Imperial College London
Department of Computing

Learning to Automatically Detect and Track Cells in Microscopic Imaging

by

Pedro Damian Kostelec

September 2014

Supervised by Ben Glocker

Submitted in part fulfilment of the requirements for the
MSc degree in Computer Science (Artificial Intelligence) of Imperial College London

Abstract

Automatic cell detection and tracking for *in vivo* microscopic image sequences enables the efficient analysis and quantification of cell behaviour in their natural environment, thus improving the study of drugs and our understanding of living beings. The technical achievements in *in vivo* image acquisition in the past few years have enabled us to study cell behaviour in a much more detailed manner without dramatically changing their natural environment. This research focused on the detection and tracking of cells in microscopic images that often exhibit motion artifacts, blurred frames, obscured cells and background noise. In fact, by using current cell detection and tracking methods, it would not have been possible to efficiently analyse these images.

To achieve this, the author improved a cell detection method that learns to classify non-overlapping candidate regions as cell or not-cell. The computation time of the original method has been significantly reduced to process each frame (of dimensions around 512-by-512 pixels) between 0.5 and 1.5 seconds. This method achieves high precision and recall values on the studied datasets and can be trained with dot-annotations – each dot corresponding to a cell. In some cases it is possible to use a pre-trained detector to detect cells in a new, previously unseen dataset and still achieve good detection rates.

The author also developed a tracking method that attempts to link the centroids of the detected cells into trajectories. The method relies on a global data association approach to reliably generate trajectories based on a global decision. Within the context of this research, previous methods have been improved by approaching the problem of defining likelihoods of linking tracklets from a machine learning point of view. Because the likelihoods are obtained directly from training examples, it was possible to produce good results even on low quality images, where several frames in a sequence can become out-of-focus, and in which cells can disappear and reappear over time, etc.

Overall, the combination of these two data driven algorithms for cell detection and tracking has been shown to be a promising approach to automatic cell tracking. The system has shown good results on the studied datasets. Within the scope of this dissertation, the author has also presented various possible additional improvements to the methods, which, if applied, might even further improve their performance.

Acknowledgement

I would like to acknowledge a number of individuals that have supported me throughout this project or in some way or another contributed to the development of the automatic cell tracker presented in this report.

First, I give my sincerest gratitude to my supervisor, Ben Glocker, for his continuous feedback, guidance and contagious motivation for the project.

Second, I would like to thank Dr. Leo Carlin for seeding the initial idea for the project as well as providing all the image sequences used to train and evaluate the system. I would also like to thank him for carefully reviewing the training datasets.

Third, I give my thanks to my friend Diego Aranda Pérez for his help in annotating the training image sequences as well as helping me stay focused, especially in the initial weeks of the project.

Fourth, my friend Stefan McCarthy for proofreading this report.

Fifth, Carlos Arteta for making publicly available the original code from his research. His cell detection code has been reused in this project and allowed me to focus on the development of the cell tracking module.

Most of all, I give my thanks to my parents and brothers, for their guidance and support during the first two decades of my life.

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.

Contents

1. Introduction

In this introductory chapter we describe the problem of tracking cells in microscopy images manually and motivate the development of an automatic cell tracker. We also outline the objectives and contributions of the project and provide a report outline for the following chapters.

1.1. Motivation

Recent advances in intravital microscopy enable us to study the behaviour of different cells without excessively modifying the natural environment in which these cells are found within the observed organism.

Neutrophils are a type of leukocyte that have a crucial role in the clearance of infections [1]. A significant reduction in the number of neutrophils in the human body (or in mice) leads to severe immunodeficiency or death. They can be found in the bone marrow, liver, spleen, lung and throughout the circulatory system. Direct observation of neutrophils should help explain their function in these organs.

Of special interest is the transit of neutrophils in the fine and specialized capillary network located in the lung. To transit, the neutrophils must come in contact with the endothelium of these small vessels. In order to protect against pathogens, neutrophils are potently cytotoxic. Therefore, their regulation in the lung is important as the consequences of their unwarranted activation would be severe.

A recent study [2] has shown that the lifespan of neutrophils is much longer than previously predicted (up to 12.5 hours for mice and 5.4 days for humans). Although this specific study is a source of doubtful criticism [3], the longevity of neutrophils increases during inflammation. This longer lifespan may permit them to perform a wider range of complex activities, beyond the clearance of infections. An analysis of their behaviour as observed through intravital microscopy could help reveal more of their roles.

Finally, there is accumulating evidence to support the existence of different lineages of neutrophils with discrete roles. Automated analysis of microscopic images could explain whether these are truly distinct lineages or if they instead all develop from the same neutrophil predecessor.

The observation of neutrophils requires the analysis of hundreds of frames of microscopy image sequences. The manual annotation of these image sequences in order to extract trajectories of

movement of neutrophils is a time consuming and error prone process. Manual annotation severely limits the amount of data that can be analysed and slows down the advancement of cell research. It would be a major advance to be able to rely on the automatic identification and tracking of cells over time in sometimes noisy complex images.

1.2. Objectives

The clarity of neutrophil image sequences obtained *in vivo* can vary considerably due to the combination of the raster scanning imaging technique and, in the case of the lung, the motion of the tissue induced by a mechanical ventilator. In the images we can observe artifacts such as out-of-focus frames and blurred images due to the jiggling tissue, etc. This can sometimes make it difficult to identify and segment individual cells.

The aim of this research is to develop methods for cell detection and tracking. This system should be able to accept image sequences of cells in tissue, identify the cells, and track their position over the entire sequence.

The identification of cells should take into account the nature of the imaging technique and the quality of the studied images. Simple methods such as thresholding would be too unreliable; more advanced methods need to be investigated.

Tracking of cells should also take into account the nature of input data. Basic frame-by-frame tracking of cells is likely to have poor results because the cells frequently disappear into the depths of tissue or lose focus. Methods that take into account the temporal behaviour of cells are likely to result in more robust tracking.

1.3. Contributions

The work in this project led to a development of a robust pipeline for automatic cell tracking. The pipeline combines a cell detection algorithm from [4] and, to the best knowledge of the author, a novel tracking method that is directly based on the observed data.

The cell tracking module uses a global data association approach to reliably generate cell detection trajectories based on a global decision. This research has improved previous methods allowing them to rely on a large set of features obtained from observed data. A machine learning approach is used to compute likelihoods for linking cell detections into increasingly longer trajectories. Because the likelihoods are obtained directly from training examples, the method is able to produce good results even from noisy datasets, where several frames in a sequence can become out-of-focus, and in which cells can disappear and reappear over time, etc.

Finally, an image annotation tool is developed that allows simple dot annotation of cells and linking of cells among frames to represent trajectories. This tool can be used for generating the necessary

training data and further allows to visualise and inspect the results produced by the automatic cell tracking method.

1.4. Report structure

The rest of the thesis is structured as follows:

Chapter 2 consists of brief literature survey outlining existing methods for cell detection and tracking. The chapter gives a background of how the methods for cell detection and tracking have evolved over time.

Chapter 3 describes in more depth the cell detector from Arteta *et al.* [4] and outlines the modifications that improved the detection speed.

Chapter 4 describes in detail the cell tracking module. It outlines the two step process of generating robust tracklets and then joining them into long trajectories using trained classifiers.

Chapter 5 represents an overview of example datasets upon which the tracking tool is tested and presents a cell annotation tool developed to ease the annotation of image sequences.

Chapter 6 quantitatively and qualitatively evaluates the performance of the cell detection and tracking modules. Experiments used to test the methods and their results are provided.

Chapter 7 includes some concluding remarks and ideas that could be implemented to continue the advancement in the field of automatic cell detection and tracking.

2. Related work

This chapter consists a survey of current methods for cell detection and tracking. It summarizes the different techniques in a structured way, and discusses the advantages and disadvantages of each method. The chapter is divided into three sections. Section 2.1 describes methods to perform cell detection on images containing cells. Section 2.2 presents methods used to track cells in a sequence of images. Finally, in section 2.3 we describe which methods seem most promising for our application of cell tracking.

2.1. Cell detection

Cell detection consists of identifying individual cells in microscopy images. Due to the different imaging techniques and the types of cells we may wish to detect, several algorithms have been developed, each to handle a specific case. This section is an overview of these techniques.

2.1.1. Cell segmentation using the Watershed technique

A basic cell detection method relies on binarizing an image to separate the background from the cells, followed by a segmentation step to extract the cells. Chen, Biddell *et al.* [5] use a flooding approach for segmentation. The entire method can be summarized as follows. First, a spatial adapter filter eliminates some noise in the image. Second, it locates the pixels with minimum intensities in a small sliding window. Third, for each minimum point it proceeds to the progressive flooding of its neighbouring points and a final post-processing step discards false regions.

A similar method by Chen *et al.* [6] consists on using a Watershed algorithm [7] to separate cell nuclei after using Otsu's thresholding method to segment nuclei from the background. Morphological operations [8] can be used to fill holes and eliminate patches that are too small to correspond to cells. Additionally they developed a nuclei-fragment merging method based on the shapes and sizes of the nuclei to deal with the problem of over-segmentation caused by the Watershed algorithm.

Disadvantages of these methods include the requirement for a similar number of pixels belonging to both the cells and background as well as the high signal to noise ratio.

2.1.2. Cell segmentation using level sets

Another interesting technique to segment cells is a contour evolution method that makes use of the general appearance of cells to segment them using level sets. Mukherjee *et al.* [9] make the observation that leukocyte shapes are nearly circular and that at least for a significant part of the border of the cells, the intensity profile is different from the cell cytoplasm as well as the background. Using this observation, identification of a leukocyte is formulated as a minimization of an energy function incorporating image gradient and intensity homogeneity within the closed contour encompassing the cell. The benefit of this method is that it can be adjusted to perform well in images with significant clutter and poor contrast by increasing the importance of the homogeneity, or for images with good contrast, where the gradient magnitude term is given more importance. The disadvantage of this method is that cells cannot overlap. The energy function can be minimized with the gradient descent method.

To reduce the solution space for the energy function, only the boundaries of connected components within the image-levels sets are evaluated with the energy function. Only the connected components satisfying the size and shape constraints of the cells are extracted. The remaining components are eliminated using area morphology operations. This level-set analysis provides a more efficient solution that is linear in the number of intensity levels in the image in contrast to the much higher complexity of a curve evolution method.

The level set method is a contour evolution approach which has good results in segmentation. Tang *et al.* [10] have successfully combined level-sets and local grey thresholding [11] for segmenting neuron stem cells in images which have been obtained by confocal microscopy.

2.1.3. Cell detection by model learning

The previous methods perform efficiently in cells with sufficiently good contrast. In images where the cell borders are unclear, images are of varying intensity, cell density is high, or cells represent different shapes, these methods may not perform as well. In such cases machine learning methods can perform better by learning a model of a cell based on a training set of annotated examples.

Arteta *et al.* [4, 12], propose an algorithm that uses a highly-efficient MSER region detector [13] to find a broad number of candidate regions. These regions are then scored depending on the similarity to the cell type of interest by a machine learning algorithm.

The authors organized the extremal regions obtained from the MSER detector into trees, so that each tree corresponds to a set of overlapping extremal regions. Each region is given a value corresponding to the similarity of the region to a cell. The non-overlapping regions which achieve high scores can then be selected via dynamic programming of the trees. The learning is performed using a structured SVM [14] which is able to take into account the non-overlap constraint, and achieves good performance. The learning is performed on weakly annotated images – a single dot is necessary on each cell.

The advantage of this approach is the tolerance to changes in image intensities, cell densities and sizes. The major downside is the non-overlap constraint. Fortunately the authors have also developed an algorithm to detect partially overlapping cells [12].

The idea is to learn to detect overlapping cells together with the number of cells in the region. The algorithm starts by generating a set of nested regions. Each region is then scored using a set of classifiers that evaluate the similarity of the region to each of the possible classes, where each class corresponds to the number of cells that the region contains. An inference procedure then selects the non-overlapping subset of regions, and assigns each a class label indicating the number of cells that the model believes lie in the region. This eliminates the non-overlap constraint, but requires a larger training set of annotated images to train the detector.

2.2. Cell tracking

After detecting each cell in each image of the sequence we need to determine which cells in an image correspond to which in the next image. This way we can generate tracks of the cells across the image sequence. We have identified four main approaches to tracking:

Tracking by model evolution Active contours or level sets are used to detect a cell in the first frame of the sequence, and the cell models are then propagated to the next frame.

Tracking by data association Cells in consecutive frames are associated according to a similarity measure, which compares features such as the cell intensity and the relative spatial location.

Tracking with a dynamics filter This method uses a dynamics filter (e.g. Kalman or Particle filter) to predict the spatial location of the cell in the following frames.

Tracking by global data association This method approaches tracking as a global optimization problem and all trajectories are generated at once. These methods aim to maximize the probabilities that the generated trajectories are correct.

The remaining of this chapter briefly describes how these methods have been implemented in different papers.

2.2.1. Tracking by model evolution

This approach consists of identifying a cell in the first frame of a sequence by means of active contours or level sets and then propagating the models to the next frame. Model evolution tracking methods handle cell deformation well, but can often get stuck in local minima. This method can be computationally expensive because of the need to evolve the cell models for each frame. It is possible to improve the tracking accuracy at the expense of computation time.

Mukherjee *et al.* [9] model the problem of cell tracking as maximizing a similarity measure between

level sets in consecutive frames. The method minimizes the energy associated with leukocyte boundary detection and then matches the boundary with that of the previous frame. To maintain spatial coherence, the authors assume that the displacement of cells between frames is marginal. Such an automatic tracker is able to handle slight rotations or deformations of the leukocytes well.

2.2.2. Tracking by frame-by-frame data association

When tracking by frame-by-frame data association, cells in consecutive frames are associated according to a similarity measure, which compares features such as the cell intensity or relative spatial location. This method is very efficient, but only effective when cells are accurately detected. It is possible to improve the association by analysing several frames in a sequence.

Chen *et al.* [6] perform the matching process based on the distances between the cells. A match is found if a feature distance is below a threshold. The authors have also developed strategies to overcome the problem of false matches and ambiguous correspondence when nuclei are touching or partially overlapping. In this case information about these nuclei is stored (nuclei size and their relative location – up, down, left, right –). When they separate, the algorithm can resolve the ambiguities by comparing their current status with the previously stored information.

To reduce the vulnerability of frame-by-frame data association it is possible to analyse several frames of the sequence. This is the approach by Huh [15]. For each new frame, a set of hypotheses is computed for each detected blob, corresponding to migration, exit, entrance, clustering and mitosis. After all hypotheses are obtained, a best combination is selected by formulating and solving an integer programming problem. If, after the best association is found, there are remaining cells, these are considered again in the following frames.

The similarity function used to perform data association between consecutive frames is of crucial importance. House *et al.* [16] propose a novel cost function that encodes the response of cells to conditions of their environment. The tracking problem is formulated as a standard Bayesian filtering problem. The model allows the state of the cells in the images to evolve in time. To solve the complete assignment they have used a probabilistic data association algorithm which produces an optimal solution.

2.2.3. Tracking with a dynamics filter

Dynamics filters are a powerful addition to tracking systems because they predict the motion of a cell, and reduce the search space in which we look for matches. These algorithms not only perform frame-by-frame tracking, but also take temporal information into account, resulting in models that are robust to long-term occlusion and cell detection error.

Xu *et al.* [17] developed an ant stochastic searching behaviour based tracking system to track multiple cells in fluorescent image sequences. The system is similar to particle filters, but the motion behaviour is modelled to be similar to how ants behave when searching for food. When ants move

around, they deposit small quantities of chemical pheromone. Once they find food, they retrace the steps deposing larger amounts of pheromone. Other ants can then use this guide to find food faster, but are allowed to deviate from the path and pursue their own paths. The author proposes a system where the food is not static, so that it can be used for cell tracking. Compared to particle filters, this algorithm is more computationally expensive, but achieves better accuracy. The algorithm is able to deal with cells entering or leaving the scene as well as occlusion.

Kalman filters are another popular method for state prediction. Tang *et al.* [10] use a Kalman filter to guide the tracking of active cells. Active cells are those that move a distance larger than a threshold (supposedly their radius) between frames. Inactive cells are tracked separately by observing their overlap region.

Cells do not always exhibit just one type of motion and a single Kalman filter cannot take this into account. Li *et al.* [18] propose an automated tracking system which runs several Kalman filters in parallel, each corresponding to a different modality: random walk, first-order and second-order linear extrapolations which correspond to Brownian motion, migration with constant speed and migration with constant acceleration. The Kalman filters are run by the interacting multiple models (IMM) filter. The transitions between models is determined by a finite state Markov chain. Their system is composed of five modules: cell detector, cell tracker, dynamics filter, track compiler and track linker. The cell detector separates the background from cell pixels. The cell tracker propagates the cell labelling to the next frame in the sequence. The track compiler compares the results of the cell tracker, cell detector and dynamics filter to create a new track for new or daughter cells, and updates or terminates existing tracks. The track linker connects two or more track segments if they belong to the same cell.

2.2.4. Cell tracking by global data association

The benefit of global data association approaches is that they aggregate the results of all frames and make a global decision rather than propagating the results of each frame to the next. This makes them less dependent on errors from the cell detection module.

Massoudi *et al.* [19] propose a tracking method that does not rely on perfect segmentation and can deal with uncertainties by exploiting temporal information and aggregating the results of many frames. The system only requires probabilities or potentials that represent cell positions. Tracking is modelled as a network flow problem and is formulated as a linear program based on the occupancy likelihood of the edges of the network. The system can detect false positives or false negatives and correct itself. The method handles division events and cells entering or exiting the screen at the boundaries.

Zhang *et al.* [20] also researched tracking of pedestrians using network flows and achieved superior performance to frame-by-frame methods. In their approach, data association is defined as a maximum a posteriori (MAP) estimation problem which is solved as a flow network. The flow paths in the network correspond to non-overlapping trajectory hypotheses, and the flow costs to the observation likelihoods and transition probabilities. The global association is found by a minimum cost flow

algorithm.

An equivalent MAP problem is defined by Huang *et al.* [21] and Bise *et al.* [22]. First, they both generate reliable tracklets by linking cell detection responses in consecutive frames. Second, the short tracklets are associated into longer and longer tracklets. This second, global data association approach is solved by Huang *et al.* iteratively using the Hungarian association algorithm [23] and by Bise *et al.* as an integer optimization problem. This approach can achieve state-of-the-art performance and surpass the accuracy of previously discussed methods.

2.3. Conclusion

This chapter presented an overview of methods used for cell detection and tracking. Some of these methods are designed to perform better under specific imaging and cell conditions. The combination of thresholding and the Watershed method for segmentation is likely to perform well in high quality images where the cells are clearly discernible. However, in the case of noisy images with varying contrasts, the machine learning approach presented by Arteta *et al.* is likely to perform much better, at the cost of computation time to compute the larger number of features for each candidate cell region. Both these methods have been briefly evaluated on the noisy image sequences that we studied in this research. The first method required a lot of manual adjustments to perform acceptably well. The second method was able to reliably detect cells in the low signal-to-noise images. For this reason, the machine learning method by Arteta *et al.* has been chosen for this application. A more in depth explanation of the method is presented in chapter 3.

Although the detection method is able to reliably detect most of the cells in the image sequences, the cells sometimes cannot be detected due to several factors. First, the cells can submerge into the depth of the tissue and reappear a few frames later. Second, due to the motion artifacts caused by the moving (lung) tissue when the images are obtained, the images can be out-of-focus for several frames. These and other artifacts make the cell tracking problem much harder. Therefore, we needed a method that would be able to handle false detections and missing observations over a few frames. A global data association method heavily inspired by [22] has been developed and is presented in chapter 4.

There are several reasons why automatic cell tracking systems are not as popular as we would expect. The main reason is probably that the level of accuracy falls short of manual annotation. Furthermore, these systems are often not robust to changes in cell type, cell culture condition and microscopy image quality. It is our hope that the state-of-the-art methods we have chosen and will develop for our tracking application will enable us to track cells in noisy images with a performance comparable to that of a human.

3. Detection of cells

This chapter presents an overview of the cell detector from [4]. The method has been chosen for its ability to learn to detect cells in the low contrast images we aim to analyse with the tracking pipeline.

3.1. Method overview

In order to track cells across frames we need to first be able to identify the cells within the images. As described in section 2.1, there is a broad range of automatic methods for cell detection and segmentation.

The requirements for detecting cells in our application are:

1. Accurate detection in sometimes noisy images. The detector should be able to robustly identify cells in datasets that include noise, motion artifacts (induced by the camera and the moving tissue) and variable lens focus. The detector does not need to detect cells that are out-of-focus, but should reliably detect those which are in focus. Missed detections are dealt with in the tracking module.

2. There is no need for accurate segmentation. We are primarily interested in the accurate positional tracking of cells over time, and analysis of the cells appearance is of secondary interest. For the purpose of tracking, it is sufficient to be able to reliably localize cells, and extract some features that quantify them.

3. The algorithm should adapt well to different microscopy modalities. The studied datasets are obtained from various organs within the body of mice that have distinct textures. The detector should be able to detect cells in any of these datasets with minimal manual adjustments.

Much of the previous work described in section 2.1 was focused on an accurate segmentation of cells. Many of the described methods would return very good segmentations, but they would fail when presented with a very noisy dataset of varying contrasts, or need major manual tuning of parameters when presented with a new dataset with a different kind of cell.

For the purpose of our application we have chosen the cell detector presented in [4], because it conforms to all three requirements. The machine learning method is able to handle noisy datasets, and the model can easily be retrained when a new, previously unseen, type of image needs to be

analysed. The method does not focus on an accurate segmentation of cells, but on the robust localization. The major drawback of this method was performance. The authors reported run-times of 30 seconds to detect cells on 400-by-400 pixel images on an i7 CPU. This slow performance would make it unusable for tracking large image sequences. However, we were able to optimize the MATLAB code for feature computation so that the detection process took only a fraction of the reported run-time.

