Karlsruhe Institute of Technology

Multiple Object Tracking in Light Microscopy Images Using Graph-based and Deep Learning Methods

Zur Erlangung des akademischen Grades einer

Doktorin der Ingenieurwissenschaften (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau
des Karlsruher Instituts für Technologie (KIT)

genehmigte
DISSERTATION

von

M.Sc. Katharina Löffler

aus Heiligenstadt

Hauptreferent: apl. Prof. Dr.-Ing. Ralf Mikut
Korreferenten: Prof. Dr. Britta Nestler
Prof. Dr. Uwe Strähle

Tag der mündlichen Prüfung: 19. Juni 2023

Zusammenfassung

Multi-Objekt-Tracking (MOT) ist ein Problem der Bildanalyse, welches die Lokalisierung und Verknüpfung von Objekten in einer Bildsequenz über die Zeit umfasst, mit zahlreichen Anwendungen in Bereichen wie autonomes Fahren, Robotik oder Überwachung. Neben technischen Anwendungsgebieten besteht auch ein großer Bedarf an MOT in biomedizinischen Anwendungen. So können beispielsweise Experimente, die mittels Lichtmikroskopie über mehrere Stunden oder Tage hinweg erfasst wurden, Hunderte oder sogar Tausende von ähnlich aussehenden Objekten enthalten, was eine manuelle Analyse unmöglich macht. Um jedoch zuverlässige Schlussfolgerungen aus den verfolgten Objekten abzuleiten, ist eine hohe Qualität der prädizierten Trajektorien erforderlich. Daher werden domänenspezifische MOT-Ansätze benötigt, die in der Lage sind, die Besonderheiten von lichtmikroskopischen Daten zu berücksichtigen. In dieser Arbeit werden daher zwei neuartige Methoden für das MOT-Problem in Lichtmikroskopie-Bildern erarbeitet sowie Ansätze zum Vergleich der Tracking-Methoden vorgestellt.

Um die Performanz der Tracking-Methode von der Qualität der Segmentierung zu unterscheiden, wird ein Ansatz vorgeschlagen, der es ermöglicht die Tracking-Methode getrennt von der Segmentierung zu analysieren, was auch eine Untersuchung der Robustheit von Tracking-Methoden gegeben verschlechterter Segmentierungsdaten erlaubt. Des Weiteren wird eine graphbasierte Tracking-Methode vorgeschlagen, welche eine Brücke zwischen einfach anzuwendenden, aber weniger performanten Tracking-Methoden und performanten Tracking-Methoden mit vielen schwer einstellbaren Parametern schlägt. Die vorgeschlagene Tracking-Methode hat nur wenige manuell einstellbare Parameter und ist einfach auf 2D- und 3D-Datensätze anwendbar. Durch die Modellierung von Vorwissen über die Form des Tracking-Graphen ist die vorgeschlagene Tracking-Methode außerdem in der Lage, bestimmte Arten von Segmentierungsfehlern automatisch zu korrigieren. Darüber hinaus wird ein auf Deep Learning basierender Ansatz vorgeschlagen, der die Aufgabe der Instanzsegmentierung und Objektverfolgung gleichzeitig in einem einzigen neuronalen Netzwerk erlernt. Außerdem lernt der vorgeschlagene Ansatz Repräsentationen zu prädizieren, die für den Menschen verständlich sind. Um die Performanz der beiden vorgeschlagenen Tracking-Methoden im Vergleich zu anderen aktuellen, domänenspezifischen Tracking-Ansätzen zu zeigen, werden sie auf einen domänenspezifischen Benchmark angewendet. Darüber hinaus werden weitere Bewertungskriterien für Tracking-Methoden eingeführt, welche zum Vergleich der beiden vorgeschlagenen Tracking-Methoden herangezogen werden.

Abstract

Multiple Object Tracking (MOT) is a problem in image analysis that involves the localization and linking of objects in an image sequence over time with numerous applications in fields such as autonomous driving, robotics, or surveillance. Besides technical fields of application, also there is a tremendous need for MOT in biomedical applications. For instance, experiments captured with light microscopy imaging over several hours or days can contain hundreds or even thousands of objects similar in appearance, which makes manual analysis infeasible. However, to deduce reliable conclusions from the tracked objects, a high tracking quality is needed. Therefore, domain-specific MOT approaches are required which are able to manage the distinctiveness of light microscopy data. Hence, this thesis proposes two novel tracking methods for the task of MOT in light microscopy images as well as provides approaches to compare the performance of tracking methods.

To distinguish the performance of the tracking method from the segmentation quality, an approach is proposed to analyze the tracking method separately from the segmentation approach, which also allows the investigation of the robustness of a tracking method when provided with degraded segmentation data. Next, a graph-based tracking method is proposed that bridges the gap between simple to apply yet inferior-performing tracking methods and parameter-heavy, well-performing tracking methods which are difficult to tune. The proposed tracking method has few manually tunable parameters and is simple to apply to 2D and 3D datasets. Moreover, by modeling prior knowledge on the shape of the tracking graph the proposed tracking is capable to correct certain types of segmentation errors automatically. In addition, a deep learning based approach is proposed which learns the task of instance segmentation and tracking simultaneously in a single neural network. Also, the proposed approach learns to predict representations that are comprehensible to humans. Moreover, to show the performance of the two proposed tracking methods compared to other domain-specific state-of-the-art methods, they are applied to a domain-specific benchmark. In addition, further evaluation criteria are proposed which are applied to compare the two proposed tracking methods.

Acknowledgments

Wow. I am a bit astonished that this long journey is coming to an end. Looking back at April 2017, when my journey with cells and my time at the institute started as a student assistant, I wouldn't have thought this topic would accompany me that long. Sometimes I have the feeling that during this journey I have learned more about people and myself than my actual research topic, so I would like to thank everyone for the many, sometimes unexpected and astounding, but in the end very valuable lessons that I have been taught.

Thanks to Benni and Hannes who sparked my fascination for cells by giving me the opportunity to start my journey at IAI as a student assistant. Thanks to Andy who patiently supervised my Master's thesis, which was of course cell related.

My tremendous gratitude goes out to my supervisor Ralf Mikut for giving me the opportunity to do a dissertation under his supervision. Thanks for the numerous intense but calm discussions, your fairness, reliability, and openness; and of course your thorough reading with plentiful comments in papers and especially this document that sometimes put me at the brink of despair but usually were worth addressing.

Thanks to the graduate school HIDSS4Health for the financial support and the numerous possibilities for further education. I especially like to thank Ines Reinartz and Nicole Merkle for their efforts in organizing events and supporting doctoral researchers in the scope of HIDSS4Health. I would also like to thank my life science PI Uwe Strähle, and my thesis advisory committee members Peter Sanders and Bogdan Savchynskyy for their efforts in advising this thesis, as well as Uwe Strähle and Britta Nestler for reviewing this thesis.

Thanks to Izhar Bar-Gad for hosting me in Israel and providing me with the opportunity to get a glimpse into neuroscience. I would also like to thank Orel, Kate, and Yuval for making me feel very welcome in Izhar's group during my short stay.

Thanks to my two students Michael and Thomas, who I had the pleasure to work with. A special thank you to all my colleagues, especially Andy, Claudia, Ines, Kaleb, Lisa, Marian, Moritz, Nicole, Oli, and Tim. Thanks for the exchange, discussions, occasional laughter, and chocolate breaks.

Thanks to my parents for accepting my career choice of becoming an engineer and allowing me to find my own path in life. My deep gratitude goes out to Chris and all of my friends, who endured my fascination for bad puns and sometimes even shared a burst of laughter about them with me. Thank you for all the walks, hikes, couch potato chocolate times, and ridiculously long phone calls.

And finally, thanks to the numerous other people that are not mentioned here explicitly who accompanied me on this journey!

Karlsruhe, February 2023 *Katharina Löffler*

Contents

List of Figures

List of Tables

1. Introduction

Image analysis provides the key to extract quantitative insights from observations. One task in image analysis is Multiple Object Tracking (MOT) which aims at localizing objects of interest in an image sequence and linking corresponding objects over time. The accurate tracking of objects provides the basis for further tasks such as behavior analysis, path planning, or forecasting of movements which have numerous applications in for instance autonomous driving, surveillance, or human-machine interaction. Besides these traditional fields of application for MOT, biomedical image processing is another application domain with a tremendous need for automated tracking methods.

One approach to acquire images in the biomedical domain is light microscopy which allows to observe objects from the size of a few dozen nanometers up to millimeters [1, 2]. However, the data captured with light microscopy are very different to the data captured in traditional MOT applications. (i) First of all, large-scale experiments running for several hours or even days acquire image data up to the terabyte range [3, 4, 5] that need to be processed. (ii) Instead of tracking a few dozen objects, biomedical experiments capture hundreds or even thousands of objects that need to be tracked simultaneously [6]. (iii) Moreover, the captured objects are very similar in appearance [7]. (iv) Objects can divide, hence, a correct linking of predecessor-successor objects is required [7]. (v) To aggravate this task further, often the temporal resolution of the image sequence and contrast of the images are low. Thus, due to the distinctiveness of biomedical data, domain-specific MOT approaches are required.

While the task of MOT in light microscopy image sequences is challenging, solving this task enables large-scale analysis of object behavior over time, and thus holds the exceptional potential to derive new insights. In the following, cell migration [8, 9], a process of particular interest in the life sciences is chosen to illustrate the potential of MOT in the biomedical domain. Cell migration is essential for numerous biological processes such as embryogenesis [10, 11], wound healing [12], or immune response [13]. Moreover, disruptions in cell migration can cause malformation [14] and diseases such as autoimmune disease [15] or metastasis [16]. The automated tracking of cells yields the potential to better understand the forces driving cell migration [17], to reveal wrong assumptions on cell lines [18], to investigate the effectiveness of cancer therapies [19], or to better understand immune response [20]. Moreover, the increased understanding of the processes influencing cell behavior paves the way to understanding illnesses better and finding cures for diseases. However, to extract reliable conclusions from the captured data, a high tracking quality is required.

The present thesis provides contributions towards the long-term aim of virtually error-free instance segmentation and tracking by proposing novel tracking methods as well as approaches to compare their performance. When comparing the performance of different tracking-by-detection approaches it is difficult to distinguish the influence of the segmentation approach from the tracking approach on the final performance. Hence, this thesis proposes a method to analyze tracking approaches separately from the segmentation method. In addition, tracking methods can be roughly grouped into simple to apply, yet inferior performing methods and tracking approaches that perform well on benchmark datasets, however, have many parameters and are therefore difficult to tune to a new analysis task for end users. Therefore, the present thesis proposes a graph-based tracking approach with few manually tunable parameters, which is simple to apply to 2D and 3D datasets and yet capable to correct certain types of segmentation errors automatically. With deep learning dominating the task of instance segmentation in recent years, this work proposes to learn the task of instance segmentation and tracking jointly in a single neural network. The representations learned by a neural network usually have no intuitive interpretation, therefore, the presented neural network learns to predict representations that are comprehensible to humans. To evaluate the performance of the two proposed tracking methods, they are compared on a benchmark dataset with other state-of-the-art MOT methods. Moreover, while the tracking quality is important, depending on the application also other selection criteria are important. Hence, additional criteria for comparison, such as the effort required to apply the MOT approach to a new dataset and the robustness of the parametrization, are proposed which are used to compare the two proposed tracking methods.

The remainder of this chapter is organized as follows. Section 1.1 provides an overview of the theoretical background and related work for this thesis covering an overview of the image analysis pipeline in light microscopy images and an overview of the challenges, evaluation metrics, and methods for MOT. Next, Section 1.2 introduces open questions in the field of MOT in light microscopy images. Finally, Section 1.3 summarizes the contributions of this thesis to address the stated questions and provides an outline of the thesis.

1.1. Theoretical Background and Related Work

In the following, the task of MOT is set in the context of the overall goal of automated image analysis. Moreover, a brief overview of different MOT approaches in the domain of light microscopy images is provided and the challenges of MOT in light microscopy images are illustrated. Next, methods of how errors in MOT can be detected and corrected are summarized, and finally, to compare MOT approaches evaluation metrics and available benchmark datasets are introduced.

1.1.1. The Image Analysis Pipeline – From Data Acquisition to Gaining Insights

To extract insights from a microscopy experiment, a multi-step pipeline is used, shown in Figure 1.1, which spans from image acquisition up to knowledge extraction. In recent years, numerous tools have been proposed to aid in the different processing steps [21, 22] and best practice guidelines to set up microscopy image analysis pipelines have been proposed [23]. In the following, the steps of the image analysis pipeline are introduced in more detail.

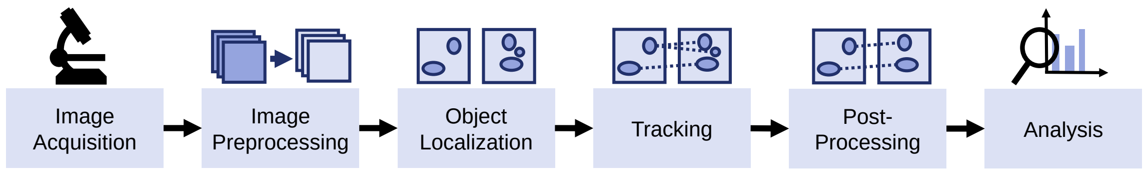


Figure 1.1.: Analysis pipeline for light microscopy images.

Image Acquisition The images captured during the experiment are the main source of information, all subsequent steps of the image analysis pipeline are dependent on it [24]. To acquire images, a variety of light microscopy approaches exist spanning from bright field microscopy over fluorescence microscopy [1, 25, 26] up to microscopy methods allowing super-resolution [27]. Introductions on the working principles of different light microscopes are given in [1, 24, 26].

In general, acquiring 'good' images means having a reasonable spatial and temporal resolution as well as having a good distinction between sample and background in each image, also referred to as a high Signal-to-Noise Ratio (SNR). Noise is introduced to the captured images through many sources. For instance in fluorescence microscopy noise occurs due to autofluorescence, shot noise, light from the surrounding environment, and noise induced by the camera sensor and during readout [24]. Two approaches to increase the SNR are binning and increasing the exposure time [24]. While binning reduces the spatial resolution [24], increasing the exposure time reduces the temporal resolution when imaging multiple samples in parallel and can cause saturation, photobleaching or even phototoxicity [28]. Phototoxicity is the effect of light-induced damage on the sample [24], which can even alter the behavior of the observed sample [29, 30]. For instance, decreasing the wavelength, which results in higher energy, and increasing the temporal resolution between frames, which results in larger exposure times, can increase phototoxicity. Especially in 3D, phototoxicity can be an even bigger challenge, as apart from light sheet microscopy, the whole image region is illuminated repeatedly instead of illuminating only the slice of interest [31, 32]. Hence, image acquisition is a tradeoff between capturing images that contain as much information as possible while on the other hand avoiding altering the sample due to phototoxicity which can potentially result in deriving spurious conclusions. Another tradeoff can occur between the temporal resolution of a single experiment and the number of experiments acquired in parallel [28].

Besides phototoxicity, further influences affect the captured images and can hence impact the conclusions derived from the experiment. For instance, the sample can distort the light by absorption, scattering, refraction, or autofluorescence [33]. An extensive overview of additional sources affecting the captured images is given in [33].

Image Preprocessing Next, the captured images can be preprocessed to correct uneven illumination, enhance contrast, and reduce noise. Uneven illumination can be caused by the optical system appearing as a bright image center and dark image borders [28]. A simple approach to correct this is by capturing a reference image and dividing the captured images by the reference image [28]. The contrast can be improved by distributing the intensity values of the image by applying a transformation such as Contrast Limited Adaptive

Histogram Equalization (CLAHE) [34], homomorphic filtering [35], high pass filtering, or phase shifting high-frequency components [36]. To reduce noise, often smoothing using a Gaussian filter or Median filter is applied, which however can reduce the local contrast [23].

Besides traditional approaches, deep learning-based methods have been proposed to transform low-quality images, captured with experimental setups causing only little phototoxicity, into high-quality images [37, 38, 39].

Depending on the experiment also image stitching [40, 41], blending several images into an image spanning over a large field of view, or image registration [42, 43, 44], transforming all images into a shared coordinate system, can be required. Image stitching can be challenging when there is little overlap between images or even gaps [45].

Object Localization After preprocessing the images, the objects of interest are localized. Object localization can be grouped into object detection, semantic segmentation, and instance segmentation. In object detection, the objects of interest are localized by predicting bounding boxes, seed points, or centroids. In semantic segmentation, the aim is to assign pixels which belong to the same object class, for instance pixels belonging to the background or foreground, the same class ID. Instance segmentation, in contrast, aims at localizing each object and assigning each pixel that belongs to the same object the same ID, which allows to the extraction of information about the object's appearance. While distinguishing instances is simple for non-touching objects, it becomes challenging in crowded scenarios with many touching objects that belong to the same object class. Object detection and instance segmentation both allow for counting instances or linking them over time, whereas semantic segmentation does not allow distinguishing different instances, however, provides the possibility to quantify collective behavior such as collective migration [46].

Approaches for object localization span from traditional image processing to machine learning-based approaches including deep learning based methods [47]. For instance, simple, traditional object localization approaches are thresholding based on intensity, blob detection for instance by using a Laplacian-of-Gaussian filter, or morphological filtering [48, 49]. Moreover, deformable models, level sets, superpixels, and Markov random fields [50] are further traditional segmentation approaches. Recently, deep learning based object localization approaches have been dominating object localization in light microscopy images [51, 52]. In the scope of deep learning based object localization approaches, the methods based on the U-Net [53] such as StarDist [54], CellPose [55], or Omnipose [56] have shown outstanding performance. Moreover, adaptions of the U-Net architecture using multiple decoders [57], or modeling temporal information [58] have been proposed. To a lesser extend also other neural network architectures, such as the Mask-RCNN [59] and ErfNet [60], have been applied for object localization in microscopy images [61, 62].

An in-depth overview on recent object localization approaches is given in [51, 52, 63], whereas Ma *et al.* provides an overview spanning from traditional object localization approaches up to recent deep learning methods [47]. Moreover, an overview of available open source tools for deep learning based object localization is available in [64, 65].

Tracking Tracking objects of interest allows to extract information concerning the movement behavior or the lineage in the subsequent analysis step. To link instances in time and thereby create tracks, two main tracking paradigms exist: tracking-by-detection and tracking-by-model-evolution [48, 66, 67]. In tracking-by-model-evolution, object localization and linking are done simultaneously, whereas in tracking-by-detection object localization and linking are split into two separate tasks which are solved subsequently. In addition, approaches exist that cannot be assigned to one of the two paradigms, for instance, methods that create sets of competing instance segmentation masks during object localization, where the most suitable masks from these sets are picked during the tracking step [68, 69, 70, 71, 72, 73].

Examples of tracking-by-model-evolution approaches are active contour models [74, 75], mean shift tracking [76], dynamic shape models [77], and deformable graph models [78]. Since the object shape of the last frame is used as initialization for the object shape in the next frame, contour evolution approaches are struggling with large changes in appearance such as due to mitosis [67]. Moreover, parametrization of active contour evolution approaches is difficult, for instance, the weighting of energy terms [79]. In tracking competitions, tracking-by-detection approaches have been dominating [66, 80]. Therefore, an in-depth overview of tracking methods used in tracking-by-detection approaches is given in Section 1.1.2.

Post-Processing After linking the objects of interest in time, a post-processing step can be applied to improve the detection or segmentation of localized objects, and tracks. For virtually error-free tracking results, a manual inspection and correction can be necessary, however, semi- and fully automated post-processing methods can assist to reduce the manual effort required. A simple step to improve instance segmentation masks automatically is by applying morphological operators to fill holes or split touching objects [23]. Moreover, tracks can be improved by removing artifacts, adding missing instance segmentation masks, splitting under-segmented objects, and merging over-segmented objects [23]. A more in-depth overview of correcting tracks is given in Section 1.1.4.

