cell speeds increases the probability of confusing their tracks, and hence makes tracking more difficult.

For the evaluation of tracking scalability in terms of the performance and
running time, we vary the number of cells $N$ in the range $30 - 200$ and use several different sets of other parameters. Overall we find that once the normalized density, $\beta_n$, is fixed, the tracking performance, $P_{\text{links}}$, does not depend on the number of cells. This is consistent with the result of Mori et al. [88].

Further, we found that the dependency of the running time on the number of cells can be fit with a degree 3 polynomial (see Appendix B.6). The empirical polynomial complexity of NENIA is consistent with the theoretical complexity estimation for the assignment algorithm [89], which is a core step of NENIA.

For testing the relationship between the tracking performance and the gating distance, we vary $\beta_n$ such that it covers a much wider range of values than the real videos. We use a few different sets of other parameters, and recorded the performance of NENIA $P_{\text{links}}$ as a function of gating distance $R_g$. All obtained empirical curves (see examples in Figure 5.13, not all curves shown) preserve the prominent features of our hypothesized curve (Figure 5.6): sharp rise from zero to maximum, maximum around the point $R_g = R_{t_{\text{max}}}$, and gradual decrease, after this point (recall that $R_{t_{\text{max}}}$ is the maximum length of a true link). We conclude that it is feasible to estimate the optimal gating distance as $R_g = R_{t_{\text{max}}}$, as proposed in our analysis (Section 5.2.3).

Synthetic videos cover both cases when $R_{t_{\text{max}}} < R_{f_{\text{max}}}$ and $R_{t_{\text{max}}} > R_{f_{\text{max}}}$ (recall that $R_{f_{\text{max}}}$ is the maximum length of a false link). The cases when $R_{t_{\text{max}}} > R_{f_{\text{max}}}$ occur at high values of $\beta_n$ ($\beta_n > 300$). These cases correspond to a situation when, between consecutive frames, cells travel distances that are large compared to the distances between cells in a frame. This results in a rapid confusion of cells and consequently in poor tracking performance. We find that the empirical performance curve in this situation ($R_{t_{\text{max}}} > R_{f_{\text{max}}}$) still resembles our analytical curve.

Finally, we use another type of synthetic video to evaluate the ability of NENIA to resolve cell divisions. These videos consist of 3 frames, and start with 100 cells out of which 30 cells divide. We generated the synthetic videos using different combinations of parameters $\beta_n$ and $P_{fp}$. See the details of our evaluation in Appendix B.6.

We find that for every parameter combination, the majority of divisions are resolved correctly: the ratio is about 100% for the easiest cases and about 72%
Figure 5.13: Empirical performance curves as a function of gating distance obtained on synthetic videos. When the normalized object density is low (left) the tracking performance remains high for different detection levels (the performance curves overlap, therefore only the case $P_{fp} = P_{fn} = 5\%$ is shown). When the object density is high (right), the performance depends on the detection quality. Dots correspond to actual data points. The bold vertical stroke marks the point $R_g = R_{t_{max}}$. At this point the performance is maximized or near the maximum. The empirical performance curves preserve prominent features of our analytical curve (compare with Figure 5.6).
for the hardest cases. We cannot compare these numbers with the performance of the LJIPDA algorithm, because LJIPDA does not address cell divisions, but we note that our results are somewhat similar to the results of Padfield et al. \cite{97}. The evaluation of Padfield et al. roughly corresponds to our case when $P_{fp} = 0\%$. This comparison is indirect, as different authors use different videos (we do not have their algorithm implementation).

Overall, we find NENIA to be scalable in terms of performance and running time. Further, we verify our hypothesis about the relation between the tracking performance and the gating distance. We also find that in our synthetic videos, NENIA correctly resolves most cell divisions.

Further, we note that incorporating a physically well-conserved observable quantity such as cell fluorescence into NENIA can improve its ability to resolve divisions (this is the subject of future work). Finally, recall that cell moves significantly outnumber cell divisions in real videos. Therefore the accumulation of errors from cell moves in between divisions needs to be taken into account when comparing a tracker’s ability to maintain cell identity between and across divisions.

5.4 Discussion

In our cell tracking evaluation, we only use cell centroid locations because it appears that this feature is available with any cell segmentation algorithm (and with our semi-segmentation algorithm). We find that the performance of NENIA is reasonably high even with this minimum amount of information. In practice, there can be more information available, such as cell size or fluorescence. These extra features can be seamlessly incorporated in NENIA to improve the performance further.

New features can be added to the Euclidean distance expression after normalizing by the features’ variances. The segmentation algorithm produces sets of features, such as centroid locations and the set of cell areas, and the variance of each feature can be calculated automatically. Therefore, after incorporating new features, NENIA does not require extra human attention.
In NENIA we use the same gating threshold and division probabilities for all cells. We note that in reality the lengths of cell moves and the division probabilities are cell and time dependent. Varying the parameters across cells might further improve the accuracy of tracking. However, estimating the division probability or the gating threshold for a particular cell requires tracking cell identities in the first place. It is possible to use tracking results for the past frames to estimate the gating threshold for a cell in the future frames [67], but the past tracking results may contain errors and therefore the estimation can be inaccurate. To the best of our knowledge, there is no evidence yet in the literature that such an adaptive method is superior to one that uses a common gating threshold (comparative analysis can be a direction for future work). A similar remark can be applied to adaptive estimation of division probabilities. Further, we note that on the range of videos tested, NENIA (with a common gating threshold and division probability) performs sufficiently well to be useful in practice.

NENIA with method B is a parameter-free algorithm in the sense that for its operation it only requires the problem input (i.e., cell detection), and does not require user attention. Internally NENIA does use some variables, such as the gating distance \( R_g \), but these are initialized automatically. Therefore the use of internal variables is immaterial for end-users. Note that this is not the case with many previously proposed algorithms, where a user is required to somehow determine the value for a gating threshold [6, 67, 76].

NENIA with method A requires the user to provide an estimation of the quality of cell detection \( (P_{fp}, P_{fn}) \). In some cases, one can assume that the detection quality remains roughly the same across a number of videos (e.g., the videos show the same cells at about the same density). In these cases, a manual estimation of the detection quality can be done for one video and then used for the other videos. In cases where the similarity assumption cannot be made, we suggest to use NENIA with method B. We emphasize that providing accurate estimations for the detection quality is not critical for our cell tracker as our experiments show that NENIA with method B can still achieve reasonable performance.

Some variables in NENIA are fixed to particular values (e.g., the form of the
cost function and the number of look-ahead frames). However, for each such variable we provide an argument as to which particular values are reasonable under the assumptions 5.1–5.3.

NENIA was designed to be portable across a variety of videos. In particular cases, specialized algorithms can achieve higher performance. For example, there can be a tracker optimized to certain cell dynamics. The problem is, given a video, to find out which particular case it represents. Generally, the cell dynamics are not known a priori. To address this problem, NENIA can be used in a bootstrap-like procedure: it can assist manual collection of the ground truth for a part of the video. The ground truth can then be used to decide which specialized algorithm to use for the rest of the video.

Finally, we would like to highlight our method for the reconstruction of the CDF of true links. This method was presented here in the context of our cell tracking algorithm. However, this technique does not depend on the tracking algorithm and has further potential applications as follows.

- The method can be used for the estimation of the average speed of cells (without doing cell tracking). This, in turn, can be used for deriving an optimal frame acquisition rate for cell videos.

- The method can be used for estimation of a cell tracker’s performance in the absence of a ground truth. (We take this research direction in Chapter 7.)

### 5.5 Conclusions

We develop and evaluate a novel cell tracking algorithm called NENIA. The algorithm can handle cell divisions and deaths, and is portable across a variety of videos. We also propose a parameter-free configuration of NENIA. Our algorithm is based on inter-frame assignments supplemented by a track management module. We first design the algorithm in such a way that it depends on a single parameter, the gating distance. Further, we analyze the effect of gating distance on the tracker’s performance, and propose methods for automated estimation of the gating distance, hence removing this parameter.
For the range of videos tested, NENIA correctly resolves on average 96% of cell moves, and when the quality of cell detection is high, NENIA correctly resolves almost 100% of cell moves. This performance is achieved across a variety of real cell videos that differ both in cell density and dynamics. The scalability of our algorithm is tested on synthetic videos with up to 200 cells per frame. The empirical complexity of NENIA is found to be a polynomial of degree 3 on synthetic videos.

NENIA is one of the major contribution of the thesis towards a more portable cell tracking pipeline. The portability of cell segmentation and tracking methods is one way to approach the goal of our thesis: minimizing the amount of manual effort. In the rest of this thesis, we explore alternative ways to achieve this goal. In the next chapter, we present a method that assists tracking results correction in a semi-automated fashion. Finally, in Chapter 7, we present a novel automated method capable of ranking cell tracking systems in the absence of a ground truth.
In this chapter, we explore a semi-automated strategy for improving tracking results. Cell tracking is a difficult task, because a single inter-frame mis-association can invalidate the entire cell track. In practical scenarios, even state of the art automated methods still do not achieve 100% of accuracy [79, 97]. At the same time, producing as many entirely correct tracks as possible can be important for reconstructing cell lineage trees [46, 61, 125]. Therefore, a manual post-processing stage can be required to improve the results further. Here we propose a novel semi-automated tracking framework that aims to facilitate this manual post-processing stage.

The key idea is that instead of looking at every video frame, a user might like to only look at the frames that are likely to contain errors. Real cell videos can be long (thousands of frames). At the same time, a good cell tracker can make only a few mistakes [79, 97]. Thus, identification of problematic frames can significantly reduce the amount of time required for the post-processing stage. The challenge is to identify the likely errors in tracking.

Recently there has been some work addressing this problem. Adanja et al. [5] propose to look at the situations when two tracks come too close to each other, because it is likely that such tracks will be confused. Chen et al. [22] propose to sort automatically constructed links according to their costs. The links with unusually high costs are likely to be incorrect. However, both methods cannot spot missing correct links. There is also no estimation of the effective-
Figure 6.1: Overview of our semi-automated tracking method. The method is based on predicting frames that are likely to contain tracking errors.

In summary, the contribution of this chapter is a novel semi-automated tracking framework SAMTRA. This framework allows us to improve the tracking quality while reducing the overall amount of manual work. In the next three sections, we present details of SAMTRA and our evaluation of this method, and draw conclusions.

6.1 Learning the Probability of Error

Tracking can be improved using human assistance, where an operator reviews the video superimposed with the tracking results. Once the operator detects a tracking error, he or she can intervene to correct those errors. However, the drawback of this approach is that the operator has to review all frames. Therefore, we suggest to identify and highlight frames with the highest probability of error.

In Chapter 5, we show that using only cell centroids (i.e., detection) as input, it is possible to achieve an accurate tracking. Furthermore, cell centroids are a feature that can be expected from any cell segmentation or cell detection algorithm. In order to make our method general, we assume that cell tracking was performed using only detections as input.

Furthermore, we assume that correcting a detection error is easier (takes less
6.1. LEARNING THE PROBABILITY OF ERROR

human time) than correcting a tracking error. For example, a typical tracking error, such as swapping of tracks, involves two subsequent frames. That is, the user must pay attention on two frames and the cell associations between them. At the same time, a detection error can be identified from a single frame, and can be resolved in a single mouse click (e.g., left click to add a missing cell, and right click to remove a spurious cell).

Evaluations of tracking algorithms (see [67], or Chapter 5 of this thesis) suggest that better cell detection leads to better tracking. In SAMTRA, we employ this observation, along with the above assumption. We train SAMTRA to identify detection errors, rather than tracking errors. Our evaluation confirms that the quality of tracking can be increased by correcting detection errors.

SAMTRA predicts the probability of detection errors in a frame using a set of features that are collected during tracking. Let $M_f$ be the number of measurements in frame $f$. Consider a transition from frame $f-1$ to frame $f$. Suppose that at this point, according to some tracker, $T_f$ existing tracks are updated, $N_f$ new tracks are initiated, $L_f$ lost tracks are terminated, $R_f$ lost tracks are recovered, and $S_f$ tracks are split.

For each updated track $t_i$, let a displacement between the track positions $x_{f,i}$ and $x_{f-1,i}$ at frames $f$ and $f-1$ be $r_{f,i} = \|x_{f,i} - x_{f-1,i}\|^2$, and the mean displacement over all updated tracks for frame $f$ be

$$R_f = \frac{1}{T_f} \left( \sum_{i=1}^{T_f} r_{f,i} \right). \quad (6.1)$$

We use 11 tracker independent features $\mathcal{F} = \{M_{f-2}, M_{f-1}, M_f, M_{f+1}, N_f, L_f, R_f, S_f, R_{f-1}, R_{f+1}, R_f/\sqrt{R_{f-1}}\}$ in a linear logistic regression with the logit

$$z = \omega_0 + \sum_{i=1}^{11} \omega_i F_i, \quad (6.2)$$

where $F_i$ enumerates the features in $\mathcal{F}$ and $\omega_i$ are feature weights.

Let $k_1$ and $k_2$ be chosen such that $k_1 + k_2$ is the number of frames that is considered feasible for manual inspection, and $k_1$ is the number of instances that is sufficient to train the logistic regression classifier. We then take $k_1$ frames from
a video and run a cell segmentation algorithm. We then manually generate the response vector. For each frame, the corresponding value in the vector equals to 1 if there is a detection error, or 0 otherwise. Using this manually collected response vector, we learn the weights $\omega_i$ using the logit in Equation 6.2 with features $\mathcal{F}$. Finally, for the rest of the video, or for other similar videos, we estimate the probability of a detection error in frame $f$ as

$$P_{err} = \frac{1}{1 + e^{-z}}. \quad (6.3)$$

Finally, we take $k_2$ frames with the highest probabilities of error, and manually correct cell detection (not tracking) in these frames. This results in improved tracking quality, as shown by our evaluations, presented in the next section.

### 6.2 Evaluation of SAMTRA

In our evaluation of SAMTRA, we simulate a practical scenario, where a user decides to improve the overall tracking quality by correcting a few frames suggested by SAMTRA. We first choose the tracking quality measure for the evaluation.

There have been a number tracking quality measures proposed (Section 2.5.3), and in Chapter 5, we used $P_{link}$ that reflects the proportion of individual inter-frame mis-associations. In the case of SAMTRA, our end goal is to improve the number of entirely reconstructed tracks. Therefore we use an appropriate measure such as follows.

Let a correct track be a track that includes all occurrences of the same cell and does not include any other measurements (the correct track may include dummy measurements). Each distinct cell in a video has a corresponding correct track.

Furthermore, let $\#corr$ tracks be the number of correct tracks produced by a cell tracker, and $\#cells$ be the number of distinct cells in a video (we count all cells that have appeared in the video in at least one frame). We then use the
6.2. EVALUATION OF SAMTRA

performance measure as proposed by Li et al. \cite{79},

\[ P_{tracks} = \frac{\#corr \ tracks}{\#cells}. \]  \hspace{1cm} (6.4)

We use this measure in the experimental setup described below.

6.2.1 Experimental Setup and Results

Our evaluation of SAMTRA was conducted using five real cell videos described previously in Chapter 5. We train SAMTRA using only one of the videos (hex.22) with one detection quality level \( (P_{fp} = P_{fn} \approx 2.5\%) \). We then use all our videos at other detection levels (other values for \( P_{fp} \) and \( P_{fn} \)) to evaluate SAMTRA. For each video, we randomly choose a detection such that the tracking quality on that detection is around 75\%. We then randomly select 10\% of the frames and corrected the cell detection, where necessary, in the selected frames. We use the random selection as our baseline. We then run the tracking again and record the tracking performance. In the evaluation, we use NENIA with method B (our parameter free tracker, see Chapter 5).

After that, we take the original detection and again select 10\% of frames. This time we select frames with the highest probabilities of errors as predicted by SAMTRA. Where necessary, we correct cell detections in these frames, run tracking again, and record the tracking performance. For each video we run the experiment 5 times (Table 6.1 shows mean values). SAMTRA is generally better than random selection. The difference in performances across all videos achieved by SAMTRA and random recovery is statistically significant (Wilcoxon rank sum test, \( p < 0.005 \)).

The results show that using SAMTRA we are able to increase the ratio of correct tracks on average by 13\% (recall that a correct track has all its links correct). This is achieved by inspecting only 10\% of frames and correcting cell detections in those frames. Also note that in most experiments, selecting frames for review using SAMTRA leads to more substantial performance gains than random selection. These results demonstrate that SAMTRA can be used for efficient semi-automated tracking correction.
Table 6.1: Tracking performance gain for NENIA by manual correction of cell detections in 10% of all frames in a video. SAMTRA is better than random selection (Wilcoxon rank sum test, \( p < 0.005 \)).

<table>
<thead>
<tr>
<th>Video</th>
<th>Frames total</th>
<th>Performance (original)</th>
<th>Performance (random recovery)</th>
<th>Performance (SAMTRA recovery)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ak</td>
<td>300</td>
<td>72.2</td>
<td>77.6</td>
<td>91.0</td>
</tr>
<tr>
<td>hex.6</td>
<td>100</td>
<td>86.6</td>
<td>86.6</td>
<td>93.4</td>
</tr>
<tr>
<td>hex.16</td>
<td>100</td>
<td>67.6</td>
<td>74.1</td>
<td>80.0</td>
</tr>
<tr>
<td>hex.22</td>
<td>100</td>
<td>77.2</td>
<td>77.2</td>
<td>92.4</td>
</tr>
<tr>
<td>square</td>
<td>50</td>
<td>66.4</td>
<td>69.8</td>
<td>92.4</td>
</tr>
</tbody>
</table>

In the next section, we present experiments that determine how much training data is needed for SAMTRA to achieve a certain gain in tracking performance.

6.2.2 Training SAMTRA

In these experiments, the setup is similar to that before. The difference is that we vary the number of training frames from the `hex.22` video. Recall that we train SAMTRA using only one of the videos (`hex.22`), and then use all our videos at different detection levels to evaluate SAMTRA. Furthermore, we use random frame selection as our baseline. Let us define a performance gain as \( G = P_{\text{samtra}} - P_{\text{random}} \), where \( P_{\text{samtra}} \) (respectively, \( P_{\text{random}} \)) are \( P_{\text{track}} \) from Equation 6.4 achieved after correcting frames selected by SAMTRA (respectively, random selection). For different numbers of training frames and different testing videos we observe different gains.

In Table 6.2, we report a weighted average of the gain for a fixed number of training frames across different testing videos. Weights are assigned to videos according to the proportion of the correct tracks in each video compared to the number of correct tracks in all videos.

As expected, a larger number of training frames tends to lead to a better performance. At the same time, in our experiments, we observe a positive gain using as few as 20 frames. Note that correct tracks in our videos almost always
6.3 DISCUSSION AND CONCLUSION

Table 6.2: Number of frames used for SAMTRA training and the corresponding tracking performance gain

<table>
<thead>
<tr>
<th>Number of Frames</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance gain (%)</td>
<td>2</td>
<td>1</td>
<td>13</td>
<td>13</td>
<td>14</td>
</tr>
</tbody>
</table>

span longer than 20 frames, and therefore in order to validate a single track manually one normally needs to review more than 20 frames. This indicates that SAMTRA can be useful in practical scenarios.

### 6.3 Discussion and Conclusion

In this chapter, we propose a semi-automated tracking method called SAMTRA. This method uses logistic linear regression to predict the probability of detection errors in a video. Our evaluation shows that SAMTRA is capable of increasing the proportion of correctly reconstructed tracks on average by 13% through manually correcting detections in only 10% of all frames in a video.

Our implementation of SAMTRA involves training from the tracking results. This approach is independent of the choice of segmentation algorithm. Moreover, it can be possible to avoid training when using a segmentation algorithm that yields the relative level of certainty in its detections as a part of its output. Here the estimation of the probabilities of errors in frames can be based on how certain the segmentation algorithm was in a particular frame. Moreover, we speculate that SAMTRA can facilitate tracking in the first place, by aiming to find a tracking that minimizes the probability of errors in all frames.

Our evaluation suggests that a small number of training frames (e.g., 20) can be sufficient to achieve improved tracking performance. However, the need for training is still a limitation. In the next chapter, we present a novel method for ranking cell trackers without manual validation, that is, without a need for training.
Chapter 7

Ranking Cell Trackers

We now introduce another major contribution of this thesis. We first recall that automated cell tracking has become an important tool in a wide range of biological studies, including anti-cancer drug screening, measuring the proliferation of immune system cells, and conducting wound healing assays [61, 72].

A cell tracking system (CTS) is a combination of algorithms that is capable of both locating cells in video frames and maintaining cell identities throughout the video. The input of the CTS is a video and the output of the CTS is a set of trajectories.

A major challenge in developing an automated CTS is the large diversity of cell morphology and motility as well as varying recording conditions (Figure 5.8). This diversity has resulted in a large number of proposed cell tracking systems, where each system has various parameters that need to be specified [108]. Examples of parameters include the process noise covariance for the Kalman filter used in the method of Li et al. [79], and the weights for association costs in the algorithm by Padfield et al. [97]. For a given video and CTS, an optimal parameter setting is a set of values that results in the most accurate cell tracking.

In order to find an optimal parameter setting for a given video, a natural solution is to annotate a part of the video manually, and then automatically search for parameter values that are optimal for that part. However, due to the variability, each new cell video is likely to have different optimal parameter settings. Furthermore, one can expect the optimal parameter settings to vary
even within a single video. For example, a video can start with a few cells seeded at a low density, but end up with a high density clutter of cells due to proliferation. In this case, the cells are likely to have different dynamics at the beginning and at the end of the video.

As a result, the parameter settings found on the annotated part of a video are unlikely to remain optimal for other videos or even for other parts of the same video. At the same time, annotation of each new video, even in part, is time consuming. Therefore, an important and open research problem is how to select algorithms and their associated parameter settings without access to the ground truth for videos. This is a challenging problem, because it is unclear how to compare the performance of trackers in the absence of reference data.

The problem of automated performance estimation has been previously addressed for medical image segmentation and tracking for surveillance cameras \[110, 124\]. The previous methods rely on knowledge specific to their respective domains and are not directly applicable to CTSs. To the best of our knowledge, we are the first to present a method for ranking CTSs in the absence of a ground truth.

Our method is based on the observation that the results of any CTS can be decomposed into a set of detections (cell centroids) and links (associations between the centroids). Given the results of two CTSs for the same video, we first match measurements reported by the CTSs and produce a common set of measurements. Each of the CTSs then has its own version of the links made between the measurements from the common set. We then use a method for comparing the quality of links (i.e., data association).

Note that we do not estimate the quality of an individual CTS. Instead, we rank CTSs according to their performance. Furthermore, our method requires a user to provide an estimate of the cell size. However, we do not require producing a ground truth that comprises manually located and linked cells. Finally, our method relies on an assumption that the distribution of the lengths of wrong associations can be estimated from the sample of distances between measurements within frames (Section 7.3.2). Our evaluation on real and synthetic videos shows that the assumption holds in practical scenarios.

The main applications for our method are selecting the best CTS for a given
cell video, or tuning the parameters of the chosen CTS. We evaluate our method in the context of cell tracking, because this is the main focus of this thesis. However, our method is sufficiently general to be applicable to other domains, such as particle tracking or vehicle tracking. This is due to the fact that we develop our formulations upon a general points association problem.