The detection process runs in three steps. First, robust candidate regions are extracted from an image. The method uses a Maximally Stable Extremal Region (MSER) region detector [13] to find a large number of candidate regions. Second, each of these regions is assigned a value which is produced by a classifier; this is a score of similarity between the region and a true cell. Finally, the non-overlapping subset of these candidate regions with high similarity to the annotated cells in the training data are selected via dynamic programming.

The detector can be trained on a small number of training images, where each cell is annotated with a single dot.

The code used in our method is based on the original code from [4]. Some of our modifications, especially in the feature computations, allow the detector to run significantly faster. Additional work was required to combine the cell detector with the tracking module described in chapter 4.

The rest of this chapter is divided as follows. In section 3.2 we describe how the candidate regions are obtained. Section 3.3 shows the inference procedures that selects the optimal subsets of candidate regions. In section 3.4 we formulate the learning method and in section 3.5 we describe the choice of features. Finally, in section 3.6 we briefly outline the main changes that improved the performance of the original algorithm.

3.2. Detection of candidate regions

The first stage of the cell detection process is the extraction of a large number of candidate cell regions. A region is extremal if the image intensity everywhere inside of it is higher than the image intensity at its outer boundary. For a greyscale image I it is the set of connected components of the thresholded image $\mathcal{I}_{>\mathrm{t}} = \{\mathcal{I} > \mathrm{t}\}$ for some threshold t. In practice we consider only regions that are maximally stable, as defined in [13], obtained using an efficient implementation of a maximally stable region detector [24].

Extremal regions are invariant to affine transformations of image intensities. They are stable, which means that their support is virtually unchanged over a range of thresholds, and they can be detected at multiple scales and be enumerated very efficiently. This makes them ideal for detecting candidates regions that correspond to cells in microscopy imaging. Figure 3.1 shows an example result of running the MSER detector on a sample set of microscopy cell images.

An important property of extremal regions is *nestedness*, which means that two extremal regions detected on the same image can be either nested or non-overlapping.

The number of extracted candidate regions is high, and an inference step determines which belong to cells.

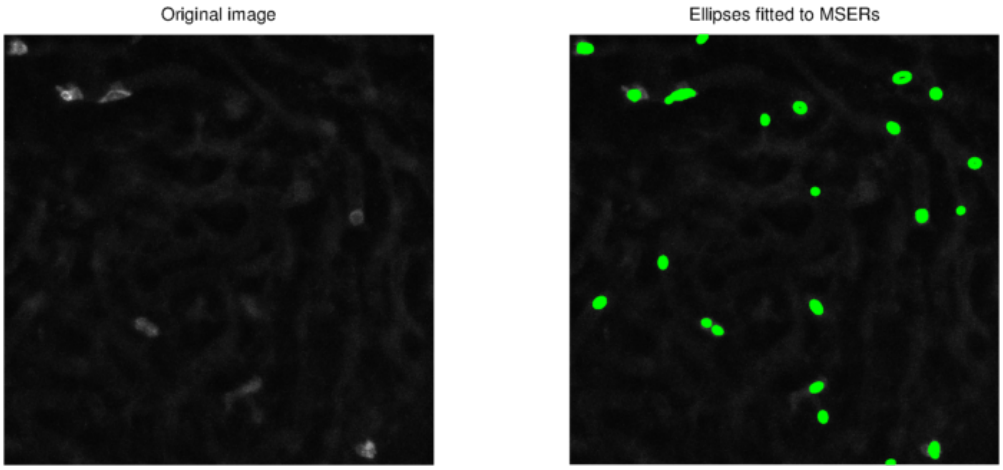


Figure 3.1.: Example original image on the left, and on the right the same image with ellipses fitted to detected MSERs.

3.3. Inference under the non-overlap constraint

The inference procedure selects a subset of candidate extremal regions that correspond to cells. Let V_i be a value assigned to each candidate region R_i indicating the appropriateness of the region to correspond to a cell. The inference procedures selects a subset of the non-overlapping candidate regions such that the sum of the values of selected regions is maximized. To make the problem computationally feasible, the regions are organized into trees according to their nestedness property. Each tree represents a set of overlapping extremal regions. The problem can then be formalized as follows. Let $\mathbf{y} = \{y_1, y_2, \dots y_N\}$ be a binary variable such that $y_i = 1$ if the region R_i is selected. Let $\mathcal{Y}$ be the set of non-overlapping regions. The maximization problem can be defined as:

$$F(\mathbf{y}) = \max_{\mathbf{y} \in \mathcal{Y}} \sum_{i=1}^{N} y_i V_i.$$

Further details on how the exact solution is obtained can be found in [4].

Although cells in the studied datasets occasionally overlap, detecting only non-overlapping cells is a good balance between the manual work required to annotate datasets, the training time and achieving accurate detections. The described framework has been expanded by the original authors to allow for the detection of partially overlapping regions [12]. However, that model requires a longer training time, and a larger set of manually annotated images to accurately detect blobs of overlapping cells. Finally, the global data association technique used in the cell tracker module should be able to correctly associate cells if the overlap time is short, such as in trajectories that are crossing and then continuing in opposite directions.

3.4. Learning the classifier

The value V_i assigned to each region R_i is a classifier score that indicates the similarity of the extremal region to a cell. The classifier is trained using dot-annotated training images $\mathcal{I}^1, \mathcal{I}^2, \mathcal{I}^3, \ldots$ as follows. Let $R_1^j, R_2^j, \ldots, R_{N^j}^j$ indicate the set of N^j candidate regions extracted from the training image $\mathcal{I}^j$. The value V_i^j of each region is computed as the dot product between a feature vector $\mathbf{f}_i^j$ computed for each region and a weight vector w: $V_i^j = \mathbf{f}_i^j \cdot w$.

The learning procedure has to learn the weight vector w so that the inference procedure tends to pick regions with a single annotated cell $n_i^j = 1$. In order to consider the non-overlap constraint of regions, a structured SVM [25] is used that directly optimizes the performance of the inference on the training set. The loss function that the learning framework tries to optimize counts the number of deviations from the one-to-one correspondence between the annotated dots and the selected regions:

$$L(\mathbf{y}^j) = \sum_{i=1}^{N^j} y_i^j |n_i^j - 1| + U^j(\mathbf{y}^j),$$

where n_i^j indicates the number of annotated dots in region R_i^j and $U^j(\mathbf{y}^j)$ counts the number of annotated dots that do not have a corresponding selected region R_i^j. Further details about the convex objective we tried to minimize and how it can be optimized using a cutting-plain algorithm can be found in [4].

3.5. Feature selection

The effectiveness of the machine learning method to detect cells depends on the quality of features computed for the regions. Good features have a lot of discriminative power, so that they can effectively distinguish regions corresponding to cells from regions that don't correspond to cells. In order to find the best feature vector, we have evaluated the performance of the classifier with several combinations of the features used in the original method [4]:

Area The area of the candidate region.

Intensity histogram : A histogram of the intensities within the region.

Region position : The position of the region centroid in x-y coordinates

Shape descriptor : A shape descriptor of the boundary of the region.

Region edges : The proportion of edge-pixels in the regions.

Invariant intensity feature : A histogram of differences of intensities between the region border and dilations of it.

Each of these features has different discriminative power, and takes some time to compute. The application of the cell detector requires that images are processed within a time limit. For this reason, we have trained and tested the algorithm with all the possible combinations of these features and identified a feature set that computes in an acceptable amount of time with a high recall and precision.

We have also developed a tool that helps select the most appropriate set of features given specific constraints, for example a maximum computation time, minimal precision and recall values, etc. A graph generated by the function is shown in figure 3.2.

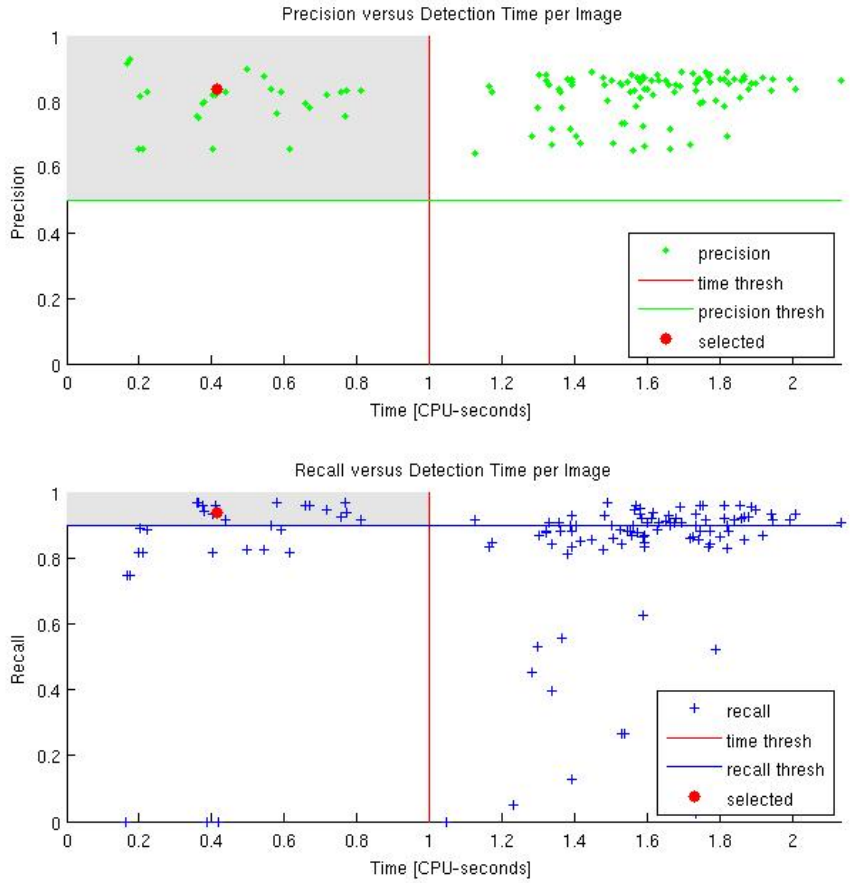


Figure 3.2.: A function helps select an appropriate feature set according to constraints, such as the computation time per image, mean precision and recall. This example shows features sets that compute within 1 CPU-second per image on average (to the left of the red vertical line), have at least 0.5 precision (above the green horizontal line) and 0.9 recall (above the blue horizontal line). The feature set corresponding to the selected parameters shown as red dots contains features 1, 3 and 4. Most importance was given to high precision followed by low computation time and high recall. The selected feature set computes in about 0.4 CPU-seconds per 512-by-512 pixel image with mean precision of 0.836 and mean recall of 0.9363 (as measured on an i7-2600 CPU wish a clock frequency of 3.40 GHz).

3.6. Performance improvements

The main drawback of the algorithm presented by Arteta *et. al.* [4] is the slow feature computation. The original paper reported detection times of 30 seconds for images of dimensions 400-by-400 pixels on an i7 CPU. Because the aim of this project is tracking cells, we expect to be processing hundreds or thousands of frames of microscopy image sequences. It was important to reduce the detection time as much as possible. The major performance improvements where achieved by addressing three things.

The algorithms computes the features for the classifier on every candidate region. The original method was configured such that thousands of MSERs were computed for each frame. While this provides more robust learning and detection, it was slowing the algorithm down excessively. First, we fine-tuned the MSER detector to detect less regions, but still return a large number of features such that the detection rate wasn't penalized.

Second, we have identified the slowest computed features and improved the algorithmic behaviour of the original MATLAB implementation. One such feature is the Contour Points Distribution Histogram. The function was frequently calling slow, built-in functions, to extract region characteristics. This, and other functions, were rewritten to call these expensive sub-procedures less frequently, without affecting the behaviour of the algorithms.

Third, we noticed that the MATLAB built-in functions were performing a lot of parameter checking, to make sure that the input parameters were valid. Many of these parameters were strings, and string manipulations are very slow in MATLAB. Several MATLAB built-in function were rewritten or replaced to remove excessive parameter checking, which in several cases represented an overhead of over 30%. These parameter checks are welcome when developing the algorithm, but once the algorithm is complete, several of these checks can be safely removed.

It is also worth mentioning that the updated original code for the detector is now more stable and better tested. A few rare bugs have been fixed, after being found when running the detector over a large number of datasets.

These optimizations resulted in a significant performance boost. The updated algorithm can perform the same computations in a fraction of the time taken by the original implementation provided by the authors of [4].

4. Tracking of cells

This chapter describes the method for tracking cell detection results to obtain cell trajectories. The chapter is a step-by-step description of the process, starting in section 4.1 with a high level overview of the tracking process. The following chapters describe each step in more detail. Section 4.2 explains how cell detection results are joined into robust tracklets. In section 4.3 we formulate a maximum a posteriori problem to link these tracklets into robust trajectories, and then in section 4.4 the problem is converted to an integer optimization problem and solved by linear programming. The final three sections describe specific details of the linear programming solution. In section 4.5 we define the set of hypotheses (false positive, linking, tracklet initialization and tracklet termination) for each tracklet and in section 4.6 we describe how the likelihoods of these hypotheses are computed. Finally, section 4.7 describes the different features that were used to train a classifier for linking trajectories.

4.1. Method overview

The cell detections obtained using the method described in chapter 3 need to be linked into trajectories. The detections contain a number of false positives and false negatives (missed detections) which make the task of associating them into trajectories difficult.

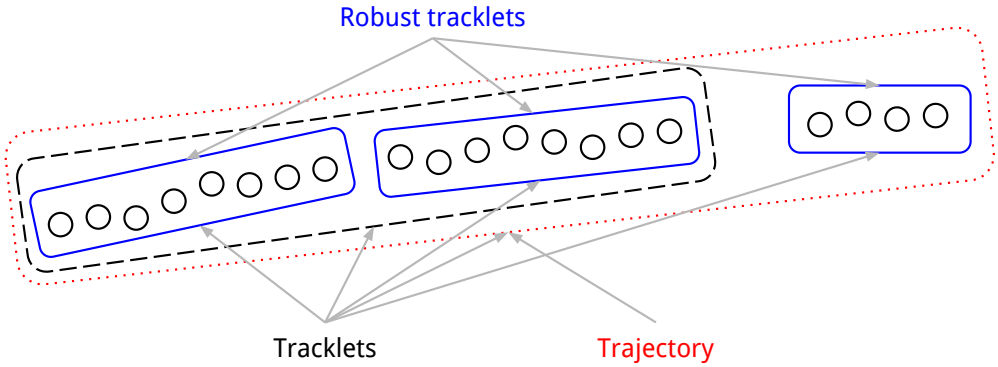


Figure 4.1.: Illustrated example of three robust tracklets which can be grouped to tracklets and all together form a trajectory.

First, we define the terminology used in the subsequent sections. A robust tracklet is a sequence of cell detections that can be linked with high confidence. These are likely to be detections that were segmented with high accuracy and their feature vectors are very similar. A robust tracklet cannot have gaps (missing detections). Similarly a tracklet is a sequence of cell detections, but differ from robust tracklets in the manner in which it may contain gaps (missed detections). A sequence of one or more robust tracklets is a tracklet, but the opposite is not necessarily true. Finally, we

call a trajectory a sequence of tracklets that cannot be effectively linked to any other tracklets. A trajectory is a maximally linked tracklet, and corresponds to the actual path performed by a cell in the image sequence. These concepts are illustrated in fig. 4.1.

Linking cell detections into tracklets is performed in two steps, as seen in fig. 4.2. First, cell detections are linked into robust tracklets by a reliable linking model. Second, the tracklets are associated into longer tracklets by closing gaps. A detailed overview of these two steps is provided in the following sections.

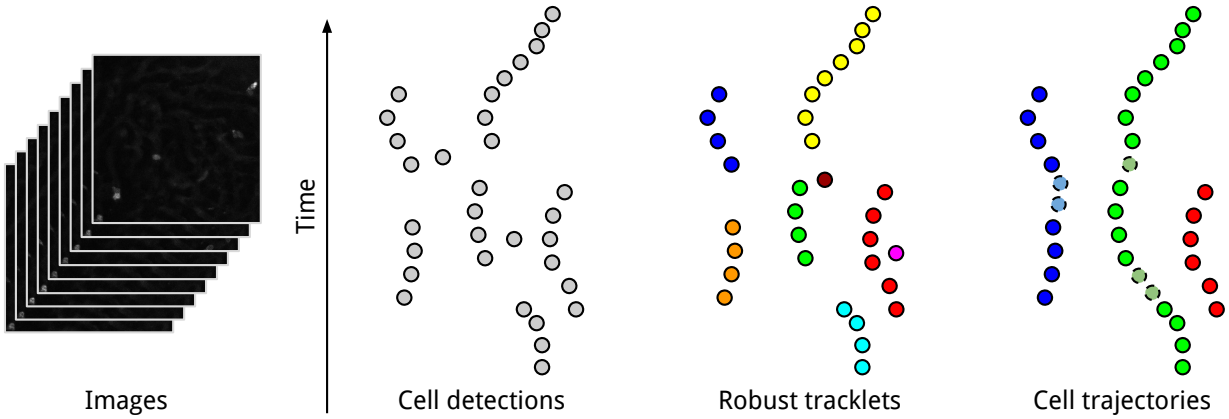


Figure 4.2.: Overview of the tracking process. First, cell detection are obtained from images. These cell detections are then grouped into robust tracklets. These tracklets are then linked into trajectories.

The process of linking robust tracklets into trajectories is performed globally, by selecting the optimal subset of tracklets to link in each iteration. This global data association approach has been chosen because it has shown significant improvements in tracking performance compared to other successful methods [22], such as [26] which performs tracking using an Interacting Multiple Models (IMM) filter that runs several Kalman filters in parallel for each trajectory.

The approach used in this project is similar to [27] in the way in which it formulates the maximum a posteriori probability (MAP) problem and solves it using linear programming, but the likelihoods of the hypotheses (the likelihood of linking two tracklets, the likelihood of a tracklet being the first tracklet in a trajectory, the likelihood of it being the last tracklet in a trajectory or the likelihood of it being a false positive) were computed using a machine learning approach. This approach makes it possible to associate cells in very noisy and low quality images, where accurate cell detections are not always possible.

Although the developed system is fully automatic, it gives some control to the user by letting him adjust the hypotheses' likelihoods by means of four intuitive parameters and thus change the way the tracklets are linked. This control eliminates the need to retrain a specific model for each new dataset, as it makes it possible to re-use a trained linking classifier to link robust tracklets in an independent dataset and adjust these parameters in order to obtain an adequate set of cell trajectories.

Finally, let's examine the importance of recall and precision values from the cell detection module and how these affect the process of generating cell trajectories. A high precision value indicates a small number of false positive detections. The cell tracker module is able to effectively eliminate

these false positives. However if the precision value is low, then these false positives could start forming structures that the cell tracker could confuse for actual cell trajectories. It is therefore important that the cell detection module should not detect a too large number of false positives. A high recall value indicates that most of the manually annotated cells have been detected, which means that the cell detections will be easily linked to form trajectories as seen in fig. 4.3. The tracker will work best with very high recall values. If the recall value is low, there will be large gaps between individual detections. The cell tracker will them have to link short tracklets that can be further apart, which is likely to increase the number of incorrect links.

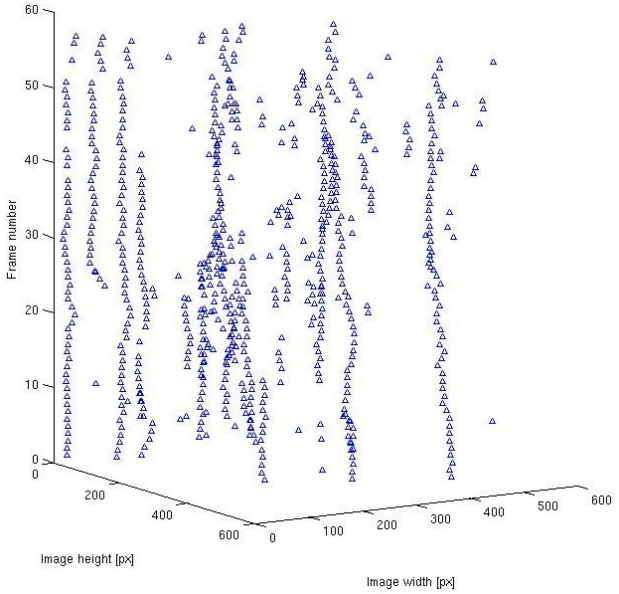


Figure 4.3.: Cells detected over 60 consecutive frames are visualized as a time series. The vertical axis corresponds to the frames. Even in this raw detection data, it is possible to see the tracks of some of these cells.

4.2. Joining cell detections into robust tracklets

The first phase of linking cell detections into trajectories consists of identifying a set of robust tracklets. Robust tracklets for dataset B are shown in fig. 4.4.

We define a cell detection $d_i = (x_i, y_i, f_i, t_i)$, where x_i and y_i are the position of the cell detections within the frame, f_i an appearance feature vector obtained from the cell detector module, and t_i the frame index of the detection. The set of all cell detections in the image sequence is $\mathbf{D} = \{d_i\}$.