Analysis Finally, the extracted information, such as the location and movement of objects, is further processed to generate insights into the object's behavior. To gain insights, different features can be extracted and compared such as the object shape using shape descriptors or analyzing the object movement by computing position or velocity-based features [49]. To visualize the extracted information, numerous possibilities exist. For instance, visualizing tracking data can be done via lineage diagrams, spatial embeddings, plotting of trajectories over time, or aggregating the trajectory data further [81]. While the visualization can impact the human interpretation of the extracted information, also how the information is aggregated can impact results for instance aggregating over an image sequence or single objects [49].

1.1.2. MOT in Light Microscopy Images

Numerous tracking-by-detection methods have been proposed to link corresponding objects between successive frames. To compare and categorize the different MOT approaches, multiple criteria are available such as (i) the operation mode of the MOT approach – online or offline, (ii) the scope in which the assigned link is optimal given a cost function – global or local, (iii) the similarity measure, also referred to as cost function, that quantifies of similar two objects are, and (iv) the underlying concept based on which objects are linked.

(i) Online methods allow tracking objects during the experiment is running, whereas offline methods process the captured image sequence after the experiment is finished. In the scope of light microscopy image analysis, the majority of approaches are used offline as there is usually no requirement for immediate decision-making during the experiment compared to other scenarios such as in surveillance or human-machine interaction.

(ii) Local methods are faster to calculate, however, can perform inferior to globally optimal approaches. Especially in image sequences covering hundreds or even thousands of objects solving a large-scale optimization problem to assign objects globally optimal can be intractable.

(iii) To find corresponding objects in successive frames a similarity measure is required. A simple similarity measure is the Euclidean distance between object positions in successive frames. Besides that, the measure of similarity can be constructed using handcrafted features that combine object position and appearance [82, 83, 84, 85], modeling information about the object's neighborhood [86], by a graph structure [87], or by learning features [88, 89, 90, 91, 92]. To weigh the extracted features in the similarity measure, weights can be learned for instance by using logistic regression [85], a structured support vector machine [69], a random forest [93], or training convolutional neural networks [90, 91]. To detect events, such as mitosis, additional detectors can be trained [91, 94, 95].

(iv) Multiple concepts for linking objects have been proposed spanning from simple, nearest neighbor based approaches, over Bayesian filters, hybrid approaches, and graph-based approaches to deep learning based methods. In the following, the different linking concepts are introduced, with a focus on graph-based and deep learning based tracking methods.

Nearest Neighbors Nearest neighbor based approaches link objects between successive frames based on the assumption that the object movement with respect to the object size is small between successive frames. The objects are linked greedy – each object in an image frame is linked to its closest neighbor in the other image frame, for instance, based on their overlap in the previous frame [94, 96] or based on the Euclidean distance between object centers [95, 97, 98]. If the temporal resolution in the image sequence is high with respect to the object movement, this linking approach can be sufficient [99]. Usually, however, the temporal resolution is restricted. Hence, large object movements between successive frames are possible resulting in linking errors.

Hybrid Hybrid MOT approaches [83, 100, 101, 102] combine simple linking concepts, such as nearest neighbors, with more sophisticated tracking methods. The idea is to apply the simple, fast to calculate linking concepts to objects that can be linked easily, whereas the more sophisticated method is applied to more challenging image regions, e.g. where the density of objects is high.

Bayesian Filtering Bayesian filtering provides a way to estimate the state of objects, for instance, their position and velocity, which is changing over time given a set of measurements [103, 104]. To estimate the object state, two probability density functions are modeled, which are updated alternatingly. The prior probability density function models any prior knowledge of the objects, e.g. their movement behavior for instance nearly constant velocity [103]. The posterior probability density function contains all information about the object state given the measurements. Unfortunately, updating these probability density functions is usually intractable. However, by making assumptions on the family of probability distributions as well as the motion and measurement models, special cases can be derived that are tractable such as the Kalman filter, the particle filter, or the Bernoulli filter. An in-depth introduction to Bayesian filtering can be found in [104], whereas a brief overview with a focus on multitarget tracking methods is given in [103].

In the scope of MOT in light microscopy images, several applications of Bayesian filtering have been proposed using the Kalman filter [105, 106, 107], the particle filter [90, 91, 108] or the Bernoulli filter [109, 110].

Graph-Based Graph-based tracking approaches allow the modeling of the object's behavior such as movement, appearance, and disappearance as well as events such as the death of the object or mitosis. In most approaches, each segmented object is modeled as a node in the graph, whereas potential links between them are modeled as edges [85, 111, 112, 113, 114]. Besides modeling each segmented object as a node in the graph, also super-nodes have been proposed introducing a layer of abstraction to the graph [70, 114]. Akram *et al.*, for instance, model the object behavior for each instance segmentation mask proposal as a super-node [70], whereas Kausler *et al.* use conditional random fields as super-nodes [114].

Linking the segmented objects over time can be formulated as the problem of finding the subset of edges in the graph that link the segmented objects best subject to a set of constraints [85, 111, 112, 113]. Some approaches assume perfect segmentation [112, 113], other approaches model segmentation errors in the graph [85, 93, 111, 114]. For instance, in [114] false positives can be handled by excluding subsets of nodes in the graph from the final solution, whereas in [93] over- and under-segmentation are modeled explicitly.

Instead of modeling segmentation errors explicitly in the graph, another approach is to select a set of instance segmentation masks from sets of competing instance segmentation masks proposals jointly with linking them over time. This can be modeled as the problem of selecting a subset of nodes and the subset of edges in the graph [68, 70, 115]. Combining the problems of selecting instance segmentation masks and linking them, results in large optimization problems, which can become intractable. Therefore often greedy algorithms are applied to find a sub-optimal solution [70, 115].

To model domain knowledge probabilistic, a subset of graph-based tracking approaches are graphical models [70, 93, 114, 115] that solve the problem of selecting a subset of edges, and nodes, in the graph by calculating the maximum a-posteriori probability.

The maximum a-posteriori probability can be calculated for instance by formulating the problem as an energy minimization problem which can be solved using integer linear programming [114], or by calculating an approximate solution using message passing [93].

The problem of selecting a subset of edges, and nodes, from the constructed graphs can be modeled as a network flow problem [68, 111, 112, 113], which can be solved, depending on the constraints, with linear programming [111, 112, 113] or integer linear programming [68]. Moreover, to make the optimization problem tractable, some approaches prune edges based on their cost [112] before solving the optimization problem, whereas other approaches solve the optimization problem greedy [70, 85, 115].

To solve the assignment problems modeled as a linear program or an integer linear program general-purpose solvers are available such as the commercial solver Gurobi [116] or non-commercial solvers such as SCIP [117] and GLPK [118]. In addition, approaches that find approximate solutions close to the optimal solution have been proposed [119, 120] that show better scaling concerning computing resources and run time for large-scale MOT.

Deep Learning Based While for instance segmentation in light microscopy images, deep learning based approaches are widely used [51, 63], the amount of deep learning based MOT approaches in light microscopy is still small. To link objects using deep learning, neural networks are either trained to extract features based on which linking costs can be calculated [88, 89, 90, 92, 121, 122, 123], or to predict similarity scores between pairs of objects [90, 124, 125, 126, 127]. A disadvantage of features and similarity functions learned by deep learning approaches is their lack of interpretability, compared to handcrafted features based on object position and appearance. Some deep learning approaches combine the tasks of object localization and tracking in a single neural network [88, 89, 121, 125], whereas others use a pipeline of deep learning methods for object localization, feature extraction, and linking [90, 122, 123, 124, 126, 127, 128, 129].

To combine object localization and tracking in a single neural network, Payer *et al.* [89] and Zhao *et al.* [121] use a Convolutional Neural Network (CNN) shaped like a stacked hourglass including convolutional gated recurrent units to predict embedding vectors for each pixel. The embeddings are learned such that embedding vectors of pixels belonging to the same instance are more similar than embeddings of pixels belonging to different instances. The learned embeddings, however, have no human comprehensible interpretation. The embeddings are then used to cluster the pixels into instances. Moreover, Hayashida *et al.* propose combining detection and tracking in a single model by predicting offsets and a magnitude that scores how likely an object is at a specific position [88]. In contrast, Lugagne *et al.* train two separate models for instance segmentation and tracking [127]. Also, the tracking can consist of several deep learning models to solve parts of the linking problem. For instance, Ben-Haim *et al.* proposed a cascade of approaches: After object localization, deep metric learning is used for extracting features, followed by a graph neural network which learns to predict linking costs based on the extracted features, and finally, graph-based matching is used to link objects based on the predicted linking costs [129].

Also, deep learning can be combined with traditional methods. For instance, a combination of particle filter and multi-task learning is proposed in [90], where deep learning is used to learn an observation model that selects in the next frame the most probable candidate. In [122] detection and tracking are learned in two separate neural networks, where the neural network trained for tracking predicts a motion flow of detected object centroids between successive image frames.

To learn similarity scores between pairs of frames, a variety of features can be used as input. For example, the object position and the Euclidean distances of the objects to their neighbors are used as input for a multilayer perceptron [124] to predict similarity scores between pairs of objects. In contrast, Xie *et al.* trains a neural network to predict similarity scores between pairs of objects based on image crops [126]. Also, handcrafted features and learned features can be combined as in [125], where a Mask R-CNN is extended with an additional Siamese branch to do segmentation and tracking, using handcrafted and learned features in the tracking branch to link corresponding objects.

As generating annotation manually is time intensive, Sugawara *et al.* use sparse annotations to train two models, one for object detection and one for tracking, where the tracking model predicts flows between pairs of successive image frames [123]. In [92], a neural network is trained to predict pseudo tracking labels from detection annotations. Moreover, Liu *et al.* uses a CycleGAN [130] to generate synthetically annotated training data [131]. The CycleGAN receives two sets of unpaired image data, raw images and simulated segmentation masks, and learns to translate raw images into segmentation masks and vice versa.

1.1.3. Causes of Segmentation and Tracking Errors in Light Microscopy Images

Extracting meaningful insights from the captured image sequences requires a high tracking quality. However, to avoid altering the behavior of the objects under observation [29, 30, 132] for instance through phototoxicity [28], the frequency and intensity of the illumination need to be limited. Hence, the captured image sequences often have low SNR and low temporal resolution which makes segmentation and tracking challenging. As a result, errors in segmentation and tracking and tracking occur. Segmentation errors can be categorized into False Positives (FP)s, False Negatives (FN)s, over-segmentation, under-segmentation, and wrong partitioning of touching objects [133]. In addition, tracking errors can be categorized into ID switches, fragmented tracks, spurious tracks, and missing tracks [80, Supplementary Note 3]. Moreover, in experiments with dividing objects, also the erroneous linking of predecessor and successor tracks occurs [134]. An overview of the different types of segmentation and tracking errors is shown in Figure 1.2.

FNs, the missing localization of an object of interest, can be caused by low SNR, low contrast, or noise [48, 66]. Moreover, objects being at the border of the field of view and therefore only partially visible can be missed by object localization approaches. In contrast, FPs, the localization of spurious objects, are caused by artifacts such as entrapped particles. Under-segmentation is the detection of several objects as a single object, whereas the wrong partitioning of objects occurs when parts of an object are attributed to another object. Both under-segmentation and wrong partitioning of objects can occur in dense scenarios when objects are touching. Lastly, over-segmentation, the detection of a single object as multiple objects, can occur when objects have irregular shapes or have a very heterogeneous brightness.

Tracking errors can be caused by challenging scenarios such as crowded scenes, low temporal resolution, and segmentation errors. ID switches, correctly detected objects that are assigned to the wrong tracks, can occur due to low temporal resolution resulting in large changes in the appearance of the object and position, crowded scenes, objects leaving and entering the field of view, or being caused by the random, undirected movement of objects [135] between successive time points. In addition, ID switches can be caused by segmentation errors. Fragmentation, a single track split into several shorter tracks, is often caused by FN errors. Moreover, an object leaving and re-entering the field of view can also result in a fragmented track. Objects not detected over their whole span of existence are referred to as missing tracks, whereas artifacts that are detected and linked over several time points, FPs, are referred to as spurious tracks. Linking errors during mitosis can occur during the orchestrated division of many objects at the same time [136], or in crowded scenes.

Usually, a combination of segmentation and tracking errors are existent in a tracked image sequence. As a result, deriving insights can be difficult. For instance, wrong assignments between predecessor and successor tracks can make conclusions on the lineage spurious, whereas due to fragmentation the resulting tracks are short which hinders long time behavior analysis [137].

1.1.4. Correction of Segmentation and Tracking Errors

As described in Section 1.1.3, segmentation errors can often cause tracking errors. Hence, to reduce tracking errors, one strategy is to reduce segmentation errors. To avoid linking erroneous instance segmentation masks, one approach is to predict competing sets of instance segmentation mask proposals and select the set of instance segmentation masks during tracking [68, 69, 70, 71, 72]. However, this approach can be computationally expensive as instance segmentation masks and their linking have to be solved simultaneously.

Another approach is to link potentially erroneous instance segmentation masks, by explicitly modeling segmentation errors or providing tools to detect and correct them in the post-processing step [85, 86, 87, 93, 102, 111, 114, 138, 139, 140]. To detect and correct segmentation and tracking errors, semi-automated methods [137, 138, 139, 141] and fully automated methods [85, 86, 87, 93, 111, 114, 140] have been proposed. Semi-automated approaches provide assistance to humans to correct errors, for instance by showing for a fragmented track the most likely successor tracks and facilitate linking them to correct fragmentation [137] or by predicting detection errors and showing frames that are likely to have detection errors to humans so they can be corrected manually [141]. While manual quality control provides the potential to acquire high-quality tracks, the time required to correct tracks manually can be infeasible.

Automated correction approaches often model prior knowledge of the data, such as the length of the mitosis cycle [114] or the expected track length [102]. Moreover, classifiers that estimate the number of objects in an instance segmentation mask can be trained,

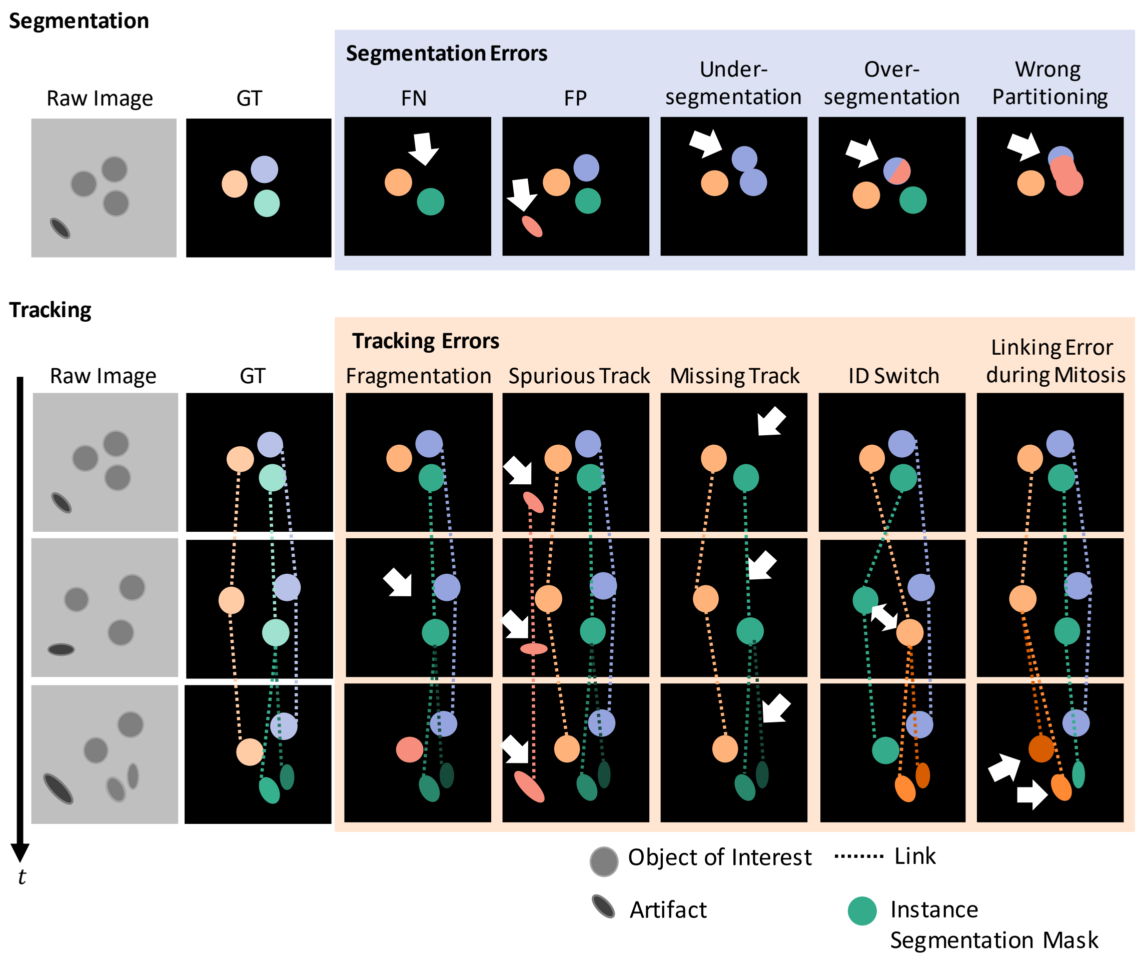


Figure 1.2.: Segmentation and tracking errors in MOT. Ground Truth (GT) and predicted instance segmentation masks are shown in color, whereas links between instance segmentation masks are shown as dashed lines. Occurring segmentation and tracking errors are highlighted with arrows.

which can then distinguish between no object predicted – a FP error, one object, or several objects – an under-segmentation error [85, 93]. For instance, to resolve instance segmentation masks that are linked to several masks in the previous time step, in [85] the instance segmentation mask is split using k-means clustering, whereas Schiegg *et al.* use spatial Gaussian mixture models to predict the number of objects in a mask [93]. To link fragmented tracks automatically, Jaqaman *et al.* apply a search window in space and time to find short tracks that potentially correspond to the same track [111]. Another challenge is to distinguish over-segmentation errors from events, such as mitosis. One approach to distinguish between the two by training a support vector machine [86]. In contrast, other approaches that are able to resolve over- and under-segmentation errors of several objects assume that no mitosis events occur [87, 140]. Moreover, to improve tracking results also uncertainty information can be incorporated in the object localization and tracking step [98].

1.1.5. Evaluation of Tracking Performance

To compare different MOT approaches, metrics are needed. Therefore, multiple metrics have been proposed that quantify the detection and linking quality of the predicted tracks. In the following, general MOT metrics as well as domain-specific tracking metrics are introduced. All introduced metrics require a set of GT annotations to quantify the tracking performance, which usually requires manual annotation effort. However, there are research directions towards ranking different MOT approaches without requiring GT data [142, 143].

General MOT Metrics

First, evaluation metrics used in general MOT applications are introduced.

MOTA The Multiple Object Tracking Accuracy (MOTA) metric [144] penalizes the detection errors FPs and FNs and penalizes linking errors as Identity Switch (IDSW). IDSWs are correctly detected objects which are assigned to different tracks in successive frames, although belonging to the same GT track. The metric is calculated as follows

$$\mathrm{MOTA} = 1 - \frac{|\mathcal{D}_{\mathrm{FP}}| + |\mathcal{D}_{\mathrm{FN}}| + |\mathcal{L}_{\mathrm{IDSW}}|}{|\mathcal{D}_{\mathrm{GT}}|}, \tag{1.1}$$

where $|\mathcal{D}_{\mathrm{FP}}|$ is the number of FP detections, $|\mathcal{D}_{\mathrm{FN}}|$ is the number of FN detections, $|\mathcal{L}_{\mathrm{IDSW}}|$ is the number of IDSWs, and $|\mathcal{D}_{\mathrm{GT}}|$ is the overall number of detected objects in the GT. Limitations of MOTA are the bias toward measuring detection quality, that tracking approaches managing to correct IDSWs at later time points are penalized, that the metric is not aligned between $[0, 1]$ as usual for other metrics, and that MOTA metric can even have a negative score [145]. More limitations of MOTA are described in detail in [145].