The main goal of this thesis is to reduce the amount of manual work while maintaining a high accuracy of cell data analysis. Towards this aim, the contributions of this chapter are as follows: (1) novel measures for ranking data association algorithms according to their performance (Section 7.3); (2) a novel method for ranking CTSs according to their performance without the need for a ground truth (Section 7.4); and (3) evaluation of our methods in several practical scenarios (Section 7.5).

We introduce the related work and the problem formulation in the next two sections.

7.1 Related Work

Over recent decades, there have been many proposed cell tracking methods [86, 108]. Each of these methods has a number of parameter values that need to be specified by the user. Rittscher (2010) remarks: “it is often not clear which particular … tracking algorithm is well suited for the given data type. It would be helpful if the decision on what type of algorithm should be used, or what particular parameter setting should be used, could be made automatically” [108]. We address this challenge by developing a method for ranking trackers in the absence of a ground truth.

More generally, object tracking is a fundamental task with applications in different domains [128]. In the domain of surveillance cameras, the problem of a reference-free tracking performance estimation was addressed by different authors (see the survey of SanMiguel et al. [110] and the references therein). For example, Wu et al. [127] propose an interesting method that estimates the tracking performance by comparing prior and posterior probability densities for object state model. The posterior density is obtained from time reversed
frame sequence after initializing the tracker with the prior density. Unfortunately, their method was developed for single object tracking.

Furthermore, previous work is not directly applicable in the cell tracking context. In the surveillance domain, such methods tend to rely on visual keys such as color and shape, or on directed object movements. In cell videos, motions tend to be more stochastic, and color and shape tend to have less discriminative power compared to objects (e.g., people or cars) in surveillance applications.

Finally, there has been work on automated parameter tuning or quality estimation in other domains. Abdul-Karim et al. [3] suggest to use the minimum description length principle for the automated selection of optimal parameter settings for vessel/neurite segmentation algorithms. Warfield et al. [124] use a modification of the expectation-maximization algorithm in order to automatically estimate a ground truth for medical image segmentation, and simultaneously estimate the performance of given segmentation algorithms. These previous methods focus on image segmentation and cannot be readily applied in the context of data association, which is a core step of tracking.

7.2 Method Overview, Scope and Preliminaries

The roadmap of this chapter is presented in Figure 7.1. Recall that a result of a CTS consists of cell measurements and associations (or links) between the measurements. Given a cell video, results of different CTSs can differ both in the set of measurements and in the set of links.

In this chapter, we divide the problem of ranking into two sub-problems. We first assume that we are given results that all have the same set of measurements, and these results can only differ in the links. Alternatively, this sub-problem can be formulated as ranking data association algorithms on the same input. Recall that a data association algorithm generates a set of links for the given set of measurements. We address this sub-problem in Section 7.3.

In reality, the sets of measurements produced by any two different CTSs are likely to be different. Therefore, in Section 7.4 we propose a method for
Figure 7.1: The roadmap of this chapter. We divide the problem or ranking CTS without ground truth (GT) into two sub-problems. We first develop measures for ranking data association algorithms (Sections 7.3.2 to 7.3.4). We then present a ranking method that is based on pairwise comparisons and uses the developed measures (Section 7.4).
matching and merging a pair of sets of measurements into a common set. Furthermore, in the same section, we propose a method for ranking an arbitrary number of CTSs according to their performance.

As we explained in Section 2.5.3, there are three aspects of the CTS performance: accuracy of outlining cells in frames, detection errors (missed cells, spurious detections), and tracking errors (swapping tracks, losing tracks). In this chapter, our definition of performance (Section 7.2.2) directly addresses the latter aspect, and indirectly accounts for the first two. For example, a “missing detection” error normally implies a “missing link” error. Furthermore, the quality of cell outlines is categorized into accepted and not accepted. Inaccurate outlining, over- and under-segmentation are reflected in the number of detection errors, which, in turn, is reflected in our performance measure.

To summarize, our method of ranking CTSs is based on pair-wise comparisons of the results of the CTSs. Within each comparison, we first match and merge two sets of detected measurements. We then employ a method for ranking data association algorithms.

In the next sections, we formally define the cell tracking process, tracking performance, and the problem that we address in this chapter.

### 7.2.1 Cell Tracking Preliminaries

Given a cell video, a cell segmentation, semi-segmentation or detection algorithm (Chapters 3 and 4) produces a set of measurements that describe identified cells in the video. A measurement is a $K$-dimensional vector $\vec{m} = \{x^{(1)}, \ldots, x^{(K)}\}$, where $x^{(i)} \in \mathbb{R}$, $i = 1, \ldots, K$. For example, the measurement can be a vector comprising the centroid position, average fluorescence and size of a cell. The set of measurements is called a detection. The detection is a set of discovered cell locations in a video. Note that this set can contain errors, such as spurious measurements and missing true cells.

A link is a tuple $\{f, \vec{m}_f, \vec{m}_{f+1}\}$, such that $\vec{m}_f$ is a measurement in frame $f$ and $\vec{m}_{f+1}$ is a measurement in frame $f + 1$. We denote the set of all possible links on detection $\mathbb{D}$ as $\mathbb{L}_{\text{all}}$. If $\vec{m}_f$ and $\vec{m}_{f+1}$ relate to the same cell then the link $\{f, \vec{m}_f, \vec{m}_{f+1}\}$ is called a true link. If a cell divides, the mother and each of the
Figure 7.2: An illustration of the concepts of all possible links, true links, and false links. Links that connect measurements of the same cell are called true links (solid black lines). Links that connect unrelated measurements are called false links (dotted red lines).

daughter cells are considered to be different cells. However, a link connecting the mother and a daughter cell is considered to be a true link. All links that are not true links are called *false links* (Figure 7.2).

$L_{true}$ (respectively $L_{false}$) denotes the set of all true (respectively false) links in $L_{all}$, so that $L_{true} \cup L_{false} = L_{all}$ and $L_{true} \cap L_{false} = \emptyset$. For a given detection $D$, the *ground truth* is a function that maps each link $l \in L_{all}$ to the set $\{true, false\}$.

A *cell tracking system (CTS)* is an algorithm that takes a cell video as input and produces a detection $D$ and a set of links $L$ defined on that detection. A cell tracker (data association algorithm) is a part of the CTS. The *cell tracker* $CT$ is an algorithm that takes a cell detection $D$ as input and produces a set of links as output: $CT(D) \rightarrow L_{trk} \subseteq L_{all}$. Cell tracking can be considered as the process of selecting links $L_{trk}$ from $L_{all}$. To avoid confusing we denote the former concept by the abbreviation CTS, and the latter by cell tracker or tracker.

Some CTSs do not have an isolated cell tracker as a module (Section 2.4). However, the results of any CTS can be decomposed into detection $D$ and a set of links $L$. The set $L$ can then be artificially represented as a result of a tracker.
7. RANKING CELL TRACKERS

CT(D).

By the CTS or the cell tracker, we mean a method with all its parameters set to certain values. For example, a nearest neighbor linking tracker with its gating distance set to 5, and a nearest neighbor linking tracker with its gating distance set to 7 are considered as two different trackers.

7.2.2 Cell Tracking Performance

Previously we introduced several measures for tracking performance (Sections 2.5.3 and 5.2.1). In this chapter, we mainly use a novel F-score based measure, because one of our methods is based on the definitions of precision and recall. At the same time, we also report some of the results with respect to other measures. Furthermore, we note that F-score and the other measures are related (Appendix C.1), and it is not surprising that in our evaluation we find correlation between different measures (Section 7.5).

On the level of inter-frame associations, the accuracy of maintaining cell identities can be measured by precision and recall, commonly used in the field of information retrieval. These measures capture, respectively, the proportion of spurious links made by the tracker and correct links overlooked by the tracker. Precision is defined as \( \text{prec} = \frac{|L_{true} \cap L_{trk}|}{|L_{trk}|} \), and recall is defined as \( \text{recl} = \frac{|L_{true} \cap L_{trk}|}{|L_{true}|} \) (we assume \( |L_{trk}| > 0 \) and \( |L_{true}| > 0 \)). Precision and recall are usually combined into a single measure called the F-score. In fact, there is a family of \( F_\beta \)-scores, and in what follows by \( F \) we denote \( F_1 \)-score.

F-score is the harmonic mean of precision and recall \( F = 2 \frac{\text{prec} \cdot \text{recl}}{\text{prec} + \text{recl}} \). Note that by definition, F-score is bounded in the range \([0, 1]\). An F-score of one corresponds to a perfect tracking without any errors. As the number of spurious and overlooked links increase, that is, as tracking gets less accurate, the F-score decreases. For convenience, when both precision and recall equal to zero we define \( F = 0 \).

\(^1\)Recall is also known as sensitivity.
7.3. RANKING ASSOCIATIONS WITHOUT GROUND TRUTH

7.2.3 CTS Selection Problem

Our main problem can be formulated as follows. Given a set of CTSs for a video, and an estimate of the cell size, our task is to rank the CTSs according to their F-scores while the ground truth is unknown. As we mentioned before, we address this problem by first solving a sub-problem of ranking cell trackers (data association algorithms). The sub-problem can be formulated in a similar fashion. Given detection D, and a set of cell trackers, we aim to rank the trackers according to their F-scores without using a ground truth.

A large part of this chapter (next section) is dedicated to the solution of the sub-problem. We approach the sub-problem by developing reference-free measures (that do not require ground truth) that correlate with F-score. We then use one of these measures to solve the main problem in Section 7.4.

7.3 Ranking Associations without Ground Truth

7.3.1 Observable Information

Our approach for solving the problem of ranking cell trackers (data association algorithms) is to develop a measure that estimates the tracking performance from the information available as a result of tracking. For example, one can know the number of links made by the tracker in every frame. Furthermore, one can know the lengths of the links made by the tracker. Our baseline measure presented in the next section, uses the number of links a tracker makes between consecutive frames.

7.3.1.1 Number of Links

Intuitively, a “good” tracker is one that is consistent in making links across frames. Consider the output L_{trk} of a tracker. Let \( L_i \subseteq L_{trk} \) be the subset of links that end in frame \( i \), that is, \( L_i = \{ l = \{ i - 1, \vec{m}_{i-1}, \vec{m}_i \} : l \in L_{trk} \} \). Now \( n_i = |L_i| \) is the number of links that end in frame \( i \) according to the cell tracker. Our naive measure of tracking performance is called VN-score. It is defined as a
sample variance $VN = \text{Var} \left( \{ n_2, \ldots, n_{N_f} \} \right)$, where $N_f$ is the number of frames in the video. Note that VN-score does not require a ground truth.

Surprisingly, such a simple measure is well correlated with F-score in our real video experiments, but it does not perform well in the synthetic video experiments where tracking conditions are harsher. In fact, VN-score has intrinsic limitations. For example, swapping tracks is a common tracking error in practice. However, after a swap, the number of cells or links in a frame as perceived by the tracker does not change compared to the real number of links. Therefore a swap error is not captured by the VN-score.

Moreover, if a cell enters or leaves the field of view, the number of true links changes. However this legitimate change will be reflected in the VN-score as an error. We further discuss the VN-score in Appendix C.2, and here we proceed with developing a more reliable measure.

7.3.1.2 Lengths of Links

Another kind of information that is available without a ground truth is the lengths of links made by the tracker. For a given link $l = \{ f, \vec{m}_f, \vec{m}_{f+1} \}$, where $\vec{m}_f, \vec{m}_{f+1} \in \mathbb{R}^K$ the length of the link is a function $R(l) : \mathbb{R}^K \times \mathbb{R}^K \rightarrow \mathbb{R}$. For example, if $\vec{m}_f$ and $\vec{m}_{f+1}$ are the cell centroids in two consecutive frames, the length of the link can be the Euclidean distance between $\vec{m}_f$ and $\vec{m}_{f+1}$. Essentially, the length of a link is a distance between two measurements. However, in our method, we treat a pair of measurements (i.e., a link) as a single object. From this perspective it is more convenient to use term “the length of the link” instead of “the distance between the end points of the link”.

In this chapter, we define the link length as the Euclidean distance between centroid locations $R(\{ f, \vec{m}_f, \vec{m}_{f+1} \}) = ||\vec{m}_f - \vec{m}_{f+1}||$. Note that, in general, the definition of the “link length” is not limited to the use of centroid locations only. In principle, the length might be calculated using other available information, such as cell fluorescence and area. Our evaluation shows that satisfactory results can be achieved using the centroids alone, but it would be an interesting direction for future work to consider other types of information.

Note that if the lengths of true and false links come from different distri-
7.3. RANKING ASSOCIATIONS WITHOUT GROUND TRUTH

butions, then given a set of links, we can look at their lengths and make some conclusions about the proportion of true links in the set. We next develop in three steps a measure that correlates with F-score. We first introduce a measure that correlates with precision (Section 7.3.2). We then introduce another measure that correlates with recall (Section 7.3.3). Finally, we combine these two measures into one (Section 7.3.4).

7.3.2 Mirrored Precision

Consider a cell tracker CT, with input \( D \) and output \( L_{trk} \). Let \( E_f(L_{trk}) \) denote the event “a link \( l \) that is randomly chosen from a set \( L_{trk} \) is a false link”. For brevity we denote \( E_f(L_{trk}) \) as \( E_f \). Our main observation is that the posterior probability \( P\{E_f|L_{trk}\} \) is correlated with the proportion of false links in \( L_{trk} \) and hence has negative correlation with the precision \( prec(L_{trk}) \).

Based on this observation, we define our first measure called the mirrored precision (also MP-measure or MP-score) as

\[
MP(L_{trk}) \equiv \frac{P\{E_f|L_{trk}\}}{P\{false\}} = \frac{1}{N} \sum_{i=1}^{N} \frac{P_f(R_i)}{P_{all}(R_i)}.
\] (7.1)

Here the middle and right parts of the equation are related via Bayes’ theorem. \( P_{all}(R_i) \) (respectively, \( P_f(R_i) \equiv P_{all}\{R_i|false\} \)) are the probabilities of observing a link (respectively, false link) with a particular length \( R_i \). That is, \( P_{all} \) (respectively \( P_f \)) is the PDF of the lengths of all (respectively, false) links. We assume that \( P_f \neq P_{all} \). \( P\{false\} \) is the a priori probability that a link is a false link. For a given input \( D \), \( P\{false\} \) is an (unknown) proportion of false links implied by \( D \), that is, a fixed value.

A useful property of the mirrored precision is that it correlates with precision as formalized in the proposition below (a proof sketch is given in Appendix C.3). This property is essential for our work.

---

\(^2\)We define the event \( E_f \) as “... a link ... being a false link”, not a true link, because it is more convenient for us to recover the distribution of lengths of false links, as we show in the next section. To reflect the fact that the correlation is negative we call our measure mirrored precision.
Proposition 7.1. Let \( X = MP(L_{trk}) \) and \( Y = prec(L_{trk}) \) be real-valued random variables (on the sample space \( \Phi \) where an outcome is a tracker). Then if \( P_f \neq P_{all} \), \( \text{Cov}(X, Y) < 0 \), where \( \text{Cov}(\ldots) \) denotes the covariance.

7.3.2.1 Estimation of PDFs of Link Lengths

We estimate the PDFs of the lengths of all links and false links from the input detection. We estimate \( P_{all} \) from a sample of all links that one can make in the given detection. That is, we connect each measurement in a frame with each measurement in the subsequent frame. We estimate the PDF from a sample using the kernel density estimation method from Botev et al. [14].

We also need to estimate \( P_f \), however, without a ground truth, it is not known which links are false links. Therefore, we build upon a previous observation that the distribution of false links can be estimated from the distances between measurements within a frame (Section 5.2.3). Let \( P_{within} \) be the PDF of a random variable that denotes the Euclidean distance between a random pair of measurements within a single frame in the video. We can estimate \( P_{within} \) from a set of pairwise distances between measurements within each frame. We then use the estimated \( P_{within} \) instead of \( P_f \) based on the following assumption.

Assumption 7.2. We assume that \( P_{within} \approx P_f \).

It is important to note that the assumption was empirically validated on a range of real and synthetic cell videos (Section 7.5). Moreover, below we provide a reasoning why we expect the assumption to hold in practical tracking scenarios.

Consider two measurements chosen at random within a single frame, let one of the measurements be randomly chosen as the origin, and let \( dx_{within} \) be a random variable denoting the difference between the \( x \) coordinates of the measurements (the same argument applies to the \( y \) coordinate). Furthermore, for a pair of consecutive frames, let \( dx_{true} \) and \( dx_{false} \) be random variables denoting the differences between the \( x \) coordinates of the measurements in the two frames. Here \( dx_{true} \) corresponds to a true link and \( dx_{false} \) corresponds to a false link. We can express the differences in coordinates for a false link as \( dx_{false} = dx_{true} + dx_{within} \) (Figure 7.3).
7.3. RANKING ASSOCIATIONS WITHOUT GROUND TRUTH

Figure 7.3: The length of a false link can be expressed in terms of the length of the corresponding true link and the distance between locations within a frame.

By construction, the mean of $dx_{\text{within}}$ is 0. We also assume zero means for $dx_{\text{true}}$ (otherwise there is a common offset that can be subtracted). Note that if $\text{Var}(dx_{\text{true}}) \ll \text{Var}(dx_{\text{within}})$ then $dx_{\text{false}} \approx dx_{\text{within}}$ and Assumption 7.2 is satisfied. For example, in our real videos, typical variances satisfy $\text{Var}(dx_{\text{true}}) < 36$ pixels and $\text{Var}(dx_{\text{within}}) > 5100$ pixels.

Furthermore, as $\text{Var}(dx_{\text{true}})$ approaches $\text{Var}(dx_{\text{within}})$, cells tend to travel larger distances between consecutive frames compared to the distances between different cells. This usually results in a degraded tracking performance. In turn, if the maximum achievable tracking performance is low, an inaccurate MP-score is a not critical problem. More generally, in order for cells to be identifiable, one can expect the existence of a property (or combined property) such that the variation in the property for the same cell in two consecutive frames is small compared to the variation across different cells in the same frame. In this case, the assumption holds with respect to this property.

In summary, in order to estimate the distribution of false links, we collect the distances between measurements within individual frames. We then use kernel density estimation to obtain the PDF of these within-frame distances, and use this PDF as an approximation of $P_f$. From our evaluation, we conclude that our estimation method is reliable in practical cell tracking scenarios.

7.3.3 Mirrored Recall

We now have a measure to estimate the precision of tracking, and we need a measure to estimate the recall. We note that under certain circumstances, recall is closely related to precision. For example, when the outputs of different
7. RANKING CELL TRACKERS

Figure 7.4: Equalizing the sizes of tracking results. (a) An output of a cell detection algorithm implies sets of true and false links. (b) Different trackers can produce different results, and different results can have different recall values. (c) Each tracker’s results are appended with false links randomly drawn from the set of false links in the video. The precision of the padded results is correlated to the recall of the original results.

trackers all have the same number of links \( N \), then the precision of such trackers depends only on the number of true links included in their output (true positives). At the same time, recall also depends on the number of true positives. Therefore, if different trackers produce outputs of the same size then precision is proportional to recall.

In practice, different trackers can produce outputs of different sizes. Therefore we equalize the sizes of different outputs, in order to express tracking recall via precision. We append shorter outputs to the size of the largest output \( N_{\text{max}} \) with dummy links with lengths independently randomly drawn from the PDF of lengths of false links \( P_f \) (Figure 7.4). In the rest of this section, we summarize the calculation of mirrored recall using the equalized output sizes. The proposed dummy link generation process is justified in Appendix C.4.

We denote the padded output as \( L'_{\text{trk}} = L_{\text{trk}} \cup L_{\text{pad}} \). Let \( \text{prec}'(\text{CT}(D)) = \frac{T}{N_{\text{max}}} \), that is, \( \text{prec}' \) is the precision of the padded output. Here \( T = |L_{\text{true}} \cap L'_{\text{trk}}| \) is the number of true links in \( L'_{\text{trk}} \), which is the same as the number of true links in \( L_{\text{trk}} \). Note that \( \text{rec}l(\text{CT}(D)) = \frac{T^*}{T} \), where \( T^* = |L_{\text{true}}| \). For a given input \( D \) and set of trackers \( \Phi \), both \( T^* \) and \( N_{\text{max}} \) are fixed. Hence
we have that \( \text{prec}'(\text{CT}(D)) \propto \text{recl}(\text{CT}(D)) \).

We now define the mirrored recall (also MR-measure or MR-score) as

\[
\text{MR} = \text{MP}(\text{L}'_{\text{trk}}),
\]

(7.2)

where \( \text{MP} \) is calculated using Equation 7.1. From Proposition 7.1 it follows that \( \text{MR} = \text{MP}(\text{L}'_{\text{trk}}) \) is correlated with \( \text{prec}'(\text{L}_{\text{trk}}) \), and hence MR-score is negatively correlated with the recall \( \text{recl}(\text{L}_{\text{trk}}) \).

### 7.3.4 Combined Measures for Cell Tracking

In order to produce a ranking of trackers, we need to combine the MP and MR-scores into a single measure. Our approach is based on representing each cell tracker as a point in a two-dimensional Euclidean space. Note that smaller values of MP and MR-scores correspond to higher precision and recall respectively. Therefore, an ideal tracker is expected to have both MP and MR-scores equal to zero. We define our second combined measure \( \text{ED-score} \) as the Euclidean distance to this ideal point

\[
\text{ED} = \sqrt{\text{MP}^2 + \text{MR}^2}.
\]

Since our scores have been motivated by F-score which is the harmonic mean of the precision and recall, the apparent solution would be to use the harmonic mean of MP and MR-scores. We discuss this approach in Appendix C.1, where we conclude that the ED-score is preferred.

An alternative approach to combine the scores is to treat the combination of measures as a dimensionality reduction problem. We apply principal component analysis (PCA) to the MP and MR-scores, and use the first principal component as a combined measure. We denote this measure as \( \text{PC-score} \). Note that PCA does not guarantee optimality in terms of correlation with F-score. For example, MP-score can have a perfect correlation with F-score and have a small variance, while MR-score can be uncorrelated with F-score and have a large variance. In this case, PC-score will be essentially equal to MR-score, which is not the best possible combination.

In our evaluation, we also tested the combination of VN, MP and MR-scores (i.e., three measures). We denote the respective combined scores as \( \text{ED3} \) and
Table 7.1: Summary of tracking performance measures that we introduce in this thesis. None of these measures requires ground truth. Details of our evaluation are given in Section 7.5.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Definition</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP</td>
<td>Equation 7.1</td>
<td>Strong negative correlation with precision</td>
</tr>
<tr>
<td>MR</td>
<td>Equation 7.2</td>
<td>Strong negative correlation with recall</td>
</tr>
<tr>
<td>ED</td>
<td>( \sqrt{MP^2 + MR^2} )</td>
<td>Strong negative correlation with F-score</td>
</tr>
<tr>
<td>PC</td>
<td>The first principal component of ([MP, MR])</td>
<td>Weak negative correlation with F-score</td>
</tr>
<tr>
<td>VN</td>
<td>Variance of the number of links made by the tracker</td>
<td>Strongly correlated in one group of experiments, but fails in another group</td>
</tr>
<tr>
<td>ED3</td>
<td>( \sqrt{MP^2 + MR^2 + VN^2} )</td>
<td>Does not perform well</td>
</tr>
<tr>
<td>PC3</td>
<td>The first principal component of ([MP, MR, VN])</td>
<td>Does not perform well</td>
</tr>
</tbody>
</table>

**PC3-scores.** ED3-score is the quadratic mean of VN, MP, and MR-scores with equal weights, and PC3-score is the first principal component after applying PCA to VN, MP and MR-scores. We summarize all our measures in Table 7.1.