Let $\mathbf{T} = \{T_k\}$ be a hypotheses set of tracklets, where each tracklet is defined as a list of robust tracklets, such that the frame index of the last detection ($t_{k_{i_n}}$) in each robust tracklet is lower than the frame index of the first detection of any following tracklet ($t_{k_{i+1_1}}$): $T_k = \{T_k^{robust} | \forall i,\, t_{k_{i_n}} < t_{k_{i+1_1}}\}$. A robust tracklet is defined as $T_k^{robust} = \{d_{k_i} | \forall i,\, t_{k_{i+1}} = t_{k_i} + 1,\, d_{k_i} \in \mathbf{D}\}$. Assuming that each detection can belong to only one tracklet we define the non-overlap constraint:

$$\forall T_i,\, T_j \in \mathbf{T},\, i \neq j,\, T_i \cap T_j = \emptyset.$$

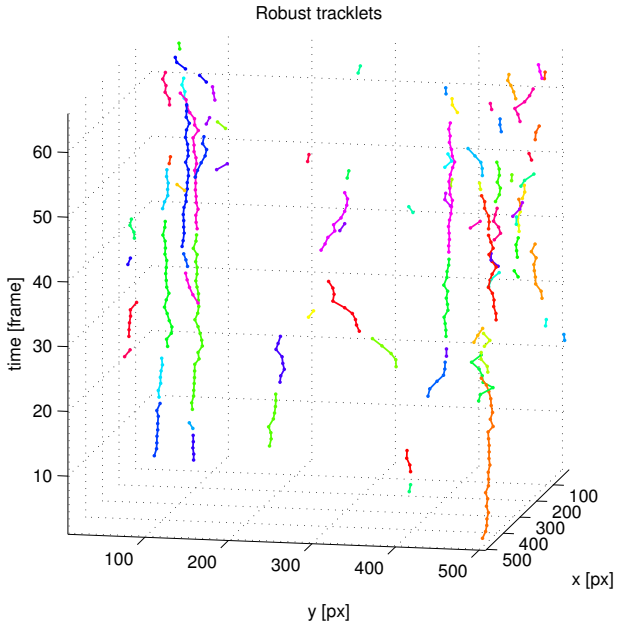


Figure 4.4.: Robust tracklets generated for dataset B.

For each cell detection we need to identify a good match in the next frame, if it exists. Let P_{link} represent the likelihood of linking two detections d_i and d_j:

$$P_{link}(d_j|d_i) = \begin{cases} V((f_i, x_i, y_i), (f_j, x_i, y_j)) & \text{if } t_j - t_i = 1 \\ 0 & \text{otherwise} \end{cases},$$

where V is an affinity function that returns the probability that the provided feature vectors and detection positions belong to the same cell. The feature vectors are obtained from the cell detection module and correspond with the feature vectors used to identify a candidate region as a cell. This avoids the need to compute new feature vectors, which could significantly slow down the tracking process.

The probability of linking detections P_{link} is computed on all pairs of cells between consecutive frames i and $i+1$ and vice versa. For each cell detection we found the most similar detection in the next frame, and for each detection in the second frame the most similar detection in the previous frame. Then, only symmetrical matches were chosen. A matching is symmetric if a detection in the first frame d_1 best matches detection d_2 in the second frame, and detection d_2 best matches detection d_1 in the first frame. This way we obtain a subset of matching pairs, such that each detection is matched to exactly one or no detection in the next frame.

To increase the robustness of the matches, a threshold θ_1 was chosen, such that only cell detection pairs whose linking probability was higher than that threshold could be linked.

The affinity function V is learned using a supervised Naive Bayes classifier, which learns to solve the binary classification problem of linking (or not linking) a pair of cell descriptors. The model is learned by comparing the appearance feature vectors of the detections as well as their positions, which are obtained from the cell detection module.

The machine learning algorithm was trained with annotated datasets. The annotations contained the position of cells in the images and links to matching cells in consecutive images. To train a linking model, cell appearance feature vectors had to be obtained for the dot-annotated cells. First, the cell detection module was trained to detect cells in a dataset. Second, the detector found a set of cell detections, which were matched to the corresponding real annotations. A detection was matched to an annotated cell if and only if the position difference was below a small threshold (10 pixels). This step was required to obtain the cell descriptors of the dot-annotated cells. Finally, the data to train the classifier was constructed as follows. Positive examples were selected as consecutive (linked) cells within each tracklet. Negative examples were chosen from all possible combinations of cells from different tracklets.

Based on the pairs of symmetric matches between cell detections, the system is able to generate a set of robust tracklets $\mathbf{T}^{robust}$. Cell detections that were not linked to any other cell are also considered robust tracklets, and are included in $\mathbf{T}^{robust}$ for possible further linking in future steps of the algorithm.

4.3. Global data association

The problem of linking robust tracklets together to form trajectories is formulated as a MAP problem [20–22]. We first present the formulation, and then in the next section we provide the linear programming implementation.

Given the robust tracklet set $\mathbf{T}^{robust}$, we maximize the posteriori probability to find the best data association:

$$\begin{aligned}\mathbf{T}^* &= \arg\max_{\mathbf{T}} P(\mathbf{T}|\mathbf{T}^{robust}) \\ &= \arg\max_{\mathbf{T}} P(\mathbf{T}^{robust}|\mathbf{T})P(\mathbf{T}).\end{aligned}$$

Assuming that the likelihood probabilities of the robust tracklets are conditionally independent given $\mathbf{T}$, and $T_k \in \mathbf{T}$ cannot overlap with each other, i.e. $\forall T_i, T_j \in \mathbf{T}, i \neq j, T_i \cap T_j = \emptyset$:

$$\mathbf{T}^* = \arg\max_{\mathbf{T}} \prod_{T_i \in \mathbf{T}^{robust}} P(T_i|\mathbf{T}) \prod_{T_k \in \mathbf{T}} P(T_k).$$

The likelihood of a robust tracklet is defined as:

$$P(T_i|\mathbf{T}) = \begin{cases} P_{TP}(T_i) & \text{if } \exists T_k \in \mathbf{T}, T_i \in T_k \\ P_{FP}(T_i) & \text{if } \forall T_k \in \mathbf{T}, T_i \notin T_k \end{cases},$$

where $P_{TP}(T_i)$ is the probability of T_i being a true positive and $P_{FP}(P_i)$ the probability of T_i being a false positive.

The probability of a tracklet $P(T_k)$ is modelled as a sequence of observations with the Markov property, namely that, given the current observation, the previous and future observations are independent:

$$\begin{aligned} P(T_k) &= P(\{T_{k_1}, T_{k_2}, T_{k_3}, \ldots, T_{k_n}\}), \text{ where } T_{k_i} \in \mathbf{T}^{robust} \\ &= P_{init}(T_{k_1}) P_{link}(T_{k_2}|T_{k_1}) P_{link}(T_{k_3}|T_{k_2}) \ldots P_{link}(T_{k_n}|T_{k_{n-1}}) P_{term}(T_{k_n}) \\ &= P_{init}(T_{k_1}) \left[\prod_{j=1:n-1} P_{link}(T_{k_{j+1}}|T_{k_j}) \right] P_{term}(T_{k_n}), \end{aligned}$$

where $P_{init}(T_{k_1})$ is the probability of T_{k_1} being the first robust tracklet in T_k, $P_{term}(T_{k_n})$ the probability of T_{k_n} being the last robust tracklet in the sequence, and $P_{link}(T_{k_{j+1}}|T_{k_j})$ transition or linking probabilities for $T_{k_{j+1}}$ and T_{k_j}. The definitions of these terms will be provided in section 4.6.

Note that the MAP problem takes into consideration the possibility of false cell detections, which makes the model ideal for very noisy and low quality microscopic image sequences where the cell detector is likely to find a number of false detections. Additionally, in the analysed image sequences there are often gaps of several frames where the image becomes out of focus and the cells disappear from the field of view. The linking probabilities permit an efficient closing of these gaps.

The benefit of the global data association approach to cell tracking is that it makes a global decision based on the probabilities defined over all the frames of the image sequence rather than propagating the results from frame to frame. This makes the algorithm more robust to errors in the cell detection module.

4.4. Implementation using linear programming

The MAP problem is converted to an integer optimization problem and solved by linear programming.

Let N be the number of input robust tracklets. Let L be a vector containing the likelihoods of all possible hypotheses: initialization, termination, false positive, and linking hypothesis between two robust tracklets. The formulation of these likelihoods is given in section 4.5, and the implementation details in section 4.6.

We also define a matrix H of dimensions $|L| \times 2N$ containing constraints to avoid selecting conflicting hypotheses. Let i represent the index of each new hypothesis and j the index over the columns of H. Then, for a robust tracklet T_k and candidate linking tracklet T_l the entries of matrix H are defined as follows for each possible hypothesis:

$$H_{ij} = \begin{cases} 1 & \text{for an initialization hypothesis if } j = N + k \\ 1 & \text{for a termination hypothesis if } j = k \\ 1 & \text{for a false positive hypothesis if } j = N + k \text{ or } j = k \\ 1 & \text{for a linking hypothesis if } j = k \text{ or } j = N + l \\ 0 & \text{otherwise.} \end{cases}$$

Once the constraint matrix H and likelihood vector L are defined, the original MAP problem from section 4.3 can be solved by selecting a subset of rows from H such that the sum of the corresponding likelihoods in L is maximized, under the non-overlap constraint of tracklets. The MAP problem can be reformulated as a binary linear problem:

$$I^* = \arg\max_I L^T I, \text{ subject to } H^T I = 1,$$

where I is a binary vector containing 1 for the selected rows of the matrix H and 0 elsewhere. The constraint $H^T I = 1$ guarantees that each robust tracklet appears in only one tracklet, or is discarded as a false positive.

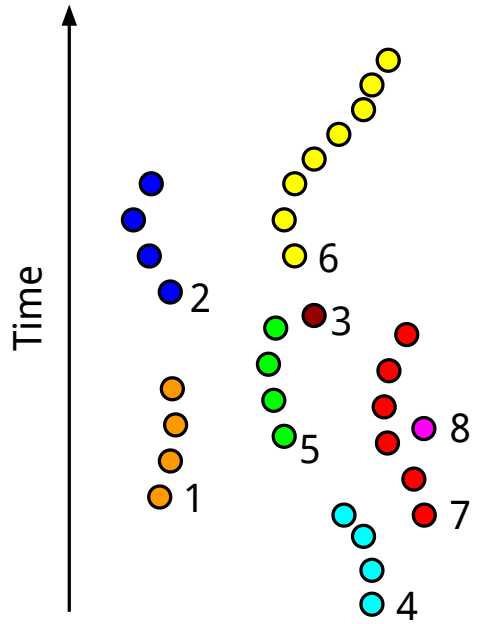
For each tracklet an initialization, a termination and a false positive hypothesis is computed. The linking hypothesis is computed for pairs of tracklets where the gap between the tail of the first and head of the next tracklet is shorter than a user specified number of frames.

Figure 4.5 shows a small artificial example of the tracklet linking process. The optimal subset of hypothesis includes initialization hypotheses for tracklets 1, 4 and 7; termination hypothesis for tracklets 2, 6 and 7; false positive hypotheses for 3 and 8, and linking hypothesis for tracklet pairs 1-2, 5-6 and 4-5.

4.5. Hypotheses likelihood definitions

In this section we define how the different hypotheses are computed. In section 4.6 we will discuss the implementation details.

Due to errors in the cell detection module, all tracklets are candidates for being false positives. The false positive hypothesis likelihood is computed as:



Description	L	I^*	H	
init 1	0.80	1	0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0
init 2	0.80	0	0 0 0 0 0 0 0 0	0 1 0 0 0 0 0 0
init 3	0.60	0	0 0 0 0 0 0 0 0	0 0 1 0 0 0 0 0
init 4	0.95	1	0 0 0 0 0 0 0 0	0 0 0 1 0 0 0 0
init 5	0.30	0	0 0 0 0 0 0 0 0	0 0 0 0 1 0 0 0
init 6	0.20	0	0 0 0 0 0 0 0 0	0 0 0 0 0 1 0 0
init 7	0.70	1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 1 0
init 8	0.70	0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 1
term 1	0.30	0	1 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
term 2	0.90	1	0 1 0 0 0 0 0 0	0 0 0 0 0 0 0 0
term 3	0.60	0	0 0 1 0 0 0 0 0	0 0 0 0 0 0 0 0
term 4	0.20	0	0 0 0 1 0 0 0 0	0 0 0 0 0 0 0 0
term 5	0.15	0	0 0 0 0 1 0 0 0	0 0 0 0 0 0 0 0
term 6	0.85	1	0 0 0 0 0 1 0 0	0 0 0 0 0 0 0 0
term 7	0.80	1	0 0 0 0 0 0 1 0	0 0 0 0 0 0 0 0
term 8	0.90	0	0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0
FP 1	0.20	0	1 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0
FP 2	0.10	0	0 1 0 0 0 0 0 0	0 1 0 0 0 0 0 0
FP 3	0.95	1	0 0 1 0 0 0 0 0	0 0 1 0 0 0 0 0
FP 4	0.30	0	0 0 0 1 0 0 0 0	0 0 0 1 0 0 0 0
FP 5	0.25	0	0 0 0 0 1 0 0 0	0 0 0 0 1 0 0 0
FP 6	0.15	0	0 0 0 0 0 1 0 0	0 0 0 0 0 1 0 0
FP 7	0.10	0	0 0 0 0 0 0 1 0	0 0 0 0 0 0 1 0
FP 8	0.90	1	0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 1
1 -> 2	0.80	1	1 0 0 0 0 0 0 0	0 1 0 0 0 0 0 0
5 -> 6	0.85	1	0 0 0 0 1 0 0 0	0 0 0 0 0 1 0 0
5 -> 3	0.90	0	0 0 0 0 1 0 0 0	0 0 1 0 0 0 0 0
4 -> 5	0.70	1	0 0 0 1 0 0 0 0	0 0 0 0 1 0 0 0
4 -> 3	0.30	0	0 0 0 1 0 0 0 0	0 0 1 0 0 0 0 0
4 -> 8	0.20	0	0 0 0 1 0 0 0 0	0 0 0 0 0 0 0 1
7 -> 3	0.10	0	0 0 0 0 0 0 1 0	0 0 1 0 0 0 0 0
8 -> 3	0.30	0	0 0 0 0 0 0 0 1	0 0 1 0 0 0 0 0

Figure 4.5.: Small artificial example of the global data optimization process including hypothesis matrix H, the likelihoods vector L, and the logical vector of the optimal subset of hypotheses I^*.

$$L_i = \log P_{FP}(T_k).$$

The linking hypothesis measures the likelihood of connecting two tracklets. Candidate tracklet pairs for linking are those for which the distance between the last detection (the tail) of the first tracklet (X_{k_n}) and the first detection (the head) on the second tracklet (X_{l_1}) is less than a specified number of frames. The likelihood of the linking hypothesis is computed as:

$$L_i = \log P_{link}(T_l|T_k) + \frac{\log P_{TP}(T_k) + \log P_{TP}(T_l)}{2}.$$

Bise *et al.* [22] dealt with cell tracking on image sequences where cell detections could be reliably detected. The authors considered tracklets close to the boundaries of the field of view as candidate for initial tracklets. This work is based on image sequences that were obtained from observing cells in thick tissue, where cells can sink into the background and reappear at any time. The initialization hypothesis indicates the likelihood that a tracklet is the first tracklet in a trajectory. Taking this into consideration new trajectories can be initialized anywhere within the field of view. The corresponding likelihood is computed as:

$$L_i = \log P_{init}(T_k) + \frac{\log P_{TP}(T_k)}{2}.$$

Similarly to the initialization hypothesis a cell can sink into the background or leave the field of view. This is taken into account in the termination hypothesis, which is also evaluated for all tracklets. The termination likelihood is computed based on the probability of the tracklet being at the end of a sequences:

$$L_i = \log P_{term}(T_k) + \frac{\log P_{TP}(T_k)}{2}.$$

A true positive tracklet appears in exactly two of the initialization, termination or linking hypotheses. For this reason the likelihood of a tracklet being true positive $\log P_{TP}(T_k)$ is divided into two halves that are included in these hypothesis.

4.6. Computing the likelihoods

In this section we describe the implementation details of the likelihoods: $P_{TP}(T_k)$, $P_{FP}(T_k)$, $P_{init}(T_k)$ and $P_{term}(T_k)$. The likelihoods are directly connected to the input observations and are estimated from training data.

The true and false positive likelihood

The true and false positive likelihoods of a tracklet T_k are defined in terms of the miss detection rate of the cell detector module α, and the number of the cell observations composing the tracklet which we denote $|T_k|$:

$$P_{FP}(T_k) = \alpha^{\frac{|T_k|}{\lambda_1}}$$
$$P_{TP}(T_k) = 1 - P_{FP}(T_k),$$

where λ_1 is empirically set to 2.

The linking hypothesis

The linking hypothesis is computed between pairs of tracklets where the distance (number of frames) between the tail of the first and the head of the second tracklet is less than a threshold. Because of the variable contrast and poor quality of the imaging technique, the measure is computed by taking into account several spatio-temporal and visual features. The different features are outlined in section 4.7. The likelihood of linking tracklet T_l with T_k is:

$$P_{link}(T_l|T_k) = \begin{cases} V(T_l, T_k) & \text{if } t_{l,1} - t_{k,n} \leq \lambda_2 \\ 0 & \text{otherwise,} \end{cases}$$

where $t_{k,n}$ is the frame index of the last detection of tracklet T_k, t_{l_1} is the frame index of the first detection of tracklet T_l, λ_2 is a threshold indicating the maximum allowed gap for linking two tracklets, and $V(.)$ a function that returns the probability that the tracklets should be linked.

The function $V(.)$ is a model incorporating appearance and spatio-temporal features of the candidate linking tracklets. It is trained using an artificial neural network (ANN) with Bayesian regularization and the number of inputs equal to the number of features and a single output that returns a value between 0 and 1. The number of hidden layers has been set to 2.

The binary ANN is trained using annotated image sequences. To generate positive training examples, the annotated trajectories were split into several parts. Then the combinations of these segments belonging to the same original trajectory were used as positive examples in the training data. Similarly, the combinations of these segments belonging to different original trajectories were used as negative examples. One of the features used in the classifier is the appearance vector, which is obtained for each cell annotation using the cell detection module.

The data used to train the classifier contains features of pairs of tracklets that can be separated by a different number of frames, from 1 to λ_2. The consequence of using a binary classifier is that it might return a larger probability of linking two tracklets that are further apart than two tracklets

that are closer together. For example, an original trajectory could be detected as three robust tracklets that should be linked sequentially. The classifier might return a larger likelihood of linking the first and third tracklet than the first and second tracklet. The data association module would then likely link the first tracklet to the third, leaving the second one as a new short trajectory. To overcome this problem the process of linking tracklets is performed iteratively, closing ever larger gaps between tracklets. Each iteration takes the tracklets obtained in the previous one and allows for further association. This way the described problem can no longer occur. A positive side effect of performing the process iteratively is a reduced peak memory usage because of the lower number of hypotheses that are evaluated in each step.

The benefits of this machine learning approach for computing the likelihood of linking two tracklets are twofold. First, it works well on noisy datasets because it uses a large number of features to train the model. Secondly, the large number of features makes it less vulnerable to inaccurate cell segmentation.

The initialization likelihood

The initialization likelihood is a measure of a tracklet being the first tracklet of a trajectory. This work is based on image sequences with high noise and contrast variance, where cell detections cannot be reliably detected over the entire trajectories of the cells. Additionally, the cells can sink into or surface from the background at any time. We take into consideration that new trajectories can be initialized anywhere within the field of view. The initialization hypothesis is based on the linking hypotheses. It is equal to the likelihood of tracklet T_k not being linked to the most likely linking tracklet in the λ_2 frames *before* its first cell detection:

$$P_{init}(T_k) = \begin{cases} 1 - \max P_{link}(T_k|T_l) & \forall T_l \in \mathbf{T}, t_{k,1} - t_{l,n} \leq \lambda_2 \\ 0 & \text{otherwise.} \end{cases}$$

This assumes that the initialization likelihood is independent of the status of any of the previous tracklets. For example, if all of the previous tracklets are selected by the algorithm to be false positives, then $P_{init}(T_k)$ should be increased. Although this assumption limits the model, it is required to limit the number of hypotheses that need to be computed. The same assumption is made for computing the termination hypotheses.

The termination hypothesis

The termination hypothesis measures the likelihood of a tracklet being the last tracklet of a trajectory. It is defined similarly to the initialization hypothesis; the likelihood of tracklet T_k not being linked to the most likely linking tracklet in the λ_2 frames *after* its last cell detection:

$$P_{term}(T_k) = \begin{cases} 1 - \max P_{link}(T_l|T_k) & \forall T_l \in \mathbf{T}, t_{l,1} - t_{k,n} \leq \lambda_2 \\ 0 & \text{otherwise.} \end{cases}$$

Scaling hypotheses' likelihoods

Each of the hypotheses' likelihoods can be scaled by an appropriate parameter π_{FP}, π_{link}, π_{init} or π_{term}. The adjustment of these parameters allows a direct manipulation of the hypotheses. For example, increasing π_{init} or π_{term} relative to π_{link} makes the system more conservative in linking tracklets. The added benefit of these scaling parameters is that they allow the reuse of the trained classifier for somewhat different datasets, without the need to annotate them and retrain the classifier.

4.7. Features for the linking classifier

In this section we present an overview of the visual and spatio-temporal features implemented and tested for the classifier for linking tracklets:

Cell feature descriptor The difference of vectors containing appearance features obtained from the cell detection module for each candidate linking tracklet. A description of these features is available in section 3.5.

Gaussian broadening distribution This feature is detailed in section 4.7.1.

Intersection of extrapolated tracklets This features makes use of a linear Kalman filter [28] to extrapolate a pair of candidate linking tracklets to their mid-point and measures how effectively they match.

Gap size The number of frames between the tail and head of a pair of tracklets.

Position distance A two dimensional vector containing the absolute distance between the head and tail of a pair of tracklets corresponding to the x and y coordinates.

Square of position distance Same as *Position distance*, but the value is squared.

Distance between points The euclidean distance between the positions of the head and tail of linking tracklets. This is similar to the previous features, but combines the distance between both coordinates into a single value.

Direction angle The difference between the tracklet orientations computed from the last few frames of the first tracklet and the first few frames on the second tracklet. The number of frames used to compute the direction is parametrized.