IDF$_1$ The IDF$_1$ metric [144] matches GT tracks to predicted tracks such that the overall amount of misdetections, FPs and FNs, as well as misalignments, a track switches its ID, between GT and prediction is minimal. This is distinct to the MOTA metric that matches GT and prediction based on their detections. To match GT tracks and predicted tracks, a bipartite matching problem is formulated which matches each GT track to at most one predicted track and vice versa. Based on the found matching between GT and predicted tracks, each of the underlying detections of the predicted tracks can be assigned to one of the three types: Identity False Positives (IDFP)s, Identity False Negatives (IDFN)s, and Identity True Positives (IDTP)s. IDFPs refer to detections of predicted tracks that have no matching counterpart in the GT, whereas IDFNs refer to detections of GT tracks that have no matching counterpart in the predicted tracks. IDTPs are detections of predicted tracks that have been successfully matched to GT tracks. The metric lies in the range $[0, 1]$ where higher scores refer to a better tracking result. The metric is calculated as

$$\mathrm{IDF}_1 = \frac{2\,|\mathcal{A}_{\mathrm{IDTP}}|}{2\,|\mathcal{A}_{\mathrm{IDTP}}| + |\mathcal{A}_{\mathrm{IDFP}}| + |\mathcal{A}_{\mathrm{IDFN}}|}\,, \tag{1.2}$$

where $|\mathcal{A}_{\mathrm{IDTP}}|$ is the number of IDTPs, $|\mathcal{A}_{\mathrm{IDFP}}|$ is the number of IDFPs, and $|\mathcal{A}_{\mathrm{IDFN}}|$ is the number of IDFNs. A limitation of the IDF_1 metric is that the metric score can be worse although the detection has been improved [145]. Moreover, as each predicted track can only be matched to at most one GT track and vice versa there can be cases where the matching is not intuitive [145]. More limitations of IDF_1 are given in [145].

HOTA The Higher Order Tracking Accuracy (HOTA) metric [145] proposes an alternative to the MOTA and IDF_1 metrics claiming to alleviate limitations of the MOTA metric.

To find predicted detections that overlap well with GT detections, an IoU threshold o is defined. Each predicted detection that has an IoU larger than o with a GT detection is referred to as a TP detection d_{TP}. Next for each TP detection d_{TP}, three types of associations are defined: True Positive Associations (TPA)s, False Negative Associations (FNA)s, and False Positive Associations (FPA)s. These associations have a resemblance to TP, FN, and FP detections, with the difference that the associations capture to which track a TP detection d^*_{TP} is mapped that belongs to the same GT track or the same predicted track as the TP detection d_{TP}. A TPA is a TP detection d^*_{TP} that is assigned to the same predicted track as d_{TP} and belongs also to the same GT track as d_{TP}. A FPA is a TP detection d^*_{TP} that is assigned to the same predicted track as d_{TP}, however, the two TP detections belong to different GT tracks. Finally, a FNA is a TP detection d^*_{TP} that belongs to the same GT track as d_{TP}, however, the two TP detections are assigned to different predicted tracks. The HOTA metric lies in the range $[0, 1]$ where higher scores refer to a better tracking result and is computed as follows

$$\begin{aligned}
a(d_{\mathrm{TP}}) &= \frac{|\mathcal{A}_{\mathrm{TPA}}(d_{\mathrm{TP}})|}{|\mathcal{A}_{\mathrm{TPA}}(d_{\mathrm{TP}})| + |\mathcal{A}_{\mathrm{FNA}}(d_{\mathrm{TP}})| + |\mathcal{A}_{\mathrm{FPA}}(d_{\mathrm{TP}})|}\,,\\
\mathrm{HOTA}_o &= \sqrt{\frac{\sum_{d_{\mathrm{TP}}\in\mathcal{D}_{\mathrm{TP}}} a(d_{\mathrm{TP}})}{|\mathcal{D}_{\mathrm{TP}}| + |\mathcal{D}_{\mathrm{FN}}| + |\mathcal{D}_{\mathrm{FP}}|}}\,,\\
\mathrm{HOTA} &= \int_0^1 \mathrm{HOTA}_o\, do\,,
\end{aligned} \tag{1.3}$$

where $|\mathcal{D}_{\mathrm{TP}}|$ is the number of TP detections, $|\mathcal{D}_{\mathrm{FN}}|$ is the number of FN detections, $|\mathcal{D}_{\mathrm{FP}}|$ is the number of FP detections, d_{TP} is a single TP detection, $\mathcal{D}_{\mathrm{TP}}$ is the set of all TP detections, $|\mathcal{A}_{\mathrm{TPA}}(d_{\mathrm{TP}})|$ is the number of all TPAs for TP detection d_{TP}, $|\mathcal{A}_{\mathrm{FNA}}(d_{\mathrm{TP}})|$ is the number of all FNAs for TP detection d_{TP}, $|\mathcal{A}_{\mathrm{FPA}}(d_{\mathrm{TP}})|$ is the number of all FPAs of TP detection d_{TP}, and o is the IoU threshold, which measures the overlap between the predicted detections and GT detections.

The HOTA metric is decomposable into several sub-metrics measuring localization, detection, and linking quality. Moreover, the metric is strictly monotonic, hence improving the tracking results leads to an improvement in the HOTA score.

Domain-Specific MOT Metrics

Next, domain-specific evaluation metrics are introduced.

SEG As the tracking performance depends on the segmentation quality, analyzing segmentation performance is important. The SEG metric [7] quantifies the segmentation quality. It is the mean of the Jaccard similarity index that measures the intersection over union between a GT instance segmentation mask and a predicted instance segmentation mask

$$\mathrm{SEG} = \frac{1}{|\mathcal{D}_{\mathrm{GT}}|} \sum_{\mathcal{I}_{\mathrm{GT}} \in \mathcal{D}_{\mathrm{GT}}} \frac{|\mathcal{I}_{\mathrm{GT}} \cap \mathcal{I}_{\mathrm{pred}}|}{|\mathcal{I}_{\mathrm{GT}} \cup \mathcal{I}_{\mathrm{pred}}|}, \tag{1.4}$$

where $|\mathcal{D}_{\mathrm{GT}}|$ is the number of GT instance segmentation masks, $\mathcal{D}_{\mathrm{GT}}$ is the set containing all GT instance segmentation masks, $\mathcal{I}_{\mathrm{pred}}$ is the set of pixels belonging to a predicted instance mask, and $\mathcal{I}_{\mathrm{GT}}$ is the set of pixels belonging to the GT instance mask. A GT instance mask is only matched to a predicted instance mask if their union is larger than half of the size of the GT instance mask. The metric lies in the range of $[0, 1]$, where a score of 1 is a perfect match between the shapes of GT and predicted instance segmentation masks.

AOGM The Acyclic Oriented Graphs Matching (AOGM) [134] metric compares the graph spanned by the GT detections and links with the graph spanned by the predicted detections and links provided by tracking approach. Each detection corresponds to a node in the graph, whereas links between objects correspond to edges in the graph. The metric penalizes deviations from the GT graph by accumulating the number of transformations needed to transform the predicted tracking graph into the GT graph. To transform the tracking graph, nodes and edges can be altered. Operations applied to the nodes are: splitting nodes (NS), e.g. to correct under-segmentation errors, adding nodes (FN), to correct FN detections, and removing nodes (FP), to correct FP detections. Edges can be added (EA), removed (ED), or corrected (EC) when they have the wrong semantic, which means a mitosis event occurred and the previous object is the mother cell instead of belonging to the same track. The AOGM metric is calculated as follows

$$\mathrm{AOGM} = w_{\mathrm{NS}}\,|\mathcal{D}_{\mathrm{NS}}| + w_{\mathrm{FN}}\,|\mathcal{D}_{\mathrm{FN}}| + w_{\mathrm{FP}}\,|\mathcal{D}_{\mathrm{FP}}| + w_{\mathrm{ED}}\,|\mathcal{L}_{\mathrm{ED}}| + w_{\mathrm{EA}}\,|\mathcal{L}_{\mathrm{EA}}| + w_{\mathrm{EC}}\,|\mathcal{L}_{\mathrm{EC}}|, \tag{1.5}$$

where $w_.$ are weights, $|\mathcal{D}_.|$ refer to the number of node transformations needed – detection errors, whereas $|\mathcal{L}_.|$ refer to the number of edge transformations required – linking errors. Matula *et al.* set the weights to $w_{\mathrm{NS}} = 5$ for splitting nodes, $w_{\mathrm{FN}} = 10$ for adding nodes, $w_{\mathrm{FP}} = 1$ for removing nodes, $w_{\mathrm{EA}} = 1.5$ for adding edges, $w_{\mathrm{ED}} = 1$ for deleting an edge, and $w_{\mathrm{EC}} = 1$ for changing the semantic of an edge [134]. Due to the weighting of the different tracking errors, the metric has a strong bias on detection, as a FN error is weighted ten times more than removing an erroneous link. Limitations of the AOGM metric concerning detection and linking errors around mitosis events where shown in [146, 147].

TRA The TRA metric [66] is a normalized version of the AOGM metric, which lies in the range of $[0, 1]$. Hence, the TRA metric measures detection and linking quality. The TRA metric is computed as follows

$$\mathrm{TRA} = 1 - \frac{\min(\mathrm{AOGM}, \mathrm{AOGM}_0)}{\mathrm{AOGM}_0}, \tag{1.6}$$

where AOGM is the AOGM score and AOGM_0 the costs to construct the GT graph from scratch – i.e. starting with an order-zero graph, a graph without any nodes and edges, and computing the costs of adding all GT nodes and all GT edges to this graph.

DET The DET metric [66] is a normalized version of a subpart of the AOGM metric, which lies in the range of $[0, 1]$. The metric penalizes only detection errors, hence the edge penalties of the AOGM metric are excluded, resulting in

$$\begin{aligned} \mathrm{AOGM_D} &= w_{\mathrm{NS}}\, |\mathcal{D}_{\mathrm{NS}}| + w_{\mathrm{FN}}\, |\mathcal{D}_{\mathrm{FN}}| + w_{\mathrm{FP}}\, |\mathcal{D}_{\mathrm{FP}}|\ , \\ \mathrm{DET} &= 1 - \frac{\min(\mathrm{AOGM_D}, \mathrm{AOGM_{D,0}})}{\mathrm{AOGM_{D,0}}}, \end{aligned} \tag{1.7}$$

where $\mathrm{AOGM_D}$ is the AOGM metric without edge penalties and $\mathrm{AOGM_{D,0}}$ the cost to construct the GT graph from scratch – i.e. starting with an order-zero graph, a graph without any nodes and edges, and computing only the costs of adding all GT nodes, no GT edges are added to the graph.

$\mathrm{OP_{CTB}}$ The $\mathrm{OP_{CTB}}$ [66] is the average between SEG and TRA measure

$$\mathrm{OP_{CTB}} = \frac{1}{2}(\mathrm{SEG} + \mathrm{TRA})\,. \tag{1.8}$$

1.1.6. Benchmark Datasets

To compare the performance of different approaches aiding in the automated analysis of microscopy image data, benchmark datasets are needed. Benchmarks based on light microscopy image data focus on different tasks such as image classification, augmented microscopy, image segmentation, event detection and classification, and MOT [51, 63].

Benchmark datasets for the task of image classification have been published, for instance, for protein classification [148] or the segmentation and classification of cells in histology images [149, 150]. Benchmarks for augmented microscopy aim to accelerate the development of approaches being able to predict labels without using fluorescent dyes [151, 152]. Moreover, image segmentation benchmarks have been proposed to compare approaches for cell segmentation [153, 154, 155], nucleus segmentation [156, 157], or foreground-background segmentation to analyze collective cell migration [46]. Event detection and classification benchmarks, for instance, allow to compare the performance of different MOT approaches to correctly predict mitosis events, or the state of cells such as cell death [158, 159, 160]. Finally, to benchmark MOT approaches, several benchmark datasets have been proposed [7, 80, 113, 161, 162]. Maška *et al.* published a benchmark with several 2D and 3D image sequences covering a variety of cell lines and microscopy imaging techniques [7, 163]. Chenouard *et al.* published a similar benchmark dataset for particle tracking consisting of simulated particles with different levels of SNR, object densities, and movement patterns [80]. Moreover, a 2D MOT benchmark dataset fully annotated with instance segmentation masks was published by Moen *et al.* [113]. Also, specialized 2D MOT benchmark datasets have been made publicly available covering sperm cells [161], yeast cells [82], and phase contrast images with fine-grained annotated cell states [164].

The quality and amount of GT annotations, however, varies between the different datasets, as providing GT annotations needs human annotators [7, 154]. While there exist approaches to at least partially automate the annotation process by iteratively training a model on a few manually annotated examples and using a human expert for inspection of the then predicted labels [149, 165, 166], large scale datasets, especially in 3D, are still only sparsely labeled [7]. Hence, utilizing deep learning based approaches that require large amounts of training data is challenging.

Therefore, instead of labeling real-world data, another direction of research is the simulation of realistic datasets. For instance, Weigert *et al.* proposed simulating light sheet microscopes to generate realistic light sheet images [167]. Kobitski *et al.* created 3D semi-synthetic models of zebrafish embryos covering several hours of early development by averaging data from several real-world experiments [3]. Malm *et al.* proposed simulating bright-field microscopy images by creating shape primitives using Fourier shape descriptors, which are placed on the image using a weighted distribution model [168]. Moreover, in [169] a conditional generative adversarial network is used to transform 3D images consisting of instance masks into realistic 3D microscopy images.

Benchmarks are an important tool to compare different approaches since they provide publicly available data. Moreover, by providing a hidden test dataset, they prevent the tweaking of an approach to the test data as well as apply the same evaluation procedure for each approach. However, benchmarks have several limitations as reported by Maier-Hein *et al.*: many benchmarks lack information concerning the challenge design, such as missing information on how the data was collected or how the final ranking is computed [170]. Moreover, the final ranking of approaches on a benchmark should be interpreted with caution since the selection of data annotation, test dataset, evaluation metric, and aggregation of the evaluation metric can substantially influence the final ranking [170].

1.2. Open Questions

Regardless of the tremendous amount of related works in MOT, there are several open questions in the domain of MOT that need to be addressed:

1. Although several MOT benchmarks have been made available to enable better comparison of tracking approaches, they are prone to heavy parameter tuning by contestants. Hence, transferring the observed performance of a tracking approach to new data is very limited. Therefore, new approaches to compare tracking performance are needed that make excessive fine-tuning more difficult.

2. As segmentation errors can cause tracking errors, it is impossible to distinguish if a tracking-by-detection approach performs well due to a superior tracking algorithm receiving a decent, yet erroneous segmentation quality as input or just due to a well-performing segmentation approach. Hence, measures are needed that allow distinguishing the performance of a tracking method from the quality of the segmentation approach. Moreover, an analysis of the behavior of the tracking method toward specific real-world segmentation errors is needed to better understand the strengths and limitations of tracking algorithms. In addition, providing such a benchmark provides the possibility to analyze tracking methods concerning their robustness and their capabilities to correct segmentation errors.

3. To extract high-quality tracking results, well-performing object localization and tracking approaches that can be combined easily are required. For object localization, instance segmentation approaches that perform well over a wide set of imaging conditions have been proposed. In contrast, tracking approaches are either simple to use and at the cost of performing inferior or perform competitively at the cost of being parameter heavy which requires careful parameter tuning. A competitive baseline tracker, however, with few parameters that is simple to apply is missing.

4. To improve tracking quality, tracking algorithms should be capable to detect and correct segmentation errors by utilizing temporal information. However, these error correction capabilities often require additional knowledge of the data e.g. concerning object sizes or object behavior. Modeling this knowledge needs either experts to carefully tune the parameters of the tracking algorithm or might need additional training, which however often requires labeled data. Hence, tracking approaches that are able to correct segmentation errors without depending on the end user to provide additional knowledge are needed.

5. With the success of deep learning for instance segmentation, joining instance segmentation and tracking into a single deep learning approach is the obvious next step. However, while deep learning outperforms traditional image processing methods in many domains, it often lacks interpretability for the user. For instance, interpreting the learned high-dimensional representations also referred to as embeddings, is usually impossible which makes further post-processing of the predictions to improve results more difficult. In addition, often recurrent parts are used which can require careful reinitialization. Hence, a deep learning based approach that

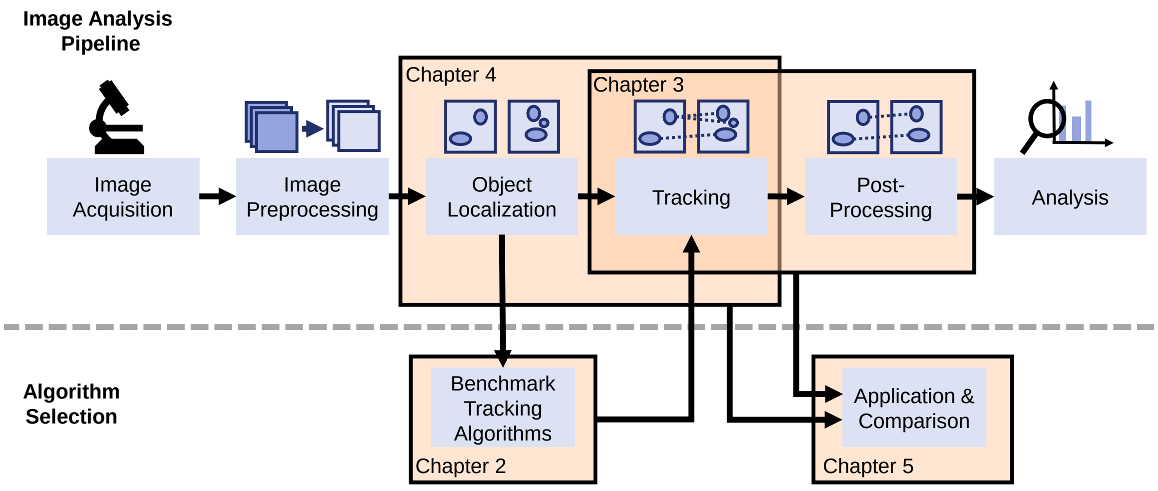


Figure 1.3.: Thesis outline. The contributions of the individual chapters with respect to the steps in the overall image processing pipeline are highlighted as boxes in orange.

combines instance segmentation and tracking by predicting human comprehensible embeddings is missing.

6. To compare different tracking methods usually only the tracking quality is considered. While tracking quality is crucial, it should not be the only criterion when selecting the most suitable algorithm for an application. For instance, an end user might prefer a simple, less performant tracking approach over a better-performing, yet very difficult to tune, parameter-heavy tracking method. Hence, tracking approaches should be compared on more criteria such as the effort to apply the selected approach to new datasets or required prior knowledge need to be considered as well.

1.3. Objectives and Thesis Outline

Based on the previously outlined open questions, the central objectives of this thesis are:

1. The proposal of an approach that allows an analysis of the tracking performance separately from the segmentation approach, thereby allowing the systematic comparison of different tracking methods concerning their robustness against segmentation errors and their capabilities to correct segmentation errors.

2. To develop a graph-based baseline tracking method with few tunable parameters that can be easily combined with an instance segmentation approach to segment and track objects in 2D and 3D.

3. The proposal of an approach to automatically detect and correct certain types of segmentation errors without requiring explicit knowledge of the object shape or additional knowledge from the end user.

4. A new deep learning based method for simultaneous instance segmentation and tracking which does not require recurrent neural network components and predicts human comprehensible embeddings.

5. The suggestion of additional comparison criteria for tracking algorithms and evaluation of the proposed graph-based tracking and the deep learning based approach based on these criteria.