### 7.4 Ranking Cell Tracking Systems

Recall, that our main problem is ranking CTSs without a ground truth. The output of any CTS can be represented as a detection \( D \) and links \( L \). Furthermore, the links can be viewed as a result of a cell tracker \( L = CT(D) \).

For a given video, we need to rank CTSs according to their performance. We address this problem by performing pairwise comparisons of CTSs. For any two CTSs that produce measurements \( D_1 \) and \( D_2 \) and links \( L_1 \) and \( L_2 \), we first match and merge their measurements into a common set \( D' \). We also augment their sets of links into \( L'_1 \) and \( L'_2 \). Note that \( L'_1 \) and \( L'_2 \) are defined on the same set \( D' \). This brings us to a sub-problem that we solved in the previous section. We then use one of the measures for ranking cell trackers in order to choose...
between $\mathbb{L}_1'$ and $\mathbb{L}_2'$. Our method is summarized in Algorithm 7.1. We first test each CTS against some internal consistency criteria, and give a low rank to CTSs that do not pass the test, as such CTSs are likely to be of poor quality. For the CTSs that pass the test, we perform pairwise comparisons based on the ED-scores introduced earlier. We count the number of wins in such comparisons and rank the CTSs according to the number of wins. We use ED-score because our evaluation (Section 7.5) suggests that this is the most reliable measure among the measures introduced in Section 7.3.4.

Algorithm 7.1 A method for ranking CTSs

Input: $S = \{CTS_1, \ldots, CTS_s\}$; $par = \{3$ parameters, see text$\}$; 
Output: $S' = S$, ordered according to the performance from high to low;

1. $S' = \emptyset$;
2. for each $CTS_i \in S$
3. \hspace{1em} if not $FaceValidity(CTS_i, par)$
4. \hspace{2em} $S = S \setminus CTS_i$;
5. \hspace{2em} $S' = S' \cup CTS_i$;
6. \hspace{1em} end
7. end
8. $\{\#wins_i\} = PairwiseComparisons(S, par)$;
9. $S = \text{order according to decreasing } \{\#wins_i\}$;
10. return $\{S, S'\}$

7.4.1 Implementation Details

Our implementation of $PairwiseComparisons(S, par)$ is illustrated in Figure 7.5. Given a pair of arbitrary CTSs, we first extract the corresponding detections $\mathbb{D}_1$ and $\mathbb{D}_2$. We then match the detections using a modification of the Hungarian algorithm for rectangular assignments (see Appendix A.3.2). In the assignment, we prohibit matching measurements that are further away from each other then
Figure 7.5: An outline of the pairwise comparison procedure. Cells are represented with circles for convenience only. Cells need not to be round in videos.

the estimate of the cell size \( d_{\text{max}} \). We combine all matched and unmatched measurements into a single synthetic detection \( D' \). In the synthetic detection \( D' \), a pair of matched measurements is represented by their mean. Here we assume that the matched measurements represent the same cell, and we want this cell to have only one centroid in the synthetic detection. Therefore, we replace the matching measurements in the two CTSs with the corresponding mean measurements. As a result, we have two sets of links \( L_1' \) and \( L_2' \) that are defined on the common detection \( D' \). We can now calculate the ED-score for each set. The CTS that results in a smaller ED-score wins (due to the negative correlation between the ED-score and the F-score).

Our baseline solution is to select a CTS according to the variance in the number of links. We implement this solution using the same Algorithm 7.1, but replacing the ED-score in \( \text{PairwiseComparisons}(S, \text{par}) \) with the variance (VN-score from Section 7.3.1).

FaceValidity(CTS\(_i\)) is a function that estimates whether a given CTS is reasonably good for the given video. In our implementation, we employ two observations. First, for a given CTS\(_i\), we derive a detection \( D_i \), and estimate the PDF of the within-frame distances \( P_{\text{within}} \) and the PDF of all links \( P_{\text{all}} \). According to Assumption 7.2, we expect \( P_{\text{within}} \approx P_f \). Therefore, we compare \( P_{\text{within}} \) and \( P_{\text{all}} \) using the Kolmogorov-Smirnov test (KST). Failing to reject the null hypothesis that \( P_{\text{within}} \approx P_f = P_{\text{all}} \) at significance level \( \alpha \), indicates that there are no true links that can be made in \( D_i \) (recall that \( P_{\text{all}} \) is a linear combination of the PDFs of true and false links). In this case, CTS\(_i\) fails our face validity check.

Second, we note that the results of certain CTSs are poor due to a large num-

174
number of spurious locations. In this case, the density of such locations can be abnormally high. Therefore, in each frame, for each location we find its nearest neighbor and compare the distance to the nearest neighbor $d_{NN}$ with our estimate of the cell size $d_{max}$. Relation ($d_{NN} < d_{max}$) indicates that the CTS reports two distinct cell locations that are suspiciously close to each other. We then look at the proportion of such suspicious locations among all locations in $D_i$. If the proportion $\beta$ is high, then $CTS_i$ fails our face validity check.

In total, our method has 3 parameters ($d_{max}$, $\alpha$, $\beta$). Here, $d_{max}$ is an estimate of the cell size in the video, $\alpha$ is the significance level for the KST, and $\beta$ is the proportion of the suspicious locations that we can tolerate. Note that $\alpha$ and $\beta$ are independent of the video. We set $\alpha = 0.1$ and $\beta = 0.5$. In our evaluation, we find our results are not very sensitive to variations of $\alpha$ and $\beta$.

7.5 Evaluation

Our method for ranking CTSs without a ground truth is based on pairwise comparisons. An essential part of the comparisons is a measure for ranking cell trackers. Such a measure must be correlated with F-score, but does not require a ground truth for computation. We proposed several such measures in Section 7.3.4.

In this section, we start with a validation of the proposed measures in practical scenarios, and with a study of the reliability of the measures under conditions that are expected to hamper our method. To this end, we use different cell trackers previously proposed in the literature, and vary the parameter values used in these trackers (Section 7.5.1). Furthermore, we use a range of real and synthetic videos as inputs (Section 7.5.2). Finally, we evaluate our method as a whole, in a scenario that involves ranking CTSs for different real videos (Section 7.5.4).
7.5.1 Real videos

7.5.1.1 Cell videos

In our evaluation, we use five real videos. These videos have been previously described in Section 5.3.1, and we only briefly summarize them here (Table 7.2). One of the videos (named ak[4]) shows neural progenitor cells, and the remaining four videos show B lymphocyte cells. The videos show cells at different densities and speeds. The difficulty of tracking cells in a video may depend on a variety of factors, such as density and relative sizes of cells. However, to our knowledge, there is no standard way to formalize the “difficulty”. As we mentioned in Section 5.3.3, an insight into the “difficulty” can be obtained using the concept of a normalized cell density \( \beta_n \) [88]. It describes how fast cells move compared to the distance between cells \( \beta_n = N \cdot \sigma^2 \cdot \pi / L^2 \). Here \( N \) is the number of cells in a frame, \( \sigma^2 \) is the variance of cell offsets between consecutive frames, and \( L \) is the side of a square frame. Higher values of \( \beta_n \) correspond to faster motions and more difficult videos. Our videos contain a few cell divisions and deaths, so the number of cells changes across frames. We use the average number of cells per frame to estimate \( \beta_n \). Furthermore, in some of our videos, cells tend to group in clusters in the middle of the frame, so we estimate \( L \) as the side of the smallest square that outlines all cells in a frame. This square can be smaller than the actual frame.

For each video, we manually produce a ground truth cell detection. We then add random noise to the ground truth, in order to simulate different levels of detection quality. We generated detection levels 0, 1, 3, 5, 10 and 20, where level \( k \) means that there are \( k\% \) of spurious measurements (false positives) and \( k\% \) of missing true cells (false negatives). Therefore, in total we have 5 real videos each at 6 detection levels, and hence 30 inputs. We denote an input as movie@\( k \). For example, hex.16@3 denotes the detection for the hex.16 video that contains 3% of false positives and 3% of false negatives.

---

3This video is an excerpt from supplementary movie 1 for the work of Al-Kofahi et al. [6]. The supplementary file is available on-line from the Cell Cycle journal website at http://www.landesbioscience.com/journals/cc/supplement/alkofahi.zip.
7.5. EVALUATION

Table 7.2: Real videos cover a variety of tracking conditions. #cells is the number of distinct cells. For example, if a mother cell divides into two daughter cells, then there are three distinct cells. $\beta_n$ is the normalized object density defined in the main text. To obtain values of $\beta_n$, divide the numbers in the table by $10^2$.

<table>
<thead>
<tr>
<th>video</th>
<th>#frames</th>
<th>#cells</th>
<th>$\beta_n \times 10^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ak</td>
<td>300</td>
<td>18</td>
<td>0.25</td>
</tr>
<tr>
<td>hex.6</td>
<td>100</td>
<td>6</td>
<td>5.94</td>
</tr>
<tr>
<td>hex.16</td>
<td>100</td>
<td>16</td>
<td>3.45</td>
</tr>
<tr>
<td>hex.22</td>
<td>100</td>
<td>22</td>
<td>3.18</td>
</tr>
<tr>
<td>square</td>
<td>50</td>
<td>35</td>
<td>0.14</td>
</tr>
</tbody>
</table>

7.5.1.2 Cell Trackers

We use three previously proposed cell tracking methods with different parameter settings (Table 7.3). We asked the authors of the corresponding methods which parameters, in their opinion, are most influential for tracking. Based on their responses we select up to three parameters. The authors of the algorithms did not suggest any specific values for the parameters, but, where appropriate, pointed out a range of reasonable values for each parameter. We then set arbitrary values for the selected parameters such that they fall into the appropriate ranges. For example, the probability should be in the range $[0, 1]$, and the maximum allowed spatial distance should be comparable with the frame dimensions. In total, different combinations of the tracking methods and selected parameter values give us $5 + 27 + 27 = 59$ different trackers (see Table 7.3).

The NENIA (version A) tracker has only one parameter. LJIPDA (hybrid) and u-track\(^4\) have more than three parameters. We set the remaining parameters to their default values as provided by the authors of the corresponding tracking methods.

LJIPDA requires a supplementary module to resolve cell divisions. We did not implement this module in our evaluation. As a result, when division occurs LJIPDA can only follow up to one of the daughter cells, and hence LJIPDA always misses the link to the second daughter. Consequently LJIPDA tends to

\(^{4}\)u-track is available online at \texttt{http://lccb.hms.harvard.edu/software.html}. Strictly speaking, this is a multiple particle tracker. However, it is often used and cited in the cell tracking domain. For convenience of narration we refer to this tracker as a cell tracker in our thesis.
Table 7.3: Real trackers and their parameters

<table>
<thead>
<tr>
<th>Method</th>
<th>Authors</th>
<th>Parameters</th>
<th>#trackers</th>
</tr>
</thead>
<tbody>
<tr>
<td>NENIA (version A)</td>
<td>Kan et al. (Chapter 5)</td>
<td>gating distance = {5, 10, 20, 50, 100}</td>
<td>5</td>
</tr>
<tr>
<td>LJIPDA (hybrid)</td>
<td>Musicki and Evans [91]; Chakravorty [unpublished]</td>
<td>process covariance = {0.1, 1, 10} detection probability = {0.7, 0.8, 0.9} termination probability = {0.1, 0.05, 0.01}</td>
<td>(3 \times 3 \times 3 = 27)</td>
</tr>
<tr>
<td>u-track</td>
<td>Jaqaman et al. [67]</td>
<td>maximum search radius = {5, 20, 100} time window = {1, 5, 10} st. dev. multiplication = {2, 3, 4}</td>
<td>(3 \times 3 \times 3 = 27)</td>
</tr>
</tbody>
</table>
perform slightly worse than the other algorithms. This is not important for our evaluation, since our aim is to rank different trackers.

7.5.2 Synthetic videos

The goal of our experiments on synthetic videos is to cover tracking conditions that were not covered in our real videos. Such conditions include (i) large numbers of cells, (ii) abrupt movements and frequent divisions, and (iii) changing cell density.

We implement two cell density changing models in two groups of synthetic videos. In the first group, the number of cells increases due to divisions, but all cells are constrained within fixed frame boundaries. In the second group, cells are not constrained and the number of cells does not change, however, cells have a directed motion outwards from the middle of the frame. This kind of motion can be observed in wound healing assays \[72\]. In a wound healing assay cells usually move towards the wound (inwards rather than outwards), but this does not make a difference for our evaluation. More details on our synthetic videos are given in the next section.

We do not test using videos with high numbers of frames, because in general a long video can be divided into parts. Some tracking algorithms are global in the sense that they can achieve higher performance given the whole video at once. Such trackers can be given the complete video as input. We then can select a part or a few parts of the video and estimate the performance of the tracker on each part.

For each synthetic video, we added random noise to the detection, such that each detection contains 3% of false positives and 3% of false negatives. In total, we have 20 synthetic inputs: 10 synthetic videos in each of the two groups described below, and one detection level for all videos.

7.5.2.1 Group One Synthetic (G1)

We generate two groups of synthetic videos. Each group comprises 10 synthetic videos. Every video has 10 frames, and starts with 100 cells uniformly placed in the first frame.
In the first group (G1), cells are moved according to a Brownian motion model. The variance $\sigma^2$ of the offset in both spatial directions is the same and takes different values in different videos in such a way that $\beta_n$ takes values from 0.1 to 1 with a step of 0.1 (in our real videos, the maximum estimated $\beta_n$ is around 0.06, i.e., the motion is much slower). By definition $\beta_n = N \cdot \sigma^2 \cdot \pi / L^2$, where we use the number of cells $N = 100$, and the side of the square frame in pixels $L = 100$.

In our G1 videos, the motion of cells is constrained within the frame boundaries. If a generated inter-frame offset for a cell leads outside the frame, we move the cell towards the direction of the offset, but leave it within the frame at some random distance from the boundary. A visual inspection of the produced videos shows that this motion amendment does not lead to dramatic skews in the spatial distribution of cells. The placement of cells remains approximately uniform.

Every odd frame, 20% of the cells divide, and so the second frame has 120 cells, the fourth frame has 144 cells, and so on. The last 10-th frame contains 200 cells. We do not recalculate the offset variance $\sigma^2$. This implies that $\beta_n$ increases further (i.e., tracking becomes harder) throughout the video.

7.5.2.2 Group Two Synthetic (G2)

In the second group (G2) of synthetic videos, there are no divisions and the number of cells is 100 in each frame. After placing the cells in the first frame, we draw an imaginary vertical line that divides the frame into halves. We then move cells to the left of the line to the left with some constant offset plus random noise, while we move cells to the right of the line to the right with the same constant offset and noise.

We characterize the constant offset in relation to the expected distance to the nearest cell denoted as $D$. It is not convenient to measure the offset in relation to the frame side $L$ because such a measurement does not take into account the initial density of cells. Different videos in the second group have different offsets $\alpha D$, where $\alpha$ takes values from 0.1 to 0.55 with a step of 0.05. We estimate $D$ as $D = L / \sqrt{\pi \cdot N}$. The intuition is that if $D$ is the distance to the nearest cell
then there is only one cell expected to be in a circle with radius $D$. On the other hand, there are $N$ cells in a square frame with side $L$, and hence $D = L / \sqrt{\pi \cdot N}$. This is only a coarse estimation of $D$, but it suffices for the purposes of our evaluation.

Note that $L$ denotes the side of the first frame where we place cells uniformly. In the subsequent frames, cells move apart and the effective frame size grows (the field of view is not restricted). In our simulations, we need control over $D$ and hence it is slightly more convenient to move cells outwards.

### 7.5.3 Summary of Results

This section summarizes our findings on the usefulness and reliability of our reference-free measures for cell tracking (data association algorithms). Recall, that we perform these evaluations on three groups of videos: real, synthetic G1 and synthetic G2.

In all three groups, each experiment has the same structure as follows. We run a set of trackers on the same input. For each tracker, we compute the F-score using the ground truth, and we compute the ED-score without the ground truth. We then select an optimal tracker based on the computed ED-scores. As our baselines we use the performance of the tracker selected using the VN-score. Another baseline is the F-score of a randomly selected tracker. Finally, in each experiment, we also compute Spearman’s correlation between the F-score and the ED-score (or other proposed measures). By the correlation being “strong” we mean that $|r| > 0.5$.

Furthermore, we compare the proposed performance measures against the random selection strategy using the Wilcoxon signed-rank test. We say that a measure is not statistically different from random selection if the p-value of the test is above 0.05.

### 7.5.3.1 Measures that Require Ground Truth

Performance measures that require ground truth are summarized in Table 7.4. In our evaluation, we find that F-score is strongly correlated with the proportion of swap errors, and with the proportion of lost tracks. On average, over all our
Table 7.4: Performance measures that require ground truth. In our evaluation, F-score is strongly correlated with three other measures.

<table>
<thead>
<tr>
<th>F-score</th>
<th>quadratic mean</th>
<th>prop. swap errors</th>
<th>prop. lost tracks</th>
</tr>
</thead>
<tbody>
<tr>
<td>(harm. mean)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\frac{2 \cdot \text{prec} \cdot \text{recl}}{\text{prec} + \text{recl}}$</td>
<td>$\sqrt{\frac{\text{prec}^2 + \text{recl}^2}{2}}$</td>
<td>$\frac{# \text{swaps}}{# \text{locations}}$</td>
<td>$\frac{# \text{lost tracks}}{# \text{all tracks}}$</td>
</tr>
</tbody>
</table>

In real and G1 experiments, correlation between the F-score and the proportion of swap errors (respectively lost tracks) is $-0.89$ (respectively $-0.62$), see also Table 7.5. Furthermore, we note that F-score is the harmonic mean of precision and recall, whereas our ED-score is the quadratic mean for mirrored precision and recall. We therefore also look at the correlation between the harmonic and quadratic means for precision and recall. In every real and synthetic experiment we find a strong correlation of approximately 0.99 between the F-score and the quadratic mean of precision and recall. Therefore, in what follows we mostly report results with respect to F-score.

7.5.3.2 F-score and ED-score

Figure 7.6 shows the tracking results measured with F-score against ED-score in a representative experiment on real videos. This and other experiments (not shown) confirm that different parameter settings can lead to significantly different performance. Furthermore, on different inputs the best performance can be achieved with different trackers. There is not necessarily a “winning” cell tracker – hence the importance of having a method to rank CTSs without a ground truth.

In general ED-score is strongly correlated with F-score. As ED-score is defined in terms of mirrored precision and recall, we also tested the relations between precision, recall and the mirrored measures. We find that in the real and G1 experiments MR-score is well correlated with recall. The weakest (respectively average) correlation coefficient for the real experiments is $-0.77$ ($-0.95$), and for G1 experiments it is $-0.65$ ($-0.83$). MP-score and precision are also well correlated, except for a few experiments. This correlation is weak only when the variance of precision is small. In all experiments, where the correlation coeffi-
7.5. EVALUATION

Figure 7.6: Experiment with the real video *hex.16* at detection level 3. There are in total 59 points in three plots. Each point is the result of a run of a tracker with its parameters fixed to certain values.

cient between mirrored precision (MP-score) and precision is weaker than $-0.3$, the standard deviation of precision is below 0.01. In such experiments, precision does not affect the variation in tracking performance, and therefore the overall correlation between ED and F-scores is strong, as we quantitatively confirm in the next section.

7.5.3.3 Real and G1 Experiments

The results of our experiments on real videos (up to detection level $k = 5$) are presented in Table 7.5. Overall, ED-score is strongly correlated with F-score. The correlation with F-score is negative because ED-score estimates the number of mistakes in the tracking results. Table 7.5 also shows the tracking performance achieved using random selection, VN and ED-scores. Both VN and ED-scores outperform random selection ($p < 0.005$).

Generally, ED-score is also strongly correlated with F-score in our real videos at detections with 10% and 20% of errors. Apart from video $ak$, the weakest observed correlation is $-0.92$ (data not shown). The correlation for $ak@10$ is
Table 7.5: The results of our real experiments (in rows). The left columns show correlation over all 59 trackers between ED-score and the proportion of swap errors (column SWP), the proportion of lost tracks (TRK), and F-score (FSC). The right columns show the F-score achieved using random selection (RND), VN-measure (VN), ED-measure (ED), and a ground truth (MAX). Both VN and ED measures outperform random selection ($p < 0.005$).

<table>
<thead>
<tr>
<th>Input</th>
<th>SWP</th>
<th>TRK</th>
<th>FSC</th>
<th>RND</th>
<th>VN</th>
<th>ED</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>ak@0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... 1</td>
<td>0.950</td>
<td>0.933</td>
<td>-0.933</td>
<td>0.984</td>
<td>0.993</td>
<td>0.993</td>
<td>1.000</td>
</tr>
<tr>
<td>... 3</td>
<td>0.787</td>
<td>0.759</td>
<td>-0.873</td>
<td>0.993</td>
<td>0.993</td>
<td>0.993</td>
<td>0.995</td>
</tr>
<tr>
<td>... 5</td>
<td>0.735</td>
<td>0.482</td>
<td>-0.807</td>
<td>0.976</td>
<td>0.990</td>
<td>0.990</td>
<td>0.990</td>
</tr>
<tr>
<td>hex.6@0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... 1</td>
<td>0.811</td>
<td>0.701</td>
<td>-0.758</td>
<td>0.817</td>
<td>1.000</td>
<td>0.987</td>
<td>1.000</td>
</tr>
<tr>
<td>... 3</td>
<td>0.884</td>
<td>0.825</td>
<td>-0.858</td>
<td>0.756</td>
<td>0.982</td>
<td>0.987</td>
<td>0.998</td>
</tr>
<tr>
<td>... 5</td>
<td>0.846</td>
<td>0.790</td>
<td>-0.868</td>
<td>0.973</td>
<td>0.967</td>
<td>0.978</td>
<td>0.985</td>
</tr>
<tr>
<td>hex.16@0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... 1</td>
<td>0.925</td>
<td>0.875</td>
<td>-0.925</td>
<td>0.693</td>
<td>1.000</td>
<td>0.970</td>
<td>1.000</td>
</tr>
<tr>
<td>... 3</td>
<td>0.924</td>
<td>0.858</td>
<td>-0.928</td>
<td>0.674</td>
<td>0.989</td>
<td>0.963</td>
<td>0.990</td>
</tr>
<tr>
<td>... 5</td>
<td>0.864</td>
<td>0.660</td>
<td>-0.886</td>
<td>0.950</td>
<td>0.950</td>
<td>0.949</td>
<td>0.975</td>
</tr>
<tr>
<td>hex.22@0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... 1</td>
<td>0.785</td>
<td>0.712</td>
<td>-0.794</td>
<td>0.979</td>
<td>0.979</td>
<td>0.968</td>
<td>0.990</td>
</tr>
<tr>
<td>... 3</td>
<td>0.790</td>
<td>0.743</td>
<td>-0.802</td>
<td>0.972</td>
<td>0.978</td>
<td>0.924</td>
<td>0.978</td>
</tr>
<tr>
<td>... 5</td>
<td>0.916</td>
<td>0.811</td>
<td>-0.969</td>
<td>0.773</td>
<td>0.848</td>
<td>0.936</td>
<td>0.937</td>
</tr>
<tr>
<td>square@0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... 1</td>
<td>0.797</td>
<td>0.760</td>
<td>-0.801</td>
<td>0.979</td>
<td>0.999</td>
<td>0.979</td>
<td>0.999</td>
</tr>
<tr>
<td>... 3</td>
<td>0.813</td>
<td>0.743</td>
<td>-0.840</td>
<td>0.968</td>
<td>0.987</td>
<td>0.956</td>
<td>0.994</td>
</tr>
<tr>
<td>... 5</td>
<td>0.844</td>
<td>0.833</td>
<td>-0.878</td>
<td>0.832</td>
<td>0.969</td>
<td>0.941</td>
<td>0.977</td>
</tr>
<tr>
<td>min</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>max</td>
<td>0.735</td>
<td>0.482</td>
<td>-0.969</td>
<td>0.674</td>
<td>0.848</td>
<td>0.924</td>
<td>0.937</td>
</tr>
<tr>
<td>mean</td>
<td>0.950</td>
<td>0.933</td>
<td>-0.758</td>
<td>0.993</td>
<td>1.000</td>
<td>0.993</td>
<td>1.000</td>
</tr>
<tr>
<td>st.dev.</td>
<td>0.845</td>
<td>0.737</td>
<td>-0.873</td>
<td>0.891</td>
<td>0.968</td>
<td>0.964</td>
<td>0.983</td>
</tr>
<tr>
<td></td>
<td>0.059</td>
<td>0.116</td>
<td>0.058</td>
<td>0.112</td>
<td>0.037</td>
<td>0.024</td>
<td>0.018</td>
</tr>
</tbody>
</table>
Table 7.6: The results of the G1 experiments (in rows). Columns show the F-score achieved using random selection (RND), VN, ED, ED3, PC, and PC3-measures, and a ground truth (MAX). ED-score outperforms random selection ($p < 0.01$). However, the performance achieved by VN, ED3, PC, and PC3-scores is not statistically different from the performance achieved using random selection.