Orientation variance Similar to the previous feature, but computes the difference of variance of

orientation change within tracklets.

Mean displacement The difference of mean displacement between cell detections within the tracklets.

Displacement variance Similar to the previous feature, but computes the difference of variances of the displacement between cell detections.

Distance from edge The difference between minimal and maximal distances of the head or tail of the tracklets from the edge of the field of view.

4.7.1. Gaussian broadening feature

This section describes a feature that is based on the observation of the motion of the trajectories. It makes intuitive sense that the probability of linking two tracklets is inversely proportional to the distance (number of frames) between tracklets. Additionally we can observe that a cell will not travel in a perfectly straight line, but might deviate from it. Using these observations, we have devised a feature inspired by Doppler broadening [1].

After a tracklet ends, its motion is extrapolated for several frames. The probability of a tracklet being at each point along the extrapolated tracklet is assumed to be normally distributed. Because of the observation that a tracklet might deviate from its temporary direction, the variance is assumed to be larger at each next extrapolated location. If we place such normal distributions along the extrapolated trajectory, and we normalize their value such that the sum of all the normal distributions is equal to one, we obtain a new distribution that describes the probability of a tracklet's position in the future.

We can use this tracklet's distribution to evaluate the likelihood that another candidate tracklet should be linked to it by summing the values of the distribution at the locations corresponding to the cell detection of the candidate tracklet.

Figure 4.6 shows an example profile of such a distribution, together with two candidate linking tracklets. The sum of values from the distribution at the location of the tracklet's cell detections indicates that the black tracklet is more likely to be linked to the red one than the green tracklet. The figure also shows that the distribution correctly adapts to tracklets composed of a single cell detection, or similarly to tracklets with very little movement.

4.8. Implementation details

In this section we describe some of the challenges encountered during the implementation of the method, and how they have been dealt with.

[1] `http://en.wikipedia.org/wiki/Doppler_broadening`

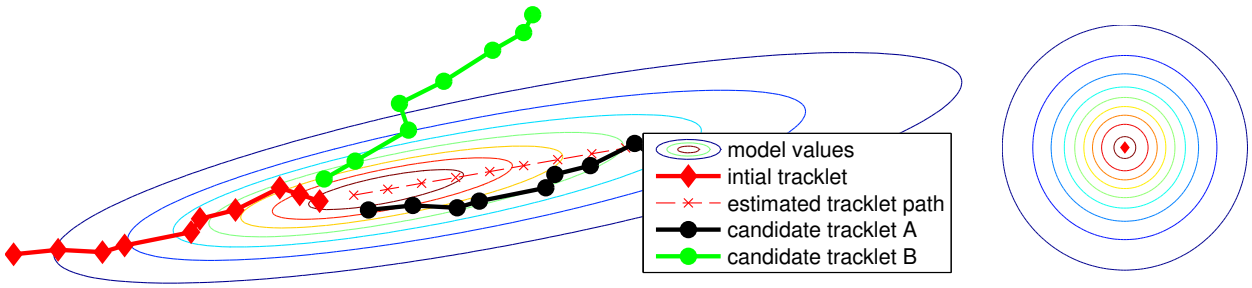


Figure 4.6.: Gaussian broadening feature. The distribution is computed for the red tracklet. The contour colours of the distribution are red for higher values and blue as the value decreases. The candidate black tracklet is given a linking likelihood of 0.20, while the green, which is angled from the direction of the red tracklet, a value of 0.06. On the plot on the right we see that if a tracklet is composed of only one cell detection the value of the feature is equal all around it.

In the previous sections we said that linking hypotheses were measured for all tracklet pairs that were less than a certain number of frames apart. We have found that by pruning the unlikely linking hypotheses we were able to increase the speed of the tracking module significantly. We have therefore skipped the computation of linking likelihoods for pairs of tracklets that were more than a configurable amount of pixels apart. The threshold is set sufficiently high so that it doesn't affect the linking of trajectories for quickly moving cells.

The choice of using a single classifier to compute likelihoods for link hypotheses can cause some incorrect connections when the size of gaps between tracklets is large (e.g. more than 10 frames). To remedy this issue, the tracker can be configured to close gaps iteratively, for example first gaps of 10 frames, then gaps of 20, etc. This not only prevents certain incorrect connections, but also reduces the amount of peak memory used by the tracker since a fewer hypotheses need to be evaluated at once.

Finally, very short robust tracklets of length one can be challenging to connect, as spatio-temporal features cannot be effectively computed for them. During development we have found that the performance of the tracking method can be slightly increased by discarding all the robust tracklets of length one as false positives. For this reason, our method does not use these tracklets to generate trajectories. This has also improved the computational time, as fewer hypotheses need to be evaluation.

5. Data acquisition and annotation

This chapter describes the data that influenced the decisions of selecting the cell detection and tracking methods. In section 5.1 we briefly describe the imaging method used to acquire the image sequences and present some example datasets. Section 5.1.2 describes some of the challenges of tracking cells in these datasets. Section 5.2 presents the data annotation tool that was developed to ease the annotation process.

5.1. Data acquisition and example datasets

As discussed in the concluding section of chapter 2, the datasets heavily influenced the choice of algorithms for cell detection and tracking. Many computer vision algorithms rely on heuristics to improve their accuracy. In cell detection methods, this is obvious from the fact that a algorithm developed for a certain type of imaging method will likely perform poorly on different types of images (e.g. different shape of cells). In cell tracking, heuristics help adjust the algorithms to the specific behaviour of cells that are being analysed. For example, a different tracking method could be used for images with cells that move slowly (where there is a large overlap between cells in consecutive frames) than for cells that move quickly (where there is little overlap between cells in consecutive frames).

The data acquisition process is not part of this research. However, for the reasons stated above, it is important to know the characteristics of the datasets. Because these are directly influenced by the imaging method, we follow by outlining the image acquisition process, and then present the datasets used in the evaluation of our methods.

The image sequences that inspired the development of the methods described in this thesis were acquired *in vivo*. This means that the images are obtained on living mice, in contrast to *in vitro* where cells are analysed on a tissue sample in a standard laboratory environment using petri dishes and other instruments. *In vivo* analysis is preferred over *in vitro* because it is better suited for observing the behaviour of cells in their natural environment.

Recent advances in intravital microscopy of the lung [29] and other organs allow us to image neutrophils and other cells of interest simultaneously by combining fluorescent reporter mice and intravenously-injected fluorescently-labelled antibodies. Briefly, the mouse is anaesthetized, mechanically ventilated and the surface of the lung is exposed surgically. Then a specialized "vacuum chamber" (as in [29]) is used to stably hold a glass coverslip against the surface of the lung. The alveoli, alveolar capillaries and associated cells can then be visualized by laser scanning confocal

microscopy. The same apparatus can be used to interrogate other organs without mechanical ventilation, e.g. the liver.

All the data was provided by Dr. Leo Carlin from the Leukocyte Biology Section at the National Heart and Lung Institute (NHLI)[1].

5.1.1. Datasets

From the datasets provided by Dr. Leo Carlin, five have been selected to use in the evaluation of this work because of their distinct characteristics. The original dimensions of the datasets were 512-by-512 pixels, but some have been cropped to reduce the number of cells that had to be annotated to train the cell detector and tracker.

Below we describe the main characteristics of the datasets, together with the number of frames that were annotated to train the cell detector. Because of the large number of trajectories in certain datasets, not all the trajectories have been annotated. Instead, in each dataset a few trajectories were chosen and annotated. We aimed to select the longest trajectories, as these could give the most insight in the analysis of cell behaviour.

Dataset A

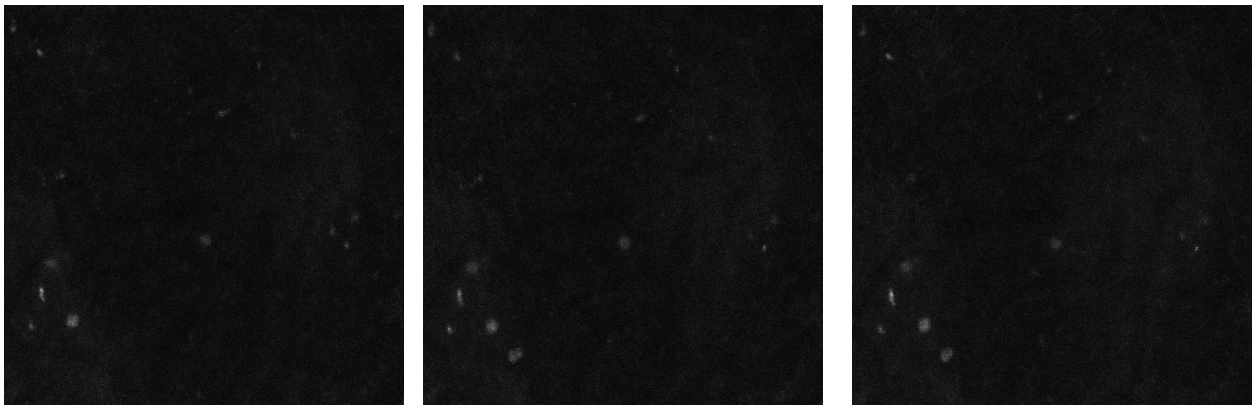


Figure 5.1.: Three consecutive frames from dataset A.

Dataset A, seen in fig. 5.1, contains 66 frames of dimensions 512-by-512 pixels. All the frames have been manually annotated for the purpose of training the cell detector. The dataset only has an average population of 1.58 cells per frame, and all four main trajectories have been annotated, although three of them can only be tracked in a dozen frames. The dataset is obtained from the liver. There is a good contrast between the cells and the background. The background is almost uniform but still contains some information about the architecture of the liver sinusoids, which is difficult to discern without adjusting the image contrast. The cells appear as medium dark shades of grey and of a circular shape; their boundaries smoothly blend with the background. The cells move slowly. Despite the small number of frames, this is an especially challenging dataset for the

[1] `http://www1.imperial.ac.uk/nhli/`

cell detection module. Frames 1–17 are radically different from frame 18 and onwards. The first 17 frames show little contrast between the cells and the background. On frame 18, the images change such that the texture from the background becomes very similar to the texture of the cells in the first 17 frames, and the intensity of the cells intensify. Finally, the last few frames contain no cells.

Dataset B

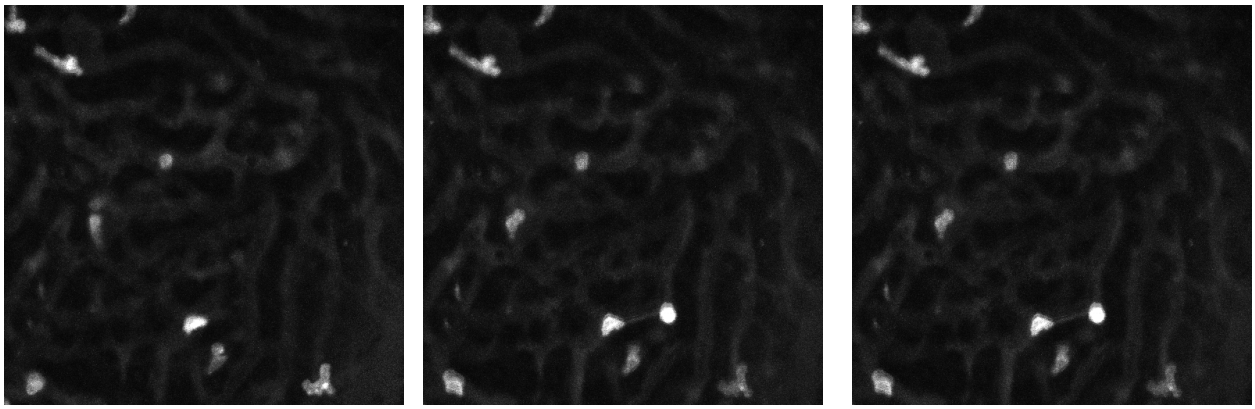


Figure 5.2.: Three consecutive frames from dataset B.

Figure 5.2 shows three consecutive frames from dataset B. The dataset contains 66 frames of dimensions 512-by-512 pixels and an average population of 8.03 cells per frame. The entire dataset has been annotated for the purpose of training the cell detector and (TODO 5) of the longest trajectories for the purpose of training the cell tracker. This dataset is also obtained from the liver sinusoids. The cells appear brighter than in dataset A, but their shapes vary. Some are round and others elongated and deformed to fit into the tight liver sinusoids in which they move. Their brightness varies from medium to very bright. The brighter ones have clearer boundaries than the medium bright ones. In the background we can clearly discern the blood vessels in a darker grey colour. Cells in this dataset are active. Some move relatively fast and change direction frequently. The images are of constant quality, and there are few significant camera artifacts.

Dataset C

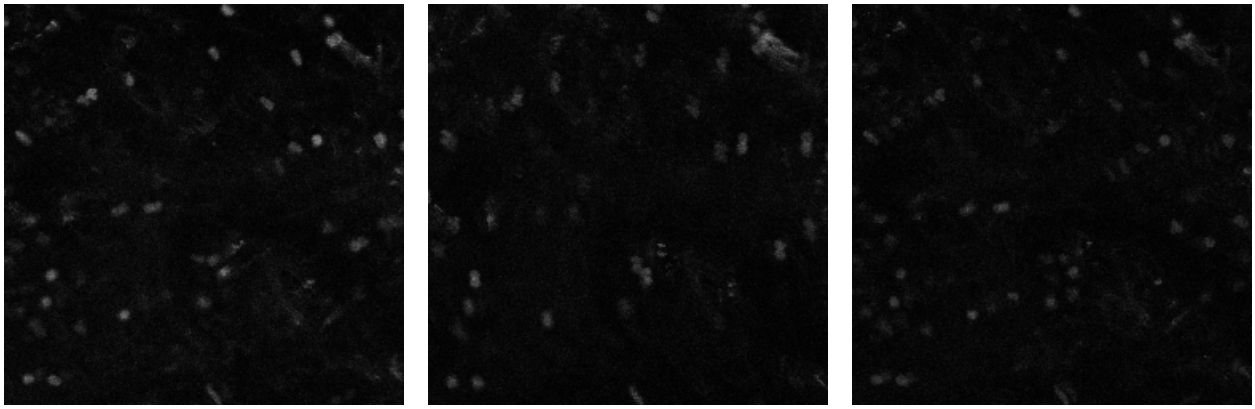


Figure 5.3.: Three consecutive frames from dataset C.

Dataset C, seen in fig. 5.3, is composed of 126 frames which have been cropped to 251-by-251 pixels. The first 58 frames have been annotated with dot-annotation for the purpose of training the cell detector, and 5 of the longest trajectories were annotated for the purpose of training the cell tracker. This dataset is also obtained from the lung. The dataset contains an average population of 19.45 cells per frame as estimated from the annotated frames. The background is dark, but contains several very faint cells which we are not interested in tracking. We are interested in tracking the brighter cells. The colours of the cells are dark to medium grey. The cell boundaries blend smoothly with the background. Most cells appear as round patches. Although the cells are mostly stagnant, the lung tissue is moving due to the breathing of the mice. For this reason, the dataset contains many motion artifacts. Some frames are blurred, and many appear to contain the duplicates of the cells shifted in position by a small amount, as seen in the middle image in fig. 5.3.

Dataset D

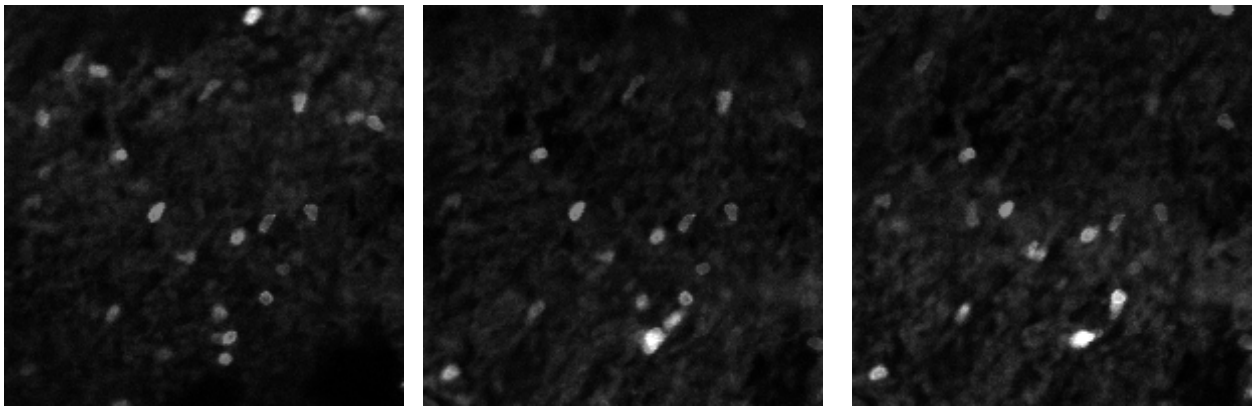


Figure 5.4.: Three consecutive frames from dataset D.

Dataset D, seen in fig. 5.4, is likely to be the most challenging for the tracking module. It was originally composed of 682 frames. Unfortunately, the dataset contained large sequences of frames that were of exceptionally low quality, such that even a human could not track the cells accurately. Thus, the dataset has been manually pruned to 377 frames and cropped to dimensions 199-by-199 pixels. 53 frames have been annotated with dot-annotations as have 7 of the longest trajectories. Dataset D is obtained from the lung. The cell density is 10.32 cells per frame. The background contains some noise, but it is still relatively different to the texture of the cells. The cells are mostly elliptical and have strong boundaries. This dataset contains several fast moving cells. As mentioned before, the dataset contains many motion artifacts, the contrast of the images is constantly changing, and often about half of the cells in each frame become invisible for a few frames, even after manually eliminating the most affected.

Dataset E

The last dataset, seen in figure fig. 5.5, contains 194 frames which have been cropped to dimensions 277-by-277 pixels. 67 frames were annotated for the purpose of training and evaluating the quality of the cell detection module. (TODO) of the longest trajectories were annotated for training the

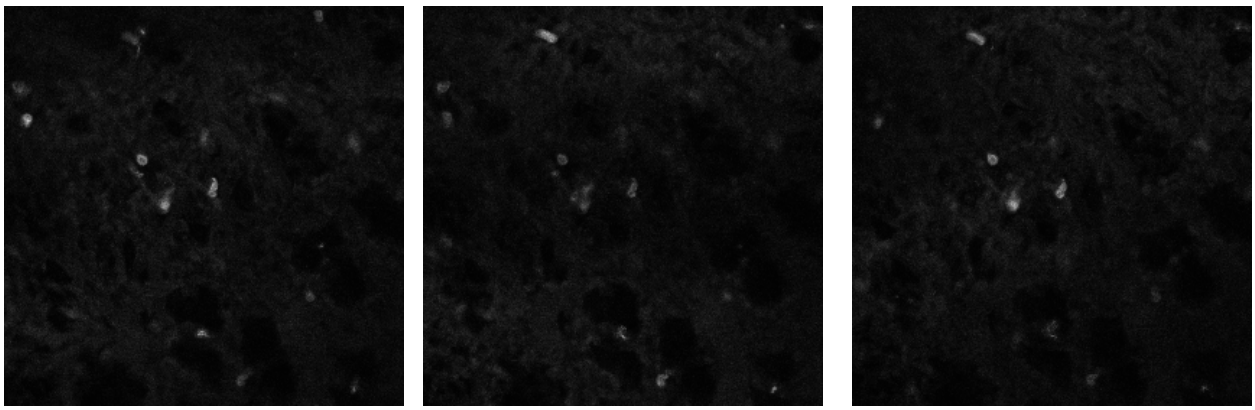


Figure 5.5.: Three consecutive frames from dataset E.

tracker. This dataset is from the lung. The density of the cells is of 7.01 cells per frame. The background contains some texture, that could sometime be confused with the cell texture. The cells in this dataset appear smaller than in the other datasets and are less elliptical. They contain more sharp edges. The intensity of the cells varies from dark to bright. The cells appear to move very slowly. The images contain many artifacts that make tracking more difficult. First, the images become darker towards the middle of the image sequence, and then during the last quarter of the sequence become clear again. Second, there is an alternating dark/bright area in all the frames caused by the raster scanning imaging method combined with the moving tissue.

5.1.2. Image analysis challenges

In this section we are going to present two phenomena caused by the combination of moving tissue due to ventilation and a raster scanned image that make these datasets especially hard to track with frame-by-frame tracking methods.

The first challenge consists of the artifacts caused by the scanned image. They appear as alternating dark/bright horizontal lines which seem to move from the top to the bottom of the images. The effect can be seen in fig. 5.6. This makes the tracking problem difficult, because the cells behind the dark areas appear fainter or disappear completely for the few frames that they are covered.

The second challenge is the movement of the tissue, especially in the case of lung imaging. The shaking can be in the x-y plane, but sometimes, as in the case shown in figure fig. 5.7 it can be primarily in the z-direction. Displacement in the x-y direction causes the cells to jiggle. Displacement of the tissue in the z-direction causes parts of all of the image to become out-of-focus for a few frames.

Several of the datasets present these phenomena to a lesser or larger extend. These artifacts were the primary reason for choosing a global optimization method for associating detection responses into trajectories instead of a frame-by-frame approach.