6. An extensive comparison of the proposed graph-based tracking and the deep learning based approach with other MOT approaches on a diverse benchmark dataset.

The remainder of this thesis is organized as follows. To compare different tracking methods independent from the segmentation approach, Chapter 2 introduces a benchmark based on synthetically degraded segmentation data modeling different types and fractions of segmentation errors. The benchmark is used to compare the robustness and error correction capabilities of different tracking methods. Next, Chapter 3 introduces a graph-based tracking approach with few manually tunable parameters that is capable to correct certain types of segmentation errors automatically by modeling prior knowledge on the tracking graph. Chapter 4 proposes a single neural network for instance segmentation and tracking, without any recurrent neural network elements required, which predicts human comprehensible embeddings. A rigorous comparison of the two proposed tracking methods is conducted in Chapter 5 by evaluating the performance of two methods on a publicly available benchmark dataset. Moreover, additional comparison criteria are proposed on which the two approaches are compared with. Figure 1.3 provides an overview how the contributions of the different chapters integrate into the overall image analysis pipeline. Finally, Chapter 6 summarizes the accomplished work and provides an outlook on potential future work.

2. A Benchmark to Compare the Robustness of Tracking Methods in Tracking-by-Detection Algorithms

With the increasing number of available MOT approaches, comparison criteria are needed to select a suitable MOT approach for an application. In the last years, several MOT benchmark datasets and tracking metrics have been proposed to compare the performance of MOT approaches [66, 171, 172]. The proposed benchmark datasets, however, usually allow for heavy parameter tuning of the MOT approaches – for instance by making the challenge dataset publicly available or by allowing multiple submissions of the same algorithm with different parametrization. Hence, deriving conclusions on how a MOT tracking approach is going to perform on new datasets is limited, as the superior performance of an MOT approach on one specific benchmark could potentially be caused by extensive parameter tuning. In addition, MOT approaches that require heavy tuning to new datasets are difficult to tune for potential, non-expert users, which can lead to worse performance of the MOT approach when applied to new datasets. Therefore, to get a more realistic and not only a best-case performance, benchmarks are needed that make excessive fine-tuning more difficult.

In tracking-by-detection methods, the performance of the tracking algorithm is dependent on the segmentation quality, as the segmentation and tracking problem are solved subsequently, which is shown in Figure 2.1. As most MOT benchmarks only score the overall performance of the whole MOT approach, it is hence not possible to distinguish if the final evaluation score is a result of bad segmentation or bad tracking. To compare tracking approaches only, there are few tracking benchmarks available that provide segmentation masks to compare different tracking methods. However, since these segmentation data are high quality, with few segmentation errors, it is impossible to examine how the tracking approach handles erroneous segmentation data. Hence, benchmarks are needed that provide insight into how tracking algorithms perform given imperfect segmentation data.

This chapter proposes the idea to compare the tracking methods of tracking-by-detection algorithms by creating synthetically degraded segmentation data. By simulating different types of segmentation errors, tracking approaches can be analyzed concerning their

This chapter extends the initial idea to simulate segmentation errors for tracking algorithm evaluation which was proposed in K. Löffler, T. Scherr, and R. Mikut. "A graph-based cell tracking algorithm with few manually tunable parameters and automated segmentation error correction". In: PLOS ONE 16.9 (2021), e0249257. doi: 10.1371/journal.pone.0249257.

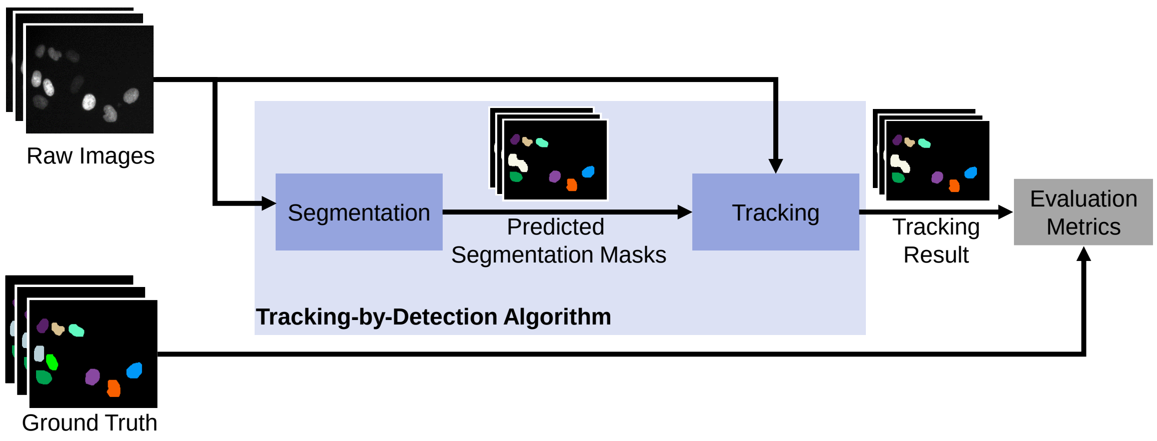


Figure 2.1.: A tracking-by-detection pipeline. First, the segmentation method predicts segmentation masks from the provided raw images. Next, the segmentation masks and raw images are forwarded to the tracking method, which yields the tracking result. Finally, the tracking quality is evaluated using metrics that compare the tracking result with the ground truth data.

robustness against specific segmentation errors and – if applicable – the performance of their segmentation error correction mechanisms.

The remainder of the present chapter is structured as follows: Section 2.1 introduces the idea of simulating erroneous segmentation data to benchmark tracking algorithms. Section 2.2 shows how the proposed method can be used to analyze evaluation metrics and investigate the strengths and weaknesses of tracking approaches. Finally, Section 2.3 concludes this chapter by providing a summary and discussion of the proposed method.

2.1. Method

Due to the challenging image conditions, which were described in Section 1.1.3, segmentation is rarely error-free. To evaluate the performance of tracking methods when provided with imperfect segmentation data, the segmentation method is replaced by modifying a fixed fraction of segmentation masks from the ground truth data, as shown in Figure 2.2. The segmentation masks are modified to model the segmentation errors under-segmentation, over-segmentation, False Negatives (FN), and a combination of the aforementioned errors, which is referred to as mixed error. The modeled segmentation errors are visualized in Figure 2.3.

The segmentation errors are generated as follows:

- **FNs** Segmentation masks are removed from the image sequence by selecting a predefined fraction of segmentation masks uniformly from the set of ground truth masks.
- **Under-Segmentation Errors** Under-segmentation errors usually occur if objects are very close or even touch. Hence, pairs of neighboring segmentation masks are calculated where each pair is assigned a sampling weight that is inverse proportional

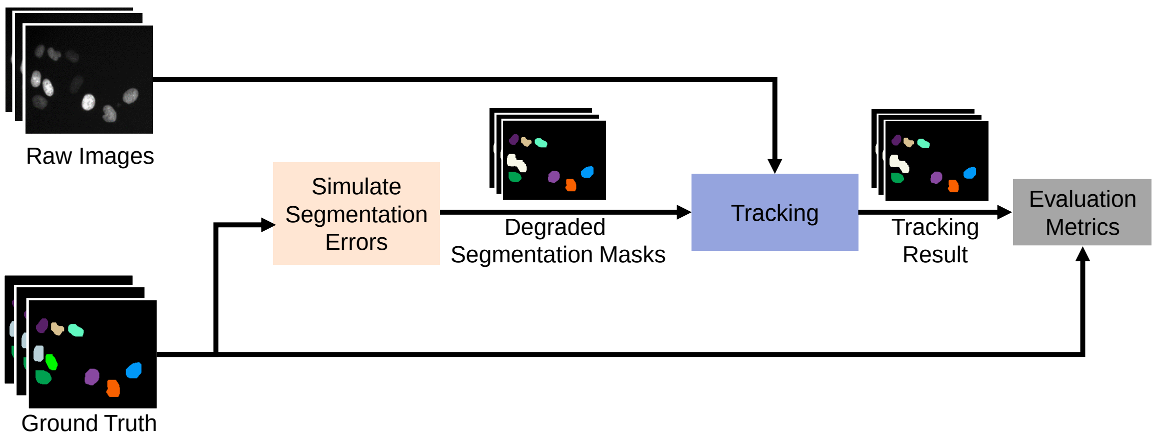


Figure 2.2.: Proposed method. The segmentation approach is replaced by a module that alters the ground truth segmentation masks to simulate segmentation errors. The synthetically degraded segmentation data is then forwarded together with the raw images as input to the tracking method, which yields the tracking result.

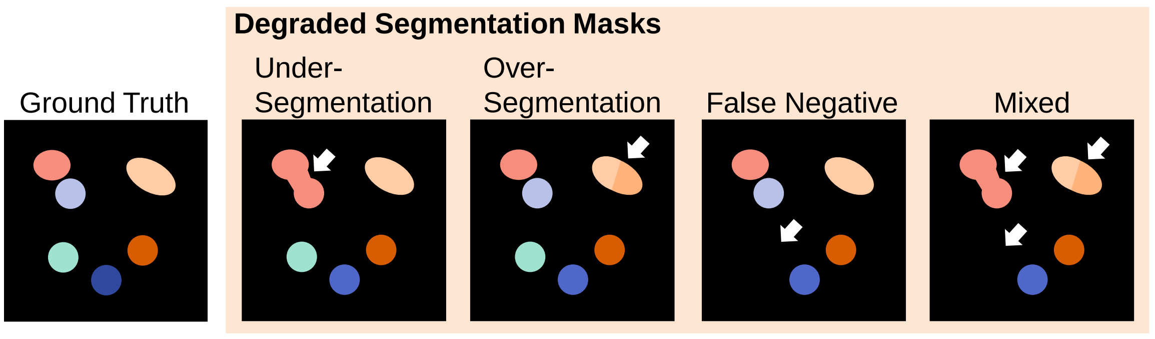


Figure 2.3.: Synthetically degraded segmentation data. The left image shows a ground truth image where each ellipsoid represents a segmentation mask. To model segmentation errors, the ground truth segmentation masks are altered by merging (under-segmentation), splitting (over-segmentation), removing (FN), and combining all three alterations (mixed error). The altered segmentation masks are marked with arrows.

to their distance, so close objects have a higher probability to be merged. Segmentation masks are merged iteratively by applying a morphological closing operation until a predefined fraction of ground truth masks are merged, where also more than two objects can be merged into a single instance mask.

- **Over-Segmentation Errors** Segmentation masks are drawn uniformly from the set of segmentation masks. Each selected segmentation mask is randomly split into two up to four segmentation mask fragments. To find a reasonable split and avoid very small segmentation mask fragments, a seed pixel is selected for each fragment. For this reason, a distance transformation is computed to find seed pixels that are far away from the segmentation mask border. The seed pixels are chosen greedily such that the next selected seed pixel is as far away as possible from the already selected ones. Therefore, the Euclidean distances between already selected seed pixels and the remaining, potential seed pixels are computed. The pixel with the largest sum of Euclidean distances is then added as a seed pixel. Finally, the segmentation mask fragments are created by assigning all pixels of the segmentation mask to their nearest seed pixel.

- **Mixed Errors** Usually multiple types of segmentation errors exist in a segmentation. Hence, the aforementioned segmentation errors are combined by modifying for each segmentation error a third of the predefined fraction of ground truth masks, so in total the predefined fraction of segmentation errors is reached.

FPs can be caused by segmenting spurious objects, which is not modeled in the benchmark since only the segmentation masks were altered and not the raw images to add spurious objects. However, FPs also occur due to over-segmentation errors. Evaluation metrics penalize predicted segmentation masks, which are not matched to a GT mask as FPs. During the evaluation, the predicted segmentation masks are matched to GT masks by selecting the best matching prediction mask. For the SEG and DET metrics, for example, the best matching segmentation mask is selected by finding the segmentation mask that has the largest IoU with the GT mask. Since for the modeled over-segmentation errors a segmentation mask is split into two up to four fragments, only one fragment is matched to the GT mask and all other fragments are penalized as FPs.

On real-world segmentation data, the fraction of segmentation errors vary – there are images without any segmentation errors and images with many segmentation errors. To simulate this variation, the fraction of segmentation errors is calculated over the whole image sequence and not per image. This also allows to simulate a specific fraction of segmentation errors on image sequences with few objects per frame, as altering only 1% of segmentation masks in an image with less than 100 segmentation masks is not possible. Hence, the fraction of simulated segmentation errors fluctuates over the images – there are images without any segmentation errors and images surpassing the defined fraction of segmentation errors.

	N Seq.	N Frames	N Tracks	N Objects		Overlap Fastest N% of Objects		
Dataset				T_{start}	T_{end}	10	25	50
HEK 293	26	30	27.0	16.5	18.0	0.869	0.916	0.952
HeLa S3	18	40	8.5	6.5	7.5	0.895	0.930	0.955
NIH 3T3	24	30	13.5	9.0	9.5	0.894	0.941	0.969
RAW 264.7	13	30	18.0	8.0	12.0	0.673	0.789	0.876

Table 2.1.: Statistics on the selected datasets from the DeepCell data. The overlap percentiles of the fastest $N\%$ of objects are calculated based on the overlap of segmentation masks referring to the same object between the successive time points t and $t-1$.

2.2. Experiment

In the following the potential of the approach is evaluated by creating erroneous segmentation data and evaluating a selection of tracking algorithms on them.

2.2.1. Experimental Setup

For evaluation, four different datasets from the DeepCell data [113] were selected: HEK 293, HeLa S3, NIH 3T3, and RAW 264.7. From each dataset, the benchmark image sequences were selected which all consist of multiple image sequences and provide fully annotated segmentation and tracking ground truth. Table 2.1 provides a summary of these datasets concerning the number of image sequences, number of objects, and object motility. To quantify the object motility, for each object at time point t the overlap to its predecessor at time point $t-1$ was computed and overlap percentiles were calculated. For instance, 10% of the objects in dataset RAW 267.4 have an overlap of less than 0.673 with their predecessor.

For each image sequence, all four segmentation error types were simulated by modifying a fraction of 1, 2, 5, 10, and 20% of the total number of objects in the image sequence. Each error simulation was repeated 5 times per image sequence, resulting in 100 degraded segmentation mask sequences per image sequence and a total of 8.100 erroneous segmentation mask sequences. Figure 2.4 shows an overview of the raw data, ground truth, and the degraded segmentation data containing different segmentation errors for the different datasets. To reuse the benchmark, the code written in Python has been made publicly available together with the graph-based tracking method of Chapter 3 at `https://git.scc.kit.edu/KIT-Sch-GE/2021-cell-tracking`.

Selected Tracking Algorithms To investigate the applicability of the proposed method, three tracking-by-tracking algorithms[1] that were submitted to the Cell Tracking Challenge are selected, which performed well on a wide set of different datasets. As just the tracking is considered in the following, only the tracking methods are described:

[1] The naming convention for the submissions to the Cell Tracking Challenge changed. The mapping of old and new submission names is provided in Table A.1

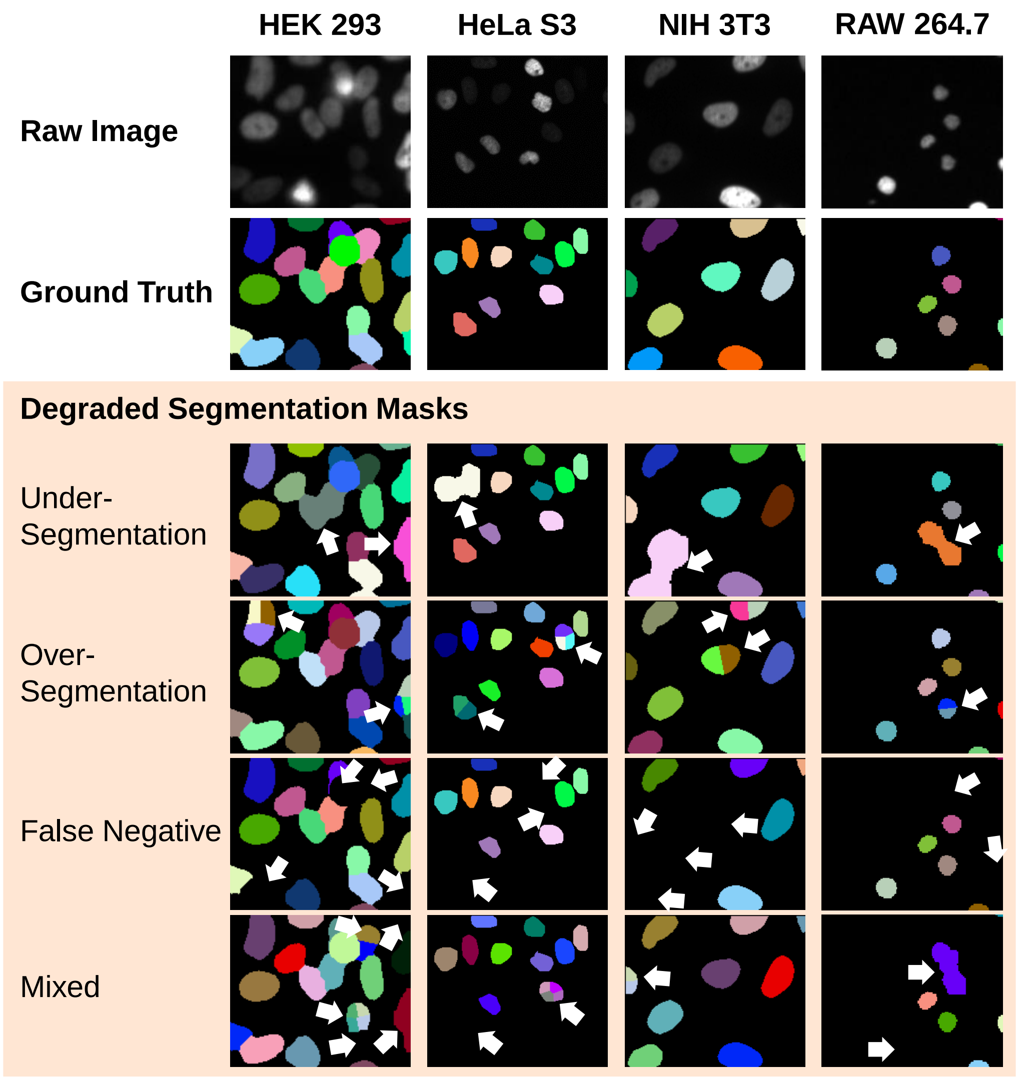


Figure 2.4.: DeepCell datasets with degraded segmentation data. Shown are raw images and ground truth annotations of different datasets from the DeepCell data, and the degraded segmentation results which were obtained by altering the ground truth segmentation masks – marked with arrows.

1. **MU-Lux-CZ** [96] A simplistic overlap-based tracking algorithm that links objects by assigning the object with the largest overlap in $t - 1$ as the predecessor to an object at t. The algorithm has no segmentation error correction capabilities and just one tunable parameter, which is the minimum overlap required to link two objects. The algorithm is implemented in Python.

2. **KTH-SE** [85, 173] A graph-based tracking algorithm that is based on the Viterbi algorithm. The approach has segmentation error correction capabilities for FP, FN, and over- and under-segmentation errors. The algorithm provides an extensive set of tunable parameters that require pre-knowledge for adapting. The algorithm is implemented in MATLAB.

3. **KIT-Sch-GE (1)** [57] A graph-based tracking algorithm based on the coupled minimum cost-flow approach. The approach has segmentation error correction capabilities for FNs and has two tunable parameters which define the maximum time span a segmentation mask can be missing to link two track fragments as one track, and a region of interest in which objects that could refer to the same track are searched. The approach is implemented in Python.

For each tracking method, the same parametrization is kept as in the submission to the Cell Tracking Challenge, however, since KTH-SE used a different parametrization for each dataset on the Cell Tracking Challenge, the parametrization of the most similar appearing dataset from the Cell Tracking Challenge was selected.