<table>
<thead>
<tr>
<th>$\beta_n$</th>
<th>RND</th>
<th>VN</th>
<th>ED</th>
<th>ED3</th>
<th>PC</th>
<th>PC3</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.605</td>
<td>0.689</td>
<td>0.715</td>
<td>0.689</td>
<td>0.689</td>
<td>0.689</td>
<td>0.721</td>
</tr>
<tr>
<td>0.2</td>
<td>0.651</td>
<td>0.596</td>
<td>0.619</td>
<td>0.596</td>
<td>0.600</td>
<td>0.596</td>
<td>0.654</td>
</tr>
<tr>
<td>0.3</td>
<td>0.447</td>
<td>0.498</td>
<td>0.520</td>
<td>0.498</td>
<td>0.492</td>
<td>0.498</td>
<td>0.545</td>
</tr>
<tr>
<td>0.4</td>
<td>0.419</td>
<td>0.406</td>
<td>0.434</td>
<td>0.406</td>
<td>0.405</td>
<td>0.406</td>
<td>0.481</td>
</tr>
<tr>
<td>0.5</td>
<td>0.382</td>
<td>0.371</td>
<td>0.393</td>
<td>0.371</td>
<td>0.371</td>
<td>0.371</td>
<td>0.438</td>
</tr>
<tr>
<td>0.6</td>
<td>0.317</td>
<td>0.338</td>
<td>0.379</td>
<td>0.338</td>
<td>0.320</td>
<td>0.338</td>
<td>0.412</td>
</tr>
<tr>
<td>0.7</td>
<td>0.276</td>
<td>0.306</td>
<td>0.368</td>
<td>0.306</td>
<td>0.353</td>
<td>0.306</td>
<td>0.373</td>
</tr>
<tr>
<td>0.8</td>
<td>0.298</td>
<td>0.290</td>
<td>0.355</td>
<td>0.290</td>
<td>0.328</td>
<td>0.290</td>
<td>0.355</td>
</tr>
<tr>
<td>0.9</td>
<td>0.277</td>
<td>0.280</td>
<td>0.302</td>
<td>0.280</td>
<td>0.302</td>
<td>0.280</td>
<td>0.322</td>
</tr>
<tr>
<td>1.0</td>
<td>0.247</td>
<td>0.240</td>
<td>0.313</td>
<td>0.240</td>
<td>0.273</td>
<td>0.240</td>
<td>0.313</td>
</tr>
<tr>
<td>min</td>
<td>0.247</td>
<td>0.240</td>
<td>0.302</td>
<td>0.240</td>
<td>0.273</td>
<td>0.240</td>
<td>0.313</td>
</tr>
<tr>
<td>max</td>
<td>0.651</td>
<td>0.689</td>
<td>0.715</td>
<td>0.689</td>
<td>0.689</td>
<td>0.689</td>
<td>0.721</td>
</tr>
<tr>
<td>mean</td>
<td>0.392</td>
<td>0.401</td>
<td>0.440</td>
<td>0.401</td>
<td>0.413</td>
<td>0.401</td>
<td>0.462</td>
</tr>
<tr>
<td>st.dev.</td>
<td>0.141</td>
<td>0.148</td>
<td>0.136</td>
<td>0.148</td>
<td>0.138</td>
<td>0.148</td>
<td>0.140</td>
</tr>
</tbody>
</table>

$-0.73$ and the correlation for $ak@20$ is $-0.39$. It turns out that in the case of $ak@20$ input, many parameter settings result in similar performance (F-score is approximately 0.95) and therefore the correlation is moderate.

In our G1 videos, ED-score is also strongly correlated with F-score. The weakest correlation observed across 10 G1 experiments is $-0.62$, and the mean value for the correlation coefficient is $-0.79$ (data not shown). ED-score also outperforms random selection in our G1 experiments ($p < 0.01$, see Table 7.6). However, in the G1 experiments, the performance achieved by VN-score is not statistically different from the performance achieved by random selection (Table 7.6). This can be explained as follows. We note that in our real videos trackers tend to have high precision, and hence the F-score is effectively determined by recall only. In this scenario, the VN-score performs well (see Appendix C.2). In the G1 videos, both precision and recall vary, and the VN-score does not estimate the performance adequately.
In the G1 videos, the performance achieved by the ED3, PC, and PC3-scores is not statistically different from the performance achieved by random selection (Table 7.6). The poor performance of the VN-score in the G1 videos contributes to the poor performance of the combined ED3 and PC3-scores. Furthermore, the G1 experiments illustrate that the first principal component is not necessarily strongly correlated with F-score.

A suggested application for the ED-score is tuning parameters of a given tracking software implementation. To test the behavior of the ED-score in this scenario, we also run separate evaluations for NENIA, LJIPDA and u-track. We use our real and G1 videos (30 inputs in total). For each input and for each of the three tracking modules, we use (i) a randomly selected parameter set from Table 7.3 and (ii) a parameter set suggested by ED-score. For each input, this results in six results grouped in three paired samples. We find that in each of the three groups, the median achieved performance is higher when using ED-score ($p < 0.001$ for LJIPDA and u-track, $p \approx 0.15$ for NENIA).

7.5.3.4 Stress Test with G2 Experiments

The ED-score relies on the assumption that the distribution of false links can be estimated from the distances between measurements in individual frames (Assumption 7.2). Our evaluation on G1 videos reveals that when cells are uniformly distributed within a frame, the assumption holds even when cells move fast, and when there is an intensive rate of division.

The G2 videos allow us to perform a stress test for another scenario when groups of cells move apart. The results of the G2 experiments are presented in Table 7.7. An increased cell motion affects Assumption 7.2. As a result, the estimation of the length distribution of false links becomes poorer (as measured by the p-value of the Kolmogorov-Smirnov test for the real and estimated distributions). Accordingly, ED-score becomes less correlated with F-score. At the same time, as the motion becomes faster, the maximum F-score achieved among our real trackers decreases. In other words, on videos with such fast motion one is unlikely to achieve reasonable tracking accuracy anyway.
Table 7.7: The results of our evaluation on the G2 videos. “alpha” characterizes the relative magnitudes of the directed motion components of cells, “r” is the correlation between ED and F-scores, “p-val” is the p-value of the Kolmogorov-Smirnov test for the real and estimated distributions of false links, and “Max” is the maximum F-score that we observe among our real trackers.

<table>
<thead>
<tr>
<th>alpha</th>
<th>p-val</th>
<th>r</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>0.990</td>
<td>-0.820</td>
<td>0.915</td>
</tr>
<tr>
<td>0.15</td>
<td>0.999</td>
<td>-0.757</td>
<td>0.905</td>
</tr>
<tr>
<td>0.20</td>
<td>0.973</td>
<td>-0.719</td>
<td>0.899</td>
</tr>
<tr>
<td>0.25</td>
<td>0.946</td>
<td>-0.700</td>
<td>0.900</td>
</tr>
<tr>
<td>0.30</td>
<td>0.949</td>
<td>-0.789</td>
<td>0.894</td>
</tr>
<tr>
<td>0.35</td>
<td>0.786</td>
<td>-0.772</td>
<td>0.892</td>
</tr>
<tr>
<td>0.40</td>
<td>0.536</td>
<td>-0.731</td>
<td>0.883</td>
</tr>
<tr>
<td>0.45</td>
<td>0.264</td>
<td>-0.568</td>
<td>0.874</td>
</tr>
<tr>
<td>0.50</td>
<td>0.119</td>
<td>-0.422</td>
<td>0.872</td>
</tr>
<tr>
<td>0.55</td>
<td>0.033</td>
<td>0.123</td>
<td>0.858</td>
</tr>
</tbody>
</table>

7.5.4 Validation of the CTS Ranking Method

The previous sections show that ED-score is a reliable measure that can be used to rank cell trackers. Therefore, we use this measure to rank CTSs, and we use the VN-score as one of the baselines. In this section, we evaluate our method for ranking CTSs.

We validate the utility of our method by considering the following practical situation. We are given a number of CTSs, constructed from algorithms previously reported in the literature and whose implementations we already have. We set most of the algorithm parameters to their default values, set some parameters arbitrarily, and some parameters based on our knowledge of video frame sizes. We construct 10 fully parametrized CTSs and we also use a random CTS that produces results irrelevant to the input video (Table 7.8).

On the other hand, we are given a set of real videos that we want to process, and we want to identify the top performing CTSs for each video, without having a ground truth. We look at two frames in each video, obtain estimates of the cell sizes, and run our method (the same set of CTSs for each input).

As our inputs, we use the 5 real videos (Section 7.5.1). From every video, we extract two sequences: with 10 and 15 frames. We use short sequences in
To establish whether a small number of frames is sufficient to rank the CTSs. If a video is long, it is possible to break it into shorter sequences and apply our method for each sequence independently. It may happen that different parts of the video require different CTSs. Our sequences are taken starting from arbitrary video frames, such that they include cell events if possible. Our sequences include events and conditions such as cell divisions, entering and leaving the field of view, high cell density, and abrupt movements.

With each video sequence as input, we run our 11 CTSs, and rank the CTSs using our method. For the purposes of evaluation, we also record the F-scores for each CTS using a ground truth. The results are presented in Table 7.9. First, note that for every input there is a large variation in the F-scores. Therefore, a proper selection of the CTS is important. Second, we find that in almost every

Table 7.8: Eleven CTSs used in our evaluation. The CTSs are constructed from previously reported algorithms: Sobel edge detection, Hough transform, Otsu thresholding, Gaussian filtering, cell tracker NENIA (Chapter 5), particle tracker u-track [67].

<table>
<thead>
<tr>
<th>CTS name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hough – 15 – nenia – 20, hough – 20 – nenia – 50</td>
<td>Sobel edge detection; Hough transform with the accumulator array threshold set to 15 and 20; NENIA tracker with the gating distance set to 20 and 50</td>
</tr>
<tr>
<td>otsu – nenia – 20, otsu – nenia – 50</td>
<td>Otsu thresholding; NENIA tracker with the gating distance set to 20 and 50</td>
</tr>
<tr>
<td>gauss – 0.22 – nenia – 20, gauss – 0.22 – nenia – 50, gauss – 0.25 – nenia – 20, gauss – 0.25 – nenia – 50</td>
<td>Gaussian filtering; locating local maxima with the fluorescence threshold set to 0.22 and 0.25; NENIA tracker with the gating distance set to 20 and 50</td>
</tr>
<tr>
<td>gauss – 0.5 – utrack – 2, gauss – 0.5 – utrack – 10</td>
<td>Gaussian filtering; locating local maxima with the fluorescence threshold set to 0.5; u-track tracker with default parameters, except for the standard deviation multiplication factor which is set to 2 and 10</td>
</tr>
<tr>
<td>rand – 30 – nenia – 200</td>
<td>30 random uniformly distributed locations; NENIA tracker with the gating distance set to 200</td>
</tr>
</tbody>
</table>
case, the performance of the top 3 CTSs selected using our method is significantly higher than the mean F-score (our random selection baseline). Moreover, our method outperforms our second baseline based on the variance in the number of links (VN-score, data not shown). Finally, we observe that in every case one of our top 3 results achieves the maximum or near maximum F-score. Recall, that our ranking method does not use a ground truth (apart from the estimate of the cell size). We conclude that our method can be used to select the CTSs in practical scenarios.

Table 7.9: The results of our ranking of 11 CTSs on 10 sequences extracted from 5 real cell videos. Subscript of the sequence indicates the number of frames. “Cells” is the average number of cells per frame. “Valid” is the number of CTSs (out of 11) that pass our validity test (FaceValidity function, see text). “Mean±SD” and “Max” show the mean, standard deviation, and maximum of the F-scores among 11 CTSs for each sequence. The columns on the right show the F-scores of the top 3 CTSs selected using our method. The maximum score is underlined (if achieved).

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Cells</th>
<th>Valid</th>
<th>Mean±SD</th>
<th>Max</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>ak\textsubscript{10}</td>
<td>10</td>
<td>8</td>
<td>0.16 ± 0.20</td>
<td>0.76</td>
<td>0.76</td>
<td>0.23</td>
<td>0.04</td>
</tr>
<tr>
<td>ak\textsubscript{15}</td>
<td>15</td>
<td>6</td>
<td>0.38 ± 0.24</td>
<td>0.83</td>
<td>0.83</td>
<td>0.06</td>
<td>0.40</td>
</tr>
<tr>
<td>hex.6\textsubscript{10}</td>
<td>60</td>
<td>2.5</td>
<td>0.80 ± 0.32</td>
<td>1.00</td>
<td>0.96</td>
<td>0.95</td>
<td>0.95</td>
</tr>
<tr>
<td>hex.6\textsubscript{15}</td>
<td>61</td>
<td>3.5</td>
<td>0.69 ± 0.35</td>
<td>0.96</td>
<td>0.94</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td>hex.16\textsubscript{10}</td>
<td>16</td>
<td>10</td>
<td>0.68 ± 0.33</td>
<td>0.94</td>
<td>0.94</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td>hex.16\textsubscript{15}</td>
<td>16</td>
<td>9</td>
<td>0.69 ± 0.34</td>
<td>0.98</td>
<td>0.97</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>hex.22\textsubscript{10}</td>
<td>22</td>
<td>19</td>
<td>0.62 ± 0.33</td>
<td>0.94</td>
<td>0.92</td>
<td>0.92</td>
<td>0.94</td>
</tr>
<tr>
<td>hex.22\textsubscript{15}</td>
<td>22</td>
<td>20.5</td>
<td>0.66 ± 0.34</td>
<td>0.98</td>
<td>0.97</td>
<td>0.97</td>
<td>0.98</td>
</tr>
<tr>
<td>square\textsubscript{10}</td>
<td>32.5</td>
<td>8</td>
<td>0.51 ± 0.45</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
</tr>
<tr>
<td>square\textsubscript{15}</td>
<td>32.5</td>
<td>9</td>
<td>0.53 ± 0.46</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
<td></td>
</tr>
</tbody>
</table>

7.6 Discussion

The advent of high throughput assays in screening applications makes it impractical to tune a cell tracker for every new video. Therefore, as such assays become cheaper to perform and hence more popular, automatic CTS selection is likely to become one of the essential steps in data analysis. Automatic CTS
selection and tuning is currently not widely adopted, yet our research demonstrates that it can be useful in many practical situations.

ED-score performs reasonably well even in the presence of abrupt movements and intensive divisions. Furthermore, ED-score performs well in our wound healing type synthetic videos, when the relative directed speed of cells satisfies $\alpha < 0.4$. We obtained three previously reported real videos of wound healing assays \[72\]. We do not have the complete ground truth for these videos, but we estimated the relative magnitude of the directed speed component. We find that, in these real videos, $\alpha \approx 0.3$ which indicates that ED-score can be applicable in real wound healing assays.

Finally, we comment on the applicability of our methods in other tracking domains. In this work, we focus on cell tracking and run experiments specific to this domain (e.g., in the presence of divisions and deaths of cells). However, our definitions of the measurement and the link are general enough to be applied to a wide range of domains including tracking people, cells or particles. According to our definition, the measurement is merely a point in some space of features that describe an object to be tracked. For a given tracking problem, one can identify the appropriate features and define the link length accordingly. The only requirement is that tracked objects satisfy Assumption \[7.2\]. As we explain in Section \[7.3.2\], if the objects are readily identifiable by the features then the assumption is likely to be satisfied.

### 7.7 Conclusions

We propose a novel method for ranking cell tracking systems, which requires minimum input (estimate of the cell size) for each new video. Our method identifies CTSs that are likely to perform well, and runs pairwise comparisons of such CTSs. A pairwise comparison is based on our novel measure for ranking cell tracking algorithms without the need for a ground truth.

We have developed several novel measures for estimating the tracking performance. Our measures do not require the ground truth, and therefore can be used to automatically select the most appropriate tracking algorithm and its
parameters for a given set of cell detection results. One of our measures (VN-score) uses the variance in the estimated number of links across frames. Other measures (ED and PC-scores) use the lengths of links in the tracker’s output. In order to derive the ED and PC-scores we introduced the concepts of mirrored precision (MP) and mirrored recall (MR).

Our results indicate that the new method can effectively assist in CTS selection and tuning in practical scenarios. Furthermore, our method is sufficiently general for potential application to tracking systems, e.g., for multiple particle tracking.

Together the method for ranking CTSs and measures for estimating cell tracking performance without a ground truth constitute one of our major contributions of the thesis. Our evaluation suggests that our methods and measures can be useful in reducing the amount of manual work while achieving an accurate cell data analysis, which is the main goal of our thesis. This chapter concludes our contributions towards developing a portable automated cell tracking system. In the next chapter, we present an example biological application that can benefit from automated cell tracking.
Chapter 8

Evaluating the Motility of Plasmodium Ookinetes

The aim of the last technical chapter of this thesis is to demonstrate a sample biological application for cell tracking systems. In the introduction of the thesis, we highlighted automated drug screening as one potential application. In this chapter, we present a possible approach for automated anti-malaria drug screening based on the use of cell tracking.

In the drug screening scenario, researchers can test large numbers of candidate chemical compounds. Malaria parasites can then be placed into different wells, and a robotic device can facilitate the distribution of different compounds into different wells [117]. An automated cell tracking and trajectory analysis system can then evaluate the effects from different compounds. For example, the desired effect can comprise inhibiting the motility of the parasites.

Previously, we described a number of cell tracking techniques (Chapter 2.4), including our novel tracking algorithm (Chapter 5). In this chapter, we focus on automated analysis of parasites’ tracks. To this end, we develop a concise description for the tracks. This description can then be used in an automated drug screening system. In addition, our methodology reveals knowledge about the parasites’ locomotion. As such, the chapter contributes (i) a set of biological results; along with (ii) the proposed numerical representation of the parasites’ tracks.
8. EVALUATING THE MOTILITY OF PLASMODIUM OOKINETES

Figure 8.1: A Plasmodium ookinete (outlined) in Matrigel. This snapshot was extracted from a supplementary video presented in [87].

We start this chapter with a brief overview of related literature in Section 8.1. We then present our methodology in Section 8.2 and our findings in Section 8.3. The main focus of this thesis is numerical analysis. Therefore, in this chapter, we omit the technical details of the assays. Instead, these details are presented in Appendix D.

8.1 Plasmodium Ookinetes

Malaria is caused by eukaryotic parasites of the genus Plasmodium. Throughout their lives, the parasites advance through several stages in a vertebrate host and in a mosquito [112]. After a mosquito bites an infected host, the Plasmodium gametocytes enter the mosquito midgut within the blood meal. Eventually they transform into motile zygotes called ookinetes (Figure 8.1). Ookinetes penetrate the midgut boundary and transform into oocysts. In turn, each oocyst produces thousands of sporozoites that migrate to mosquito salivary glands. These sporozoites then infect a vertebrate host during the next blood meal.

In order for ookinetes to survive and proceed to the next life stage they need to reach the boundary of the midgut and penetrate the midgut epithelium [119]. The motility of ookinetes plays one of the key roles in fulfilling this task. Furthermore, the motility of the parasites is essential during some other life stages such as hepatic stages in the host. It is therefore an appealing idea to develop anti-malaria drugs that inhibit the parasite motility. Unfortunately, the under-
8.1. PLASMODIUM OOKINETES

lying mechanisms of parasite motility are still not fully understood.

Previously Vlachou et al. [119] presented an in vivo analysis of malaria ookinete locomotion. However, their analysis tends to be qualitative. Hellmann et al. provided statistical evidence that environmental constraints can guide the migration of Plasmodium sporozoites (another life stage of Plasmodium parasites) in vitro and in vivo [62]; and Kudryashev et al. explored the structural basis for chirality and directional motility of Plasmodium sporozoites [78].

Here we present a methodology that can lay the foundation for automated anti-malaria drug screening. Our method is based on measuring properties of the ookinete motility in response to a certain treatment. We study over 220 trajectories of ookinetes of the Plasmodium berghei species in Matrigel (the trade name for a gelatinous protein mixture marketed by BD Bioscience). Matrigel simulates the environment within the mosquito midgut. We use automated software routines to track the ookinetes over time and measure various properties of their tracks. We study the effect of the toxin Cytochalasin D on the ookinete motility. Our experiments suggest that this methodology can be used for automated identification of a set of chemical compounds with desirable properties (e.g., inhibiting motility), out of a large number of candidate compounds. The selected compounds can then form a basis for anti-malaria drugs.

In this study, we consider Plasmodium ookinetes, which occur in the mosquito stages of the parasite life cycle. After ookinetes reach the midgut boundary and develop into oocysts, the number of parasites increase dramatically, as each oocyst produces thousands of sporozoites [119]. As such, interrupting the midgut invasion by ookinetes is an attractive target for drug intervention. Such a drug can be distributed via vaccination of infected people. Note that this would not have a direct benefit from such a drug for an infected person. However, the vaccine can reduce the chances of passing the infection to other people, and eventually re-infecting the person [77].