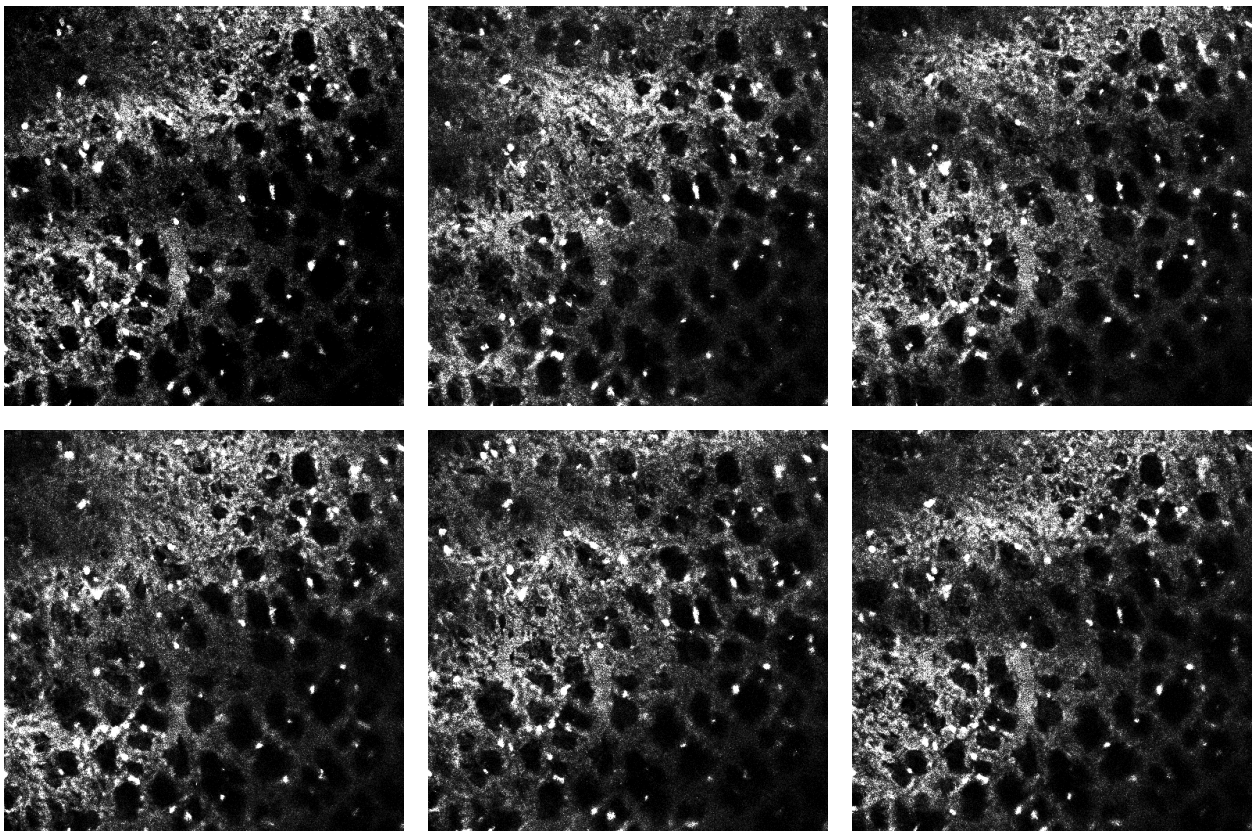


Figure 5.6.: Six consecutive frames from a sequence displaying the alternating dark/bright regions caused by the camera shutter. The contrast and sharpness of the images has been adjusted to dramatize the effect.

5.1.3. Manual data annotation

Training the detector requires a set of annotated image sequences. Precise and consistent dot-annotations allow the model to learn well. Precise annotations are those where the annotation dots are placed within the actual cell. If the dot-annotation is placed nearby, yet outside of cell, the detector will incorrectly learn to discriminate cells from the background. Consistent annotations are those where the annotator takes special care to annotate cells that look alike. For example, if only bright cells are annotated in one frame, but bright and dark cells in the next frame, the detector will not be able to reliable detect both bright and dark cells.

The major challenge of annotating the datasets is consistency, due to the human ability to easily identify cells from the background, even if their relative intensities vary significantly. The detection module is able to learn well even with the presence of a few outliers.

It was important to correctly annotate a certain number of frames on each dataset. Annotating the entire datasets would represent a lot of unnecessary manual work, because the detector can be trained on just a few dozen annotated frames. We have manually annotated and reviewed about 50 frames from each dataset. The counts of annotated frames and cell density metrics are presented in table 5.1.

In the next section we present the tool developed to facilitate the annotation process.

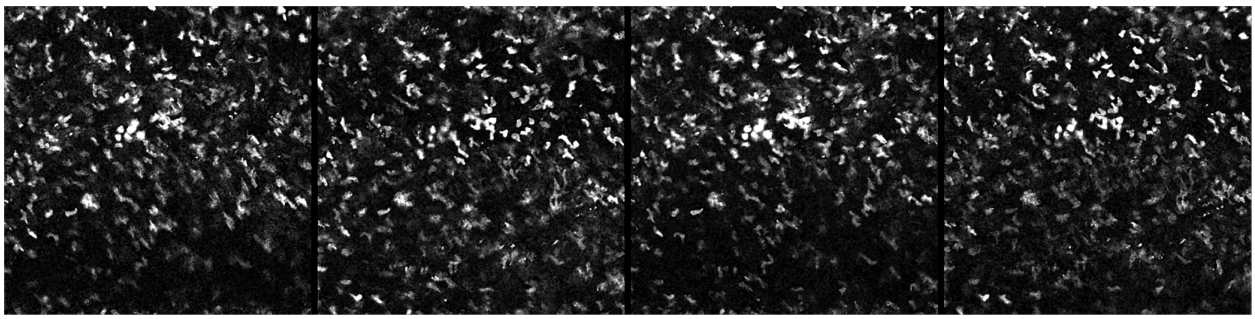


Figure 5.7.: Four consecutive frames taken from the lung displaying varying cell clarity due to the displacement caused by the lung movement. The contrast of the images has been adjusted to dramatize the effect.

Dataset	Number of annotated frames	Number of annotated cells	Average number of cells per annotated frame	Standard deviation of number of cells per annotated frame
A	66	104	1.58	0.86
B	66	530	8.03	2.44
C	58	1128	19.45	4.19
D	53	547	10.32	2.54
E	67	470	7.01	2.25

Table 5.1.: Summary of the number of frames and cells manually annotated in each dataset.

5.2. The annotation tool

In order to train the cell detector and tracker it is necessary to annotate a few frames from the datasets. The cell detector requires dot annotations - a single dot within each cell - that indicate whether an extremal region corresponds to a cell. The detector is then trained to recognize extremal regions similar to the ones that were annotated. Similarly, the cell tracker needs annotations that indicate whether two cells from different frames belong to the same tracklet. The cell tracker is then trained based on the similarity of the appearance and spatio-temporal features extracted from these cells.

To train a good detector and tracker it is important that the annotations are consistent. For example, several datasets contain cells that appear bright and some that are darker. If we are only interested in tracking the cells that consistently appear bright, the annotations should only include the bright cells. The models may not be trained correctly if only part of the bright cells were annotated. Note that this paragraph makes observations about brightness because the concept is easy to understand for a reader. However, the detector and tracker rely on many more features for detection and tracking.

To facilitate the annotation of datasets a new tool has been developed. The graphic user interface of the annotation tool is visible in fig. 5.8. The tool is able to load a sequence of images, display them and permits the user to annotate them. The user is able to preform the following actions: adding a

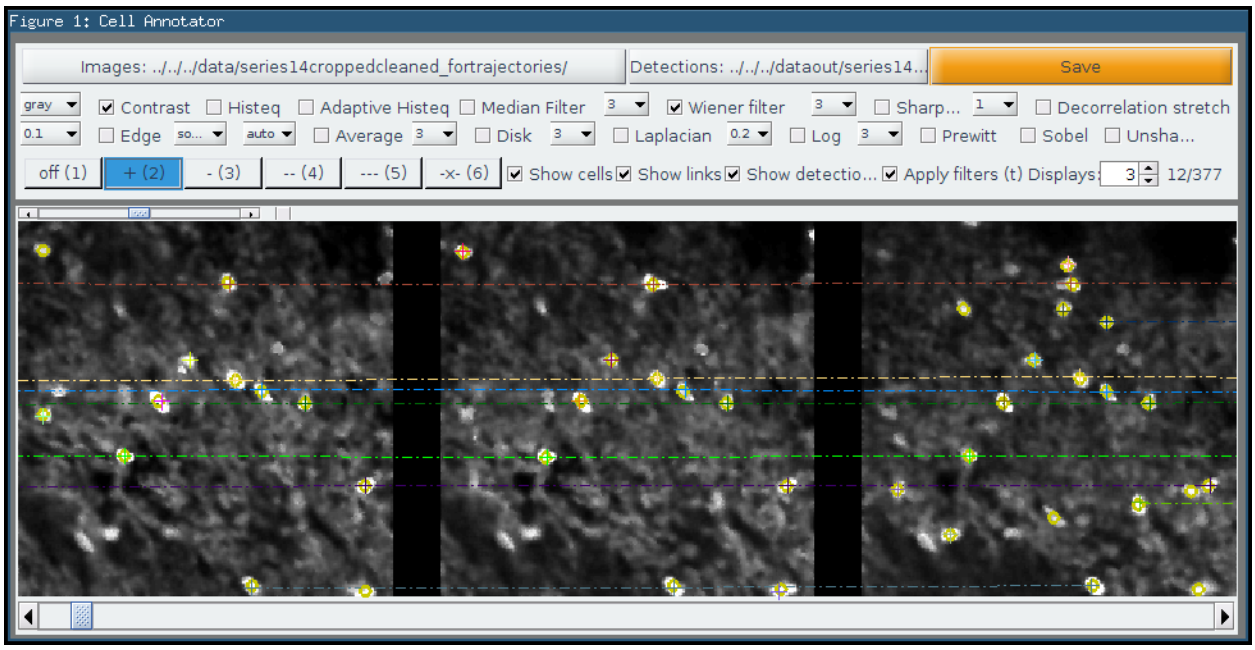


Figure 5.8.: Screenshot of the annotation tool. User annotations are shown as crosses and dashed links between them, while detection responses are are shown as yellow circles.

dot detection, deleting a dot detection, adding a link between cells, deleting a link between cells. The tool also features zooming and panning.

Furthermore, the tool includes a broad set of filters that can be applied on the images to increase the clarity of the cells and ease the annotation process. These filters include contrast adjustments, smoothing, sharpening, edge detection, etc. To reduce the clutter on densely populated datasets, the user is able to only display annotations from a specific part of the image, hide trajectories that include more than three dot annotations, hide the part of the GUI containing the buttons and filters in order to focus on just the images (*full-screen mode*), and more.

The annotation tool includes detection of potentially bad annotations. These include a warning system that displays a triangle next to a cell if two annotations are very close together and a different display style for links that deviate too much from the average (in terms of between-frame displacement). These help to eliminate accidental human errors, which would be very difficult to discover otherwise.

Finally, the tool can also be used to compare the detection responses obtained from the cell detector by overlaying them over the images. In fig. 5.8 these detections are shown as yellow circles. This can be very useful to visually evaluate the quality of the detections, and possibly reveal cells that the user might have missed when annotating.

In addition to the annotation tool, another small GUI has been developed that shows the annotations in an interactive 3D view, where each tracklet colour is different. This tool is shown in fig. 5.9. It is helpful to recognize potentially bad annotations (for example an outlier cell within a trajectory) or discover missed links.

All the annotations have been reviewed by Dr. Leo Carlin, who provided the datasets. However, due

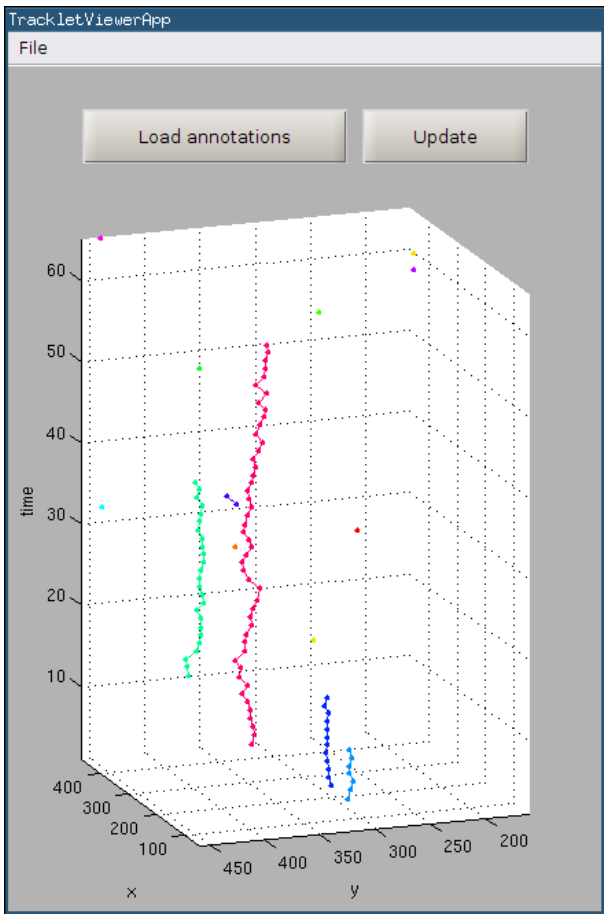


Figure 5.9.: Screenshot of the interactive 3D tracklet viewer showing annotations for Dataset A.

to the noisy nature of the images, some of the dataset are difficult to correctly annotate even for a human. It is expected that a small number of annotations are erroneous.

Both tools were developed and packaged as standalone executables with MATLAB 8.3.0.532 (R2014a). The tools do not depend on MATLAB to run, but require the free MATLAB Compiler Runtime 8.3[2] which is automatically downloaded and installed when the applications are installed on the system. Installation guides for both applications are provided in appendix A.

[2] `http://www.mathworks.co.uk/products/compiler/mcr/`

6. Experimental results

In this chapter we quantitatively and qualitatively analyse the performance of the automatic cell detector and tracker. Although some evaluation of the performance of the detection method is performed by the original authors in [4] it is useful to see how the method performs on the studied datasets in order to understand how much of the tracking accuracy is lost due to cells missed by the detection module. First, in section 6.1 we evaluate the performance and computation time of the cell detector and in section 6.2 that of the cell tracker. Finally, in section 6.3, we explore the limitations of the methods and in section 6.4 summarize the results.

6.1. Cell detector

In this section we evaluate the performance of the automatic cell detection module. First, we introduce the performance metrics used to evaluate the accuracy of the cell detector. Then we present detection accuracy results. To evaluate the accuracy and generalizability of the detection module we performed two sets of experiments. First, we trained the cell detector on a number of frames from each individual dataset, and measured the accuracy on the same dataset. Second, we trained the detector on combinations of datasets in order to judge the performance degradation caused by the focus placed on multiple types of cells. Due to the varying size of the cells in the datasets, and their varying brightness, we expect that such a trained detector will perform poorly compared to when it is trained and tested on individual datasets, sometimes mistakenly detecting small artifacts in the background as cells. Finally, we computed the average detection time per frame for each dataset.

The aim of this research was to develop an automatic cell detection and tracking pipeline that would require as little manual work as possible. This implies that a balance between accuracy and level of manual work had to be established. There is also an direct relationship between accuracy and computation time. In order to reduce the amount of manual work, we aimed to configure the cell detection module such that it would perform well on all the tested datasets without any manual adjustment of parameters. The consequences of this decision are twofold:

1. The features computed on the candidate cell regions are the same for all datasets and have been presented in section 3.5. Although some datasets could be analysed faster or more accurately with a different subset of features, using the same features for all dataset eliminates the complicated feature selection process for the user and makes the system generalizable to a large number of different cell types.

2. The parameters of the MSER detector should be adequately set to perform well on all datasets. This means that the MSER detector should be able to detect cells of varying size and contrast in the varying datasets. The consequence of this limitation for datasets with large cells and some background noise is that a potentially much larger number of candidate regions will be detected than is necessary. Since each candidate region has to be evaluated, this results in an increased computation time.

We were able to identify features that compute in an acceptable time for all these datasets (see section 3.5). However, it should be noted that in the case of testing the detector on very large datasets with thousands of frames, some adjustments of the parameters could result in a significant reduction in computation time and increased accuracy.

6.1.1. Performance metrics

We measured the performance of the cell detector in terms of precision and recall. The metrics are defined in terms of:

True Positive instances (TP): candidate cell regions that are manually annotated as cells and which the detector successfully classifies as cells.

False Positive instances (FP): candidate cell regions that are not manually annotated as cells, but the detector incorrectly classifies them as cells.

False Negative instances (FN): candidate cell regions that are manually annotated as cells, but the detector incorrectly classifies them.

We then define precision as:

$$\mathrm{PRE} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}},$$

and recall (also known as sensitivity) as:

$$\mathrm{REC} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \ .$$

6.1.2. Detection accuracy

As mentioned previously, we performed two different experiments to measure the performance of the cell detection module. The first experiment consisted of training and testing the algorithm on the same dataset. The training set was 70% and the testing set 30% of the entire dataset. The training and testing datasets were created from images in randomized order from the entire dataset. This allowed us to measure, for each dataset, the maximum precision and recall values we could expect

from the algorithm. In the second experiment the training was performed on combined datasets. The goal of this experiment was to observe how well the algorithm is able to generalize, and still return acceptable results.

Training and testing on individual datasets

The training was performed on 70% of the manually dot-annotated images from each dataset, and tested on the remaining 30%. Table 6.1 displays the computed precision and recall values.

Dataset	Precision	Recall
A	25.0	22.5
B	90.1	89.1
C	76.5	84.2
D	86.1	85.3
E	93.4	78.6

Table 6.1.: Precision and recall values for the cell detector trained on each dataset individually.

The detector tested on datasets B to E achieved precision and recall values above 75. A manual comparison of the annotations confirms that the results are good, with few false detections. Most of the differences in detection were caused by minor inconsistencies in annotations. The importance of consistent annotations was stressed in section 5.1.3.

Dataset A is an outlier with extremely low precision and recall values. This is likely caused by the specific characteristics of this image sequence, which we described in section 5.1.1. Briefly, between frame 17 and 18 there is an abrupt change of image clarity. The background from the threshold frame onwards has a texture that is very similar to that of the cells in the first 17 frames. This means that the negative candidate regions from frame 18 and onward clash with the positive candidate regions from the previous frames. The result is that the detector is unable to learn to discriminate cells from their background.

These results show us that the detector can correctly detect most of the (annotated) cells. Further manual reviews of the annotations would likely improve the performance, but this is not done here as it should have been tested on a separate validation dataset. Additionally, it is unlikely that future users of this detection method will always have perfect annotations available. In the next experiment, we measured the performance of the detector when trained on a composite dataset. This told us whether it would be possible to train a single, general detector and use it to test on new, possibly unforeseen datasets.

Figures C.1 to C.5 in appendix C display a temporal view of the detected results in each dataset. The vertical axis represents the consecutive frame number of the image sequence. The figures show that “cell tracks” are discernible, even if the number of outliers is significant. The detectors used to detect the cells on the entire dataset were trained on all the annotated frames.

Training on combined datasets

In this second experiment we were interested in measuring how well the algorithm was able to generalize when trained on a larger, combined dataset. For this purpose, several of the datasets were grouped, and the detector was trained to recognize cells of all types. The detector was trained on a random 70% of all annotated images in the combined dataset. It was then tested on a random 30% of annotated frames from each individual dataset separately. This means that sometimes the same frames could be used for training and testing. However, we also ran combinations of datasets where one dataset was left out of the training on each occasion. Testing on that dataset should then reveal if the algorithm generalizes well.

Table 6.2 summarizes the precision and recall values for all tested combined datasets. The column denoted γ contains testing performance as tested on 30% of the combined dataset. The values shown in bold correspond to the performance of testing on the dataset that was left out from the combined training dataset. The row denoted "Individual" indicates the results when the datasets where trained and tested individually (these are the same results as in table 6.1). The values shown in red indicate a decrease in precision/recall by at least 1 point compared to the results when trained and tested on individual datasets. Similarly, blue colour indicates an increase in precision/recall compared to the results on the row denoted "Individual".

Dataset	Precision						Recall					
	γ	A	B	C	D	E	γ	A	B	C	D	E
Individual		25.0	90.1	76.5	86.1	93.4		22.5	89.1	84.2	85.3	78.6
ABCDE	74.5	49.2	76.0	85.2	69.2	94.5	79.9	58.3	90.8	57.1	98.9	79.7
ABCD_	69.7	49.2	77.0	85.8	69.9	**93.4**	80.0	58.3	90.3	58.7	98.4	**80.7**
ABC_E	72.8	45.9	75.0	82.6	**41.1**	87.2	83.2	65.0	96.4	73.6	**99.4**	89.8
AB_DE	75.5	47.5	75.8	**85.7**	73.1	96.5	73.1	50.8	88.9	**46.1**	97.1	68.3
A_CDE	75.9	41.3	**63.1**	85.5	71.2	94.6	70.4	58.3	**89.2**	57.0	99.0	80.2
_BCDE	86.3	**49.2**	76.4	85.9	71.8	94.4	76.4	**58.3**	89.5	56.0	97.4	75.7
BCD	82.8	**52.5**	78.0	87.5	72.7	**95.2**	74.5	**55.8**	89.5	54.7	97.4	**74.0**
_B_DE	85.9	**42.5**	80.3	**84.8**	72.1	97.7	82.6	**45.8**	91.2	**44.8**	97.4	66.3

Table 6.2.: Precision and recall values for the cell detector trained on combined datasets. Values typed in bold indicate the testing datasets that were not included in the combined training dataset. Red/blue colour indicates a decrease/increase in performance compared to training and testing on each individual dataset by at least 1 precision/recall point.

The values in the table show us some significant insights. First of all, the recall values increased for most datasets when detecting cells using a detector trained with almost any combination of data. Using more data thus improves the recall. This means that in general the algorithm will detect a larger number of true positive instances, but also some false negatives. However, given the high recall value we can expect the number of false negatives to remain relatively small. For the tracking module it is easy to discard short, isolated detections. The high recall values indicate that we will

likely detect most of the real cell trajectories.