2.2.2. Evaluation

Influence of Segmentation Errors on SEG and DET Metric

First, the impact of erroneous segmentation data on the evaluation metrics SEG and DET is investigated, which is shown in Figure 2.5. The SEG metric penalizes FNs strongest, followed by over-segmentation and mixed errors which are roughly penalized similarly, and under-segmentation penalized least. The DET metric penalizes FNs strongest, followed by over-segmentation, then mixed errors, and finally under-segmentation which is again penalized least. Although both metrics rank the segmentation errors in the same order, the assigned scores differ substantially. This effect is most distinct for under-segmentation errors.

In the SEG metric, the shape of the segmentation masks is important since it is based on the Jaccard index. While FNs always result in a Jaccard index of 0, under- and over-segmentation errors show a spread of the SEG score, since the shape of the segmentation masks is altered compared to the shape of the ground truth segmentation masks. For under-segmentation, objects are connected with a morphological closing operation which adds pixels to the original objects, whereas for over-segmentation, the segmentation masks are split in segments of arbitrary size. Since the mixed errors include over- and under-segmentation errors they also show a spread in the SEG score.

The large differences between different segmentation error types in the DET score are due to the assigned penalty weights in the AOGM measure: an under-segmentation error

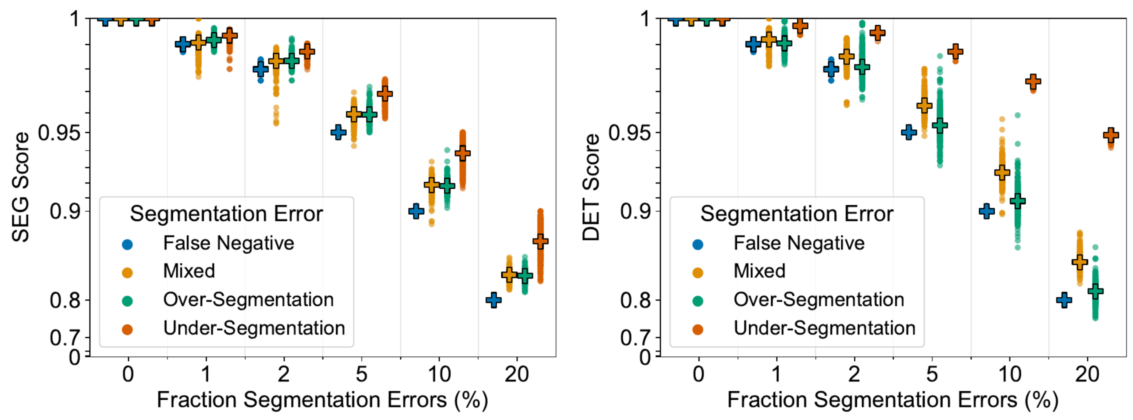


Figure 2.5.: SEG and DET score on DeepCell datasets with different types and fractions of segmentation errors. The score of a single image sequence is shown as a circle, whereas + indicates the median calculated over $N = 405$ erroneous image sequences.

is assigned a penalty weight of 5, whereas a FN error is assigned a penalty weight of 10. Over-segmentation errors are penalized as FPs with a penalty weight of 1 and, if the largest segment of a split object is less than 0.5 of the original ground truth mask, an additional FN penalty is added. This results in a comparably large spread in the DET scores for over-segmentation and mixed errors.

Evaluating the Performance of a Single Tracker

Next, it is shown how a tracking approach can be analyzed in more detail using the proposed method. Therefore, the TRA scores of the MU-Lux-CZ tracker are shown in Figure 2.6 concerning the different segmentation errors and the performance of the tracker on different datasets. As the DET and TRA metrics are closely related and the MU-Lux-CZ tracking method does not alter the segmentation masks, the TRA scores on datasets with different segmentation errors – Figure 2.6 left plot – are very similar to the DET scores – Figure 2.5 right plot. Evaluating the tracking performance split by dataset – Figure 2.6 right plot – shows only slight differences between the datasets, where the spread in results is caused by averaging over the different segmentation errors which are penalized differently.

Comparing Different Trackers

In the following, three tracking methods are compared using the degraded segmentation data. Figure 2.7 shows the results on the SEG and TRA metric for the three tracking methods MU-Lux-CZ, KTH-SE, and KIT-Sch-GE (1). Concerning SEG performance, KIT-Sch-GE (1) performs better than the other tracking algorithms on under-segmentation, FNs, and mixed errors, and even shows an improvement in the SEG score compared to no tracking. However, on over-segmentation errors, MU-Lux-CZ performs best, as the FN correction mechanism of KIT-Sch-GE (1) adds segmentation masks since the split

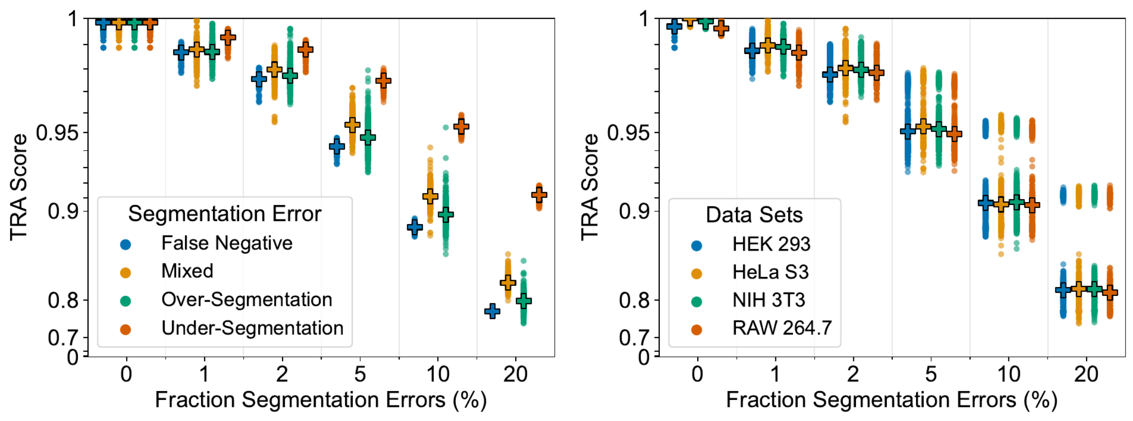


Figure 2.6.: TRA score of MU-Lux-CZ tracking split by the different error types – right plot – and different datasets – left plot on DeepCell datasets. The score of a single image sequence is shown as a circle, whereas + indicates the median calculated over a set of erroneous image sequences. For the left plot, averages were calculated over $N = 405$ tracking results, whereas for the right plot the number of erroneous datasets depends on the available image sequences: $N = 520$ for HEK 293, $N = 360$ for HeLa S3, $N = 480$ for NIH 3T3, and $N = 260$ for RAW 264.7.

segmentation masks are interpreted as individual objects with missing segmentation masks. Although not developed to correct under-segmentation errors, KIT-Sch-GE (1) can sometimes correct this error. For example, having two objects at time point t, one at $t + 1$ and two at $t + 2$ the algorithm treats this as a FN error and adds a segmentation mask which overlaps with the merged segmentation mask and hence results in two objects at $t + 1$.

For over-segmentation errors, MU-Lux-CZ and KIT-Sch-GE (1) show a similar spread in results, which is mostly due to the random splitting of objects into two up to four segments, resulting in different penalties in the metrics. The large spread in metric scores in KIT-Sch-GE (1) on the other three segmentation error types, especially on FN error image sequences, results from the error correction capabilities for FNs. On some image sequences, the algorithm seems to better correct the segmentation errors. For instance, if the object moves little between frames and has little change in its shape, the algorithm can interpolate the missing segmentation mask well.

Comparing MU-Lux-CZ and KIT-Sch-GE (1) on perfect data – the fraction of segmentation errors is 0 – it appears that MU-Lux-CZ performs slightly better. This might be due to the error correction capabilities of KIT-Sch-GE (1) that sometimes when an object leaves the field of view and then enters it again segmentation masks are interpolated at the frames in between.

KTH-SE, however, shows on both metrics very low performance, due to tending to remove segmentation masks from the image sequence up to the case that all segmentation masks are removed, which results in a metric score of 0 on all metrics. An example of how KTH-SE removes segmentation masks is shown in Figure 2.8. While for some image

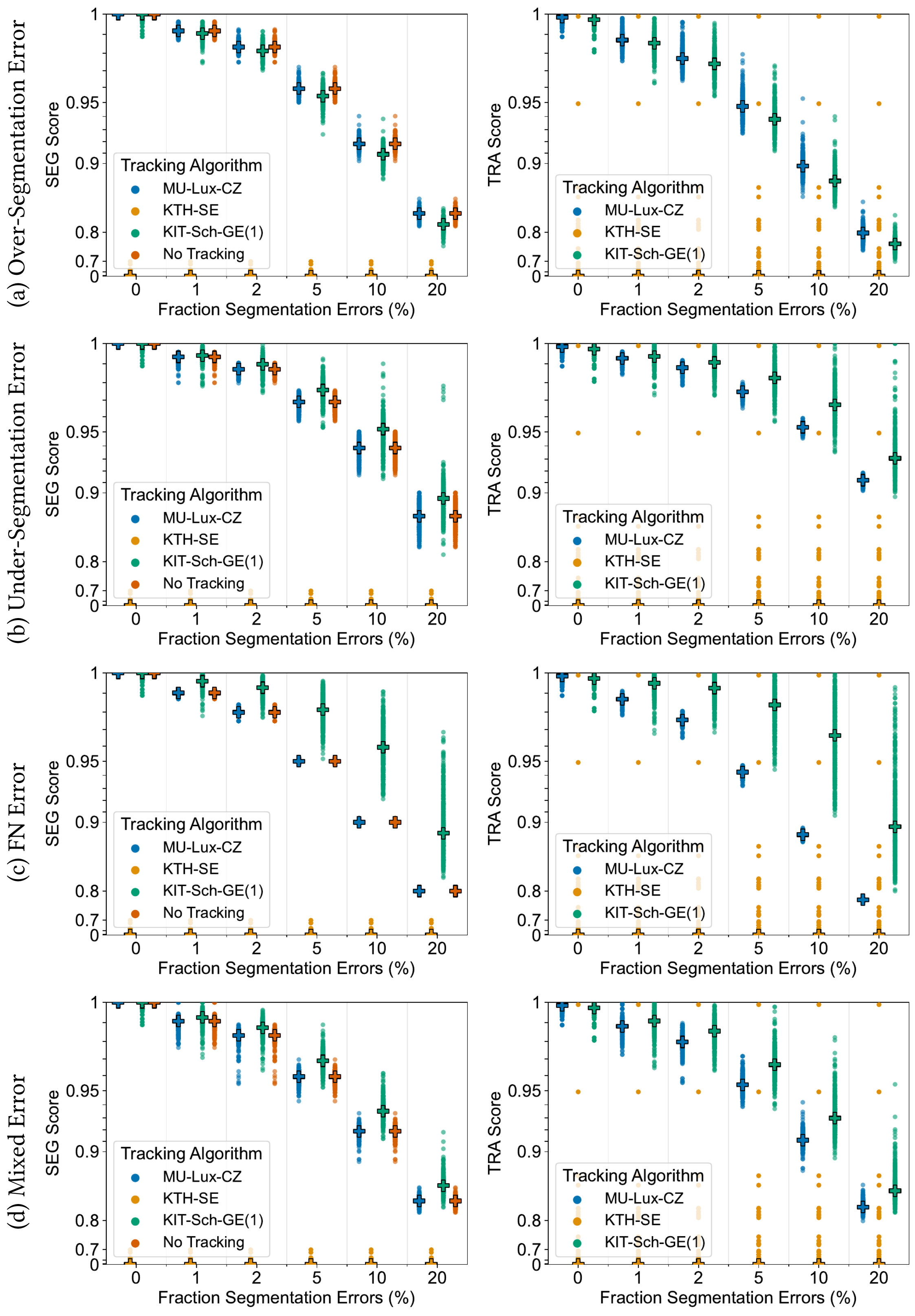


Figure 2.7.: SEG and TRA score of different tracking methods evaluated on DeepCell datasets with different types and fractions of segmentation errors. The score of a single image sequence is shown as a circle, whereas + indicates the median calculated over $N = 405$ erroneous image sequences.

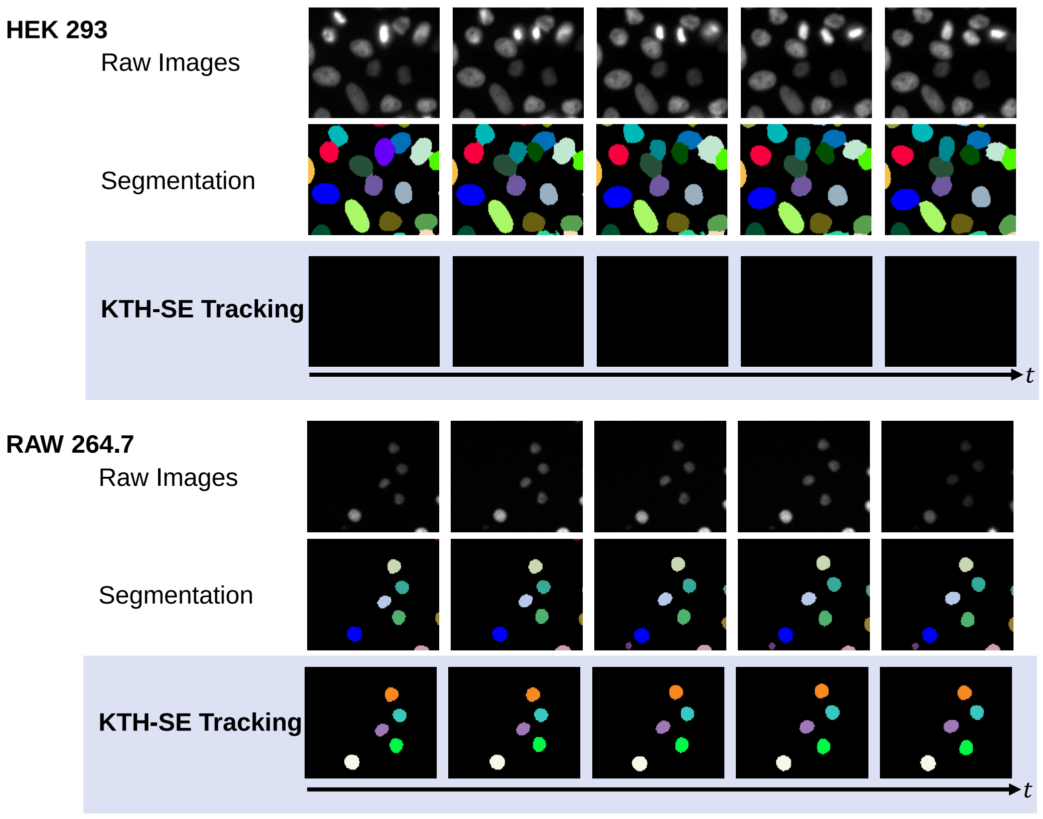


Figure 2.8.: Tracking results of KTH-SE on two different image sequences from the DeepCell dataset when provided with error-free segmentation data. On the RAW 264.7 dataset, the KTH-SE tracking just removes segmentation masks close to the image border, whereas on the HEK 293 dataset the tracking algorithm removes all segmentation masks resulting in empty frames.

sequences only objects at the image border are removed, for other image sequences all segmentation masks are removed. A possible explanation for this behavior is a strong tuning of the tracking method towards the originally used segmentation algorithm that might have been prone to FPs.

2.3. Discussion

This chapter presented a method to compare different tracking algorithms concerning their strengths and weaknesses in handling different types of segmentation errors. Therefore, it was proposed to replace the segmentation algorithm of a tracking-by-detection algorithm with degraded segmentation data covering different types and fractions of segmentation errors. To evaluate the approach, four datasets with full ground truth annotations, each consisting of multiple image sequences, were synthetically degraded. Next, the degraded segmentation data were used to analyze the influence of different types of segmentation errors on the metric score and how tracking approaches handle different types and quantities of segmentation errors.

The approach can be applied to degrade 2D and 3D segmentation data as well as real-world segmentation data and synthetic segmentation data. The analysis showed that the SEG and DET metrics both penalize FNs strongest, while under-segmentation is penalized the least. Concerning the tracking approaches, KTH-SE showed a tendency to remove segmentation masks independent from the segmentation error type but dependent on the dataset. Therefore, to use this approach careful tuning is needed, that however requires time and prior knowledge that not each non-expert user has. KIT-Sch-GE (1) can improve the SEG score on FNs, under-segmentation, and mixed errors, however, the segmentation quality decreases compared to no tracking when over-segmentation errors are dominant. Depending on the dataset, however, also simple overlap-based tracking approaches such as MU-Lux-CZ can perform competitively.

While this chapter proposed a method to compare different tracking methods, the simulated errors only partially emulate real-world errors. For instance, FNs usually occur when the visibility of an object is degraded e.g. due to low contrast or occlusion, whereas here segmentation masks were randomly removed. While several runs with different error types and fractions can be created, which makes fine-tuning more difficult as there is no single dataset, other interesting characteristics such as different movement patterns or a varying number of objects are missing. Future works could extend this approach including simulated datasets with different movement characteristics and numbers of objects in 2D and 3D which would make fine-tuning even more difficult as new datasets could be created easily without requiring time-intensive manual annotation.

Besides using the proposed approach to compare different tracking algorithms, the approach can also be used during tracking algorithm development. For instance, to find a robust parametrization of a tracking algorithm, which performs well over different types and fractions of segmentation errors. Providing such a robust parametrization of the tracking approach helps non-expert users to apply a promising tracking approach to their application easily.

3. A Graph-based Tracking Algorithm with few Tunable Parameters and Segmentation Error Correction

Currently, MOT is dominated by tracking-by-detection approaches [66, 172]. As tracking methods in tracking-by-detection approaches, numerous methods have been proposed: simplistic overlap-based tracking [95, 97], hybrid approaches combining simplistic nearest neighbor tracking approaches with more sophisticated tracking methods for challenging image regions [83, 100, 102], Bayesian filters [105, 108, 109], graph-based tracking [71, 85, 111], and deep learning based approaches [88, 90, 129].

MOT benchmarks usually only consider the overall tracking performance – the needs of non-expert users, which apply the MOT approaches for their tasks, are not taken into consideration. Since MOT is just one step in the image analysis pipeline, ideally non-expert users should be able to integrate the MOT approach with little overhead and prior knowledge in their pipeline. Hence, from the perspective of non-expert users, tracking approaches can be categorized as on the one hand, sophisticated tracking algorithms with high accuracy on benchmark datasets, which however have an extensive set of parameters that need careful manual tuning or annotated data is needed to fine-tune the approaches to the new application. On the other hand, tracking algorithms that are easy to apply, however, performed inferior on benchmark datasets. While there are segmentation approaches with a strong performance on a diverse set of data, which can be easily applied to new datasets [55, 56], a simple to apply, yet well-performing tracking algorithm is currently missing.

To close the gap, this chapter proposes a graph-based tracking algorithm with only two manually tunable parameters, which are easy to interpret for non-expert users. The proposed tracking algorithm can process 2D and 3D image sequences and can be combined with an arbitrary segmentation algorithm, which predicts instance segmentation masks to create a tracking-by-detection approach. Moreover, a post-processing step is proposed that can detect and correct certain types of segmentation errors without requiring training data or tuning.

The proposed idea to model and correct segmentation errors automatically can be applied to different domains. In this thesis, the domain-specific splitting of objects due to mitosis is modeled in the tracking algorithm. Moreover, the algorithm can be extended to

The method section is an expansion of the method description from K. Löffler, T. Scherr, and R. Mikut. “A graph-based cell tracking algorithm with few manually tunable parameters and automated segmentation error correction”. In: PLOS ONE 16.9 (2021), e0249257. doi: 10.1371/journal.pone.0249257.

model other domain-specific properties as well, which is however not the focus of this thesis.