Moreover, our methodology deepens the understanding of the ookinetes motility. We find that ookinetes move in a helical manner. Helical swimming is common across a range of living organisms [27, 56, 129]. Helical motion of the ookinetes has been previously appreciated qualitatively, by visual inspection of video frames [87, 120].
8. EVALUATING THE MOTILITY OF PLASMODIUM OOKINETES

Figure 8.2: Fragment of a visualization of P. berghei ookinete tracks generated by Bitplane Imaris software. Curves show ookinetes’ tracks in *Matrigel™*. Color encodes time that runs from a dark to a bright color. Green spots mark ookinetes’ locations at some point in time. The numbers below the scale bar show time elapsed since the recording has started. Image courtesy of Baum Lab, WEHI.

To the best of our knowledge, we are the first to provide quantitative (automated) statistical evidence that ookinetes of *P. berghei* move in a right-handed helical manner (Figure 8.2), and that the ability of the ookinetes to travel (span) large distances is related to the helicity of their tracks. We speculate that the right-handed helix is a consequence of the arrangement of ookinete microtubules.

8.2 Problem Statement and Methods

The work presented in this chapter is a part of a larger study that can be divided into stages as follows: (1) developing a systematic approach for preparation and imaging of the parasites in Matrigel in the presence of different compounds; (2) segmenting and tracking the parasites from recorded videos; (3) designing a meaningful and concise numerical description for the population of tracks; and
8.2. PROBLEM STATEMENT AND METHODS

(4) analyzing the results.

The first stage (motility assay) has been addressed by one of our collaborators, and we present the details in Appendix D. The second stage (segmentation and tracking) has been performed using Bitplane Imaris. Finally, the latter two stages (numerical analysis) of the study are described in this chapter.

From the computational point of view, the problem that we address at the third stage is as follows. Given a set of 3-dimensional coordinates of a population of ookinetes grouped into tracks, the task is to develop a numerical description for the population. The description must (a) be interpretable, i.e., provide insights into the nature of the motility of the population; and (b) reflect the changes in the motility of populations under a treatment.

We address the problem by developing a number of features. To improve the stability of our features, we use a data preprocessing step presented below.

8.2.1 Data Preprocessing

Recall that in the course of the imaging, ookinetes are placed in wells. An individual video comprises an observation of ookinetes from a single well. A replicating experiment is an observation of ookinetes in a different well, but under the same treatment. All videos were segmented and tracked using the Imaris software. After the tracking, each track is represented with a set of points, where each point has x, y and z coordinates in a right handed Cartesian coordinate system.

The tracks produced by Imaris were processed using scripts written in MATLAB 7.11.0. First, ookinetes that do not remain in the field of view (FOV) for the whole duration of the experiment were discarded. Second, the ookinetes that spend more than 5% of video time at the upper or lower boundary of the FOV were discarded as well. This was done because many ookinetes outside the FOV along the z-direction were wrongly detected as being located at the FOV boundary, due to the out-of-focus fluorescent cross-talk (registration of unwanted signal) across neighboring z-planes. After the preprocessing, each

---

1For historical reasons, in this study, segmentation and tracking had been performed using Imaris software. The author was given a set of tracks (coordinates) for further analysis.
of the remaining tracks comprised a sequence of 300 3-dimensional locations (from 300 video frames). Randomly selected tracks were manually verified for correctness.

Given the set of tracks, we develop six features that describe the set (the population) as a whole. We present the features in the next section.

8.2.2 Speed, Displacement and Period

The 6 numerical features were computed independently for each track using routines implemented in MATLAB 7.11.0. The first feature, the apparent speed, was computed as the sum of the Euclidean distances between consecutive track locations divided by 10, which was the duration of the experiment in minutes. The second feature, the magnitude of total displacement, is the Euclidean distance between the first and the last location of the track.

A visual inspection of ookinetes in the videos has suggested a periodic motion. Specifically, loops are common (see also Section 8.2.3). Therefore, our third feature is the period. The period is the time needed for an ookinete to make a loop.

The period was computed based on the Fast Fourier Transform (FFT) as outlined below. Recall that a track is represented as a set of 3-dimensional points \( \{x_1, y_1, z_1\}, \ldots, \{x_n, y_n, z_n\} \). In what follows, the three coordinates are processed in a similar manner, so we only show computations for the x-coordinate.

1. Compute signed differences between subsequent coordinates: \( d_x = \{(x_2 - x_1), \ldots, (x_n - x_{n-1})\} \).

2. Compute the FFT: \( f_x = \text{FFT}(d_x) \).

3. Compute the mean value, maximum, and ratio: \( r_x = \max(f_x) / \text{mean}(f_x) \).

4. Choose the highest ratio \( r = \max(\{r_x, r_y, r_z\}) \). If \( r < r_\theta \) then the period is undefined. Stop.

5. Set the period to \( 1/\text{freq} \), where \( \text{freq} = \text{argmax}(f) \). Here \( f \) is either \( f_x, f_y \), or \( f_z \).
In the above procedure, low-level frequencies (resulting in the periods larger than half of the duration of the experiments) were ignored. Furthermore, if none of the ratios was above a threshold $r_\theta$ (empirically set to 5) the period estimation was not performed. The latter case indicated a lack of prominent frequency components, i.e., an erratic track rather than a periodic track.

Note that the first two features have been previously used in the literature \cite{62}, and there have been similar methods for period computation \cite{56}.

### 8.2.3 Chirality, Radius and Pitch

Qualitatively, it has been noted that ookinetes move in a helical fashion \cite{87, 120}. A possible approach could be to develop a global model based on the helix equations. However, while the “helicity” pattern is indeed often present in the tracks, the tracks tend to be noisy and irregular. It can be difficult to successfully fit the global model. For example, Gurarie et al. modeled trajectories of algae with noisy helices, but for over the half of the tracks model estimation failed \cite{56}. Therefore, we took a non-parametric approach that allowed us to reliably capture the underlying helical pattern.

Our approach is based on the chirality feature that we computed as follows. For the track with points $P_1, \ldots, P_N$, and given a fixed step $d$, a sequence of four points was taken along the track: $P_i, P_{i+d}, P_{i+2d}, P_{i+3d}$ for each $i = 1, \ldots, N - 3d$. The four points define three vectors $\vec{A} = P_{i+d} - P_i$, $\vec{B} = P_{i+2d} - P_{i+d}$, and $\vec{C} = P_{i+3d} - P_{i+2d}$. A cross product $\vec{X} = \vec{A} \times \vec{B}$ was then computed, and the angle between $\vec{X}$ and $\vec{C}$ is compared to $\pi/2$ (Figure 8.3). If the angle was smaller than $\pi/2$ then the sequence of points formed a right helix primitive. The proportion of right primitives was recorded for a fixed step $d$. Finally, different steps $d = 1, \ldots, K$ were used, and the maximum proportion was reported as the track chirality. Choosing different steps allowed us to handle noise in the tracks. However this step should be low compared to the period of the helix.

In our experiments, the tracks consisted of 300 points, and a visual inspection

\footnote{In fact, we use a parameter that is an approximate lower bound estimate of the period. Our definition is non-parametric in the sense that it does not rely on a global motion model, but it looks at local point configurations along the track. This approach is convenient when the tracks are stochastic, and are far from perfect helices.}
identified around 6 loops in a typical track, i.e., the period was expected to be in the order of 50 frames. Occasionally tracks could have lower or higher periods, and therefore $K$ was set to 10, so that it was reliably smaller than half of the possible period.

Note that our computation was similar to that of Crenshaw et al. [27] who used the concept of a Frenet trihedron. In contrast to their work, we derive a single measure for a track.

Real ookinete tracks are noisy. Therefore, in order to estimate the last two features, the radius and pitch, we treated a track as a sample of individual helical loops. A number of three point samples $P_i$, $P_i + T/2$, $P_i + T$ were taken from the track for each $i = 1, \ldots, N - T$, where $T$ was the estimated track period. For each three point sample, the radius and pitch were estimated from the point coordinates as illustrated in Figure 8.4. The median values of the radius and pitch across all three point samples for $i = 1, \ldots, N - T$, were reported as the estimated radius and pitch for the track.

We applied our features, as well as some other common numerical methods to the tracked ookinete populations. We present our findings in the next section.
Figure 8.4: Estimation of the radius and pitch of a single helical loop represented with three points $P_i, P_i + T/2, P_i + T$, where $T$ is the estimated track period. The gray line shows a hypothetical helical loop going from point $P_i$ to point $P_i + T$. The geometry of the loop indicates the way to calculate the radius and pitch from the given points.

8.3 Results

8.3.1 Basic Properties of Ookinete Motility

Most ookinetes move along directed curved lines. However, there is no apparent direction that is common to all ookinetes (Figures 8.5 and 8.6). This can be confirmed by looking at track displacements along different axes. A track displacement vector is the signed difference between the last and first observed track coordinates ($x, y$ or $z$).

In a representative experiment (28 tracks from 3 fields of view), the distributions of the total track displacements along the $x$, $y$ and $z$ dimensions are likely to have zero median (two-sided sign test, $p > 0.2$). A similar observation can be made for other replicating experiments.

Recall that our first feature is the apparent speed, which is the sum of the distances between subsequent locations divided by the total duration of the experiment. Note that the apparent speed underestimates the real speed of ookinetes. Ookinetes are unlikely to go in perfectly straight lines between points, thus the real traveled distance is larger than the apparent distance. However, it is assumed that the apparent speed still gives a useful approximation of the real speed.

In total, 98 ookinetes from all replicating experiments have a mean speed of 8.09 $\mu$m/min and a standard deviation of the speed of 3.06 $\mu$m/min (Figure 8.7). A visual inspection suggests that the distribution looks like a normal dis-
8. EVALUATING THE MOTILITY OF PLASMODIUM OOKINETES

Figure 8.5: Reconstructed tracks of 35 ookinetes from 3 fields of view from a representative experiment. Circles mark initial locations of the ookinetes. Most of the tracks have a helical shape. There is no apparent direction that is common for all tracks.

Figure 8.6: Reconstructed tracks of 35 ookinetes from 3 fields of view from a representative experiment. The tracks are translated to a common initial position. Most of the tracks have a helical shape. There is no apparent direction that is common for all tracks.
8.3. RESULTS

Figure 8.7: The distribution of apparent speeds of 98 ookinetes. The mean speed is 8.09 \( \mu m/min \) and the standard deviation is 3.06 \( \mu m/min \). This distribution resembles a normal distribution (Lilliefors test, \( p > 0.3 \)).

The speed cannot be negative, so the distribution must be skewed. However, in our case, the mean value covers approximately 3 standard deviations, and skewing is not severe. Therefore, we believe it reasonable to apply a normality test. We find that the distribution of speeds resembles a normal distribution (Lilliefors test, \( p > 0.3 \)).

The second basic feature of ookinete tracks is the magnitude of the total displacement. This feature characterizes the directionality of the ookinetes’ motion. Assuming a fixed instant speed, ookinetes that turn in random directions would tend to have shorter total displacements than ookinetes that consistently follow one direction. The mean displacement for ookinetes is 43.8 \( \mu m \) and the standard deviation is 23.1 \( \mu m \).

In general, the apparent speed and the total displacement are orthogonal features. For example, an arbitrarily fast ookinete moving in a circular fashion will have a limited total displacement. However, in our experiments we find a strong correlation between the speeds and displacements (Spearman’s rho is 0.81 for 98 tracks). This suggests that ookinetes tend to move in a directed fashion.

We now look at the period of the tracks. Note that for some tracks, the
period cannot be reliably estimated, and such tracks are ignored (16 tracks). The distribution of the 82 estimated periods is shown in Figure 8.8. The mean period is 106.69 sec and the standard deviation is 43.87 sec. Note the presence of a group with abnormally large periods (14 tracks with estimated periods over 140 sec). The distribution is right-skewed, and we fit a log-normal distribution to the sample (Kolmogorov Smirnov test, $p > 0.17$). The estimated mean and standard deviation of the fitted log-normal distribution are, respectively, 4.6 sec and 0.37 sec.

As expected, there is a mild negative correlation between the apparent speed and periods (Spearman’s rho is $-0.5$ for 82 tracks), i.e., faster ookinetes tend to complete their loops in a shorter time.

Directionality and periodicity of ookinete motion, along with a visual inspection of the ookinete tracks in 3 dimensions (Figure 8.5), suggests that ookinetes move in helical manner. This suggestion is verified in the next section.

### 8.3.2 Ookinetes Move in a Right-handed Helical Manner

A right-handed helix is a spiral-like 3-dimensional curve oriented such that a clockwise screwing motion moves the helix away from the observer with the
8.3. RESULTS

Figure 8.9: A sample ookinete track. The ookinete moves in a right-handed helical fashion, that is, it tends to turn clockwise while moving towards. However the motion towards does not follow a straight line. The circle marks the starting point of the track. The distance between the starting and the ending points of this track is $107.47 \, \mu m$.

line of sight along the helix axis. Recall that we propose to use helical primitives to measure the extent to which ookinete tracks follow a right-handed helical motion (Section 8.2). We analyze 98 ookinetes’ tracks in the Matrigel, and obtain statistical evidence that ookinetes move in a right-handed helical manner.

In the sample of 98 tracks, the mean chirality is 80.64% and the standard deviation is 14.99%. The chirality in each replicating experiment is significantly larger than 50% (two-sided sign test, $p < 0.001$). This indicates that the right-handed primitives are not randomly appearing within the tracks. Instead, the tracks have an intrinsic tendency to form right-handed primitives. Therefore, we conclude that ookinetes tend to move in a right-handed helical fashion. At the same time, the tracks might comprise very noisy helices, with curved helix axes (Figure 8.9).

We plot the apparent speeds versus chirality in Figure 8.10. There is a strong correlation between the apparent speed and the chirality (Spearman’s rho is 0.82 for 98 tracks). In general, there seem to be a continuous range of speed-chirality pairs. We also present the simplest possible grouping of the data points. This grouping can be interpreted in terms of two sub-populations: (i) slow and er-
Figure 8.10: Apparent speeds and chiralities of 98 ookinete tracks. Note that for all tracks the chirality is above 50%. This suggests that ookinetes tend to move in a right-handed helical manner. There seem to be a continuous range of speed-chirality pairs. We also present the simplest possible grouping of the data points into: (i, green diamonds) slow and erratic, and (ii, blue circles) relatively fast and helical.

Finally, we estimate the radii and pitches of the helical tracks. All features are summarized in Table 8.1. Note that the estimation of the radii and pitches depends on the estimated periods. Therefore, we estimate the last two features for only 82 tracks out of 98.

We clustered the 82 tracks with all 6 features, for 2 and 3 clusters using the k-means and the expectation maximization algorithms. The features were transformed into z-scores (normalized to zero mean and unit variance). A grouping of 2 clusters looks similar to that shown in Figure 8.10. A grouping of 3 clusters is hard to interpret and not shown here.
8.3. RESULTS

Table 8.1: Mean values and standard deviation (in brackets) of our 6 features that describe 98 ookinete tracks or 82 tracks in the case of estimated periods, radii and pitches.

<table>
<thead>
<tr>
<th>Apparent speed, $\mu m/min$</th>
<th>Total displacement, $\mu m$</th>
<th>Chirality, $%$</th>
<th>Period, $min$</th>
<th>Radius, $\mu m$</th>
<th>Pitch, $\mu m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.09 (3.06)</td>
<td>43.80 (23.10)</td>
<td>80.64 (14.99)</td>
<td>1.78 (0.73)</td>
<td>1.84 (0.74)</td>
<td>9.67 (4.48)</td>
</tr>
</tbody>
</table>

8.3.3 Cytochalasin D Affects the Structure of Ookinete Tracks

In the previous sections, we introduced 6 features to describe the ookinete motility. We now estimate the reliability of our feature description for a population. To this end, we group (untreated) ookinetes by the four replicating experiments in which they were observed. For each feature, we find that in all populations, the values are likely to come from distributions with the same mean (one-way ANOVA, $p > 0.05$ for all features, except total displacement, $p > 0.03$ for the displacement feature). Furthermore, the joint distributions of features in different replicating experiments are likely to have the same mean point (multi-way ANOVA, $p > 0.07$). We conclude that our features provide a consistent description of the motility of ookinete populations.

We now look into the effects of a treatment on ookinete motility. As an example, we study the effects of 100 $nM$ of Cytochalasin D ($IC_{50}$, we omit drug response curves here) on different features (Figure 8.11). As expected, Cytochalasin D inhibits the motility of the ookinetes, as can be seen from the changed distributions of the apparent speeds and total displacements. This is accompanied by the effect on ookinete chirality. In the treated population the median chirality is lower and this difference is statistically significant at the 5% level. However, the treated population still exhibits a right-handed helical motion. The median chirality after treatment is still larger than 50% (two-sided sign test, $p < 0.001$). We plot the ookinete speeds and chiralities in control and treated populations in Figure 8.12. Cytochalasin D shifts the population towards slower and more erratic tracks.
8. EVALUATING THE MOTILITY OF PLASMODIUM OOKINETES

Figure 8.11: Effect of 100 $nM$ of Cytochalasin D on the ookinete motility. For all features, except radii and pitches, the difference in parameter medians is statistically significant at the 5% level. For the speeds, displacements and chirality, there are 98 control tracks (control) and 124 treated tracks (cytoD). For periods, radii and pitches, there are 82 control tracks (control) and 66 treated tracks (cytoD). On each box, the central mark is the median, the edges of the box are the 25-th and 75-th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually.
8.4. CONCLUSIONS

Figure 8.12: Apparent speeds and chiralities of 98 control ookinete tracks (blue circles) and 124 tracks of ookinetes treated with 100 nM of Cytochalasin D (green diamonds). Cytochalasin D shifts the population towards slower and more erratic tracks.

The decrease in the apparent speed coincides with the increasing periods. However, there is no effect on the radii and pitches of the tracks. The distributions of the radii and pitches are likely to remain the same after the treatment (Kolmogorov-Smirnov test, \( p > 0.2 \) for radii, \( p > 0.5 \) for pitches). We conclude that 100 nM of Cytochalasin D inhibits the speed of the ookinetes and make their tracks more erratic. At the same time, the tracks after the treatment still exhibit the helical pattern, and the structure of the motion (measured with the estimated radii and pitches) remains unchanged. We further reflect on the importance of our results in the next section.

8.4 Conclusions

In this chapter, we focus on numerical cell trajectory analysis for application to automated drug screening. Experiments performed in vitro generally allow one a higher degree of control over the environment compared to in vivo experiments. As such, in vitro experiments might be better suited for a high-throughput automated screening. Therefore, in this work, we consider ookinete
trajectories in Matrigel. We have also applied our methods to tracks in vivo and observed similar patterns (data not shown).

Therefore, we took a non-parametric approach that allowed us to reliably capture the underlying helical pattern. To the best of our knowledge, we are the first to statistically confirm that *P. berghei* ookinetes move in a right-handed helical manner. Furthermore, we observe that the helical tracks, albeit directed, do not spin around straight lines. Instead, in the long term, the tracks might exhibit another, higher level motion pattern. We speculate that the higher level pattern can be related to the foraging strategy inherent to ookinetes. Recall that in order to survive and proliferate, ookinetes must find the mid-gut boundary.

Our experiments with the Cytochalasin D toxin treatment confirm the usefulness of our numerical features. The experiments show that our features are capable of describing the nature of the influence of the treatment. From this description, one can obtain a better understanding of the mechanics of ookinete locomotion. Moreover, the features can be used in high-throughput anti-malaria compounds screening.

In our study, we do not explore the reasons why ookinetes tend to form groups (slow and erratic; fast and helical), whereas one could expect a continuous spectrum of speeds and chiralities. Here we only note a mild negative correlation between the apparent speeds and the number of ookinetes in the field of view, that is, density (Spearman’s rho is $-0.24$ for 98 tracks). We leave a deeper investigation for future work.
Chapter 9

Discussion and Conclusions

This thesis focuses on automated analysis of cell data in time lapse microscopy imaging. The topic has been gaining increasing importance as high throughput cell assays become cheaper. In such assays, hundreds and thousands of cells can be observed under a variety of treatment conditions, and the major bottleneck in fast and large scale data analysis is automated cell segmentation and tracking.

Major challenges for automated segmentation and tracking include the large variability in cell morphology and dynamics across experiments, and even within a single experiment. This has resulted in a wide range of proposed segmentation and tracking algorithms. Each such algorithm usually fits a particular type of cell video better than other types. Furthermore, previous approaches tend to have numerous parameters, and the accuracy of the approaches tends to depend on the right choice of parameter values.

The choice and subsequent tuning of an algorithm for a particular cell video often involves a considerable amount of manual work. Moreover, due to the variations in cell morphology and dynamics across the videos, the next video is likely to require another set of parameter values or another algorithm. Therefore, the amount of manual work tends to grow linearly with the number of videos that need to be processed. Alternatively, one can decide to use the same settings for different videos, but this usually results in sub-optimal algorithm performance. In this thesis, we aim to reduce the amount of manual work required while maintaining a high accuracy of the analysis.
9. DISCUSSION AND CONCLUSIONS

9.1 Summary of Contributions

We attack this aim from different angles and from different stages of a cell tracking pipeline. We first propose portable cell semi-segmentation and segmentation algorithms (Chapters 3 and 4). To the best of our knowledge, we are the first to introduce the concept of semi-segmentation of cells, and the first to introduce the use of sampling from an image so that the cell detection can be implemented as a clustering based on pixel locations and intensity at the same time. Another novel aspect of our semi-segmentation and segmentation algorithms is the notion of the acceptance function that encapsulates the description of cell appearance. The acceptance function is a key feature that allows us to achieve the portability of the algorithms. As a consequence, our semi-segmentation and segmentation algorithms can be more easily adopted to different cell shapes compared to many previous approaches. At the same time, our evaluation shows that our methods are capable of outperforming major baseline methods in the accuracy of cell segmentation and detection.

We also propose a novel portable cell tracking algorithm (Chapter 5). In this case, the portability of the algorithm is achieved through a design that minimizes the number of required parameters. In one of the versions, our cell tracking algorithm has only one parameter. Furthermore, this algorithm appears to be the first in the cell tracking literature that requires a user to provide only one (method A) or none (method B) parameters. On the range of videos tested, the parameter-free version of the algorithm was shown to correctly resolve the majority of the inter-frame cell associations (over 90%). Therefore, given a new cell video, our algorithm can be used to generate the initial ground truth set of links. Only a small number of these links then need to be corrected by a human.

We then present a method for semi-automated tracking results correction (Chapter 6). A novel aspect of this work is an approach to learn the probability of errors made by an automated tracker. Once such probabilities are estimated, the user can focus on only a small subset of the video frames, and correct tracks in those frames. The remaining part of a possibly long video (e.g., 2000 frames) can avoid manual verification, because the probability of having an error in that part is low. Our evaluation indicates that this can be a useful approach.
9.1. SUMMARY OF CONTRIBUTIONS

in practical scenarios. For example, we were able to increase the proportion of completely correct tracks on average by 13% through a manual revision of only 10% of all video frames after the automated tracking.

Moreover, we find it possible to rank the results of automated cell tracking systems by accuracy without having a ground truth at all. This ranking of cell tracking systems is possible using our approach presented in Chapter 7. This ranking is based on our new measure of cell tracking accuracy. Our measure does not require a ground truth for calculation. At the same time it is highly correlated (e.g., Spearman’s rho $\rho > 0.8$) with traditional measures, such as F-score, that do require the ground truth. The results in this chapter are particularly important, because they are not specific to the cell tracking domain. We speculate that these results can be widely used in a range of object tracking applications.