Second, we notice that while most recall values increase, precision tends to decrease in some datasets. This means that the number of false positives increased. As said beforehand, the cell tracking module can deal effectively with short, isolated false positives. However, if the number of false positives increases too much, the tracking module might detect cell trajectories erroneously (or correspond to background noise). However, given that in most examples the precision values did not fall below 70, we should still expect to achieve good tracking results.

The results from Dataset C went against these observations. Its recall values decreased significantly whenever a combined dataset was used to train the detector. This dataset contains many motion artifacts as well as high variance in the intensity of the cells, some of which smoothly blend into the background. This makes it very difficult to consistently annotate. It is possible that the inconsistencies in annotation caused it to perform in such a way when combined with other datasets.

Another interesting observation is that the precision and recall values for dataset A improved significantly compared to training the detector only on dataset A. We have previously presented some possible reasons for its poor performance when trained individually. The increased performance when using a cell detector trained on other datasets (even excluding dataset A itself) can be attributed to two factors. First, the detector learns to detect a wider range of cell types. Second, dataset A contains much fewer annotated cells than the other datasets (about five times less). It is possible that a detector trained on only dataset A overfits the annotations, and cannot generalize to detect cells in other frames.

The performance on dataset E remained almost constant, with very few exceptions (e.g. smaller recall value when the detector was trained on combined dataset containing datasets A, B, D and E).

In this section we aimed to understand whether a detector trained on a single, combined dataset could be used to detect cells in new, previously unseen images. The results have shown that a general detector will not perform well on *all* new unseen datasets. For example, we have measured a significant drop in precision values when testing on unseen datasets B and D, and a significant drop in recall values when testing on dataset C. However, in some cases training on a combined detector either improved the performance or reduced it insignificantly (such as in datasets A and E). What this tells us is that when presented with a new dataset it is worth trying to use a pre-trained detector and review its results. In some cases this could provide acceptable results, and manual annotation of the new dataset might be unnecessary.

It is worth keeping in mind that all five datasets analysed in this thesis show distinct characteristics (both in cell type and image clarity). In practice, datasets will often be similar and it might be sufficient to annotate a single dataset to train a detector that can be used on other similar samples.

6.1.3. Computations time

In order to positionally track cells in an image sequence, the computation time of extracting the cell position from each frame is just as important as the accurate identification of cells. Table 6.3 displays the average detection times per frame for each dataset. We also measured the average detection time per annotated cell in each dataset and the average ratio between the number of all candidate regions that were detected with the MSER detector and the number of annotated cells in each frame.

The detection was performed on a PC with an Intel(R) Core(TM) i7-2600 CPU with a clock frequency of 3.40GHz and 8GB RAM. The MATLAB version used to measure the detection speed was 8.1.0.604 (R2013a) running in Ubuntu Linux 13.04 x64. Althought the detector can be easily configured to use several workers to process the image sequences in parallel, the measurements were performed using a single worker, i.e. all images was processed sequentially.

Dataset	Time per frame [s]	Time per annotated cell [s]	Candidate-Annotation ratio
A	0.7458	0.4733	36:1
B	1.4466	0.1801	18:1
C	1.2730	0.1422	11:1
D	1.3607	0.9378	21:1
E	0.6273	0.2589	13:1

Table 6.3.: Average computation times per frame and annotated cell.

The results show that we were able to optimize the cell detector to a point where it's speed was no longer an issue for many cases. As a reminder, the original paper by Arteta *et. al.* [4] reports detection speeds of 30 seconds per 400-by-400 pixels image on an i7 CPU. Most of the computation time is spent by the MSER detector and the feature computation.

The measurements show that the detection speed is dependent primarily on the number of cells in each frame. Datasets B, C, and D contain 5 to 10 times more cells per frame than dataset A. Dataset E also contains 5 times more cells than dataset A, but it's computation time per frame was lower. This could be attributed to the smaller fraction of candidate regions that had to be evaluated. In contrast, dataset A, which contains larger cells than the other datasets, many more candidate cells were evaluated relative to the number of annotated cells. It is likely that, since the MSER detector was configured equally for all datasets, it identified a larger number of small noise artifacts as cell candidates.

6.2. Cell tracker

After obtaining the cell detection results we went on to determine the number of trajectories and which detection results belong to which trajectory. In the analysis of the performance of the tracking system, we assume that the detection results are already available. The cell detector had been

trained on the annotated images from each dataset, and the cells which existed within the entire image sequence had been identified. Similarly to the experiments ran on the cell detection module, we also run two sets of experiments for the cell tracker. First, we measured the performance of the cell tracker after it had been trained and tested on each dataset individually. In the second experiment we trained the cell tracking module on combined datasets and evaluated its performance on previously unseen trajectories.

The aim of this research was to develop an automatic cell detection and tracking pipeline that would require as little manual work as possible. However, there are a few cases when an automated algorithm might not perform with the desired accuracy, for example when testing on a new dataset and reusing a tracker trained on a different dataset. For this reason four intuitive parameters can be adjusted to modify the tracking results. We discussed the benefits of this configuration in section 4.6.

In order to evaluate the cell tracking module, we were interested in two factors. First, how well do the generated trajectories fit with the ground truth annotations. This measure is dependent on the effectiveness of the cell tracker module as well as the cell detection module. This is a measure of how well the automatic tracking pipeline works. However, if the cell detection module misses too many cells, it might be impossible for the tracking module to compensate for the errors. Therefore, we were also interested in how well the developed tracking module worked independently of the cell detection module. This gave us an idea of how much of the performance is lost in the tracking module, given perfect detection results. For this reason the following scenarios were evaluated:

1. Performance relative to the Ground Truth annotations (GT-P). This is a measure of how well the combination of the cell detector and tracker perform together.

2. Performance relative to *Detected Ground Truth annotations* (DGP-P). *Detected ground truth annotations* are obtained by automatically mapping the ground truth annotations to the cell detection results. For example, a ground truth annotated trajectory is mapped to detection results by finding for each annotated cell observation of the trajectory a detection result that is less than a certain number of pixels away (e.g. 10 pixels). This measure compares the *detected ground truth annotations* to the generated trajectories and gives a measure that is less dependent on the performance of the cell detection module.

3. Comparison of Ground Truth annotation and *Detected Ground Truth annotations* (GT-DGT-P). This measure does not measure the quality of the generated trajectories, but measures the performance lost due to errors in the cell detection module.

These performance evaluations measure the performance of the longest ground truth trajectories. Annotating all the trajectories in the full datasets would require weeks of manual work and review in order to establish the correctness of the annotations. For this reason testing is limited to a certain number of long fully annotated trajectories which have been manually identified and annotated. The counts of the annotated trajectories together with their length is available in table 6.4.

All the measurements of the cell tracker performance have been obtained by cross validation. Because the number of annotated trajectories is relatively small, we have used leave-one-out cross validation.

Dataset	Annotated trajectories	Length (number of frames) of annotated trajectories
Dataset A	2	49 25
Dataset B	14	66 65 60 51 46 41 37 29 26 19 18 15 14 10
Dataset C	8	118 118 97 83 81 74 71
Dataset D	6	267 164 116 112 107
Dataset E	8	117 110 105 93 73 51 37

Table 6.4.: Count and length of manually annotated trajectories in each dataset.

This involves repeatedly partitioning the set of annotated trajectories into two subsets, training the tracker on the training subset, and measuring its performance on the testing subset. As the name suggests, in each round we train the tracker on all but one annotated trajectory, using the left-out trajectory for measuring the performance. This is repeated for each annotated trajectory and the results are averaged over all the rounds.

6.2.1. Performance metrics

To quantify the performance of the cell tracking module we used two metrics originating from radar tracking literature [30], recently used in other works [15, 22, 31]: *target effectiveness* and *track purity*.

Let us call a *target* a true cell trajectory as provided by manual ground truth annotations and *track* the trajectory generated by the cell tracker module. Then, for a target and track pair, target effectiveness can be defined as:

$$\text{Target Effectivenes} = \frac{100 \times \text{Number of frames where the target is correctly followed by the track}}{\text{Total number of frames of the target}}.$$

Note that the multiplication by 100 is simply to represent the output from 0 to 100 instead of 0 to 1. Target effectiveness therefore tells us how effectively a target is followed by a track. For each target we find the track that follows it correctly for the longest number of frames. The measure penalizes tracks that are initialized late or terminated early relative to the target. However, it does not penalize tracks for being initiated early, or terminated late. Therefore, we need an additional measure, track purity, which does just that. In the case of track purity the roles of track and target are inverted. It measures how well the target follows a track, and is defined as:

$$\text{Track Purity} = \frac{100 \times \text{Number of frames where the track is correctly followed by the target}}{\text{Total number of frames of the track}}.$$

This measure will penalize early initiated or late terminated tracks. It is similar to precision in that it penalizes false detections. We measure it by assigning to each track the target that follows it for

the longest number of frames.

6.2.2. Tracking accuracy

As mentioned previously, we performed two different experiments to measure the performance of the cell tracking module. The first experiment consisted of training and testing the algorithm on the same dataset. The performance evaluation was performed by leave-one-out cross validation where we repeatedly train the model on $n-1$ annotated trajectories and tested it on the trajectory that was left out. This first experiment allowed us to measure, for each dataset, how well the tracker could link cells with a specific mode of motion. In the second experiment the training was performed on a combined dataset. The goal of this experiment was to observe how well the algorithm is able to learn from a large number of trajectories exhibiting different modes of motions and apply the learned model to track a previously unseen trajectory.

Although a large set of features have been implemented to train the classifier, only two have been used in the following test because they have shown results that were sufficiently good (not necessarily best) and could be computed quickly. The first of the features is the squared euclidean distance between the last cell detection of the first tracklet and the first cell detection of the following tracklet. The second, is the Gaussian broadening distribution explained in section 4.7.1.

Training and testing on individual datasets

The performance measurements were obtained by leave-one-out cross validation. Table 6.5 displays the computed target effectiveness and track purity values for each individual dataset.

Dataset	Target Effectiveness			Track purity		
	GT-P	DGT-P	GT-DGT-P	**GT-P**	DGT-P	GT-DGT-P
B	**90.652**	92.463	98.041	**99.735**	99.735	100
C	**92.906**	93.385	99.463	**87.741**	87.712	100
D	**98.351**	98.542	99.794	**77.169**	77.169	100
E	**94.051**	94.051	100	**92.52**	92.52	100

Table 6.5.: Target effectiveness and track purity for the cell tracker as measured on individual datasets.

Note that table table 6.5 does not include dataset A because the detection module was unable to detect cells in at least two trajectories. This prevented us from training the robust tracklets linker classifier due to the inability to generate negative training examples (i.e. examples of robust tracklets that should not be linked).

The tracker was configured and trained to close gaps of up to 9 frames, with the parameters set to the default values of $\pi_{init} = 1$, $\pi_{term} = 1$, $\pi_{link} = 1$ and $\pi_{FP} = 1$. The exception is dataset E, where the tracker was configured to perform two runs, the first one closing gaps up to length 9, and the

second one up to 20, and the parameters π_{init} and π_{term} were set to 3. This extra gap closing was required for this dataset, because the motion artifacts ensured that the number of very short robust tracklets (of length 1 and 2) was very high. The robust tracklets and generated trajectories for this dataset can be seen in fig. 6.1. Since all robust tracklets of length 1 are simply eliminated, there were few gaps longer than 9, which could be effectively closed in the second iteration.

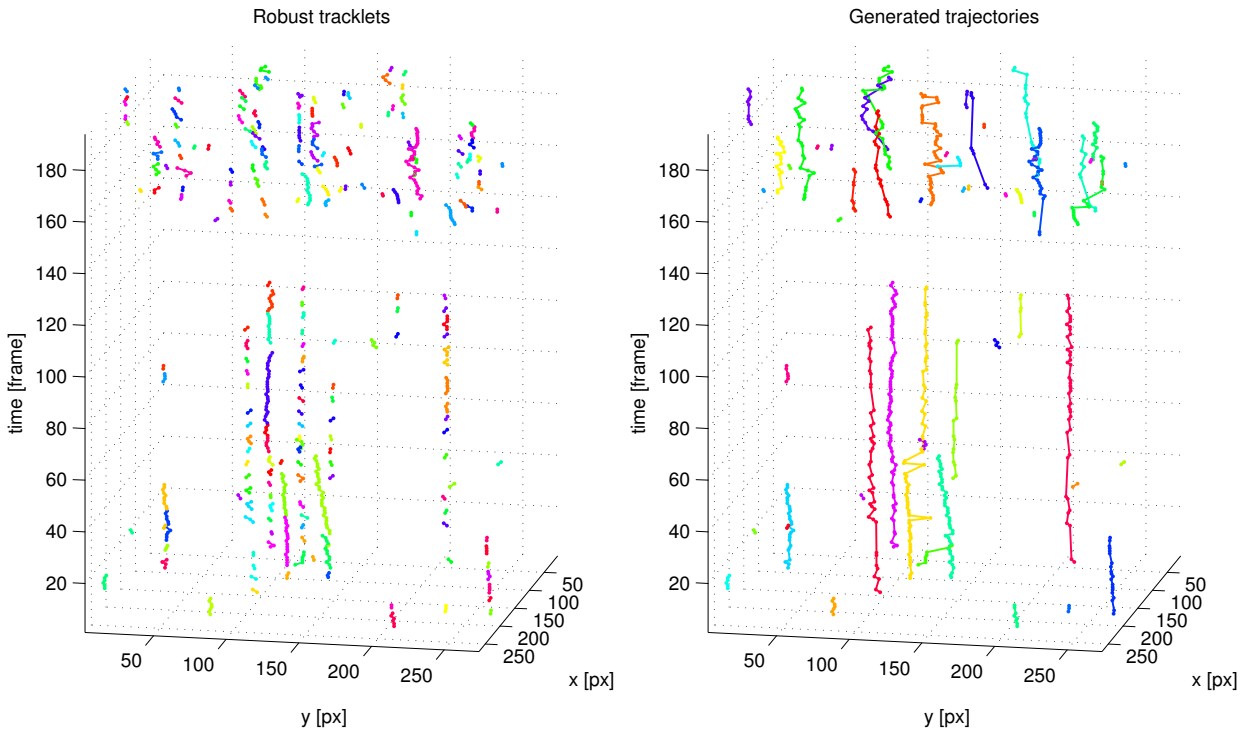


Figure 6.1.: The tracker was able to close gaps up to length 20 in dataset E and connect a large number of robust tracklets into coherent trajectories.

All four datasets exceeded the target effectiveness of 90. Dataset B achieved a track purity of 99.7, dataset C achieved 87.7, dataset D achieved 77.2 and dataset E, 92.5. The lower track purity in dataset D can be attributed to two related reasons. First, this dataset contains a significant number of motion artifacts, arguably more than the other datasets. The second and more likely reason for the low target effectiveness is that several frames had been manually deleted from this dataset. In the original dataset more than 30% of the frames were completely unusable; no cells were clearly discernible. These frames had been simply deleted before any analysis was performed on them. The result was that the original cell trajectories were cut, and these cuts are clearly noticeable in a few places in the image sequence. Cell trajectories where the cut is clearly visible were annotated as 2 distinct trajectories. However, since the tracker wasn't aware of this deletion it linked these individual track segments, resulting in a lower track purity value. This can be clearly seen in fig. 6.2 which shows the most affected trajectories.

The comparison of ground truth annotations and *detected ground truth annotations* is very high in all datasets, which tells us that the cell detector has in general detected most cells within the evaluated trajectories (note that the position of missing detections were completed by linearly interpolating the positions of nearby detections; the high values do not mean that there were very few missed detections). The track purity for DG-DGP-P is always 100 because of the design of *detected ground truth annotations* which dictates that they cannot ever be longer than the ground truth annotations.

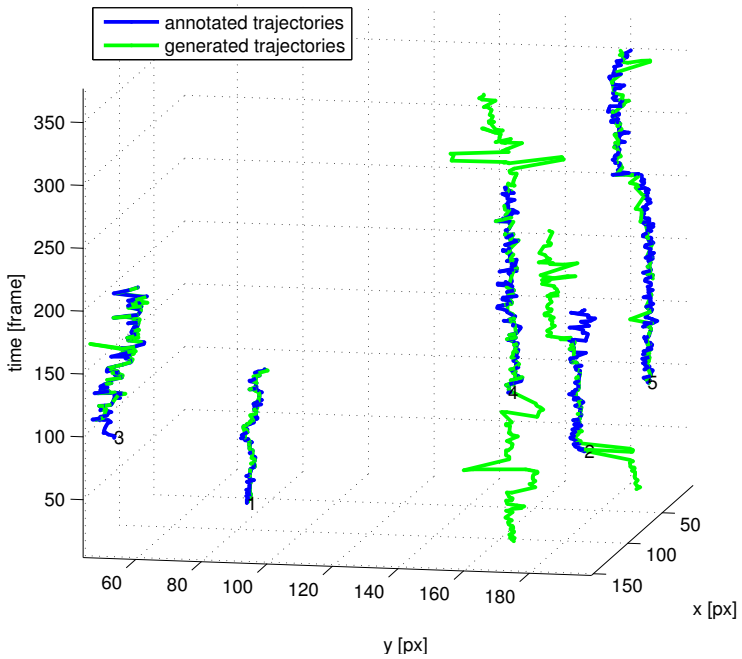


Figure 6.2.: The tracker has linked together trajectory segments that were annotated as independent trajectories in dataset E. The sharp changes in the trajectories are due to manual deletion of a number of frames in this dataset.

Finally, the measurements confirm the expectation that the performance of the generated trajectories relative to the *detected ground truth annotations* is higher than their performance relative to ground truth annotations.

The generated trajectories for each of these datasets are available in figs. D.1 to D.4 in appendix D. These trajectories were obtained from trackers that were trained on all the annotated trajectories.

Training on a combined dataset

The first experiment showed us the ability of the tracker to learn to correctly link trajectories from a single dataset. Often, most cells in the same dataset move in a similar mode of motion (e.g. mostly stagnant, slow translation in one direction, fast translation, frequent direction changes, etc). In this second experiment we trained the tracker on several combined datasets (at each round of the cross validation we only left a single tracklet out of the training set for testing). We were interested in how well the tracker was be able to learn from this vast range of modes of motions, and whether it would be able to successfully track an independent cell that could exhibit any of these modes of motion. The results are shown in table 6.6.

A difficulty in setting up this experiment comes from the fact that the training and testing is performed on the longest tracklets of the entire dataset. Therefore, shorter image sequences are likely to be under-represented when training and testing on the dataset composed of all five datasets. Therefore the experiment was also evaluated on different combinations of combined datasets. Furthermore, the lower performance values can also be attributed to the fact that the adjustments of scaling parameters π_{FP}, π_{link}, π_{init} and π_{term} cannot improve the tracking in the case of testing on combined

Dataset	Target Effectiveness			Track purity		
	GT-P	DGT-P	GT-DGT-P	**GT-P**	DGT-P	GT-DGT-P
ABCDE	**82.1**	82.8	97.8	**81.5**	81.3	100
ABCE	**81.2**	82.9	96.8	**87.4**	87.3	100
ABC	**80.7**	83.0	95.8	**88.2**	88.2	100
AB	**79.5**	82.1	94.9	**99.6**	99.6	100
BCE	**83.7**	85.1	97.6	**85.4**	85.3	100

Table 6.6.: Tracker performance on combined datasets.

datasets, because each of the individual datasets exhibits different characteristics.

Let the combined dataset ABCDE be the basis for our analysis. This dataset achieved a target effectiveness of 82.1 and track purity of 81.5. By eliminating dataset D, which had a lower track purity when training individually, we were able to increase the track purity from 81.5 to 87.4. Peak track purity was achieved when training on the combined datasets A and B. Target effectiveness remained mostly constant in all the experiments.

6.2.3. Computation time

When testing the tracker on individual datasets we also measured the time required to link all the robust tracklets into trajectories. The results are shown in table 6.7.

The detection was performed on a PC with an Intel(R) Core(TM) i7-2600 CPU with a clock frequency of 3.40GHz and 8GB RAM. The MATLAB version used to measure the tracking time was 8.1.0.604 (R2013a) running in Ubuntu Linux 13.04 x64.

Dataset	Total time [seconds]	Time per frame [seconds]	Number of frames per second
B	2.64	0.038	26.5
C	9.79	0.078	12.9
D	19.25	0.050	19.6
E	5.78	0.030	33.6

Table 6.7.: Total computation time, time per frame, and number of frames tracked per second for each dataset.

The average time per frame over all the datasets was 0.049 seconds, which is equivalent to processing 23.1 frames per second. This short computation time is negligible compared to the time taken by the cell detection module. The short tracking computation permits the adjustment of the parameters π_{init}, π_{term}, π_{link} and π_{FP} and the regeneration of the trajectories in just a few seconds.

The slowest part of the cell tracking module is the computation of spatio-temporal features, especially the unoptimized code that computes the Gaussian broadening features (see section 4.7.1).

6.3. Limitations and areas of improvement

Testing the tracker on the studied datasets has given good results as seen in fig. 6.3 which compares the annotated with the generated trajectories in dataset B.