The remainder of this section is organized as follows: Section 3.1 provides the mathematical preliminaries for this chapter. Then, Section 3.2 introduces the proposed graph-based tracking method including its post-processing step. The tracking approach is compared with other tracking methods in Section 3.3. Finally, Section 3.4 concludes this chapter by providing a summary and discussion of the proposed method.

3.1. Mathematical Preliminaries

This section introduces the minimum cost flow problem based on [174, Chapter 3] and [175, Chapter 9]. An in-depth introduction to graph theory and optimization problems can be found in [174, 175].

Minimum Cost Flow Problem Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed graph, where $\mathcal{V} = \{u, v, w, \dots\}$ is the set of nodes and $\mathcal{E} = \{(u, v) \mid u, v \in \mathcal{V}\}$ the set of edges connecting pairs of nodes. Moreover, let q^- be a source node and q^+ be a sink node. Each node v has a balance $b(v)$, where nodes with $b(v) > 0$ are called sinks and nodes with $b(v) < 0$ sources, otherwise v is called balanced $b(v) = 0$. The flow of units $f(u, v)$ that can be send over an edge (u, v) are bounded by a lower capacity $k_l(u, v)$ and an upper capacity $k_u(u, v)$ with $k_u(u, v) \geq k_l(u, v) \geq 0$. Sending a unit over an edge has a cost $c(u, v)$. The problem of finding a feasible flow of units through this graph at an overall minimal cost can then be modeled as follows:

$$\min_{f(u,v)} \sum_{(u,v)\in\mathcal{E}} c(u, v) f(u, v) \tag{3.1}$$

subject to:

$$\text{Capacity constraint:} \quad k_l(u, v) \leq f(u, v) \leq k_u(u, v) \quad \forall v \in \mathcal{V} \tag{3.2}$$

$$\text{Flow conservation:} \quad \sum_{u\in V} f(u, v) - \sum_{w\in V} f(v, w) = b(v) = 0 \quad \forall v \in \mathcal{V} \setminus \left\{q^+, q^-\right\} \tag{3.3}$$

$$\text{Flow requirement:} \quad \sum_{v\in V} f\left(v, q^+\right) = b\left(q^+\right) = \sum_{v\in V} f\left(q^-, v\right) = -b\left(q^-\right) > 0\,. \tag{3.4}$$

The problem can be solved with linear programming [175, Chapter 3]. Moreover, the integrality theorem [174, p. 133] states for the case that all capacities and vertex balances are constrained to be integers, then there exists an optimal solution that is integer-valued. Hence, although the problem is now an integer program it can be solved efficiently using linear programming.

Coupled Minimum Cost Flow Problem The term coupled minimum cost flow problem was introduced by Padfield *et al.* who modeled different types of object behavior in a graph formulation for MOT [112]. The approach models object movement, appearance, and disappearance as well as splitting and merging pairs of objects. While object movement, appearance, and disappearance can be modeled directly using minimum cost flow, the formulation needs to be extended to model the merging and splitting of objects. Therefore, Padfield *et al.* proposed to add coupling constraints to the original minimum cost flow problem to model an object splitting into two objects or two objects merge into one object. The added coupling constraints enforce that if an edge, which is connected to a split or merge vertex, is chosen, additional edges need to be selected as well. A detailed description of the approach can be found in [112].

3.2. Method

The proposed method builds up on the coupled minimum cost flow approach proposed by Padfield *et al.* [112]. In the following, the approach is extended by modeling FNs, merges of arbitrary numbers of objects into a single object, and splitting a single object into an arbitrary number of objects. Moreover, to reduce the number of edges and hence flow variables in the graph, a tracklet step is introduced. In addition, simple, position-based costs are proposed that allow linking also erroneously segmented objects. Finally, a post-processing step is proposed that can use the additional information of many-to-one and one-to-many links of the tracking graph to correct segmentation errors.

The proposed tracking algorithm is based on the following assumptions: (i) The object movement is small compared to the overall image size and (ii) the majority of segmentation masks are segmenting single objects correctly. (i) The object movement assumption is motivated by the need for a reasonable temporal resolution of the image sequence for a detailed analysis of the lineage or object behavior. (ii) The segmentation assumption is motivated by the availability of reasonably well-performing segmentation approaches [53, 54, 55, 56, 57, 176].

It is vital to emphasize that not each segmented object refers to an entity. Due to segmentation errors, the segmented objects can contain detected artifacts, only parts of an entity, a single entity, or several entities.

The tracking is split into three steps: tracklet step, matching step, and post-processing step. In the tracklet step, the segmented objects are coarsely followed over time to find potential objects belonging to the same track. In the matching step, the segmented objects are assigned to tracks by solving a graph-based optimization problem. The graph models object behavior including appearance, disappearance, movement, and mitosis as well as splitting and merging of objects, due to over- and under-segmentation errors, and FNs. Lastly, a post-processing step is applied to correct segmentation errors. Figure 3.1 shows an overview of the proposed tracking pipeline. In the following, all steps of the tracking method are introduced based on the example dataset shown in Figure 3.1.

3.2.1. Step 1: Tracklet Step

Based on the object movement assumption, segmented objects belonging to the same track should be spatially close between successive time points. Similar to in [57], for each segmented object a rectangular-shaped Region Of Interest (ROI) is defined, which size is derived from the median size of the segmentation masks, to find objects which could belong to the same track at successive time points. The ROI is propagated over

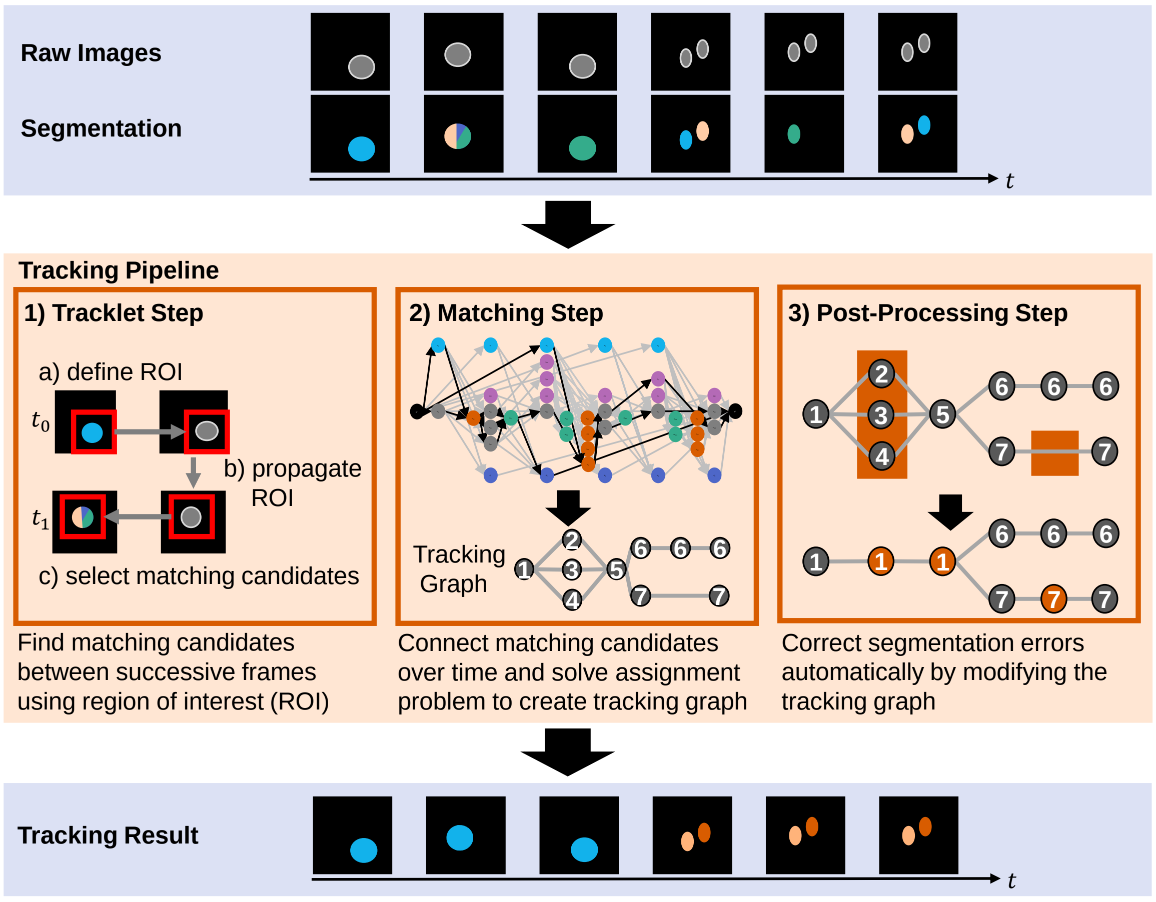

Raw Images
Segmentation
t
Tracking Pipeline
1) Tracklet Step
a) define ROI
t_0
b) propagate ROI
t_1
c) select matching candidates
2) Matching Step
Tracking Graph
3) Post-Processing Step
Find matching candidates between successive frames using region of interest (ROI)
Connect matching candidates over time and solve assignment problem to create tracking graph
Correct segmentation errors automatically by modifying the tracking graph
Tracking Result
t

Figure 3.1.: Proposed tracking method. A sequence of raw images and erroneous segmentation results is fed into the tracking pipeline. During tracking the segmentation masks are linked over time and segmentation errors are corrected. Derivative of Figure 1 by Löffler *et al.* [177] licensed under CC BY 4.0; rearranged plot and changed colors.

time by estimating a displacement between successive frames using a phase correlation [178]. Segmented objects which overlap with the propagated ROI are considered matching candidates which are linked in the matching step.

3.2.2. Step 2: Matching Step

Segmented objects and their matching candidates are represented as nodes in a directed graph. The graph models the object behavior: appearance, disappearance, movement, and mitosis as well as the segmentation errors: FNs, under-, and over-segmentation. The segmented objects are linked over time by solving a coupled minimum cost flow problem.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed graph with a set of nodes $\mathcal{V} = \{u, v, w, \dots\}$ and a set of edges $\mathcal{E} = \{(u, v)\}$ connecting pairs of nodes u and v. Edges (u, v) are directed, starting from node u and ending in node v.

Nodes The following node types are defined to model object behavior and segmentation errors:

- q^-: source node
- q^+: sink node
- $o_{\cdot,\cdot}$: object nodes modeling segmented objects
- $s_{\cdot,\cdot}$: split nodes modeling splitting objects
- $m_{\cdot,\cdot}$: merge nodes modeling merging objects
- $x_{\cdot,\cdot}$: skip nodes modeling FNs
- $d_{\cdot}$: disappearance nodes modeling disappearing objects
- $a_{\cdot}$: appearance nodes modeling appearing objects

A specific node in the graph is referred to as $v_{i,t}$, where v is the node type and i is a unique identifier referencing a segmented object, and t is a time point.

For each segmented object i at time point t a corresponding object node $o_{i,t}$ is added to the graph.

To link tracks with missing segmentation masks, skip nodes are added for each segmented object at time point t at the successive $\Delta t - 1$ time points. For example, if $\Delta t = 3$, tracks can be linked that have missing segmentation masks for a maximum of two subsequent time points. The modeling of missing segmentation masks can be dropped by setting $\Delta t = 1$. For each time point t an appearance node a_t is added to model appearing objects at time point $t + 1$, whereas a disappearance node d_t is added for each time point t to model disappearing objects at time point $t - 1$. Mitosis and splitting objects are modeled by adding for each object node and skip node at time point t a split node $s_{i,t+1}$ at time point $t + 1$. Merging objects are modeled by adding a merge node $m_{i,t-1}$ at time point $t - 1$ for each object node and each skip node at time point t. The source node q^- is added before the first time point and a sink node q^+ is added after the last time point of the considered set of time points $\mathcal{T}$.

It is emphasized that the merge and split nodes both model under- and over-segmentation errors depending on the context: Multiple objects at time point t which are being under-segmented as a single object at the next time point $t + 1$ are modeled by a merge node as well as an over-segmented object at time point t which is being correctly segmented as a single object at the next time point $t + 1$. Similarly, an object at t which is over-segmented as multiple objects at the next time point $t+1$ is modeled by a split node as well as multiple under-segmented objects, which are correctly segmented as a single object at the next time point $t + 1$.

Edges The nodes are connected by directed edges to model events, such as linking segmented objects between successive time points. Directed edges are allowed between the following node types, where $u : \{v, w\}$ means edges starting from node type u can end in the node types v and w:

- $q^- : \left\{a_\cdot, o_{\cdot,\cdot}\right\}$
- $q^+ : \{\}$
- $o_{\cdot,\cdot} : \left\{d_\cdot, m_{\cdot,\cdot}, o_{\cdot,\cdot}, s_{\cdot,\cdot}, q^+, x_{\cdot,\cdot}\right\}$
- $s_{\cdot,\cdot} : \left\{o_{\cdot,\cdot}\right\}$
- $m_{\cdot,\cdot} : \left\{d_\cdot, o_{\cdot,\cdot}\right\}$
- $x_{\cdot,\cdot} : \left\{m_{\cdot,\cdot}, o_{\cdot,\cdot}, s_{\cdot,\cdot}, q^+, x_{\cdot,\cdot}\right\}$
- $d_\cdot : \{q^+\}$
- $a_\cdot : \left\{d_\cdot, o_{\cdot,\cdot}, s_{\cdot,\cdot}\right\}$

Figure 3.2 shows the constructed graph based on the image sequence with erroneous segmentation from Figure 3.1.

Connecting all object nodes and skip nodes at time point t naïvely to all other object nodes at time point $t + 1$, would result in a quadratically growing number of edges. To reduce the number of edges in the graph, the matching candidates from the tracklet step are used to connect object nodes only to the object nodes in successive frames that correspond with its matching candidates. This is applied to the split and merge nodes as well, by connecting a split node $s_{i,t+1}$ only to the object nodes at $t + 1$, the object node $o_{i,t}$ or skip node $x_{i,t}$ is connected to. A merge node $m_{i,t}$ is only connected to the object nodes and skip nodes at t, the object node $o_{i,t+1}$ is connected to. A visualization how nodes are connected is shown in Figure 3.3 and Figure 3.2. The used costs functions $c(\cdot, \cdot)$ are introduced in more detail subsequently.

In theory, the graph could be spanned over the full-time span of an image sequence, however, for image sequences with many objects and time points, this would lead to large optimization problems that need to be solved. Therefore, smaller optimization problems

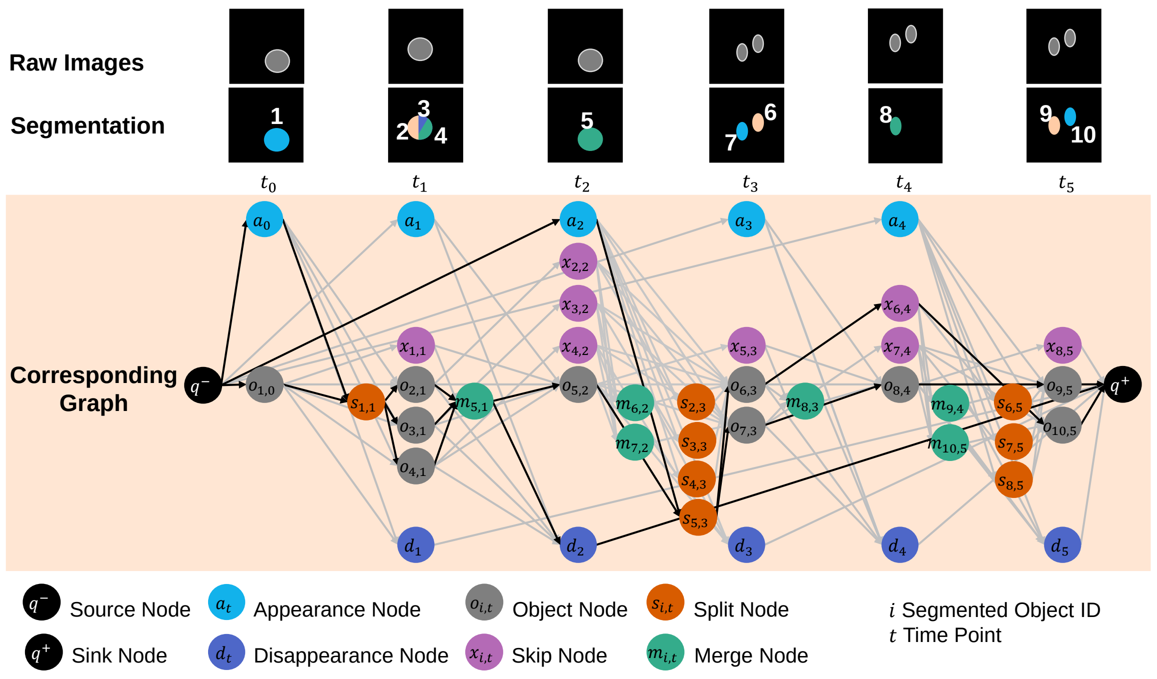


Figure 3.2.: Graph constructed for an image sequence with erroneous segmentation, where each segmented object is assigned a unique ID i. Nodes corresponding to a segmented object share the same ID i, however, depending on the node type they are assigned to different time points t in the graph. Segmented objects are linked over a maximum time span of $\Delta t = 2$ frames by adding for each object node $o_{i,t}$ a skip node $x_{i,t+1}$, which models a missing segmentation mask. The segmented objects are assigned to tracks by solving the coupled minimum cost flow problem – the optimal solution is highlighted in black. The optimal solution contains additional flows from the source node over appearance to disappearance nodes to the sink node, which were omitted for better visibility. Derivative of Figure 2 by Löffler *et al.* [177] licensed under CC BY 4.0; changed colors.

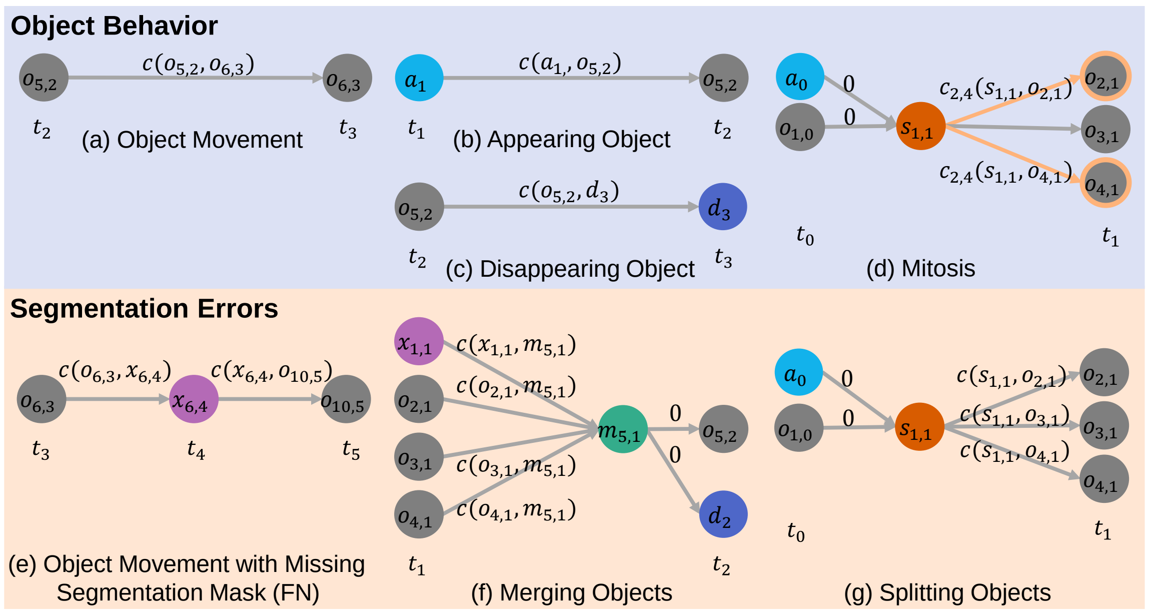


Figure 3.3.: Modeling object behavior and segmentation errors in the graph example from Figure 3.2. Annotations $c(\cdot,\cdot)$ on the edges are assigned edge costs. To model mitosis, edges that are connected to pairs of “daughter" nodes are pairwise coupled – highlighted in orange. Derivative of Figure 2 by Löffler *et al.* [177] licensed under CC BY 4.0; rearranged plot and changed colors.

are solved by dividing the image sequence into smaller time spans and constructing graphs that overlap in time. The size of these graphs is defined by the parameter Δt. Hence, every $T = n \cdot \Delta t$ time points, with $n = \{0, 1, 2, \dots,\}$, a graph is constructed, which spans over the time points $\mathcal{T} = \{T - (\Delta t - 1), \dots, T, \dots, \min(T + \Delta t, T_{\max})\}$, where $T_{\max}$ is the last time point of the image sequence. If no tracks with missing segmentation masks were detected in the graph constructed for the previous time point, no skip nodes need to be added for time points $\mathcal{T} = \{T - (\Delta t - 1), \dots, T - 1\}$, which are linked to object nodes at time point T. Hence, in these cases, the graph spans just over the time points $\mathcal{T} = \{T, \dots, T + \Delta t\}$.