Finally, in Chapter 8, we present a possible application for cell segmentation and tracking. Specifically, we focus on developing an assay that can be used for an automated drug screening. Automated drug screening involves observing cells under different treatments (candidate drugs) and identifying the most appropriate drug with minimum human intervention. Automated drug screening is an important future technology, as it has the potential to significantly speed up the process of developing new drugs. In Chapter 8, we develop a numerical description for the motility of parasites of the genus *Plasmodium* (that cause Malaria). We show that our description is sufficiently sensitive to detect relevant changes in parasite behavior under a treatment.

Together, the contributions of this thesis can either form the core of a next generation software package for cell data analysis, or become plug-ins that extend the functionality of existing packages such as ImageJ and CellProfiler. Either way, we hope that our contributions can define a starting point towards a higher level of automation of cell data analysis, larger scale of assays, cheaper drug screening techniques, and eventually towards a better world.
9.2 Notes on Future Work

Our results open several directions for future work, and in this final section of the thesis, we highlight selected potential avenues of future research.

First, one can further explore the semi-segmentation approach for cell data analysis. A semi-segmentation algorithm does not produce complete cell outlines in images, but produces a range of features that can approximate cell location, size and shape. The advantages of semi-segmentation compared to traditional segmentation methods, can be speed, or in some cases, a higher accuracy of analysis.

Second, a possible future work direction is combining the semi-automated tracking approach SAMTRA with the (semi-)segmentation algorithm, such that the probability of an error in a video frame could be automatically estimated from the segmentation results. Furthermore, SAMTRA can be combined with a cell tracker and facilitate the tracking process in the first place.

Third, one can investigate our method for ranking cell tracking systems in other object tracking domains. This is possible because the formulation of our tracking accuracy measure does not use any cell specific heuristics. Moreover, we observe that our method reliably works on videos where a reasonably accurate tracking is possible. A direction for future work can focus on the theoretical foundations of this observation.
Appendix A

Established Numerical Methods

In this appendix, we present a selection of well established mathematical tools and concepts that are widely used at different stages of automated cell tracking systems. We group the tools by areas of application.

A.1 Image Processing

A.1.1 Pixel Connectivity

Pixel connectivity refers to a method in which different image pixels are treated as neighbors [52]. According to 4−connectivity, a pixel at \((x, y)\) has 4 neighbors located at \((x - 1, y)\), \((x + 1, y)\), \((x, y - 1)\), \((x, y + 1)\). According to 8−connectivity the same pixel has 8 neighbors located at \((x - 1, y - 1)\), \((x - 1, y)\), \((x - 1, y + 1)\), \((x, y - 1)\), \((x, y + 1)\), \((x + 1, y - 1)\), \((x + 1, y)\), and \((x + 1, y + 1)\). In other words, pixels that are connected with location \((x, y)\) along a diagonal direction are neighbors according to 8−connectivity, but are not neighbors according to 4−connectivity.

4−connectivity is a more conservative type of connectivity, and it can be more beneficial to use this type for cell tracking applications. Cells often have convex elliptical shapes, and 8−connectivity tends to result in touching cells being merged into one object.
A.1.2 Gradient Transform and Distance Transform

A gradient transform of an image $I(x, y)$ is an image $J(x, y)$, where each pixel $J(x, y)$ reflects the variation in intensity levels in some neighborhood of $(x, y)$. There can be different specific definitions of the gradient transform [52]. For example, this can be defined via the Sobel operator, where the image is convolved with two predefined $3 \times 3$ kernels resulting in two images that reflect gradients along the x and y axes, denoted $J_x(x, y)$ and $J_y(x, y)$ respectively. The final gradient image is defined as a combination of the convolution results $J(x, y) = \sqrt{J_x^2(x, y) + J_y^2(x, y)}$, where the mathematical operators are applied element-wise.

A distance transform of a binary image $B(x, y)$, where 0 denotes background and 1 denotes an object pixel, is an image $D(x, y)$, where the value of each pixel in $D(x, y)$ is the Euclidean distance to the nearest 1 pixel from the point $(x, y)$ in image $B(x, y)$.

A.1.3 Laplacian of Gaussian Filter

The Laplacian of Gaussian filter is often used in locating image regions that differ in certain properties (e.g., intensity levels) from surrounding regions [81]. In this section, for ease of narration, we assume that an image and a kernel is a continuous function of two arguments. In practice, discrete versions of the operators presented below are used.

An input image $I(x, y)$ is first convolved by a Gaussian kernel at a certain scale $\sigma$: $K(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \cdot \exp\left\{-\frac{x^2 + y^2}{2\sigma^2}\right\}$ (A.1) in order to obtain the so-called scale-space representation $S_\sigma(x, y, \sigma) = I(x, y) * K(x, y, \sigma)$, where $*$ denotes convolution. The Laplace operator is applied to the convolution result $L(x, y, \sigma) = \nabla^2 S_\sigma(x, y) = \frac{\partial^2 S_\sigma}{\partial x^2} + \frac{\partial^2 S_\sigma}{\partial y^2}$. (A.2)

The Laplacian has extrema for dark and bright blobs of the size of $\sigma$. In
A.2. OBJECT TRACKING

order to detect blobs of different sizes, one can compute normalized Laplacians
\( L_N(x,y,\sigma) = \sigma^2 \cdot L(x,y,\sigma) \) for different scales \( \sigma \). This results in a function of
3 arguments, where local extrema represent dark or bright blobs of different scales.

A.2 Object Tracking

A.2.1 Kalman Filter

The Kalman filter is a method for reconstructing a signal at discrete time points from a series of noisy measurements [70]. This method is widely used in tracking applications where the signal is a time series of object locations (i.e., a track) and measurements come from an object detector (e.g., cell segmentation algorithm).

In object tracking, the Kalman filter is used to smooth a track of a single object, given measured object positions along with estimates of measurement errors, and the motion model. The smoothed track is a compromise between the two sources of information (the model and the measurements). Furthermore, one can adjust the balance between the two sources. In extreme cases a track can comprise unchanged measured positions (we assume no measurement error), or positions according to the model ignoring the measurements (do not trust the measurements at all).

Let an object at time point \( t \) be described with a vector \( \vec{x}_t \) (e.g., a vector of coordinates of cell centroid). In the Kalman filter method, it is assumed that the evolution of the object state can be described as

\[
\vec{x}_t = F \cdot \vec{x}_{t-1} + \vec{w}_t. \tag{A.3}
\]

Here, \( F \) is the transition model (i.e., the motion model) that is assumed to be known \textit{a priori}, and \( \vec{w}_t \sim \mathcal{N}(0, Q) \) is the process noise which is assumed to have a multivariate normal distribution with zero mean and covariance \( Q \).
Furthermore, it is assumed that measurements $\bar{z}_t$ are derived as

$$\bar{z}_t = H \cdot \bar{x}_t + \bar{v}_t, \quad (A.4)$$

where $H$ is the measurement model that relates true states $\bar{x}_t$ to measurements, and $\bar{v}_t \sim \mathcal{N}(0, R)$ is the measurement noise which is assumed to have a multivariate normal distribution with zero mean and covariance $R$.

Finally, it is assumed that the initial position $\bar{x}_0$ and noise vectors $\bar{w}_t$ and $\bar{v}_t$ at each step $t = 1, \ldots, k$ are mutually independent.

The Kalman filter recursively estimates the position of the object at each time step. Given an estimator for the previous step, the filter estimates the next position in two steps: prediction and update. The prediction step takes into account the motion and measurement models, whereas the update step also considers the observed measurement at the next step.

Let $\hat{x}_{t-1|t-1}$ be a position estimate at step $t-1$ and let $P_{t-1|t-1}$ be the error covariance matrix (that estimates the accuracy of the position estimate). The predicted (a priori) position at step $t$ is defined as

$$\hat{x}_{t|t-1} = F \cdot \hat{x}_{t-1|t-1}, \quad (A.5)$$


and the predicted (a priori) error covariance matrix at step $t$ is defined as

$$P_{t|t-1} = F \cdot P_{t-1|t-1} \cdot F^T + Q, \quad (A.6)$$

where $F^T$ denotes matrix transpose.

The updated (a posteriori) position and error estimates are computed as

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t \cdot y_t \quad (A.7)$$

and

$$P_{t|t} = (I - K_t \cdot H) \cdot P_{t|t-1}. \quad (A.8)$$

Here $y_t$ is the measurement residual: $y_t = z_t - H \cdot \hat{x}_{t|t-1}$, and $K_t$ is the optimal Kalman gain. The gain is optimal with respect to minimizing the error covari-
A.2. OBJECT TRACKING

ance, and can be shown to be

\[ K_t = P_{t|t-1} \cdot H^T \cdot (H \cdot P_{t|t-1} \cdot H^T + R)^{-1}. \]  

(A.9)

A.2.2 Particle Filter

A particle filter is a method to obtain an approximate solution to the single object tracking problem under a Bayesian formulation [35].

Consider a single object tracking problem, where the object at each discrete time step \( t \) (e.g., video frame) is represented by a state vector \( \vec{s}_t \). Furthermore, at each time step an object detector produces a measurement of the object state \( \vec{m}_t \). The Bayesian approach for single object tracking consists of recursive estimation of the posterior distribution \( p(\vec{s}_t | \vec{m}_{1:t}) \), where \( \vec{m}_{1:t} \) is a shorthand for \{\vec{m}_1, \ldots, \vec{m}_t\}. This estimation is obtained by assuming that the cell state \( \vec{s}_t \) evolves as a known Markov process \( D(\vec{s}_t | \vec{s}_{t-1}) \). Moreover, it is assumed that the measurement model is given in the form of a likelihood \( L(\vec{m}_t | \vec{s}_t) \). Under these assumptions, the estimation of the posterior distribution is performed in two steps: prediction and update.

In the prediction step, the estimated posterior densities from the previous frame \( p(\vec{s}_{t-1} | \vec{m}_{1:t-1}) \) are combined to obtain a prior density using the Chapman-Kolmogorov equation

\[ p(\vec{s}_t | \vec{m}_{1:t-1}) = \int D(\vec{s}_t | \vec{s}_{t-1}) \cdot p(\vec{s}_{t-1} | \vec{m}_{1:t-1}) \, d\vec{s}_{t-1}. \]  

(A.10)

In the update step, Bayes rule is applied to the prior density in order to obtain the posterior density

\[ p(\vec{s}_t | \vec{m}_{1:t}) \propto L(\vec{m}_t | \vec{s}_t) \cdot p(\vec{s}_t | \vec{m}_{1:t-1}). \]  

(A.11)

The above method is computationally intractable. A particle filter delivers an approximation of the posterior by means of \( N \) random weighted samples.
where $\mathbf{\delta}(\cdot)$ is the Dirac delta function.

The particles are drawn according to the principles of important sampling. The key observation is that for sampling, it is possible to use an arbitrary function $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{m}_t)$, as long as it has the same support as $p(\mathbf{s}_t|\mathbf{m}_{1:t})$. Indeed, let $\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, \mathbf{m}_t)$, and let $f(\mathbf{x}_t)$ be an arbitrary function. Then the expectation of $f(\mathbf{x}_t)$ with respect to $p(\mathbf{s}_t|\mathbf{m}_{1:t})$ can be approximated as

$$
\hat{f}(\mathbf{x}_t) \approx \sum_{i=1}^{N} f(\mathbf{x}_t^{(i)}) \cdot w_t^{(i)},
$$

(A.13)

where $w_t^{(i)} \propto p(\mathbf{x}_t^{(i)}|\mathbf{m}_{1:t})/q(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, \mathbf{m}_t)$. Therefore the Chapman-Kolmogorov equation can be approximated with

$$
\sum_{i=1}^{N} f(\mathbf{x}_t^{(i)}) \cdot w_t^{(i)}.
$$

If $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{m}_t)$ is chosen to be factorisable then after certain algebraic manipulations the following sampling algorithm can be derived: for each $i = 1$ to $N$, draw $\mathbf{x}_t^{(i)}$ from $q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, \mathbf{m}_t)$ and set the corresponding weight according to

$$
w_t^{(i)} \propto w_{t-1}^{(i)} \frac{L(\mathbf{m}_t|\mathbf{x}_t^{(i)}) \cdot D(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, \mathbf{m}_t)}.
$$

(A.14)

A common problem with the above filter is degeneracy, that is, a situation when after a few iterations, all but one particle will have negligible weights. In order to alleviate degeneracy, particles with small weights can be discarded and new particles can be re-sampled. The decision on discarding low weight particles can be based on an estimate of the effective sample

$$
N_{\text{eff}} \approx \left( \sum_{i=1}^{N} (w_t^{(i)})^2 \right)^{-1}.
$$

(A.15)

Re-sampling can be applied when $N_{\text{eff}}$ falls below a certain threshold $N_T$. 
A.2. OBJECT TRACKING

A.2.3 Mean Shift Algorithm

Mean shift is a simple iterative technique for estimating a mode of a probability distribution given a sample from that distribution [47].

Let $S = \{ \vec{x}_1, \ldots, \vec{x}_n \}$ be the sample from a multinomial probability density function (PDF) $f(\vec{x})$ in $d$ dimensions; and let $h$ be a bandwidth parameter. Given a kernel $K_h(\vec{x})$, a kernel density estimation of the PDF is given by

$$\hat{f}(\vec{x}) = \frac{1}{n \cdot h^d} \sum_{i=1}^{n} K_h \left( \frac{\vec{x} - \vec{x}_i}{h} \right). \quad (A.16)$$

Radially symmetric kernels can be expressed as $K_h(\vec{x}) = c_{k,d} \cdot k(||\vec{x}||^2)$, where $c_{k,d}$ is a normalization constant used to enforce integration of $K_h(\vec{x})$ to 1. The gradient of the above density estimator can be then expressed as

$$\nabla \hat{f}(\vec{x}) = \frac{2 \cdot c_{k,d}}{n \cdot h^{d+2}} \left[ \sum_{i=1}^{n} g \left( \frac{||\vec{x} - \vec{x}_i||^2}{h^2} \right) \cdot \left[ \frac{\sum_{i=1}^{n} \vec{x}_i \cdot g \left( \frac{||\vec{x} - \vec{x}_i||^2}{h^2} \right)}{\sum_{i=1}^{n} g \left( \frac{||\vec{x} - \vec{x}_i||^2}{h^2} \right)} - \vec{x} \right] \right], \quad (A.17)$$

where $g(\vec{x}) = -k'(\vec{x})$, and the second term

$$\vec{m}_h(\vec{x}) = \frac{\sum_{i=1}^{n} \vec{x}_i \cdot g \left( \frac{||\vec{x} - \vec{x}_i||^2}{h^2} \right)}{\sum_{i=1}^{n} g \left( \frac{||\vec{x} - \vec{x}_i||^2}{h^2} \right)}$$

is called the mean shift. The mean shift points towards the direction of maximum increase of the density.

In the mean shift algorithm, given an initial estimate $\vec{x}_k$, the next estimate is computed as $\vec{x}_{k+1} = \vec{x}_k + \vec{m}_h(\vec{x}_k)$, and this iterative procedure is guaranteed to converge.

221
A.3 Other Methods

A.3.1 Hierarchical Clustering

Consider a sample of \(N\) instances \(\vec{m}_1, \ldots, \vec{m}_N\), where each instance is represented with a vector of features \(\vec{m}_i = \{f_{i,1}, \ldots, f_{i,n}\}\). The task is to group instances in \(C\) clusters such that the instances within a cluster are similar and instances in different clusters are different with respect to a given similarity (or distance) measure, such as the Euclidean distance.

Hierarchical clustering is a widely used approach to clustering [60]. There are two main variations of hierarchical clustering: agglomerative and divisive. In divisive hierarchical clustering all instances are first put into a single cluster, and then at each iteration one of the existing clusters is split. Divisive clustering tends to be more complicated and has a higher computational complexity than agglomerative clustering. In the rest of this section, we focus on agglomerative hierarchical clustering.

A high level outline of agglomerative hierarchical clustering is shown below.

1. Create singleton clusters for each of the instances.

2. Find the two most similar clusters, and merge these clusters.

3. If there are more than \(C\) clusters, go to step 2.

There are different possible strategies to decide how to compute the similarity between clusters. Furthermore, cluster similarity measures are based on instance similarity measures, and there are different ways to define the similarity between two instances \(\vec{m}_i\) and \(\vec{m}_j\). Popular choices include the Euclidean distance \(D_E = \sqrt{\sum_{k=1}^{n}(f_{i,k} - f_{j,k})^2}\); Mahalanobis distance \(D_M = \sqrt{(\vec{m}_i - \vec{m}_j)^T S^{-1} (\vec{m}_i - \vec{m}_j)}\), where \(S\) is the covariance matrix; and cosine similarity \(D_C = \vec{m}_i \cdot \vec{m}_j / \|\vec{m}_i\| \cdot \|\vec{m}_j\|\).

Once an instance similarity measure \(D(\vec{m}_i, \vec{m}_j)\) is defined, the similarity between clusters \(A\) and \(B\) (also called a linkage criterion) \(L(A, B)\), can be defined as a centroid linkage \(L(A, B) = D(\vec{a}, \vec{b})\), where \(D\) is applied to the centroids of the two clusters. Other popular choices include complete linkage.
\( \mathcal{L}(A, B) = \max \{ D(\vec{m}_i, \vec{m}_j), \vec{m}_i \in A, \vec{m}_j \in B \} \), and single linkage \( \mathcal{L}(A, B) = \min \{ D(\vec{m}_i, \vec{m}_j), \vec{m}_i \in A, \vec{m}_j \in B \} \).

The choice of the instance and cluster similarity measures is usually guided by the application. The complexity of the algorithm is \( O(N \times N \times C) \), and therefore agglomerative hierarchical clustering can be slow for large datasets.

### A.3.2 Hungarian Assignment

An optimal assignment problem can be defined as follows. Consider two groups of objects, each group with \( N \) elements. These can be, for example, detected cells in two subsequent frames, or workers and jobs. Furthermore, consider a non-negative \( N \times N \) cost matrix \( A \), where each matrix element \( a_{i,j} \) denotes a cost of assigning an object \( i \) from the first group to object \( j \) from the second group. In an example with cells, these costs can depend on spatial proximity and similarity between cell measurements in subsequent frames: the more similar the measurements are, the more likely it is that they originate from the same cell, and the lower is the association cost. In an example with workers, the costs can represent the salary that worker \( i \) demands in order to accomplish job \( j \). The problem is then to assign each object from the first group to each object from the second group such that the sum of the individual assignment costs is minimized.

The Hungarian algorithm (also known as the Kuhn-Munkres algorithm or Munkres assignment algorithm) is a polynomial method to solve the above problem [89]. The original algorithm had the complexity \( O(N^4) \). The algorithm was later improved to have \( O(N^3) \) complexity.

Below we outline a \( O(N^3) \) version of the algorithm.

1. Subtract the minimum row element from each row.

2. Subtract the minimum column element from each column.

3. After the first two steps there will be some zero elements. Cover all zeros with the minimum number of lines. A line can be either horizontal (covers a row) or vertical (covers a column). Some elements can be covered twice...
(at lines crossing). Minimum line coverage can be found with different methods (not given in this Appendix).

4. Add the minimum uncovered element to every covered element. If an element is covered twice, add the minimum value to it twice.

5. Cover the zero elements with lines again. If the number of lines is less than $N$, go to 4.

6. Now every row and every column has a zero element. Select $N$ zero elements such that each row and each column has exactly one zero element. The positions of the selected zero elements give the required assignment.

A rectangular assignment problem (when the number of rows is not equal to number of columns) can be converted to a square assignment problem above by adding dummy rows (or columns) to make the matrix square. Each element of a dummy row (or column) is then set to the maximum cost value.
Appendix B

Cell Tracker NENIA

In this appendix, we present proofs, additional details and supporting arguments for Chapter 5 (the automated cell tracker NENIA).

B.1 Assumption about the Cell Dynamics

We only consider the distributions of $P(i \rightarrow j)$ with zero mean, because before doing any assignment we can subtract the superposition of offsets over all pairs of measurements (e.g., as proposed by Chowdhury et al. [24]).

Let $P(i \rightarrow j)$ be the probability that a cell moves between the locations $\vec{x}_i$ and $\vec{x}_j$, $\vec{x}_i, \vec{x}_j \in \mathbb{R}^n$. This probability can be written as $P(i \rightarrow j) = P(\vec{\theta})P(r^2|\vec{\theta})$, where $\vec{\theta}$ is a $n$ dimensional vector of angles between the line $\vec{x}_i \vec{x}_j$ and each of the coordinate axes, and $r^2$ is the squared Euclidean distance $r^2 = \|\vec{x}_i - \vec{x}_j\|^2$.

We do not restrict our algorithm to any specific distribution of $\vec{\theta}$, and we estimate $P(i \rightarrow j)$ only on the basis of the expected probability for the distance

$$P(i \rightarrow j) \propto \int_0^{2\pi} \cdots \int_0^{2\pi} P(\vec{\theta})P(r^2|\vec{\theta}) \, d\theta_1 \cdots d\theta_n. \tag{B.1}$$

As defined by Gomez et al. [51], the $n$ dimensional power exponential dis-
B. CELL TRACKER NENIA

The distribution with zero mean is

\[ P(\vec{x}) = k(n, \beta) |\Sigma|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} \left[ \vec{x}^T \Sigma^{-1} \vec{x} \right]^{\beta} \right\}, \]  

(B.2)

where \( k(n, \beta) \) is a parameter that determines the shape of the distribution, \( \Sigma \) is a positive definite covariance matrix, and superscript \( T \) means the transpose of a matrix, e.g., \( A^T \) is a transpose of matrix \( A \).

Here \( \vec{x} \in \mathbb{R}^n \) can be written as \( \vec{x} = r \cdot \cos \vec{\theta} \), and the expression \( \vec{x}^T \Sigma^{-1} \vec{x} = r^2 \left[ (\cos \vec{\theta})^T \Sigma^{-1} (\cos \vec{\theta}) \right] \), where \( \cos \vec{\theta} \) denotes the \( n \) dimensional vector with elements \( \cos \theta_i \). \( \Sigma \) is a positive definite matrix, hence \( \Sigma^{-1} \) is also positive definite and thus \( \left[ (\cos \vec{\theta})^T \Sigma^{-1} (\cos \vec{\theta}) \right] > 0 \) for all \( \vec{\theta} \). We have that for all \( \vec{\theta} \), \( \vec{x}^T \Sigma^{-1} \vec{x} \) increases with \( r^2 \) and thus expression B.2 decreases with \( r^2 \).

B.2 Two Round Assignment Approach

Suppose that at frame \( f \) we have a set of previously established tracks \( T_f = \{ t_i \} \), where a track \( t_i \) has the location \( \vec{m}_f, i \). Further, suppose that at frame \( f + 1 \) we have a set of measurements \( \vec{m}_{f+1} = \{ \vec{m}_{f+1, j} \} \).

Consider assigning measurements in \( \vec{m}_{f+1} \) to tracks in \( T_f \) such that a measurement \( \vec{m} \) can be assigned to 0 or 1 tracks, and 0, 1, or 2 measurements can be assigned to a track \( t \). Figure 5.3 (Section 5.2.2) explains the choice of this assignment setup with an example. We refer to an individual assignment of measurement \( j \) to track \( i \) as a link \( i \rightarrow j \). The link \( i \rightarrow j \) represents a potential cell offset from the location \( \vec{m}_{f,i} \) to the location \( \vec{m}_{f+1,j} \). The length of a link \( i \rightarrow j \) is the Euclidean distance \( r_{ij} = \| \vec{m}_i - \vec{m}_j \| \) between the locations of \( \vec{m}_i \) and \( \vec{m}_j \). Where unambiguous, we use term link to denote the length of a link.