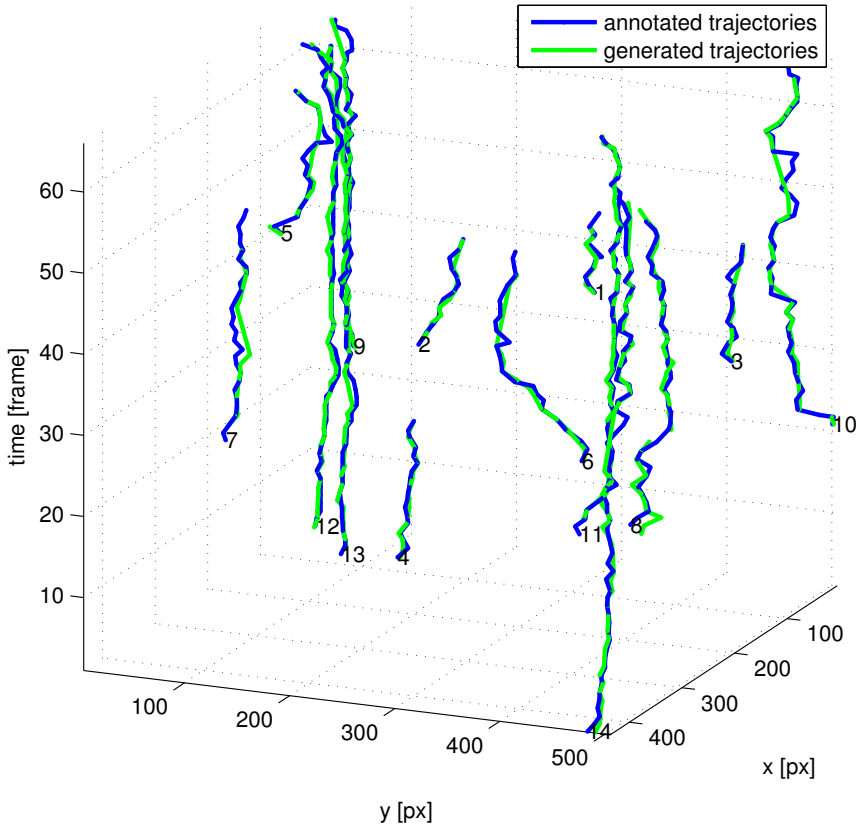


Figure 6.3.: Comparison of annotated and generated trajectories in dataset B.

As we have shown in fig. 6.1 it is also able to efficiently close very large gaps. The tracker attempts to close gaps up to the maximum length which it had been trained to achieve. Since a classifier is used to determine whether two tracklets should be linked, generally equal linking likelihoods could be computed for tracklets that are e.g. 2 frames apart as opposed to tracks that are 7 frames apart. It is possible to train the tracker to close very large gaps, e.g. more than 30 frames, but this increases the likelihood of erroneous connections occurring. For example, if a cell trajectory is composed of three segments (robust tracklets) where the middle one is of length less than 30, the tracker might decide to link the first and third segments of the trajectory, making the middle segment a false positive or a short tracklet, which would be false. To work around this problem the closing of gaps can be performed iteratively, e.g. first for gaps of length 10, then 20 and then 30. This solution works, but remains prone to small errors. An alternative implementation of the system could use a regression model instead of a classifier such that small gaps are given a precedence over long gaps. However, this method requires the identification a suitable way to score the hypotheses.

The tracker is able to deal with some limited jiggling of the frames. However, as we have seen in fig. 6.2, excessive jiggling (in this case caused by the manual deletion of several frames) can result in lower performance. This issue could be resolved with a pre-processing step that would stabilize the images based on the background information present in them before inserting them into the cell

tracker.

Most of the analysed cells move slowly. Although the moving cells were tracked successfully, the tests do not thoroughly test the performance of the tracker on rapidly moving cells, due to the unavailability of such data.

It is worth pointing out that the tests performed on the tracker only measured the performance of the longest trajectories. The reason for this is that a careful annotation and review of all the trajectories in the datasets would be too time consuming and would likely include some significant number of incorrectly linked cells.

Finally, the tracker can sometimes link two independent trajectories that are close together incorrectly, as seen in fig. 6.1. This could be solved by computing more advanced spatio-temporal features or with a post-processing step that would detect these outliers and correct them.

6.4. Summary

In this chapter we have demonstrated the performance of the cell detection and tracking modules. Several experiments on individual and combined datasets were performed to measure how well the algorithms are able to generalize to independent datasets.

The cell detector module was able to correctly detect most of the cells in the input images in all but one dataset. The cell tracker module was then able to use these detection results to generate cell trajectories with good results. The limitations and good points of the tracker are also described and demonstrated.

Additionally, we have measured the computation time of both the detector and tracker. We have shown that our system is able to process at an average of 53 frames per minute (96% of the time spent by the cell detector). Most of the time is spent on the extraction of extremal regions and feature computation of the detection module. Considerably less time is spent on the cell tracker module, where most time is spent on the unoptimized spatio-temporal feature computation.

7. Conclusions and future work

In this final chapter of the report we present some concluding remarks and enumerate a list of possible upgrades to improve the cell detection and tracking modules.

7.1. Conclusion

Improved microscopy imaging techniques allow us to gather large amounts of cell microscopy images. The manual analysis of these images would be an error prone and slow process, requiring days of manual work to review some hundreds of frames. The advances of computer vision algorithms for cell detection and tracking over the past decades magnified by the increased computational power of modern computers allow for an efficient analysis of these datasets in a fraction of the time compared to manual analysis.

The large amount of data that can be analysed with these new methods improve the quality of cell research. They allow for new insights into drug development and a better understanding of the living body. Specifically, this research was focused on enabling the efficient analysis of neutrophils which have a crucial role in the clearance of infections. Their careful analysis could help explain their prominent presence in certain organs, such as the lung. It could also help discover any additional activities that these leukocytes perform, and clarify whether they develop from a single or several neutrophil predecessors.

In this three month project we have identified a pipeline of algorithms that enables the automated analysis of neutrophil behaviour in sometimes noisy images of varying contrasts. This required identifying a robust algorithm to detect cells in these images, and develop a tracking method that would perform well with imperfect cell segmentation and a certain amount of missed detections.

We have upgraded a cell detector developed by Arteta *et al.* [4]. The detector was able to learn how to discriminate between candidate cell regions as either cell or not-cell. We have successfully applied the method to our datasets and improved its speed to make it usable for detecting cells in hundreds of frames. The method was able to robustly detect cells (albeit with some false positive and false negatives) after being trained with a small number of dot-annotated images.

We have developed a tracking method inspired by Bise *et al.* [22] that performs a global decision to join robust tracklets into longer ones. We have modified the original method to heavily rely on the input data, thus eliminating the need for a number of heuristics, which would likely have made the algorithm perform worse when presented with a new dataset. The new approach only requires the

algorithm to be retrained using a small number of annotated trajectories. Although the method is automatic, the user is presented with four parameters, which can be adjusted to improve the generated trajectories. Experiments designed to test the performance of the method have shown promising results.

We have also developed efficient image annotation tools to annotate images with dots and links connecting them. These tools can be used for the annotation of any point-like objects and include features specifically designed to increase the clarity of noisy and low contrast images to facilitate the annotation. Furthermore, the annotation tool can be used to review the cell detection results.

Overall, the detection method was able to detect cell detections with high precision and recall values, thus enabling the tracking method to effectively connect the cell detection results and link robust tracklets into longer trajectories by closing gaps of up to 20 frames.

7.2. Future work

The work developed in this thesis is promising in the manner in which it can deliver automatic cell detection and tracking; however, the method can be further improved, and alternative methods researched. Below we present a list of possible improvements that would likely make the algorithm more robust.

The process of obtaining cell images *in vivo* is challenging especially in moving organs such as the lung, where the motion of the tissue causes the images to jiggle or lose focus. The jiggling can be eliminated using a pre-processing step that would stabilize the images using information hidden in the background, such as blood vessels. This would result in smoother trajectories. Furthermore, this would simplify the computations of spatio-temporal features to train the cell tracker module, as it would be easier to predict the velocity of the cells.

This research was focused on tracking cells in order to facilitate the analysis of cell behaviour. The better suit this goal, it would be beneficial if the system returned a profile for each tracked cell including their appearance and mode of motion. The cell detection module did not focus on an accurate segmentation of the cells. Upgrading the system to accurately segment the cells after they have been identified would not only provide an appearance profile of the cells, but also improve the tracking system.

From the datasets we analysed, the original images for dataset D (described in section 5.1.1) include a large portion that are completely unusable for the tracker because the cells were blurred or invisible for a large number of frames. These frames have been manually removed. It could be beneficial to automate this process by automatically detecting images that are of too low a quality to be usable and subsequently discard of them, whilst leaving a mark with the number of frames skipped. If this step could be performed quickly, the total computation time would be reduced as such frames wouldn't need to be analysed by the cell detector.

The accuracy of the tracking module is heavily dependent on the quality of features that are

computed for the tracklets. We have implemented and tested a broad range of spatio-temporal features, including a linear Kalman filter. Instead of assuming that the tracklet's velocity would be linear with respect to the last few frames of a trajectory, it would be beneficial to use a model that would be able to predict the cell direction more accurately. For example, an interacting multiple models motion filter running several Kalman filters in parallel has been proved to better predict future cells positions [26].

In section 4.8 we mentioned the difficulty of computing spatio-temporal features for robust tracklets with a single cell observation. For this reason, they have not been considered as candidate linking tracklets. However, these short tracklets include information that could improve the detail of generated trajectories. Further research into how to best use these short tracklets could results in trajectories that more accurately conform to the real cell motion.

The developed tracking method has been tested on hundreds of frames of microscopy images. However, as the number of frames is increased, the method is likely to reach a bottleneck due to memory usage. In order to improve the space requirements of the tracker and permit the tracking of thousands of image frames, it would be beneficial to ensure that the processing of tracklets is performed in windows, a few hundred frames at a time. Thus the tracker would first generate tracklets within each window, and then link these tracklets between windows.

In terms of user experience, a simple graphic user interface to load the image sequences and start the tracking process would greatly improve the approachability of the method to a larger non-technical audience. A more advanced user interface could also include options to train and evaluated a new model.

Finally, whilst the above improvements relate to the software, it is expected that the imaging technique will also improve. This could alleviate the problem of jiggling cells and out of focus images, thus reducing the need to overcome these imaging limitations in the software.

Appendices

A. Installation instructions for the image annotations tools

Both the Image Annotation Tool and the Trajectory Annotation Viewer are provided as Windows executables that need to be installed. The installation only requires an active internet connection to automatically download the MATLAB Compiler Runtime[1].

[1] http://www.mathworks.co.uk/products/compiler/mcr/

B. Installation instructions and usage example for the cell detector and tracker

B.1. Installation instructions

The source code of the cell detector and tracker requires the manual installation of several dependencies which have to be manually downloaded and added to the MATLAB search path[1]. The dependencies, including the tested version numbers, are:

1. VLFeat[2] (version 9.18), an open source library of popular computer vision algorithms.
2. svm-struct-matlab[3] (version 1.2), a MATLAB wrapper for SVM$^{\text{struct}}$.
3. The MATLAB code for the inference in the pylon model[4]
4. QPBO[5] (version v1.31) from Vladimir Kolmogorov.

A Bash script is available in the source code in *cell_ tracker/+detector/setup/setup.sh* to automate the installation of the required dependencies. Before running the script it is advisable to review it and configure the installation path in the first few lines of the script.

The code has been tested in MATLAB R2014a under Ubuntu 14.04.1 LTS. However, the code should work also in version R2013b. Additionally the following MATLAB toolboxes are needed:

1. Statistics Toolbox
2. Image Processing Toolbox
3. Computer Vision System Toolbox
4. Neural Network Toolbox
5. Optimization Toolbox

[1] `http://www.mathworks.co.uk/help/matlab/matlab_env/what-is-the-matlab-search-path.html`
[2] `http://www.vlfeat.org/`
[3] `http://www.robots.ox.ac.uk/~vedaldi/code/svm-struct-matlab.html`
[4] `http://www.robots.ox.ac.uk/~vilem/`
[5] `http://pub.ist.ac.at/~vnk/software.html`

6. Parallel Computing Toolbox

7. System Identification Toolbox

B.2. Usage example

In this section we describe how to configure the system to train and test the cell detector and tracker on your own image sequences.

First, configure the data directories in *dataFolders.m*. This will include appending a block of code similar to this:

```
% dataFolders.m
% ...
case 7 % dataset ID: use the next number of the sequence
    dotFolder = '<folderNamesWithDotAnnotations>';
    linkFolder = '<folderNamesWithLinkAnnotations>';
    outFolder = '<outputFolderName>';
    numAnnotatedFrames = 30;      % the number of dot-annotated frames
    numAnnotatedTrajectories = 4; % the total number of annotated trajectories
% ...
```

The *dotFolder* should contain images in the *pgm* format named sequentially as *im001.pgm* and correspondingly named *mat* files with dot-annotations for the first frames of the sequence, as indicated by *numAnnotatedFrames*. The *mat* files can be generated by the Image Annotation Tool.

The *linkFolder* should contain images and annotations files in the same format as the *dotFolder*. The *mat* files should not only contain annotated dots but also at least a few fully annotated cell trajectories (as indicated by *numAnnotatedTrajectories*) required to train the tracker. Note that in general *dotFolder* and *linkFolder* can be the same.

The *outFolder* is the folder where the detector and tracker will output temporary and final results.

Second, in the file named *runner.m* it is required to insert the dataset ID as above, and select which of the actions should be executed:

```
% runner.m
% ...
datasetIDs   = [7];    % Look into dataFolders.m

trainDetector = true;  % Use annotations in dotFolder to train a new detector
trainTracker = true;   % Use annotations in linkFolder to train a new tracker

testDetector = true;   % Evaluate the trained detector on the full image sequence
testTracker  = true;   % Evaluate the trained tracker on the full image sequence
```

```
showTracks   = true;   % Display a figure containing the generated trajectories
% ...
```

Note that it is possible to insert several dataset IDs and the code will be executed on each of them sequentially. The file *runner.m* can then be executed using MATLAB.

C. Cell detection results

This chapter includes three-dimensional figures of the results of the cell detection module on each of the studied datasets.

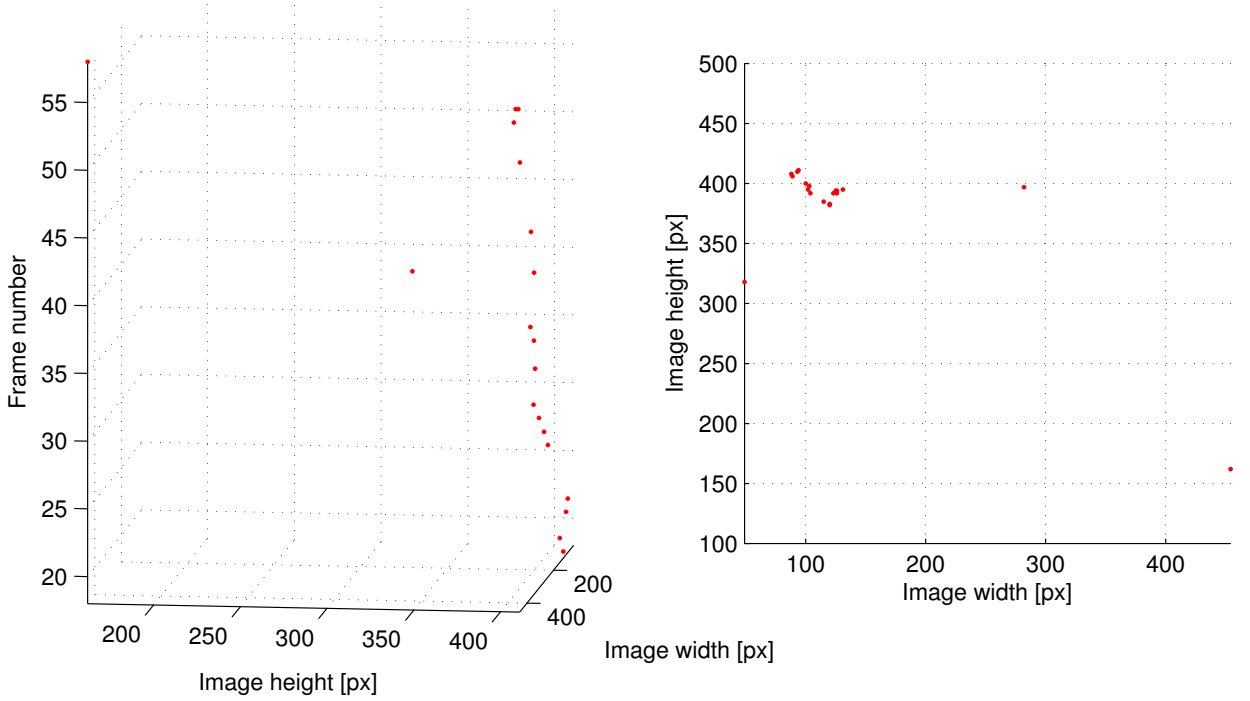


Figure C.1.: Three-dimensional view and orthographic projection from the top of the detection results for dataset A.

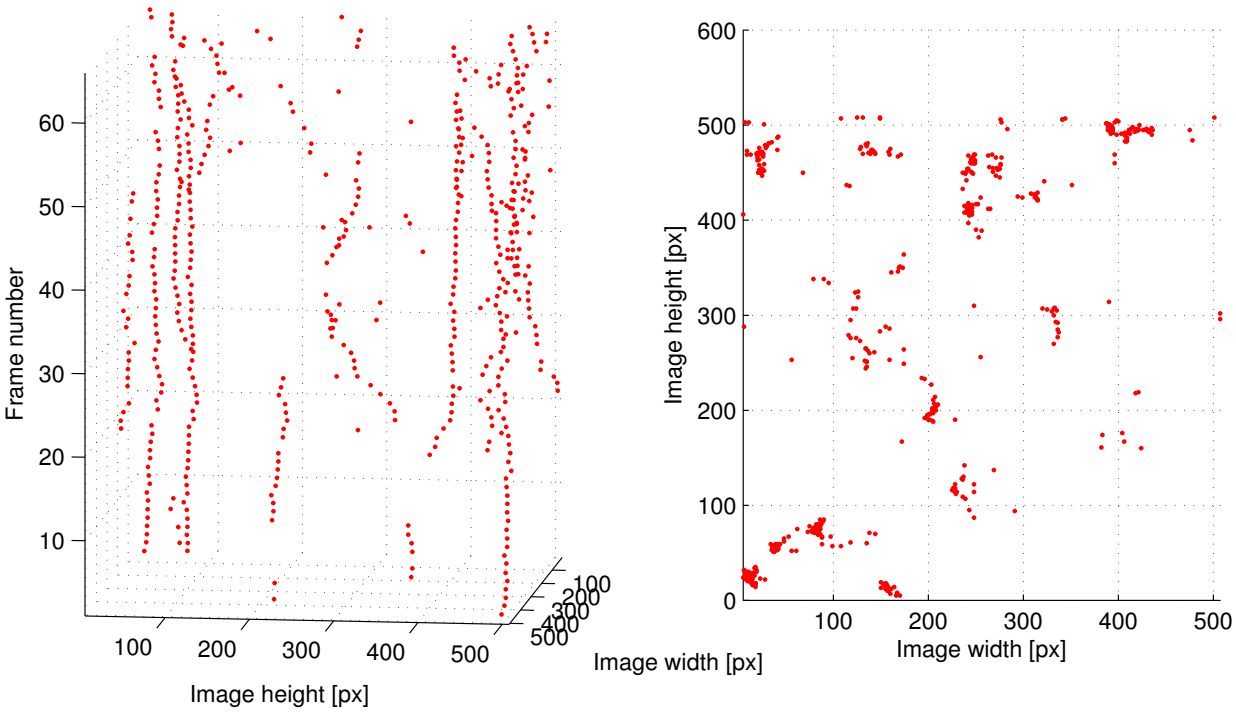


Figure C.2.: Three-dimensional view and orthographic projection from the top of the detection results for dataset B.

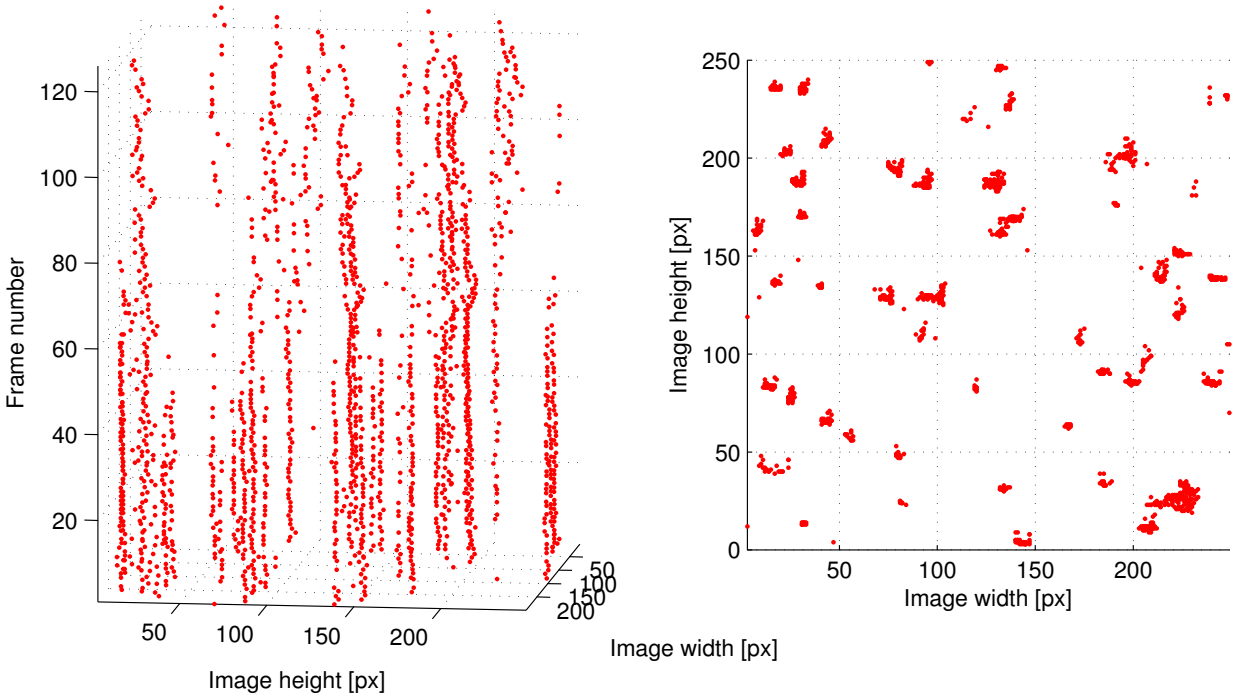


Figure C.3.: Three-dimensional view and orthographic projection from the top of the detection results for dataset C.