Formulation as Coupled Minimum Cost Flow Problem

To link the segmented objects over time, a coupled minimum cost flow problem as described in Section 3.1 is constructed. The formulation builds up on the coupled minimum cost flow problem [112], which is extended such that many-to-one and one-to-many links are possible to model over- and under-segmentation errors of two or more objects as well as introducing skip nodes to model FNs. The optimization problem is given as:

$$\begin{aligned} \min_{f(u,v)} \quad & \sum_{(u,v)\in\mathcal{E}} c(u,v)\, f(u,v) \\ \text{subject to:} \quad & \\ & g_{i'}\left(f(u,v)\right) = 0\,,\ i' = 0,\dots,N \\ & h_{j'}\left(f(u,v)\right) \leq 0\,,\ j' = 0,\dots,M \end{aligned} \tag{3.5}$$

where $c(u, v)$ is a cost, $f(u, v)$ the flow variables which are constrained to be integer-valued, $g_{i'}$ are equality constraints, and $h_{j'}$ inequality constraints, which are introduced next.

A flow conservation constraint, as introduced in Equation 3.3, is added for all nodes apart from the source node and sink node:

$$\sum_{u \in \mathcal{V}} f(u, v) = \sum_{w \in \mathcal{V}} f(v, w) \ . \tag{3.6}$$

Flow requirements, as introduced in Equation 3.4, enforce a flow of a fixed number of units through the graph. To ensure that each segmented object is assigned to a track, a flow of one is enforced through each object node $o_{i,t}$ and the flow from the source node q^- is set to the total number of segmented object nodes

$$\begin{aligned} \sum_{u \in \mathcal{V}} f(q^-, u) &= \sum_{t' \in \mathcal{T}} |O_{t'}| \,, \\ \sum_{u \in \mathcal{V}} f\left(u, o_{i,t}\right) &= 1 \,, \\ f(q^-, a_t) &= |O_{t+1}| \,, \end{aligned} \tag{3.7}$$

where $\mathcal{T}$ is the set of all time points in the graph and $|O_t|$ is the number of object nodes at time point t.

Due to the capacity constraint, Inequality 3.2, the flow over an edge (u, v) is restricted by a maximum capacity constraint $k_u(u, v)$, whereas the lower capacity constraint is set to 0

$$0 \le f(u, v) \le k_u(u, v) \ . \tag{3.8}$$

Edges connected to at least one skip $x_{i,t}$ or object node $o_{i,t}$ have a capacity of one

$$k_u\left(u, o_{i,t}\right) = k_u\left(u, x_{i,t}\right) = k_u\left(o_{i,t}, v\right) = k_u\left(x_{i,t}, v\right) = 1 \,. \tag{3.9}$$

To model the splitting and merging of more than two objects, the capacity of edges connecting merge nodes $m_{i,t-1}$ to disappearance nodes d_t and appearance nodes a_t to split nodes $s_{i,t+1}$ depends on the number of edges ending in the merge node and edges starting from the split node, respectively:

$$\begin{aligned} k_u\left(m_{i,t-1}, d_t\right) &= \left|\left\{\left(v_{\cdot,t-1}, m_{i,t-1}\right) \;\middle|\; v_{\cdot,t-1} \text{ connected to } m_{i,t-1}\right\}\right| \,, \\ k_u\left(a_t, s_{i,t+1}\right) &= \left|\left\{\left(s_{i,t+1}, v_{\cdot,t+1}\right) \;\middle|\; v_{\cdot,t+1} \text{ connected to } s_{i,t+1}\right\}\right| \ . \end{aligned} \tag{3.10}$$

The capacity of edges connecting the source node q^- to appearance nodes a_t depends on the number of segmented objects at time point $t + 1$, whereas the capacity of edges connecting disappearance nodes d_t to the sink node q^+ depend on the number of segmented objects at time points $\{t - \Delta t, \ldots, t\}$

$$\begin{aligned} k_u(q^-, a_t) &= |O_{t+1}| \,, \\ k_u(a_{t-1}, d_t) &= |O_t| \,, \\ k_u\left(d_t, q^+\right) &= \sum_{t'=t-\Delta t}^{t} |O_{t'}| \ , \end{aligned} \tag{3.11}$$

where $|O_t|$ is the number of object nodes at time point t. The sum of the capacity constraint $k_u(d_t, q^+)$ results from the added skip nodes which enable linking segmented objects over a maximum time span Δt. For $\Delta t = 1$, no skip nodes are added resulting in $k_u(d_t, q^+) = |O_t| + |O_{t-1}|$, providing an upper bound. For $\Delta t = 2$ for each object node a skip node is added, allowing a flow from an object node at $t-2$ over its skip node to a merge node at $t-1$, which is connected to the disappearance node d_t. To provide a large enough upper bound, the number of object nodes from time point $t-2$ is added to the maximum capacity:

$$k_u\left(d_t, q^+\right) = \sum_{t'=t-2}^{t} |O_{t'}| = |O_t| + |O_{t-1}| + |O_{t-2}| \,. \tag{3.12}$$

To model the merging of two or more objects into a single object, for each merge node $m_{i,t-1}$ the following constraints are added:

$$\begin{aligned} f\left(m_{i,t-1}, o_{i,t}\right) - f\left(m_{i,t-1}, d_t\right) &\leq 0\,, \\ f\left(v_{\cdot,t-1}, m_{i,t-1}\right) - f\left(m_{i,t-1}, o_{i,t}\right) &\leq 0 \;\forall\; v_{\cdot,t-1} \text{ connected to } m_{i,t}\,. \end{aligned} \tag{3.13}$$

By combining Equation 3.8 and Equation 3.13

$$0 \leq f\left(v_{\cdot,t-1}, m_{i,t-1}\right) \leq f\left(m_{i,t-1}, o_{i,t}\right) \leq f\left(m_{i,t-1}, d_t\right)\,, \tag{3.14}$$

can be derived.

For a flow $f\left(v_{\cdot,t-1}, m_{i,t-1}\right)$ from a node $v_{\cdot,t-1}$ to the merge node $m_{i,t-1}$ larger than zero, the flow from the merge node to the object node $f\left(m_{i,t-1}, o_{i,t}\right)$ and the flow from the merge node to the disappearance node $f\left(m_{i,t-1}, d_t\right)$ need to be at least as large. The flow conservation constraint Equation 3.6 enforces the same flow into a node and from a node, resulting in a flow of at least two through the merge node $m_{i,t-1}$ or zero.

To model the splitting of an object into two or more objects, for each split node the following constraints are added:

$$\begin{aligned} -f\left(a_t, s_{i,t+1}\right) + f\left(o_{i,t}, s_{i,t+1}\right) &\leq 0\,, \\ -f\left(o_{i,t}, s_{i,t+1}\right) + f\left(s_{i,t+1}, v_{\cdot,t+1}\right) &\leq 0 \;\forall\; v_{\cdot,t+1} \text{ connected to } s_{i,t+1}\,. \end{aligned} \tag{3.15}$$

Similar to before, by combining Equation 3.8 and Equation 3.15

$$0 \leq f\left(s_{i,t+1}, v_{\cdot,t+1}\right) \leq f\left(o_{i,t}, s_{i,t+1}\right) \leq f\left(a_t, s_{i,t+1}\right)\,, \tag{3.16}$$

can be derived.

For a flow $f\left(s_{i,t+1}, v_{\cdot,t+1}\right)$ from the split node $s_{i,t+1}$ to a node $v_{\cdot,t+1}$ larger than zero, the flow from the object node to the split node $f\left(o_{i,t}, s_{i,t+1}\right)$ and the flow from the appearance node to the split node $f\left(a_t, s_{i,t+1}\right)$ need to be at least as large. The flow conservation constraint Equation 3.6 enforces the same flow into a node and from a node, resulting in a flow of at least two through the split node $s_{i,t+1}$ or zero.

Mitosis events differ from splitting objects, as one “mother" entity evolves into two “daughter" entities. In contrast, the splitting of objects is caused by the over-segmentation of an entity or the under-segmentation of entities being correctly segmented as different objects in the next time point. To assign different edge costs to mitosis events, all pairs of “daughter" nodes $o_{j,t+1}$ & $o_{l,t+1}$ are constructed to which the “mother" node $s_{i,t+1}$ is connected to and pairwise coupled flow variables are added. Those pairwise coupled flow variables are referred to as $f_{jl}\left(\cdot,\cdot\right)$, where jl refers to the indices of the pair of coupled daughter nodes. Since these flow variables are coupled, they always are assigned the same value

$$f_{jl}\left(s_{i,t+1}, o_{j,t+1}\right) = f_{jl}\left(s_{i,t+1}, o_{l,t+1}\right) \; . \tag{3.17}$$

During mitosis, from each mother object at most one pair of daughter objects can emerge, which is modeled by connecting the split node $s_{i,t+1}$ to at most two daughter nodes

$$\sum_{j} \sum_{\substack{l \\ l\neq j}} f_{jl}\left(s_{i,t+1}, o_{j,t+1}\right) \leq 2 \, . \tag{3.18}$$

In addition, a split node $s_{i,t+1}$ can either model mitosis or the splitting of objects. This is enforced by adding for all pairs of flow variables that correspond to edges starting from $s_{i,t+1}$ an inequality constraint:

$$f_{jl}\left(s_{i,t+1}, o_{j,t+1}\right) - f\left(s_{i,t+1}, o_{l,t+1}\right) \leq 1 \, . \tag{3.19}$$

The number of daughter pairs, and hence the number of pairwise coupled flow variables, grows quadratically with the number of potential daughter objects. For instance, if an object has ten potential daughter objects, the number of potential daughter pairs is $\binom{10}{2} = 45$. As each mitosis is modeled by two coupled flow variables, 90 coupled flow variables need to be added to the optimization problem. To reduce the number of pairwise coupled flow variables, and hence the size of the optimization problem, for each segmented object $o_{i,t+1}$ only ten potential mitosis pairs are considered. These potential mitosis pairs are chosen by selecting the ten potential mitosis pairs with the smallest mitosis costs, which is given in Equation 3.23. As the number of potential daughter objects is limited by the number of objects overlapping with the ROI and the mitosis costs are based on the Euclidean distances of three vectors, which are 2- or 3-dimensional, the mitosis costs can be calculated quickly.

Cost Functions

In the following, edge costs $c\left(u, v\right)$ are defined based on positional features only, which can be used over a variety of image sequences in 2D and 3D. This has the advantage that no additional labeled data or fine-tuning is needed, which might otherwise be required to learn features or tune the weighting between different types of features such as visual appearance and position. The cost functions are derived using domain-specific knowledge: (i) the object movement between successive frames is small and objects tend to perform random movements, (ii) after mitosis the emerging daughter objects are close to the former position of the mother object, and (iii) objects appear and disappear when they enter or leave the field of view at the image border. In general, arbitrary costs can be chosen for the tracking approach to model complex domain-specific behavior such as mitosis. However, proposing cost functions to model domain-specific knowledge, in general, is outside of the scope of this thesis.

For each segmented object based on its segmentation mask the following features are extracted: the mask centroid $\mathbf{c}_{i,t}$, a bounding box, and a set of mask points. The bounding box $\mathcal{B}_{i,t}$ is spanned by the top left and bottom right coordinates of the segmentation mask i at time point t and contains all points within the spanned rectangle. The set of mask points $\mathcal{Q}_{i,t}$ is derived by calculating a distance transformation on the segmentation mask, where a single point is referred to as $\mathbf{q}_{i,t}$. A visualization of the extracted features is shown in Figure 3.4.

The features are propagated over time by using the estimated displacement $\hat{\mathbf{d}}_{i,\cdot,\cdot}$, from the tracklet step:

$$\begin{aligned} \hat{\mathbf{c}}_{i,t+1} &= \mathbf{c}_{i,t} + \hat{\mathbf{d}}_{i,t,t+1}\,, \\ \widehat{\mathcal{B}}_{i,t+1} &= \left\{ \mathbf{b}_{i,t} + \hat{\mathbf{d}}_{i,t,t+1} \;\middle|\; \mathbf{b}_{i,t} \in \mathcal{B}_{i,t} \right\}\,, \\ \widehat{\mathcal{Q}}_{i,t+1} &= \left\{ \mathbf{q}_{i,t} + \hat{\mathbf{d}}_{i,t,t+1} \;\middle|\; \mathbf{q}_{i,t} \in \mathcal{Q}_{i,t} \right\}\,. \end{aligned} \tag{3.20}$$

Costs between object nodes model the movement of an object between successive time points:

$$c\left(o_{i,t}, o_{j,t+1}\right) = \left\|\hat{\mathbf{c}}_{i,t+1} - \mathbf{c}_{j,t+1}\right\|_2\,, \tag{3.21}$$

where $\hat{\mathbf{c}}_{i,t+1}$ is the estimated mask centroid of object i at time point $t+1$ and $\mathbf{c}_{j,t+1}$ the mask centroid of object j at time point $t+1$. The costs are small if the estimated position, which is derived in the tracklet step, is close to the position of a segmented object at the subsequent time point.

As skip nodes model FNs, no position of the segmented object is available since the segmentation mask is missing. Hence, edge costs involving skip nodes are based on the propagated position of the segmented object

$$\begin{aligned} c\left(o_{i,t}, x_{i,t+1}\right) &= \begin{cases} \left\|\mathbf{c}_{i,t} - \hat{\mathbf{c}}_{i,t+1}\right\|_2 = \left\|\hat{\mathbf{d}}_{i,t,t+1}\right\|_2 & \text{if } \hat{\mathbf{c}}_{i,t+1} \notin \mathcal{B}_{j,t+1}\ \forall j \\ \theta & \text{else} \end{cases}\,, \\ c\left(x_{i,t+1}, x_{i,t+2}\right) &= \begin{cases} \left\|\hat{\mathbf{c}}_{i,t+1} - \hat{\mathbf{c}}_{i,t+2}\right\|_2 = \left\|\hat{\mathbf{d}}_{i,t+1,t+2}\right\|_2 & \text{if } \hat{\mathbf{c}}_{i,t+2} \notin \mathcal{B}_{j,t+2}\ \forall j \\ \theta & \text{else} \end{cases}\,, \\ c\left(x_{i,t+1}, o_{j,t+2}\right) &= \left\|\hat{\mathbf{c}}_{i,t+2} - \mathbf{c}_{j,t+2}\right\|_2\,, \end{aligned} \tag{3.22}$$

where θ is a large constant.

The mitosis costs for the pairwise coupled flow variables are defined as

$$\begin{aligned} c_1 &= \left\|\mathbf{c}_{i,t} - \frac{1}{2}\left(\mathbf{c}_{j,t+1} + \mathbf{c}_{l,t+1}\right)\right\|_2\,, \\ c_2 &= \left|\left\|\mathbf{c}_{i,t} - \mathbf{c}_{j,t+1}\right\|_2 - \left\|\mathbf{c}_{i,t} - \mathbf{c}_{l,t+1}\right\|_2\right|\,, \\ c_3 &= \left\|\mathbf{c}_{j,t+1} - \mathbf{c}_{l,t+1}\right\|_2\,, \\ c_{jl}\left(s_{i,t+1}, o_{j,t+1}\right) = c_{jl}\left(s_{i,t+1}, o_{l,t+1}\right) &= \begin{cases} c_1 + c_2 & \text{if } c_3 \leq 1.5 b_{i,t} \\ \theta & \text{else} \end{cases}\,, \end{aligned} \tag{3.23}$$

where $b_{i,t}$ is the length of the diagonal spanned by the top left and bottom right coordinate of the bounding box. The costs enforce that daughter objects have a similar distance to the mother object, their average position is close to the previous position of the mother

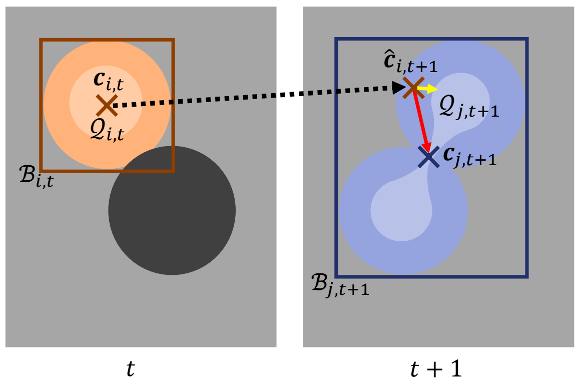


Figure 3.4.: Extracted features to link the two segmented objects at t to the segmented object at $t+1$. For each segmentation mask i at time point t the mask centroid $\mathbf{c}_{i,t}$ – shown as a cross –, a set of mask points $\mathcal{Q}_{i,t}$ – shown in a lighter shade – and a bounding box $\mathcal{B}_{i,t}$ – shown as a rectangle – are extracted. While the Euclidean distance between the mask centroid $\mathbf{c}_{j,t+1}$ and the propagated mask centroid $\hat{\mathbf{c}}_{i,t+1}$ is large – red arrow, which can result in wrong links, the minimal Euclidean distance between the propagated mask centroid $\hat{\mathbf{c}}_{i,t+1}$ and the set of mask points $\mathcal{Q}_{j,t+1}$ – yellow arrow – is small. Derivative of Figure 3 by Löffler *et al.* [177] licensed under CC BY 4.0; rearranged plot and changed colors.

object and the distance between the daughter objects is small. An estimated position of the mother object is not used, as the displacement estimation which is based on the appearance of image crops is unreliable, when one image crop shows a single mother object and the other shows two daughter objects.

In the case of splitting and merging of objects, costs based on mask centroids can lead to large cost terms, as the Euclidean distance between the propagated mask centroid of a correctly segmented object and the mask centroid of merged objects can be large, which is shown in Figure 3.4. Therefore, the set of mask points and the centroid position are used to derive cost terms. For splitting of objects, the following costs are defined

$$c\left(s_{i,t}, o_{j,t}\right) = \begin{cases} \min\left(\left\{\left\|\mathbf{q}_{i,t-1} - \mathbf{c}_{j,t}\right\|_2 \mid \mathbf{q}_{i,t-1} \in \mathcal{Q}_{i,t-1}\right\}\right) & \text{if } \mathbf{c}_{j,t} \in \mathcal{B}_{i,t-1} \\ \theta & \text{else} \end{cases}, \tag{3.24}$$

where i is a segmented object at time point t and $\mathcal{Q}_{i,t-1}$ is the set of mask points and $\mathbf{q}_{i,t-1}$ a mask point of the segmented object i propagated at the previous time point $t-1$.