We refer to a set of tracks, measurements, and established links between them as an assignment. Our task is to find the assignment that corresponds to the true set of cell events that occur between frames \( f \) and \( (f+1) \). A popular approach for solving this kind of task is based on Bayesian inference. Here the likelihood of an assignment is expressed using the probabilities of individual events implied by the assignment [76, 88]. The four possible events in the
assignment are as follows.

1. A one-to-one assignment \( i \rightarrow j \) (e.g., link 1 \( \rightarrow \) 1 in Figure 5.3) implies a cell move between locations \( \vec{m}_i \) and \( \vec{m}_j \), and we express the probability of such an event as \( P_{m} \cdot P(i \rightarrow j) \).

2. A two-to-one assignment \( i \rightarrow j \) and \( i \rightarrow k \) (e.g., links 2 \( \rightarrow \) 2 and 2 \( \rightarrow \) 3 in Figure 5.3) implies the following set of events: {a cell division, a displacement between \( \vec{m}_i \) and \( \vec{m}_j \) of the first daughter cell, and a displacement between \( \vec{m}_i \) and \( \vec{m}_k \) of the second daughter cell}. We express the probability of this events as \( P_{s} \cdot P(i \rightarrow j) \cdot P(i \rightarrow k) \).

3. A zero-to-one assignment (e.g., point 3 at frame \( f \) in Figure 5.3) implies a cell disappearance: death or a false negative error. The probability of this event is \( P_{dis} = P_d + P_{fn} \).

4. A one-to-zero assignment (e.g., point 4 at frame \( f+1 \) in Figure 5.3) implies a false positive error with the probability \( P_{fp} \).

Under the independence assumption, we can express the likelihood of assignment \( A \) as

\[
L(A) = \prod_{a \in M(A)} P_{m} P(i_a \rightarrow j_a) \times \prod_{b \in S(A)} P_{s} P(i_b \rightarrow j_b) P(i_b \rightarrow k_b) \times \prod_{c \in D(A)} P_{dis} \times \prod_{d \in FP(A)} P_{fp}
\]  

(B.3)

Here, \( M(A) \), \( S(A) \), \( D(A) \) and \( FP(A) \) are the sets of links and measurements corresponding to a move, division (split), disappearance and false positive configurations respectively. At least one of the sets is non-empty. Now our problem is to find assignment \( A' \) that maximizes expression (B.3), that is, find any \( A' = \arg \max_A L(A) \). In what follows, we shorten expressions like \( L(A) \) to \( L \).
Figure B.1: Possible configurations of links between two measurements and two tracks. Lemma B.1 states that configuration in the Figure B is more likely.

A straightforward solution of the problem (e.g., using integer programming) requires values for parameters $P_s$, $P_d$, $P_{fp}$, and $P_{fn}$. In what follows, we propose an approach that delivers an approximate solution, but does not require knowledge of the values for these parameters. The idea is to replace the four parameters by a single parameter, called the gating distance (introduced later), and then use our technique to estimate the value of this remaining parameter. Our approach is based on a simplification of the objective function (Equation B.3) in three steps.

### B.2.1 Simplification Step 1

The first simplification step is based on the following lemma.

**Lemma B.1.** Consider an arbitrary assignment, such that there exist links $i \rightarrow j$, $i \rightarrow l$ and unassigned track $t_k$ (Figure B.1 a). If $P(i \rightarrow l) < 81 \cdot P(k \rightarrow l)$ then removing the link $i \rightarrow l$ and adding the link $k \rightarrow l$ (Figure B.1 b) increases the likelihood of the assignment.\footnote{Proofs of all lemmas are given in Appendix B.3}

For a random pair of move probabilities $P(i \rightarrow l)$ and $P(k \rightarrow l)$, it is likely that $P(i \rightarrow l) < 81 \cdot P(k \rightarrow l)$, because the second term is multiplied by a large coefficient. The particular value here is not important (the number 81 arises from the probabilities in Assumption 5.1, see the proof in Appendix B.3). Thus it is likely that the second configuration is favorable. As an approximation, we consider that there cannot be divisions, unless all tracks have been assigned a measurement. Therefore, we propose to seek an assignment that maximizes...
B.2. TWO ROUND ASSIGNMENT APPROACH

Equation B.3 using a two-round assignment approach, where, in each round, assigning two measurements to one track is not permitted. We then find optimal assignments in each round using modified objective functions, presented below, and then combine the two found assignments into one (Figure B.2). The advantage of our two round approach is that the divisions are eliminated from a single assignment round. A division event occurs when there are measurements assigned to the same track in both rounds. (This is a copy of Figure 5.4, repeated here for clarity.)

Figure B.2: Two-round assignment approach. In a single round two-to-one links are not permitted, hence division events are eliminated from a single round. However, a division event can occur when there are measurements assigned to the same track in both rounds. (This is a copy of Figure 5.4, repeated here for clarity.)

Let \( A'_1 \) be an optimal assignment in the first round, and \( A'_2 \) be an optimal assignment in the second round. We approximate an optimal inter-frame assignment by the assignment \( A' \) that is derived by adding the links and measurements from \( A'_2 \) to \( A'_1 \) (Figure B.2).

B.2.2 Simplification Step 2

We now present the objective functions for the two assignment rounds. Recall that the probability of a division event is \( P_s \cdot P(i \rightarrow j) \cdot P(i \rightarrow k) \). This can be written as \( [P_m P_f P(i \rightarrow j)] \cdot \left[ \frac{P_s P(i \rightarrow k)}{P_m P_f} \right] \). We use this decomposition to divide the likelihood in Equation B.3 into two parts \( L = L'_1 L'_2 \), where one part contains one of the terms with \( P(i \rightarrow j) \) or \( P(i \rightarrow k) \) for each division.

Using this decomposition, in the first round we assign measurements in
\( \tilde{m}_{f+1} \) to tracks in \( T_f \), and seek an assignment \( A'_1 \) that maximizes

\[
L'_1 = \left[ \prod_{a \in \tilde{m}'_1} P_m P(i_a \rightarrow j_a) \right] \times \left[ \prod_{b \in D'_1} P_{\text{dis}} \right] \left[ \prod_{c \in FP'_1} P_{fp} \right].
\] (B.4)

Here \( \tilde{m}'_1 \), \( D'_1 \), and \( FP'_1 \) are the sets of links and measurements corresponding to the selection of moves, disappearances and false positives in the first round. Note that at least one of the sets is non-empty.

In the second round, we assign the measurements that are left unassigned after the first round (i.e., measurements in \( FP'_1 \)), to tracks in \( T_f \). Here we seek an assignment \( A'_2 \) that maximizes

\[
L'_2 = \min \left[ 1, \prod_{a \in \tilde{m}'_2} \frac{P_s P(i_v \rightarrow k_v)}{P_m P_{fp}} \right].
\] (B.5)

Here the measurements or tracks that are left unassigned imply disappearance or false positive events that have already been implied in the first round. Thus we omit the corresponding terms in Equation \( \text{B.5} \). \( \tilde{m}'_2 \) corresponds to the selection of links. Note that if \( \tilde{m}'_2 \) is empty, \( L'_2 = 1 \).

**Lemma B.2.** Let \( A'_1 \) and \( A'_2 \) be optimal assignments in the first and the second rounds respectively. If there is a link \( i \rightarrow k \) in \( A'_2 \), then there must be a link \( i \rightarrow l \) in \( A'_1 \).

This lemma shows that in the first assignment round we select the links that correspond to cell moves, and we also follow one of the daughters in divisions. In the second round, from this set of links we select those that correspond to cell divisions (we follow the second daughter).

### B.2.3 Simplification Step 3

Finally, we simplify both rounds further by introducing gating. Recall that each link \( i \rightarrow j \) has a corresponding probability \( P(i \rightarrow j) \). By gating we mean that we
prohibit making the links that have \( P(i \rightarrow j) \leq P_g \), where \( P_g \) is some probability gating threshold. We now want to express the gating threshold in terms of other parameters. If we set \( P_g = \sqrt{P_m P_f} \), then the following property holds.

**Lemma B.3.** If the links that have \( P(i \rightarrow j) \leq \sqrt{P_m P_f} \) are prohibited, then the optimal assignment in the first or second round has the maximum possible number of allowed links.

Each link \( i \rightarrow j \) has a length denoted \( r_{ij} \), and under Assumption 5.3 about cell dynamics, \( P(i \rightarrow j) \) is decreasing with \( r_{ij}^2 \). Hence there exists some distance threshold \( R_g \) such that the links that have \( P(i \rightarrow j) \leq P_g \) also have \( r_{ij}^2 \leq R_g \) and vice versa. Threshold \( R_g \) is called the gating distance. Let \( R_g \) be a parameter of our tracking problem. Then in both assignment rounds, we can follow the same procedure: first, we prohibit the links where \( r_{ij} \geq R_g \); next, we only consider assignments with the maximum number of possible links; finally, we select an assignment \( A'' \) that maximizes

\[
L'' = \prod_{a \in \text{links}} P(i_a \rightarrow j_a),
\]

or, taking the log likelihood, minimizes assignment cost

\[
C = \sum_{a \in \text{links}} - \ln P(i_a \rightarrow j_a) = \sum_{a \in \text{links}} c(i_a \rightarrow j_a).
\]

Here, \( c(i \rightarrow j) \) is the cost of link \( i \rightarrow j \). Under Assumption 5.3 about the cell dynamics, we propose to set \( c(i \rightarrow j) = r_{ij}^2 \), where \( r_{ij}^2 \) is the squared Euclidean distance between locations \( \vec{m}_i \) and \( \vec{m}_j \).

We have reduced our inter-frame association problem to the global nearest neighbor assignment task. This problem can be solved using the polynomial time Hungarian algorithm (see [89] or Appendix A.3.2). After the simplification the solution does not require values of \( P_s, P_d, P_{fp}, \) and \( P_{fn} \), that is we have replaced all the parameters from the assignment task by a single parameter \( R_g \).
B.3 Proofs of Lemmas

Proof of Lemma 5.4

Proof. Equation 5.10 can be written as

\[ P_f(r \leq R) + \frac{N_{all}}{N_t}(P_{all}(r \leq R) - P_f(r \leq R)) = 1. \]  

(B.8)

Let \( \hat{R}_{t_{\text{max}}} \) be the first point such that Equation (B.8) holds, and let \( R_{f_{\text{max}}} \) be the first point such that \( P_f(r \leq R_{f_{\text{max}}}) = 1 \).

If \( \hat{R}_{t_{\text{max}}} \geq R_{f_{\text{max}}} \), then \( P_f(r \leq \hat{R}_{t_{\text{max}}}) = 1 \). This implies \( P_{all}(r \leq \hat{R}_{t_{\text{max}}}) = P_f(r \leq \hat{R}_{t_{\text{max}}}) = 1 \), and thus \( \hat{R}_{t_{\text{max}}} \geq R_{t_{\text{max}}} \).

If \( \hat{R}_{t_{\text{max}}} < R_{f_{\text{max}}} \) and \( \hat{R}_{t_{\text{max}}} < R_{t_{\text{max}}} \) then we have that \( P_f(r \leq \hat{R}_{t_{\text{max}}}) < 1 \), hence \( P_{all}(r \leq \hat{R}_{t_{\text{max}}}) > P_f(r \leq \hat{R}_{t_{\text{max}}}) \). We also have that

\[ \frac{N_{all}}{N_t}(P_{all}(r \leq \hat{R}_{t_{\text{max}}}) - P_f(r \leq \hat{R}_{t_{\text{max}}})) > \frac{N_{all}}{N_t}(P_{all}(r \leq \hat{R}_{t_{\text{max}}}) - P_f(r \leq \hat{R}_{t_{\text{max}}})). \]  

(B.9)

This implies \( \hat{N}_t < N_t \) which contradicts the initial statement.

We have that in all possible cases, if \( \hat{N}_t \geq N_t \) then \( \hat{R}_{t_{\text{max}}} \geq R_{t_{\text{max}}} \).

\[ \square \]

Proof of Lemma 5.5

Proof. Consider a false negative error. An error chain is a set of false negative errors from the same track and from consecutive frames. The chain must contain at least 2 errors, and two consecutive chains from one track are considered to be one chain.

Let \( n \) be the total number of occurrences of all cells in all frames in a video. Then the number of false negative errors in the video is \( n_{f_n} = P_{f_n} \cdot n \) (defined by Equation 5.5). If follows that the maximum number of chains for the video is \( \lfloor 0.5 \cdot n_{f_n} \rfloor \), and hence the probability of a false negative that starts a chain is
less than $|0.5 \cdot P_{fn}|$. Using assumption 5.1 about the detection quality, we have that the probability of a false negative starting a chain is less than 0.05.

Similar reasoning applies to false negative error. Here a chain is a set of false positive errors that are linked by NENIA.

Proof of Lemma B.1

Proof. Let $A$ be an arbitrary assignment $A$, such that there exist links $i \rightarrow j$, $i \rightarrow l$ and unassigned track $t_k$ (Figure B.1 A). Here the two links and the track implies two events: a split of track $t_i$, and a disappearance of track $t_k$, with the total probability of the two events $P_{dis} \cdot P_s \cdot P(i \rightarrow j) \cdot P(i \rightarrow l)$.

Let $B$ be the assignment derived from $A$ by removing the link $i \rightarrow l$ and adding the link $k \rightarrow l$ (Figure B.1 B). Here the division and disappearance events are replaced by two move events with the total probability $P_m \cdot P_m \cdot P(i \rightarrow j) \cdot P(k \rightarrow l)$.

We now compare $P_{dis} \cdot P_s \cdot P(i \rightarrow j) \cdot P(i \rightarrow l)$ and $P_m \cdot P_m \cdot P(i \rightarrow j) \cdot P(k \rightarrow l)$, or equivalently, we compare $P(i \rightarrow l)$ and $P_m \cdot P_m \cdot P(k \rightarrow l)$. Under assumption 5.1 about cell event probabilities we have $P_{fn} \leq 0.1$ and $P_s + P_d \leq 0.1$, hence $P \cdot P_{dis} \leq (0.1 - P_d)(0.1 + P_d) \leq 0.01$. Under the same assumption we also have $P_m \cdot P_m \geq 0.81$, thus $P_m \cdot P_m \cdot P_{dis} \geq 81$.

We have that if $P(i \rightarrow l) < 81 \cdot P(k \rightarrow l)$ then the likelihood of assignment $B$ is greater than the likelihood of assignment $A$.

Proof of Lemma B.2

Proof. Let $A'_1$ be an assignment that maximizes B.4 and $A'_2$ be an assignment that maximizes B.5, and there is a link $i \rightarrow k$ in $A'_2$. Suppose, that there is no measurement assigned to track $t_i$ in $A'$. Note that $\frac{P(i \rightarrow k)}{P_m P_{fp}} \geq 1$, otherwise link $i \rightarrow k$ would not be included in $A'_2$. This can be rewritten as $P(i \rightarrow k) > \frac{P_m P_{fp}}{P_s}$. Under assumption 1 about cell and detection events $\frac{P_m P_{fp}}{P_s} > \frac{P_{dis} P_{fp}}{P_m}$, hence $P(i \rightarrow k) > \frac{P_{dis} P_{fp}}{P_m}$. 

233
We have that, in $A'_1$, $t_i$ and $m_k$ were unassigned, but $P_m \cdot P(i \rightarrow k) > P_{\text{dis}} \cdot P_{fp}$. Hence adding a link $i \rightarrow k$ increases the likelihood of $A'_1$. But $A'_1$ is an optimal assignment. We have a contradiction.

\[
\square
\]

**Proof of Lemma B.3**

**Proof.** In this proof we denote any probability $P(i \rightarrow j)$ as $P(\text{link})$.

Consider an arbitrary assignment $A_1$ in the first round after the gating. Suppose there is a potential link that is allowed, but not included in $A_1$. Including the link to $A_1$ implies adding a move event and removing disappearance and false positive events. Thus after the link inclusion the likelihood of $A_1$ (Equation B.4) is multiplied by $T_{11} = \frac{P_m P(\text{link})}{P_{\text{dis}} P_{fp}}$. Because the link is allowed, we have that $P(\text{link}) > \sqrt{P_m P_{fp} P_s}$, hence $T_{11} > \frac{P_m}{P_{\text{dis}}} \sqrt{\frac{P_m}{P_s P_{fp} P_s}}$, and, under assumption 5.1 about cell event probabilities, $T_{11} > 1$. Hence adding the link increases the likelihood of $A_1$.

Now suppose that there are no allowed links to add to $A_1$. Notice that removing any of the links already in $A_1$ can introduce two other allowed links that can be added to $A_1$. Removing one link and adding two other links implies that the likelihood of $A_1$ is multiplied by $T_{12} = \frac{P_m P(\text{link}_1) P(\text{link}_2)}{P_{\text{dis}} P_{fp} P(\text{link}_3)}$. Note that $T_{12} \geq \frac{P_m P(\text{link}_1) P(\text{link}_2)}{P_{\text{dis}} P_{fp} P_s}$. For any allowed link we have that $P(\text{link}) > \sqrt{P_m P_{fp} P_s}$, and thus $T_{12} > \frac{P_m P_{fp}}{P_{\text{dis}} P_s}$, and under assumption 1 about cell event probabilities, $T_{12} > 1$. Hence increasing the number of links increases the likelihood of $A_1$.

Consider an arbitrary assignment $A_2$ in the second assignment round after the gating. Suppose there is a potential link that is allowed, but not included in $A_2$. Including the link to $A_2$ implies that the likelihood of $A_2$ (Equation B.5) is multiplied by $T_{21} = \frac{P_s P(\text{link})}{P_m P_{fp}}$. Because the link is allowed, we have that $P(\text{link}) > \sqrt{P_m P_{fp} P_s}$. This implies $T_{21} > \sqrt{\frac{P_m P_{fp}}{P_m P_{fp} P_s}}$. If $\frac{P_m P_{fp}}{P_s} > 1$ then $P(\text{link}) > 1$ and there must be no allowed links, which contradicts the first statement in this paragraph. If $\frac{P_m P_{fp}}{P_s} \leq 1$ then $\sqrt{\frac{P_m P_{fp}}{P_m P_{fp} P_s}} \geq 1$, hence $T_{21} > 1$, and adding the link increases the likelihood of $A_2$.

Now suppose that there are no allowed links to add to $A_2$. Notice that re-
B.4 Median Filtering

In Section 5.2.3, we propose a method for reconstruction of the CDF of cell displacements. When using this method, the reconstructed CDF in general follows the true CDF curve, however, it is subject to noise (Figure B.3, left). Low pass filtering reduces the noise but introduces distortions of sharp edges in true CDF curves. Therefore, we use median filtering (Figure B.3, right).

This filtering algorithm has only one parameter (the window length). We make this algorithm parameter-free as follows. We set the parameter of the filtering to $2W$, where $W$ is the maximum distance between two consecutive peaks (minimum or maximum) in the reconstructed CDF.
B.5 Details for Real Cell Videos

The differences in cell dynamics in our real videos were found as follows. We compared distributions of cell displacements between consecutive frames using a Kolmogorov-Smirnov (KS) test with the null hypothesis that the distributions are the same. Between each pair of videos, the null hypothesis was rejected at 5% significance level.

The normalized distributions of cell displacements (with zero mean and unit variance) also differed for all pairs of videos, except for pair hex.6 and square, where the hypothesis was rejected at the 11% significance level. This comparison shows that the distributions of cell displacements vary in their shape.

Further, in our videos, we found that in at least four of them, cells are unlikely to follow Brownian motion. Brownian motion in two dimensions (i.e., a bivariate normal distribution of cell displacements) results in a Rayleigh distribution of the cell displacements [101]. Therefore, using the maximum likelihood method, for each video, we estimated the parameters of the Rayleigh distribution. For all videos, except hex.6, the distribution of cell displacements differed from the fitted Rayleigh distribution.

For each video, we generated detections with different qualities as follows. First, we manually produced the ground truth. Next, for a given detection quality as specified by \( \{ P_{fp}, P_{fn} \} \), we calculated the corresponding number \( n_{fp} \) of spurious measurements and the number \( n_{fn} \) of missed cell occurrences. We then randomly added \( n_{fp} \) spurious measurements and removed \( n_{fn} \) cell occurrences. For each video, and each pair \( \{ P_{fp}, P_{fn} \} \) we generated 5 random versions of the detections. For the level \( \{0, 0\} \) we generated 5 random variations of the ground truth, by adding normally distributed offsets to each cell occurrence in the ground truth.

B.6 Evaluation on Synthetic Videos

In this section, we provide details for our synthetic evaluations presented in Section 5.3.

The synthetic videos of the first type consist of pairs of frames, where each
pair is generated as follows. In the first frame, we placed $n$ cells randomly with a uniform distribution in a square with side $L$. We then produced random offsets of these cells using the bivariate normal distribution with a diagonal covariance matrix having the same variance $\sigma^2$ in each dimension. In the second frame, we placed cells at the offset positions of the cells in the first frame. A synthetic video is 100 random pairs with the same set of parameters {$n, \sigma, L$}.

For the evaluation of tracking performance scalability, we used fixed values $L = 100$, $P_{fp} = P_{fn} = 3\%$, and took values of $N$ (number of cells in each frame) in the range 30 – 200. We chose $\sigma$ accordingly such that $\beta_n = 3$. We found that the tracking performance is similar for all $N$ ($P_{\text{links}} \approx 96\%$). We repeated the experiment for $\beta_n = 10$ and varied $N$ again. Again the observed performance was similar for all $N$ ($P_{\text{links}} \approx 86\%$). We found that, once the normalized density $\beta_n$ is fixed, the tracking performance does not depend on the number of cells.

For the evaluation of the running time, we used fixed values $L = 100$, $P_{fp} = P_{fn} = 3\%$ and took values of $N$ (number of cells in each frame) in the range 30 – 200. We chose $\sigma$ accordingly such that $\beta_n = 10$. We run the algorithm on a desktop PC, with the gating distance set to infinity, that is we evaluated the worst case when all possible links needed to be considered. The running time was about 1.17 seconds per frame for $N = 100$ cells, and 13 seconds per frame for $N = 200$ cells. We found that the dependency of the running time on the number of cells can be fit with a degree 3 polynomial (mean root square error is about 0.02 seconds per frame, the data is not shown).

For testing the hypothesis about the tracking performance as a function of gating distance, we used fixed values $N = 30$ and $L = 100$, and chose $\sigma$ such that $\beta_n$ takes values in the range $(0.01 \sim 100\,000) \times 10^{-2}$, i.e., covers a much wider range of values than real videos. Recall that our real videos have $\beta_n$ in the range $(0.14 \sim 5.94) \times 10^{-2}$ (Table 5.3).

For each video, we produced detections with quality levels $P_{fp} = P_{fn}$ in {$0\%, 3\%, 5\%$}. For each detection quality, we calculated the required number of spurious observations and missed cell locations and distributed these errors uniformly across all frame pairs in the video. We then used NENIA to perform inter-frame assignments on each frame pair, and report an average performance $P_{\text{link}}$ across all pairs for a given video with a given detection quality. We used
NENIA with different gating distances $R_g$, to obtain the empirical performance curve.