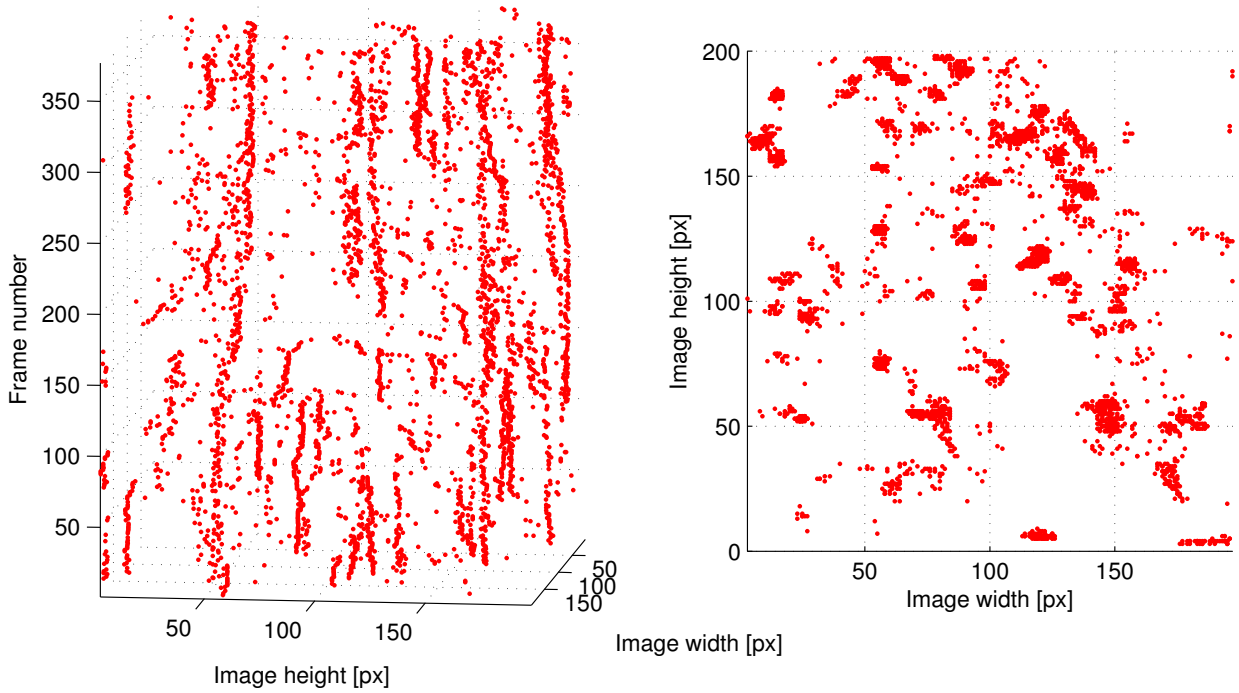


Figure C.4.: Three-dimensional view and orthographic projection from the top of the detection results for dataset D.

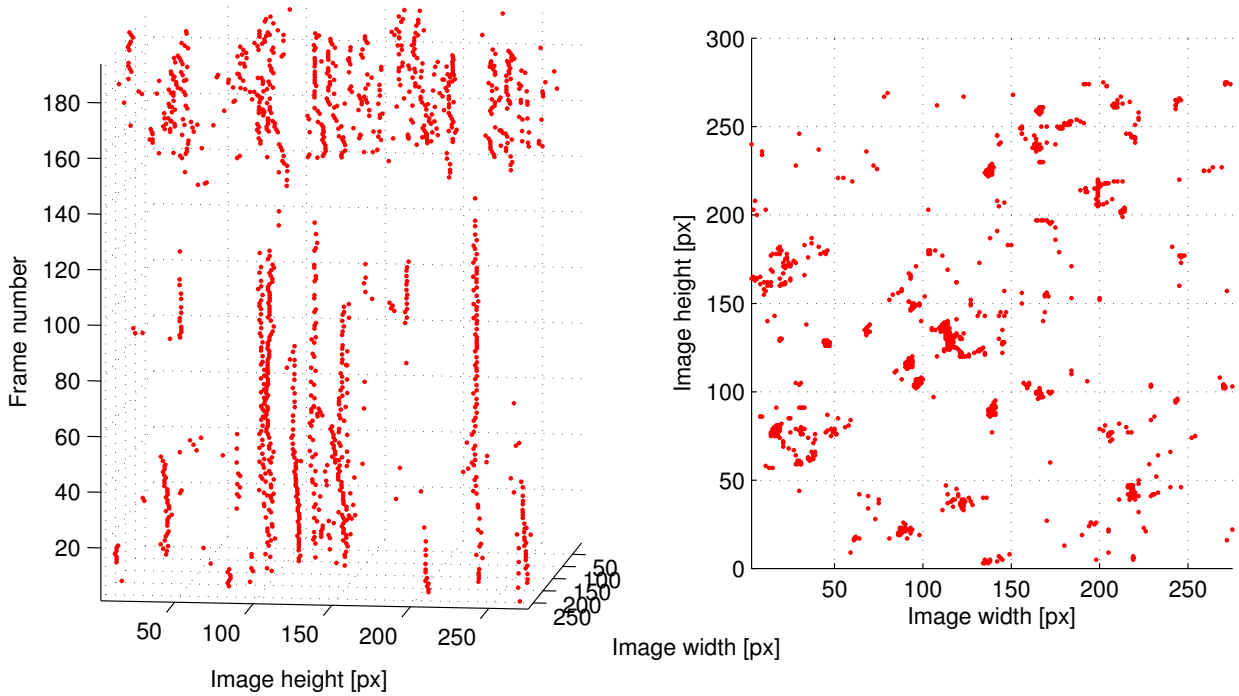


Figure C.5.: Three-dimensional view and orthographic projection from the top of the detection results for dataset E.

D. Cell tracking results

This chapter includes three-dimensional figures of the trajectories generated by the cell tracking module on four of the studied datasets.

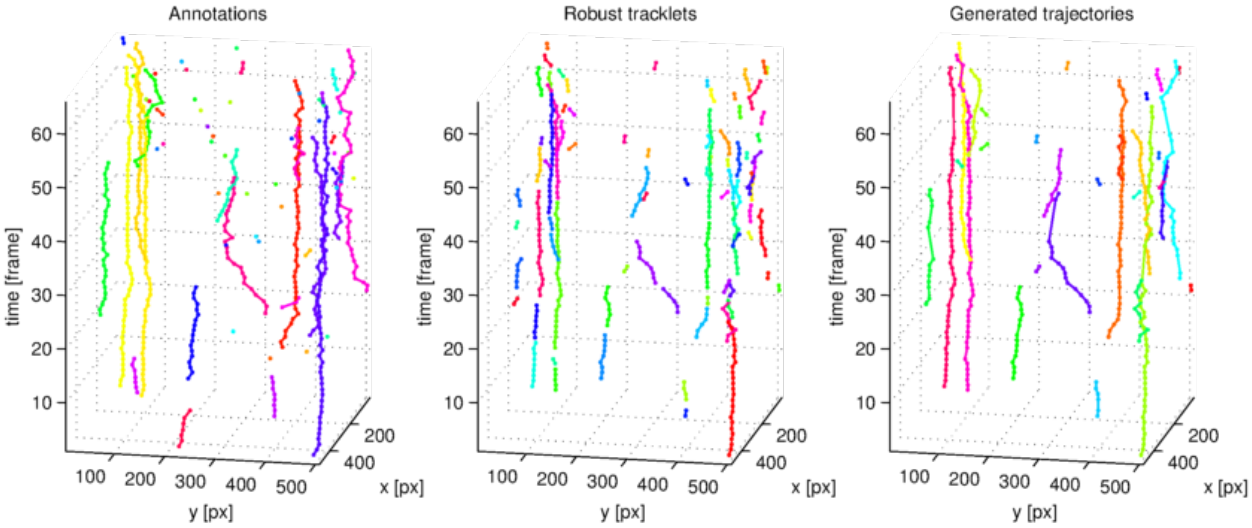


Figure D.1.: Generated trajectories for dataset B. The parameters of the tracker were set to $\pi_{init} = 1$, $\pi_{term} = 1$, $\pi_{link} = 1$ and $\pi_{FP} = 1$ and configured to close gaps up to size 9.

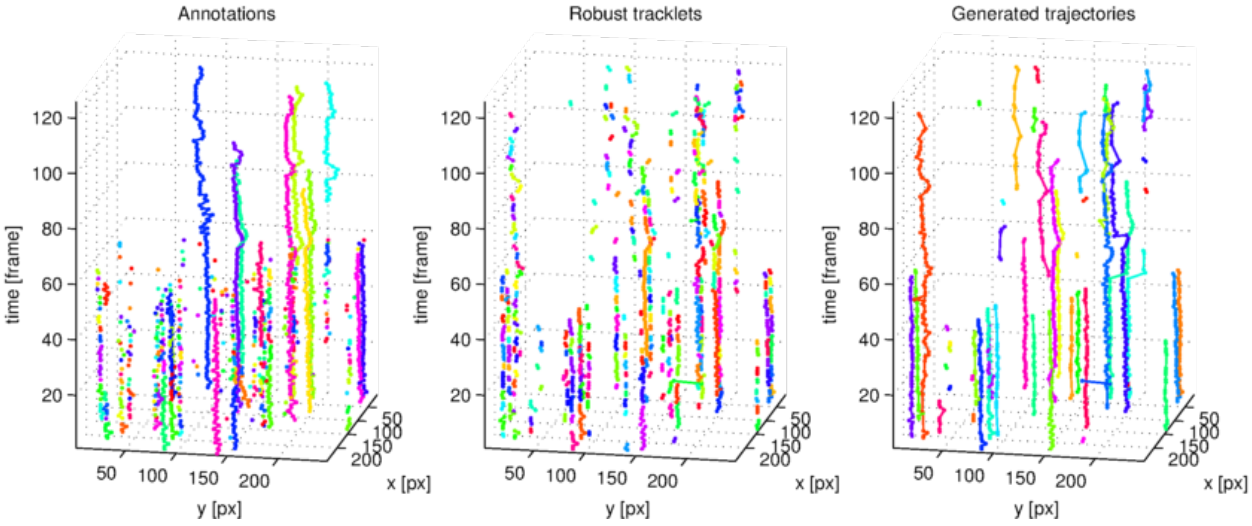


Figure D.2.: Generated trajectories for dataset C. The parameters of the tracker were set to $\pi_{init} = 1$, $\pi_{term} = 1$, $\pi_{link} = 1$ and $\pi_{FP} = 1$ and configured to close gaps up to size 9.

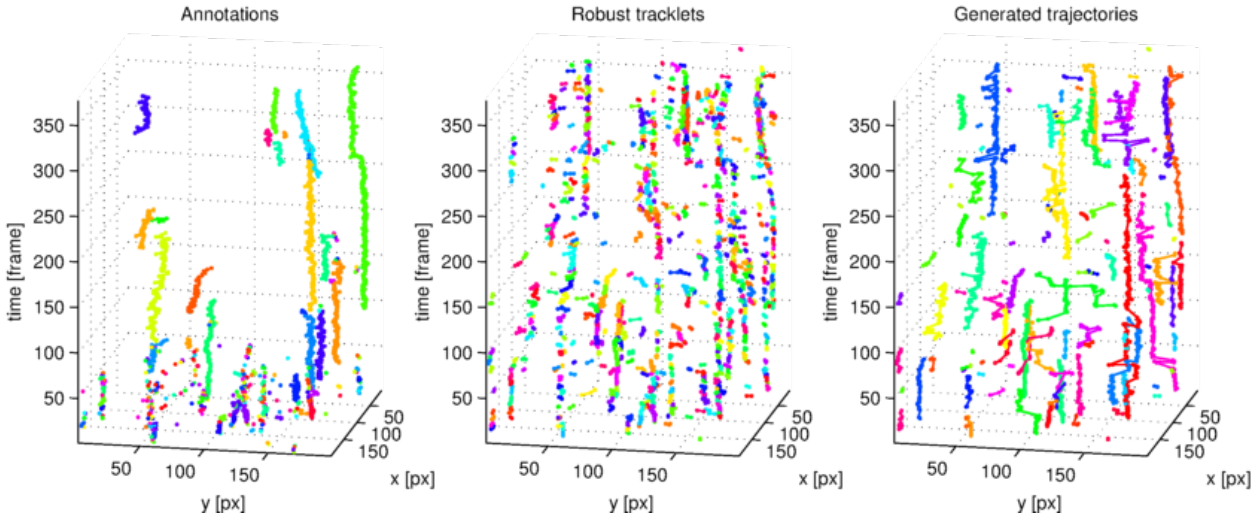


Figure D.3.: Generated trajectories for dataset D. The parameters of the tracker were set to $\pi_{init} = 1$, $\pi_{term} = 1$, $\pi_{link} = 1$ and $\pi_{FP} = 1$ and configured to close gaps up to size 9.

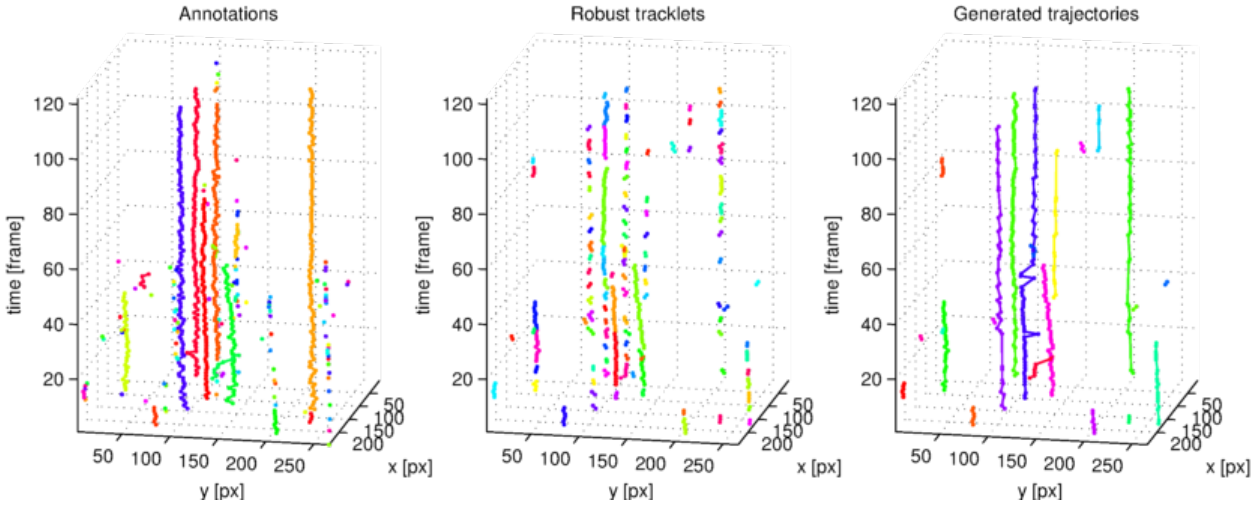


Figure D.4.: Generated trajectories for dataset E. The parameters of the tracker were set to $\pi_{init} = 3$, $\pi_{term} = 3$, $\pi_{link} = 1$ and $\pi_{FP} = 1$ and configured to close gaps iteratively up to size 9 and then up to size 20.

Bibliography

[1] P. K. Elzbieta Kolaczkowska, "Neutrophil recruitment and function in health and inflammation," 2013. 7

[2] J. Pillay, I. den Braber, N. Vrisekoop, L. M. Kwast, R. J. de Boer, J. A. M. Borghans, K. Tesselaar, and L. Koenderman, "In vivo labeling with 2h2o reveals a human neutrophil lifespan of 5.4 days," *Blood*, vol. 116, no. 4, pp. 625–627, 2010. 7

[3] P. S. Tofts, T. Chevassut, M. Cutajar, N. G. Dowell, and A. M. Peters, "Doubts concerning the recently reported human neutrophil lifespan of 5.4 days," *Blood*, vol. 117, no. 22, pp. 6050–6052, 2011. 7

[4] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman, "Learning to detect cells using non-extremal regions," in *Proceedings of the 15th International Conference on Medical Image Computing and Computer-Assisted Intervention - Volume Part I*, MICCAI'12, (Berlin, Heidelberg), pp. 348–356, Springer-Verlag, 2012. 8, 9, 11, 16, 17, 18, 19, 21, 45, 50, 59

[5] Y. Chen, K. Biddell, A. Sun, P. Relue, and J. Johnson, "An automatic cell counting method for optical images," in *[Engineering in Medicine and Biology, 1999. 21st Annual Conference and the 1999 Annual Fall Meetring of the Biomedical Engineering Society] BMES/EMBS Conference, 1999. Proceedings of the First Joint*, vol. 2, pp. 819 vol.2–, Oct 1999. 10

[6] X. Chen, X. Zhou, and S. Wong, "Automated segmentation, classification, and tracking of cancer cell nuclei in time-lapse microscopy," *IEEE Transactions on Biomedical Engineering*, vol. 53, pp. 762–766, April 2006. 10, 13

[7] L. Vincent, "Morphological grayscale reconstruction in image analysis: applications and efficient algorithms," *IEEE Transactions on Image Processing*, vol. 2, pp. 176–201, Apr 1993. 10

[8] J. Serra, *Image Analysis and Mathematical Morphology*. Orlando, FL, USA: Academic Press, Inc., 1983. 10

[9] D. Mukherjee, N. Ray, and S. Acton, "Level set analysis for leukocyte detection and tracking," *IEEE Transactions on Image Processing*, vol. 13, pp. 562–572, April 2004. 11, 12

[10] C. Tang, Y. Wang, and Y. Cui, "Tracking of active cells based on kalman filter in time lapse of image sequences of neuron stem cells," in *The 2011 International Conference on Bioinformatics and Computational Biology*, BIOCOMP'11, pp. 301–307, 2011. 11, 14

[11] D. Xu and L. Ma., "Segmentation of image sequences of neuron stem cells based on level-set algorithm combined with local gray threshold," Master's thesis, Harbin Engineering University, 2010. 11

[12] C. Arteta, V. S. Lempitsky, J. A. Noble, and A. Zisserman, "Learning to detect partially overlapping instances," in *CVPR*, pp. 3230–3237, IEEE, 2013. 11, 12, 18

[13] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," in *Proceedings of the British Machine Vision Conference*, pp. 36.1–36.10, BMVA Press, 2002. doi:10.5244/C.16.36. 11, 17

[14] T. Joachims, T. Finley, and C.-N. J. Yu, "Cutting-plane training of structural svms," *Machine Learning*, vol. 77, pp. 27–59, Oct. 2009. 11

[15] S. Huh, *Toward an Automated System for the Analysis of Cell Behavior: Cellular Event Detection and Cell Tracking in Time-lapse Live Cell Microscopy.* PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March 2013. 13, 52

[16] D. House, M. Walker, Z. Wu, J. Wong, and M. Betke, "Tracking of cell populations to understand their spatio-temporal behavior in response to physical stimuli," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009*, pp. 186–193, June 2009. 13

[17] B. Xu, M. Lu, P. Zhu, Q. Chen, and X. Wang, "Multiple cell tracking using ant estimator," in *2012 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pp. 13–17, Nov 2012. 13

[18] K. Li and T. Kanade, "Cell population tracking and lineage construction using multiple-model dynamics filters and spatiotemporal optimization," in *Proceedings of the 2nd International Workshop on Microscopic Image Analysis with Applications in Biology (MIAAB)*, September 2007. 14

[19] A. Massoudi, D. Semenovich, and A. Sowmya, "Cell tracking and mitosis detection using splitting flow networks in phase-contrast imaging," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pp. 5310–5313, Aug 2012. 14

[20] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, June 2008. 14, 26

[21] C. Huang, B. Wu, and R. Nevatia, "Robust object tracking by hierarchical association of detection responses," in *Computer Vision - ECCV 2008* (D. Forsyth, P. Torr, and A. Zisserman, eds.), vol. 5303 of *Lecture Notes in Computer Science*, pp. 788–801, Springer Berlin Heidelberg, 2008. 15, 26

[22] R. Bise, Z. Yin, and T. Kanade, "Reliable cell tracking by global data association," in *ISBI*, pp. 1004–1010, IEEE, 2011. 15, 23, 26, 30, 52, 59

[23] H. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955. 15

[24] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms." `http://www.vlfeat.org/`, 2008. 17

[25] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, (New York, NY, USA), pp. 104–, ACM, 2004. 19

[26] K. Li, E. D. Miller, M. Chen, T. Kanade, L. E. Weiss, and P. G. Campbell, "Cell population tracking and lineage construction with spatiotemporal context," *Medical Image Analysis*, vol. 12, no. 5, pp. 546 – 566, 2008. Special issue on the 10th international conference on medical imaging and computer assisted intervention. 23, 61

[27] R. Bise, T. Kanade, Z. Yin, and S. I. Huh, "Automatic cell tracking applied to analysis of cell migration in wound healing assay," in *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pp. 6174–6179, Aug 2011. 23

[28] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960. 33

[29] M. Looney, E. Thornton, D. Sen, W. Lamm, R. Glenny, and M. Krummel, "Stabilized imaging of immune surveillance in the mouse lung," *Nature Methods*, vol. 8, no. 5, pp. 91–6, 2011. 36

[30] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Artech House radar library, Artech House, 1999. 52

[31] S. Eom, S. I. Huh, D. F. E. Ker, R. Bise, T. Kanade, and P. Campbell, "Tracking of hematopoietic stem cells in microscopy images for lineage determination," vol. 6, no. 01-09, p. 9, 2007. 52