All obtained empirical curves (see examples in Figure 5.13, not all curves shown) preserve the prominent features of our hypothesized curve (Figure 5.6): a sharp rise from zero to maximum, a maximum around the point $R_g = R_{t_{\text{max}}}$, and a gradual decrease after this point. We found that as the detection quality decreases, the distance $R_g$ that maximizes the performance shifts further away from $R_{t_{\text{max}}}$. This can be attributed to a higher proportion of false links compared to true links on a poorer detection quality. However, in the part of the curve around $R_g = R_{t_{\text{max}}}$, the exact value of $R_g$ makes little difference. As a result, in all observed curves the difference $P_{\text{max}} - P_{\text{link}}(R_{t_{\text{max}}}) < 2\%$.

We evaluated the ability of NENIA to resolve cell divisions using synthetic videos of the second type generated as follows. Each synthetic video consists of 3 frames, where the first frame contains $N_{\text{cells}} = 100$ cells randomly placed with a uniform distribution in the square with side $L = 10,000$ units. $N_{\text{div}} = 30$ random cells out of 100 divide and there are 130 cells in the second and third frames. Between each two frames a cell that does not divide moves with a random 2D offset drawn from a bivariate normal distribution with the same variance $\sigma^2$ in each dimension. Cells that divide do not appear in the second frame. Instead, the two daughters of the divided cell appear. Each daughter is independently shifted from the mother cell’s position by a random 2D offset drawn from a bivariate normal distribution with the same variance $\sigma^2$ in each dimension. Finally, a certain number of false positive measurements are added to the synthetic video. The number of false positives depends on the false positive rate parameter $P_{\text{fp}}$. We do not add false negative errors, because in this case some divisions can be missed due to missing cell locations rather than due to the abilities of NENIA.

In our evaluation, we fixed all parameters, except $\beta_n$ and $P_{\text{fp}}$. Recall that $\beta_n$ is the normalized object density that characterizes the “difficulty” of a video (Section 5.3.3). Using the definition of $\beta_n$, we put $\sigma^2 = \frac{\beta_n L^2}{(N_{\text{cells}} + N_{\text{div}}) \pi}$.

For each $(\beta_n, P_{\text{fp}})$ pair, we generated 10 random variations of a synthetic video and ran NENIA with method B to resolve divisions. In Table B.1 we report the mean and the standard deviation for the ratios of correctly resolved
Table B.1: Ratio of correctly resolved cell divisions for synthetic videos with different parameters $\beta_n$ and $P_{fp}$. For every parameter combination, we tested 10 different synthetic videos, and report the mean and standard variation across 10 runs. For every parameter combination, the majority of cell divisions are resolved correctly.

<table>
<thead>
<tr>
<th>$\beta_n \times 10^2$</th>
<th>$P_{fp} = 0%$</th>
<th>3%</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>99.67; 1.05</td>
<td>88.67; 4.22</td>
<td>80.00; 6.48</td>
</tr>
<tr>
<td>1</td>
<td>95.00; 5.27</td>
<td>83.33; 4.16</td>
<td>80.67; 8.28</td>
</tr>
<tr>
<td>5</td>
<td>80.00; 6.08</td>
<td>71.00; 8.76</td>
<td>71.67; 6.33</td>
</tr>
</tbody>
</table>

divisions observed on 10 videos. We found that the ratio of correctly resolved divisions depends on both parameters $\beta_n$ and $P_{fp}$. Further, we found that for every parameter combination, the majority of cell divisions are resolved correctly.
Appendix C

Ranking Cell Tracking Systems

In this appendix, we present proof sketches, additional details and supporting arguments for Chapter 7 (ranking cell tracking systems).

C.1 Cell Tracking Performance

In this section, we reflect on two types of measures for cell tracking performance.

C.1.1 Measures that require ground truth

There have been several proposed tracking performance measures, such as the proportion of swap errors (Section 5.2.1), and the proportion of lost tracks [79]. Informally, a swap error is a situation where the same cell in consecutive frames is given different labels by the tracker. The proportion of swap errors is defined as $\frac{\text{# swaps}}{\left| D \right|}$, where $\left| D \right|$ is the number of measurements in the detection $D$. The proportion of lost tracks measures how many tracks were not completely recovered throughout the video (i.e., without any errors). In Chapter 7, we also define a measure that is based on F-score.

Each of the three measures has its limitations. For example, F-score cannot distinguish between correct and incorrect track recovery after a missing detection (Figure C.1(a, b)). The proportion of swap errors can be insensitive to small
improvements in the tracking algorithm (Figure C.1c, d). The proportion of correct tracks is even less sensitive to improvements in the tracking algorithm, because a single inter-frame error invalidates the whole track.

Rather than trying to identify any of the measures as being superior, we note that all measures are related. For example, a spurious link or an overlooked link implies a swap error. Furthermore, a swap error generally results in a wrong track. Therefore, it is not surprising that in our evaluation we find correlation between different measures (Section 7.5).

Finally, we note that the limitation of F-score as illustrated in Figure C.1 (a, b) is unlikely to hamper the accuracy of F-score in practice, as state-of-the-art detection algorithms tend to make few “missing cell” errors [79, 97]. Furthermore, it is reasonable to expect that the ability of a tracker to recover tracks after missing detections correlates with the ability of the tracker to make correct inter-frame assignments. Therefore, as confirmed by our evaluation, the estimation of precision and recall can be a feasible method for estimating tracking performance.

C.1.2 Measures that do not require ground truth

In Section 7.3, we introduce the MP and MR-scores as proxies for precision and recall. Since our scores have been motivated by F-score which is the harmonic mean of the precision and recall, the apparent solution to derive a single measure is to use the harmonic mean of MP and MR-scores.

However, note that the original F-score supports the principle of decreasing marginal effectiveness [116]. According to this principle, it should be increasingly more difficult to compensate the loss of precision as the absolute value of the loss increases (the same applies to the recall). Consider an arbitrary pair \((\text{prec}, \text{recl})\) and its score \(F_1(\text{prec}, \text{recl})\). Now suppose that we decrease the precision and compensate this by increasing the recall so that

\[
F_1(\text{prec}, \text{recl}) = F_1(\text{prec} - dp, \text{recl} + dr). \tag{C.1}
\]

This equation is satisfied by a set of points \((dp, dr)\). Alternatively this set can
C.1. CELL TRACKING PERFORMANCE

Figure C.1: Limitations of F-score (a, b) and the proportion of swap errors (c, d) as measures for tracking performance. In all frames, the same numbers label the same cells, and letters show the labeling according to a cell tracker. (a, b) F-score is unable to distinguish between correct and incorrect track recovery, after a missing detection occurred in the middle frame. In the examples (a, b), the 3 frames constitute only a part of the video, and F-score is determined by the tracking on the rest of the video. In both cases, F-score can have the same value, denoted as $X$. (c, d) The proportion of swap errors can be insensitive to small tracking improvements. In panel (d), the tracker decided that cell B divides into cells $B_1$ and $B_2$. In both cases, there are two swap errors, but the numbers of spurious links (red solid lines), and overlooked links (dotted lines) differ. At the same time, one can argue that the tracking in panel (d) is slightly better as it makes fewer spurious links and overlooks fewer correct links. The thick line shows a correctly made link.
be viewed as a function \( dr = \psi(dp) \). The principle of decreasing marginal effectiveness states that the function \( \psi \) must be super-linear (e.g., quadratic or exponential). A similar definition applies to the recall. For simplicity, we ignore constraints \( \text{prec} - dp, \text{recl} + dr \in [0,1] \).

If the above principle does not hold then it is always possible to sacrifice \( dp \) units of precision at the cost of \( dr < dp \) units of recall. If gaining each unit of precision is equally “difficult” as gaining each unit of recall, then it is always beneficial to consider trackers with \( \text{prec} = 0 \) or \( \text{recl} = 0 \). In order to avoid this situation, one requires the combined measure to support the principle of decreasing marginal effectiveness.

Since our mirrored scores have the opposite semantics to the original precision and recall (i.e., for our scores, less is better), the above principle should be applied to the transformed coordinate axes corresponding to \(-MP\) and \(-MR\). After substituting \((-MP)\) and \((-MR)\) into the formula for the harmonic mean and performing an algebraic simplification, we obtain a possible combined measure that does not require a ground truth:

\[
MF = \frac{-MP \cdot MR}{MP + MR}
\]  

(C.2)

In the main text, we introduce an alternative approach which is called the ED-score and defined as the Euclidean distance \( ED = \sqrt{MP^2 + MR^2} \). Note that ED-score satisfies the principle of decreasing marginal effectiveness.

We evaluated the performance of both MF and ED-scores. Both measures perform well, while ED-score slightly outperforms MF-score. Moreover, unlike MF-score, ED-score is defined at the point \( MP = MR = 0 \), and ED-score has a convenient geometrical interpretation. Therefore, we suggest to use ED-score as the combined measure that does not require ground truth.

Essentially, ED-score is the quadratic mean with equal weights given to the MP and MR-scores. The use of equal weights is partly justified by the fact that both the MP and MR-scores have the same scale: both MP and MR are probabilities divided by the same constant. Refer to Equation 1 in the derivation of MP-score, and note that MR-score is also defined by the same equation. An alternative justification for equal weights can be derived by treating the weights.
as uninformative priors. Without a ground truth there is no apparent way to derive these weights. Therefore we can set the weights to be equal according to the principle of indifference.

C.2 Analysis of the VN-score

An intrinsic limitation of the VN-score can be illustrated as follows. Consider an inter-frame association between frames $i-1$ and $i$. Between these frames, let the number of links made by a cell tracker be $n_i$, and let the number of true links be $n_{True_i}$. Note that the value of $n_{True_i}$ is determined by the ground truth for the video, and $n_{True_i}$ is not related to the tracker. Furthermore, let the number of spurious links made by the tracker be $n_{Spur_i}$ and the number of missed links be $n_{Miss_i}$. Then we have that $n_i = n_{True_i} + n_{Spur_i} - n_{Miss_i}$. If we now consider all frames in the video then $n_i$, $n_{True_i}$, $n_{Spur_i}$, and $n_{Miss_i}$ can be viewed as random variables. Here $n_{Spur_i}$ and $n_{Miss_i}$ can be dependent. For example, if $n_{Spur_i} \approx n_{Miss_i}$, then $VN = Var(n_i) \approx Var(n_{True_i})$. In this case, VN-score depends only on $Var(n_{True_i})$ which characterizes the video, but not the tracker, and hence, in this case, VN-score does not reflect the performance of the tracker.

At the same time, the VN-score works well in our real video experiments, but it does not perform well in the synthetic video experiments. For a given video, and hence a fixed $Var(n_{True_i})$, a high VN-score indicates high variance in $n_{Spur_i}$ and $n_{Miss_i}$, which can be regarded as poor tracking performance. In contrast, a low VN-score does not necessarily imply low variance in $n_{Spur_i}$ and $n_{Miss_i}$. In our experiments on real videos, real trackers tend to have $n_{Spur_i} \approx 0$, and this explains why VN-score is well correlated with F-score. However, our synthetic videos represent harsher tracking conditions, where for different trackers both $n_{Spur_i}$ and $n_{Miss_i}$ vary. In these experiments, the correlation between VN-score and F-score is weaker.
C.3 Proof Sketch for Proposition 7.1

Let \( S_i \) be the set of all possible trackers that yield outputs with \( N_i \) links and that all have \( T_i \) true links in their output. We choose two arbitrary such sets and without loss of generality index them with \( i = 1, 2 \). Now let \( \mathcal{E}P_i = E[MP(\mathbb{L}_i)] \) be the expected value for the MP-score, where the expectation is taken over all possible outputs \( \mathbb{L}_i \) from trackers in \( S_i \). From Equation 7.1 (main text) we have that

\[
\mathcal{E}P_i = \frac{1}{N_i} \left( \sum_{t \in T_i} E[\zeta(R_t)] + \sum_{f \in F_i} E[\zeta(R_f)] \right). \tag{C.3}
\]

Here \( \zeta(R_t) = \frac{P_t(R_t)}{P_{all}(R_t)} \) is a summation term from Equation 7.1; \( t \) and \( f \) enumerate individual links in \( \mathbb{L}_i \), and \( T_i \) and \( F_i \) are the sets of indices for true and false links respectively. The expectations are taken over all possible outputs \( \mathbb{L}_i \) from trackers in \( S_i \). For example, \( E[\zeta(R_1)] \) is the expected value of \( \zeta(R_1) \) where \( R_1 \) is the random variable denoting the length of the first link in the outputs from trackers in \( S_i \).

Given a tracker \( s \in S_i \) there exists another tracker \( s^* \in S_i \) that takes the result of \( s \) and permutes it. Consequently, for any two indices \( k, l \in T_i \), we have that \( E[\zeta(R_k)] = E[\zeta(R_l)] \). The same reasoning applies to indices from \( F_i \). We can now let \( \mathcal{E}T_i = E[\zeta(R_t)], t \in T_i \) and \( \mathcal{E}F_i = E[\zeta(R_f)], f \in F_i \), and simplify Equation C.3 to

\[
\mathcal{E}P_i = \frac{1}{N_i} \left( T_i \cdot \mathcal{E}T_i + (N_i - T_i) \cdot \mathcal{E}F_i \right) \tag{C.4}
\]

Note that \( \mathcal{E}T_i = \int_0^{\infty} P(R_t = r) \zeta(r) \, dr, t \in T_i \). Further, assuming that the trackers in \( S_i \) are equally likely (according to the principle of indifference), we have that \( P(R_t = r) = P_t \). We can then put \( \mathcal{E}T = \mathcal{E}T_1 = \mathcal{E}T_2 \). The same reasoning applies to false links, and so we can put \( \mathcal{E}F = \mathcal{E}F_1 = \mathcal{E}F_2 \).

Proposition 7.1 states a negative correlation between the MP-score and the precision \( T_i/N_i \). Alternatively this can be formulated as a condition

\[
\mathcal{E}P_1 > \mathcal{E}P_2 \iff \frac{T_1}{N_1} < \frac{T_2}{N_2}. \tag{C.5}
\]
C.4. LENGTHS OF DUMMY LINKS

From Equation C.4, we have the difference

\[ \mathcal{E}P_1 - \mathcal{E}P_2 = \frac{T_1N_2 - T_2N_1}{N_1N_2} \cdot (\mathcal{E}T - \mathcal{E}F). \tag{C.6} \]

For Proposition 7.1 to hold, we require \( \mathcal{E}F > \mathcal{E}T \), which can be further expanded as

\[ \mathcal{E}F - \mathcal{E}T = \int_0^\infty \zeta(r) \cdot (P_f(r) - P_t(r)) \, dr > 0. \tag{C.7} \]

The left part of the inequality can be written as \( \int_0^\infty \frac{P_f(R_j) \cdot (P_f(r) - P_t(r))}{a \cdot P_f(r) + (1 - a) \cdot P_t(r)} \cdot dr, 1 > \alpha > 0 \). Here we assume \( P_t(r) \neq P_f(r) \). We first define \( S_0 \equiv \int_0^\infty \frac{P_f(R_j) \cdot (P_f(r) - P_t(r))}{1 \cdot P_f(r) + \gamma \cdot P_t(r)} \cdot dr = 0 \). We now express \( \alpha \) as \( \alpha = 1 - \gamma, 1 > \gamma > 0 \), and rewrite the inequality as

\[ S \equiv \int_0^\infty \frac{P_f(R_j) \cdot (P_f(r) - P_t(r))}{1 \cdot P_f(r) + \gamma \cdot P_t(r)} \cdot dr = 0. \]\n
In \( S \), the integrand is positive when \( P_f(r) > P_t(r) \), which corresponds to \( \gamma \cdot (P_t(r) - P_f(r)) \leq 0 \). Therefore, on the intervals where the integrand of \( S \) is positive, it is greater than or equal to the value of the integrand of \( S_0 \) on the corresponding points. Similarly it can be shown that on the intervals where the integrand of \( S \) is negative, its absolute value is smaller than or equal to the value of the integrand of \( S_0 \) on the corresponding points. Given that \( P_t(r) \neq P_f(r) \), we have that \( S > S_0 = 0 \) and hence \( \mathcal{E}F - \mathcal{E}T > 0 \). Therefore condition C.5 is satisfied and Proposition 7.1 holds. \( \square \)

C.4 Lengths of Dummy Links

In this section, we justify the proposed selection of lengths for dummy (padding) links introduced in Section 7.3.3. Let \( S_i, i = 1, 2 \), be the set of all possible trackers that yield output with \( N_i \) links and that all have \( T_i \) true links in their output. Padding involves adding \( \delta N_i \) links to the outputs of trackers from \( S_i \). Note that \( (N_1 + \delta N_1) = (N_2 + \delta N_2) \).
Now let $\mathcal{E}R_i = E[MR(I_i)]$ be the expected value for the MR-score, where the expectation is taken over all possible outputs $I_i$ from trackers in $S_i$. From the definition of MR-score (Equations C.1 and C.2), we have that

$$\mathcal{E}R_i = \frac{1}{N_i + \delta N_i} \left( \sum_{t \in T_i} E[\zeta(R_t)] + \sum_{f \in F_i} E[\zeta(R_f)] + \sum_{d \in D_i} E[\zeta(R_d)] \right). \quad (C.8)$$

Here $\zeta(R_j) = \frac{p_j(R_j)}{p_{all}(R_j)}$ is a summation term from Equation 7.1; $t$, $f$, and $d$ enumerate individual links in $I_i$, and $T_i$, $F_i$, and $D_i$ are the sets of indices for true, false and dummy links respectively. The expectation is taken over all possible outputs $I_i$ from trackers in $S_i$. For example, $E[\zeta(R_1)]$ is the expected value of $\zeta(R_1)$ where $R_1$ is the random variable denoting the length of the first link in the outputs from the trackers in $S_i$.

Following similar reasoning as in Appendix C.3, we can have $\mathcal{E}T = \mathcal{E}T_1 = E[\zeta(R_k)] \approx \mathcal{E}T_2 = E[\zeta(R_i)], k \in T_i, l \in T_2$. The same applies to $\mathcal{E}F$, and we can simplify

$$\mathcal{E}R_i = \frac{1}{N_i + \delta N_i} \left( T_i \cdot \mathcal{E}T + (N_i - T_i) \cdot \mathcal{E}F + \delta N_i \cdot \mathcal{E}D_i \right). \quad (C.9)$$

We want the MR-score to have a negative correlation with the number of true links (denoted as $T_i$) in the trackers’ outputs, and therefore we want to satisfy

$$\mathcal{E}R_1 > \mathcal{E}R_2 \iff T_1 < T_2. \quad (C.10)$$

So the problem is to select the lengths of dummy links such that with the corresponding $\mathcal{E}D_i$ in Equation C.9, condition C.10 is satisfied.

To simplify the problem we find the lengths of dummy links under the constraint $\mathcal{E}D = \mathcal{E}D_1 = \mathcal{E}D_2$. Further, let $(N_1 + \delta N_1) \cdot (\mathcal{E}R_1 - \mathcal{E}R_2) = K$. We now have that

$$K = (T_1 - T_2) \cdot (\mathcal{E}T - \mathcal{E}F) + (N_1 - N_2) \cdot \mathcal{E}F + (\delta N_1 - \delta N_2) \cdot \mathcal{E}D \quad (C.11)$$

248
Note that if we set $ED = EF$ then Equation C.11 simplifies to

$$K = (T_1 - T_2) \cdot (ET - EF).$$

From Appendix C.3, we have that $ET - EF < 0$, and hence having $ED = EF$ satisfies condition C.10. We therefore suggest to generate the dummy links by drawing their lengths from the PDF of false links $P_f$, so that the expected value of $\zeta(R)$ for dummy links satisfies $ED = EF$. 

249
Appendix D

Motility of Plasmodium Ookinetes

In this Appendix, we present the details of the motility assays conducted in Chapter 8. Sections D.1 and D.2 of this Appendix are largely based on the description provided by Yan Hong Tan from the Baum Lab at the Walter and Eliza Hall Institute, who conducted the experiments.

D.1 Parasites and Mice

Six to eight week-old BALB/c or Swiss mice were used throughout the study. They were kept in a room with temperatures around 22°C and 25°C, and a 14-hour light/10-hour dark cycle or 12-hour light/12-hour dark cycle. All mice were fed on standard mouse chow and drinking water. The mice were treated with phenylhydrazine 2/3 days prior to infection by intraperitoneal inoculation with the P. berghei ANKA expressing green fluorescent protein (GFP) CON [44] obtained from a donor mouse between the first and sixth passage from cryopreserved stock. Parasitemia monitored on Giemsa stained blood films and exflagellation was quantified 3 days post infection by adding a drop of blood from a tail vein to 20 µL ookinetes medium. The percentage of gametocymia and number of exflagellation were assessed in 10 viewing fields. Infected blood was collected from mice with a parasitemia more than 4% via cardiac puncture into a heparinised syringe. Ookinetes were produced in vitro by culturing in T25 culture flasks with ookinetes media at 19°C for 18 hours. All experiments were
conducted in compliance with local Animal Ethics Committee requirements.

D.2 Motility Assay

The Giemsa staining was performed to check the conversion and quantity of ookinetes prior to the setup of the motility assay. Ookinetes were added into 96-well imaging plates (with clear polystyrene bottom, BD Bioscience) pre-coated with Matrigel (BD Bioscience) diluted 1:1 with ookinetes medium. The parasites were maintained at 19°C for 1 hour. After 1 hour incubation, the drug candidate was added onto the plates and incubated at 19°C for another 1 hour. 1% DMSO was the negative control for all the experiments. The imaging was performed on either a live cell observer Zeiss microscope or a line-scanning confocal LSM 5 live Zeiss microscope in an air-conditioned imaging suite at room temperature.

For the low definition imaging, the live cell observer microscope was used. The time-lapse video was collected with a Zeiss Axiocam HRm 1 frame every 5 seconds for 10 minutes using a 20X objective lens and 5% LED light source. The videos were controlled by Axiovision software. The videos were imported to Imaris (Bitplane) for analysis.

The confocal microscope was used for assessing the high definition imaging. The time-lapse video was collected with a Zeiss Axiocam HRm 1 frame every 2 seconds for 10 minutes using a 20X objective lens and Zen software. The laser used was 5% for all the videos.

All videos were imported to Imaris (Bitplane) for tracking the movement of ookinetes. The setting we used in Imaris was spot tracking with the diameter of the spot (which correspond to an ookinete) is around 5 µM, and automatically tracking the motile ookinetes in the time-lapse videos using Brownian random motion. The speeds of ookinetes were recorded by the Imaris software and were exported into Excel (Microsoft) sheet format. The figures were generated using the MATLAB and Prism software.
Bibliography


MICCAI Workshop on Microscopic Image Analysis with Applications in Biology, New York, New Jersey, USA, 2008.


[65] T. Inglis, H. De Sterck, G. Sanders, H. Djambazian, R. Sladek, S. Sundararajan, and T. J. Hudson, “Multilevel space-time aggregation for bright


[83] N. Malpica, C. Ortiz de Solorzano, J. Vaquero, A. Santos, I. Vallcorba, J. Garcia-Sagredo, and F. del Pozo, “Applying watershed algorithms to


264