For merging objects the following costs are defined

$$c\left(o_{j,t}, m_{i,t}\right) = \begin{cases} \min\left(\left\{\left\|\mathbf{q}_{i,t+1} - \hat{\mathbf{c}}_{j,t+1}\right\|_2 \mid \mathbf{q}_{i,t+1} \in \mathcal{Q}_{i,t+1}\right\}\right) & \text{if } \hat{\mathbf{c}}_{j,t+1} \in \mathcal{B}_{i,t+1} \\ \theta & \text{else} \end{cases}, \tag{3.25}$$

where $\hat{\mathbf{c}}_{j,t+1}$ is the predicted position of the segmented object j at time point $t+1$ and $\mathcal{Q}_{i,t+1}$ the set of mask points of the segmented object i at the subsequent time point $t+1$.

The appearance and disappearance an object is mainly caused by the object entering or leaving the field of view. In image sequences, the field of view is spanned by the image borders. To model objects entering and leaving the field of view, appearance costs depend on a threshold α and the minimum distance of the mask centroid $\mathbf{c}_{i,t}$ to the image border

$$c\left(a_{t-1}, o_{i,t}\right) = \min\left(\alpha, \min\left(\min\left(\mathbf{a} - \mathbf{c}_{i,t}\right), \min\left(\mathbf{c}_{i,t}\right)\right)\right), \tag{3.26}$$

disappearance costs are defined similarly

$$c\left(o_{i,t}, d_{t+1}\right) = \min\left(\alpha, \min\left(\min\left(\mathbf{a} - \mathbf{c}_{i,t}\right), \min\left(\mathbf{c}_{i,t}\right)\right)\right), \tag{3.27}$$

where $\mathbf{a}$ is the image size written as a vector and $\min\left(\min\left(\mathbf{a} - \mathbf{c}_{i,t}\right), \min\left(\mathbf{c}_{i,t}\right)\right)$ the minimal distance to the image border.

The parameter θ is set to $\theta = 1000\alpha$, where α is derived from the longest border of the rectangular-shaped ROI, which is set to a multiple of the size of the median segmentation mask in an image sequence. All other edges are assigned 0 cost. However, there are no paths a unit can take through the graph from source to sink node without causing a cost. For instance, edges between source and appearance nodes have assigned a cost of 0. Next, since the flow requirement needs to be full-filled and the unit has to end in the sink node, the unit can either flow to an object node or a split node, as an appearance node can be only connected to object or split nodes. The flow from an appearance node to an object node adds an appearance cost, whereas the flow to a split node adds a splitting cost when the unit flows from the split node to the object node. Figure 3.3 provides examples of the assigned edge costs between different nodes.

To reduce the number of flow variables further, edges with costs that surpass the appearance or disappearance costs of a node are pruned. An edge, and hence its flow variable, is pruned by setting the upper capacity constraint of this edge to zero. Finally, the formulated problem can be solved using mixed integer linear programming with a standard optimization toolbox such as Gurobi [116].

After solving the mixed integer linear program, the tracking graph can be extracted by following the found paths through the graph and assigning segmented objects to tracks if their corresponding object nodes are connected by the same path. On nodes where several paths start/end, new tracks are created and the predecessor/successor information is kept.

3.2.3. Step 3: Post-Processing Step

After the matching step, the segmented objects are linked over time, however, segmentation errors are not corrected. In the post-processing step, over- and under-segmentation errors are resolved and missing segmentation masks are added to resolve FNs. Figure 3.5 illustrates the tracking graph and segmentation masks after the different processing steps.

An Intuitive Introduction to the Untangling Problem

After the matching step, tracks can be assigned to more than one predecessor and can have more than two successors as shown on top in Figure 3.6. These many-to-one and one-to-many assignments are now resolved, so each track has at most one predecessor and at most two successors to model mitosis. As the tracks are "untangled", this step is referred to as untangling step. The tracking graph is transformed by applying a set of modifications

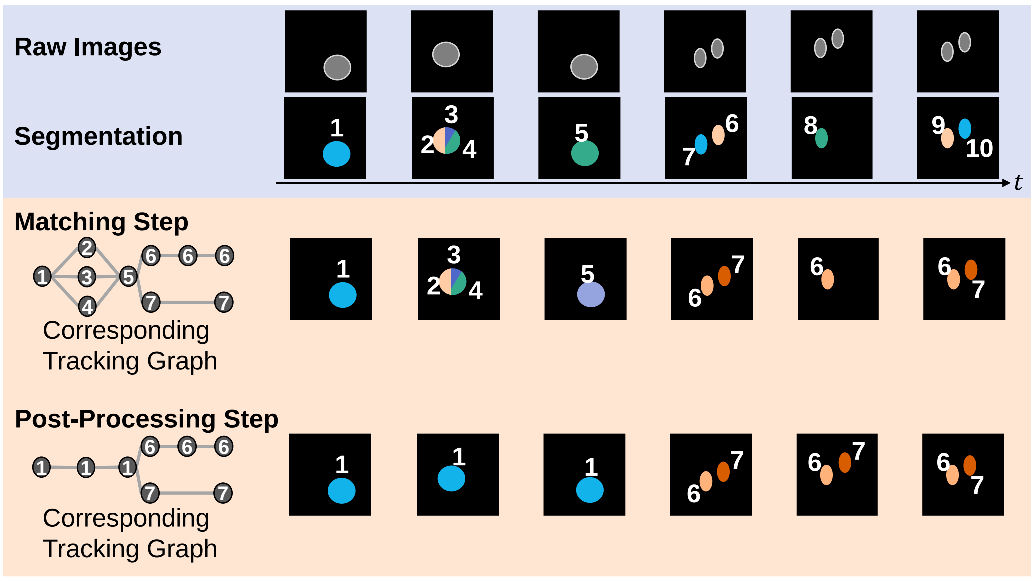


Figure 3.5.: Segmentation masks and tracking graphs after matching step and post-processing step, respectively. During the matching step, the segmented objects are linked over time, however, tracking errors still exist. In the post-processing step, the many-to-one and one-to-many links of the tracking graph are resolved and missing segmentation masks are added. Finally, after the post-processing step, the segmented objects are linked and segmentation errors are corrected.

to it, which are referred to as untangling operations: remove an edge, split a track, and merge tracks. The edge remove operation removes a single predecessor-successor link. The split operation splits a track into several tracks – to correct under-segmentation errors, whereas the merge tracks operation merges several tracks resulting in a single track – to correct over-segmentation errors. A visualization of the untangling operations is shown on the left of Figure 3.6.

Different combinations of untangling operations lead to valid tracking graphs, which are shown on the right of Figure 3.6. Based on the assumption that the objects are the majority of time points correctly segmented, the idea is to select the smallest number of untangling operations to transform the tracking graph.

Based on the prior knowledge that the image sequence can contain mitosis events, each track should have at maximum one predecessor and at maximum two successors. However, applying an untangling operation also affects the predecessor and successor tracks of track ω_n since the number of their respective successors and predecessors change. Let in the following be $\mathcal{P}_n$ the set of predecessor tracks ω_n is connected to and $\mathcal{S}_n$ the set of successor tracks the track ω_n is connected to. In addition, let ω_p be a predecessor track of ω_n and ω_s a successor track of ω_n. If an edge is removed between the predecessor track ω_p and ω_n, the number of successors of track ω_p reduces by one as well as the number of predecessors of track ω_n reduces by one. Splitting the track ω_n into N tracks increases the number of successor tracks by $N-1$ for each of its predecessor tracks in $\mathcal{P}_n$. Similarly, the number of predecessor tracks increases by $N-1$ for each of its successor tracks in $\mathcal{S}_n$.

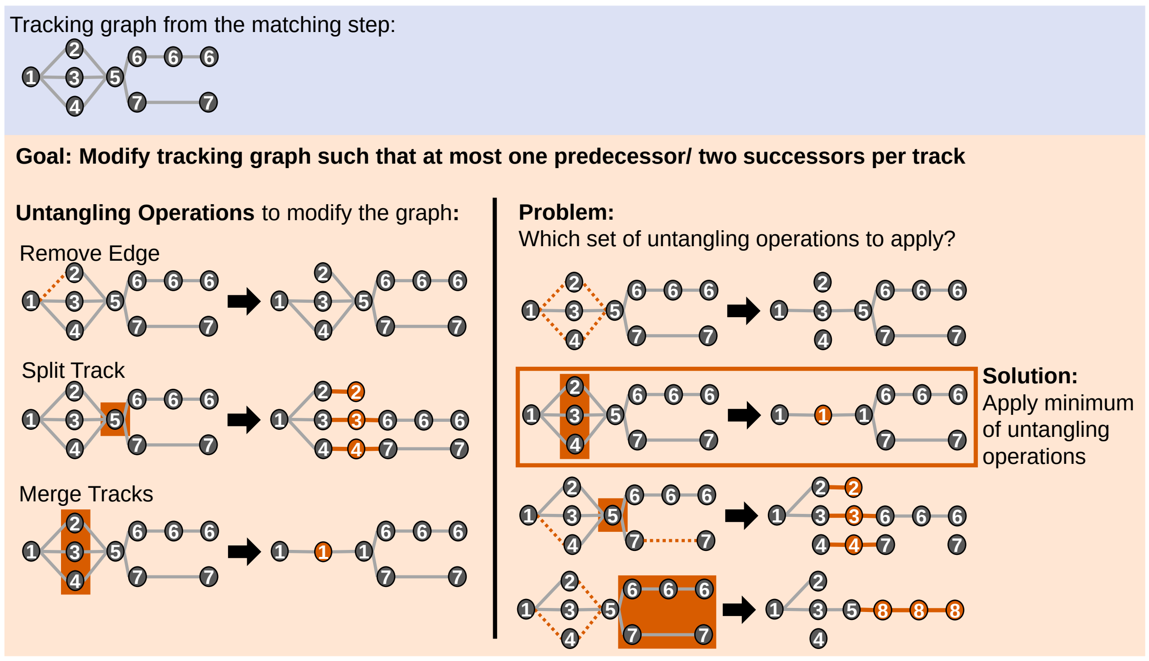

Tracking graph from the matching step:
Goal: Modify tracking graph such that at most one predecessor/ two successors per track
Untangling Operations to modify the graph:
Remove Edge
Split Track
Merge Tracks
Problem:
Which set of untangling operations to apply?
Solution:
Apply minimum of untangling operations

Figure 3.6.: Modifying the tracking graph to correct segmentation errors. The tracking graph is modified by applying untangling operations such that each track has at most one predecessor and at most two successors – to model mitosis. Different combinations of untangling operations, however, all lead to valid tracking graphs. Therefore, the solution that requires the smallest set of untangling operations is selected – highlighted with a box. Derivative of Figure 4 by Löffler *et al.* [177] licensed under CC BY 4.0; rearranged, extended plot and changed colors.

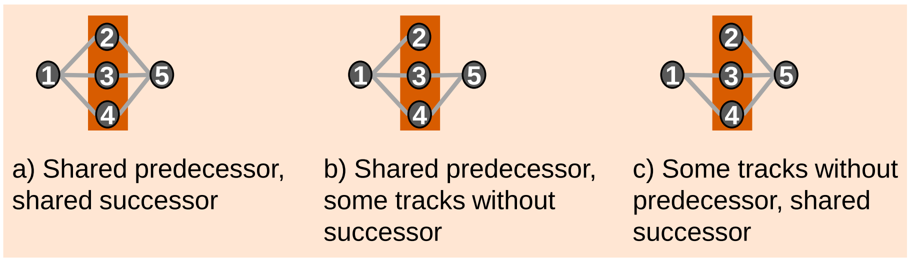

1
2
3
4
5
a) Shared predecessor, shared successor
b) Shared predecessor, some tracks without successor
c) Some tracks without predecessor, shared successor

Figure 3.7.: Cases under which tracks can be merged in the post-processing step. The three tracks with the IDs 2, 3, and 4 can be merged into a single track, although track 2 has no successor (b) or predecessor (c).

The tracking graph models one-to-many and many-to-one links, hence, several tracks can share the same successor – a many-to-one link – or several tracks can share the same predecessor – a one-to-many link. However, no many-to-many links exist, hence, there are no sets of tracks that share multiple predecessors or successors. However, there can be sets of tracks that share the same successor but a subset of tracks has no predecessor or a set of tracks that share the same predecessor but a subset of tracks has no successor. This can be summarized as conditions under which tracks can be merged: a) tracks share the same predecessor and successor, b) tracks share the same successors and some tracks have no predecessor but begin after the track with a predecessor starts, or c) tracks share the same predecessors and some tracks have no successors but end before the track with successors end. Figure 3.7 provides a visualization of the different cases under which tracks can be merged.

In addition, there are cases where only a subset of tracks that fulfill the merging conditions should be merged. An example for such a case is a track with three successors, where two successors refer to an over-segmented object which are the tracks $\{2, 3, 4\}$ in the tracking graph in Figure 3.6. By just merging two of the successors, the over-segmentation error could be resolved. Therefore, all subsets of tracks are constructed that could potentially be merged which are referred to as possible sets of mergeable tracks. In the following, let ω_a and ω_b be tracks that share the same predecessor track ω_p and successor track ω_s with ω_n. As the tracks ω_n, ω_a, and ω_b fulfill the merging conditions, the potential set of tracks ω_n can be merged with is $\mathcal{M}_n = \{\{\omega_a, \omega_n\}, \{\omega_b, \omega_n\}, \{\omega_a, \omega_b, \omega_n\}\}$. However, from the perspective of the predecessor and successor track, further sets of mergeable sets can exist, which do not include ω_n. For the predecessor track ω_p the set that consists of all possible sets of mergeable tracks that contain successor tracks of it is given by $\mathcal{M}\left(\mathcal{S}_p\right) = \{\{\omega_a, \omega_b\}, \{\omega_a, \omega_n\}, \{\omega_b, \omega_n\}, \{\omega_a, \omega_b, \omega_n\}\}$. Since all three tracks in this example, share the same predecessor and successor, the set that consists of all possible sets of mergeable tracks that contain predecessor tracks of the track ω_s, $\mathcal{M}\left(\mathcal{P}_s\right)$, is identical to $\mathcal{M}\left(\mathcal{S}_p\right)$.

Merging a set of tracks reduces for a predecessor track its number of successors by the number of its successors that were merged minus one. Similarly, for a successor track, its number of predecessors is reduced by the number of predecessors that were merged minus one. For instance, if the tracks ω_a and ω_b are merged, the number of successors of the predecessor track reduces from three, $\mathcal{S}_p = \{\omega_a, \omega_b, \omega_n\}$, to two, $\mathcal{S}_p = \{\omega_{ab}, \omega_n\}$, where ω_{ab} refers to the track created by merging the tracks ω_a and ω_b. Similarly, the number of predecessors of the successor track reduces from three, $\mathcal{P}_s = \{\omega_a, \omega_b, \omega_n\}$, to two, $\mathcal{P}_s = \{\omega_{ab}, \omega_n\}$.

Formulation of the Untangling Problem as an Optimization Problem

The task is to select a combination of untangling operations from the many possible options such that the tracking graph is modified as little as possible, which is modeled as a minimization problem

$$\min_z \sum_k c(z_k) z_k \, , \tag{3.28}$$

where $c(z_k)$ are costs and z_k variables refer to the applied untangling operations on the graph. The untangling operations z_k are denoted as follows: an edge remove operation is denoted as z^e_{pn}, where the predecessor track is ω_p and the successor track is ω_n. Splitting a track ω_n into several tracks is denoted as z^s_n, whereas merging a set of tracks is denoted as z^m_r, where r is an index that references a set of tracks in the set $\mathcal{M}_n$.

Based on the prior knowledge about the maximum number of predecessors and successors and how the untangling operations affect the number of predecessors and successors, for each track, two inequality constraints are added to the optimization problem – one modeling the predecessor side and the other one modeling the successor side of the track. The number of predecessors of a track ω_n is referred to as $|\mathcal{P}_n|$, whereas the number of successors of a track is referred to as $|\mathcal{S}_n|$. For each set of tracks r that can be merged, modeled by z^m_r, the number of tracks sharing the same predecessor as track ω_n is denoted as $P_{n,r}$ and the number of tracks sharing the same successor as track ω_n as $S_{n,r}$. Furthermore, for each predecessor track of track ω_n, all sets of tracks the predecessor track can be merged with are computed, where the set $\mathcal{M}\left(\mathcal{P}_n\right)$ consists of all possible sets of mergeable tracks that contain predecessor tracks of track ω_n. Analogously, for each successor track of track ω_n, all sets of tracks the successor track can be merged with are computed, where the set $\mathcal{M}\left(\mathcal{S}_n\right)$ consists of all possible sets of mergeable tracks that contain successor tracks of track ω_n.

As a track is influenced by the untangling operations applied to its predecessor or successor tracks, the predecessor inequality constraint has to include the untangling operations on the predecessor tracks, whereas the successor inequality constraint has to include the untangling operations on the successor tracks. Furthermore, as several tracks can share the same predecessor or successor, these tracks need to be considered in the inequality constraints as well. The predecessor inequality constraint for track ω_n is given as

$$\begin{aligned}
&\overbrace{\sum_{r\in\mathcal{M}_r}\left(P_{n,r}-1\right)z^m_r}^{\text{merge tracks}}-\overbrace{\sum_{w\in\mathcal{W}_n} z^s_w}^{\text{split tracks}}-\overbrace{\sum_{w\in\mathcal{W}_n}\sum_{p\in\mathcal{P}_w} z^e_{pw}}^{\text{remove edges to predecessors}}\\
&+\underbrace{\sum_{p\in\mathcal{P}_n} z^s_p}_{\text{split predecessors}}-\underbrace{\sum_{q\in\mathcal{M}(\mathcal{P}_n)}\max\left(0,S_{q,n}-1\right)z^m_q}_{\text{merge predecessors}}\leq-\left|\mathcal{P}_n\right|+\max\left(1,\left|\left\{\mathcal{S}_p \mid p\in\mathcal{P}_n\right\}\right|\right)\,,
\end{aligned} \tag{3.29}$$

where r and q are indices that each reference a set of tracks in the sets of mergeable tracks $\mathcal{M}_r$ and $\mathcal{M}\left(P_n\right)$ respectively, and w and p are indices referring to a single track. The variables to be optimized are the merge track variables z^m_r and z^m_q, the split track variables z^s_w and z^s_p, and the edge remove variables z^e_{pw}, where z^m_r denotes merging the set of tracks r into a single track, z^s_w denotes splitting the track ω_w, creating z^s_w additional tracks, and z^e_{pw} denotes removing the predecessor-successor link between the predecessor track ω_p and the successor track ω_w. The set $\mathcal{M}_n$ contains all sets of tracks that can be merged with the track ω_n, $P_{n,r}$ is the number of tracks of the set of mergeable tracks r that share the same predecessors as ω_n, $\mathcal{W}_n$ is a set, which contains all tracks that can be merged with track ω_n including the track ω_n. $|\mathcal{P}_n|$ is the number of predecessors of track ω_n, whereas $\mathcal{P}_w$ is the set of predecessors of track ω_w. $S_{q,n}$ is the number of tracks of the set of mergeable tracks q that have track ω_n as a successor. The total number of successors of the predecessors of track ω_n is given by $\left|\left\{\mathcal{S}_p \mid p\in\mathcal{P}_n\right\}\right|$.

The successor inequality constraint is given as:

$$\begin{aligned}
&\overbrace{\sum_{r\in\mathcal{M}_n}\left(S_{n,r}-1\right)z^m_r}^{\text{merge tracks}}-\overbrace{\sum_{w\in\mathcal{W}_n} z^s_w}^{\text{split tracks}}-\overbrace{\sum_{w\in\mathcal{W}_n}\sum_{v\in\mathcal{S}_w} z^s_{wv}}^{\text{remove edges to successors}}\\
&+\underbrace{\sum_{v\in\mathcal{S}_n} z^s_v}_{\text{split successors}}-\underbrace{\sum_{q\in\mathcal{M}(\mathcal{S}_n)}\max\left(0,P_{q,n}-1\right)z^m_q}_{\text{merge successors}}\leq-\left|\mathcal{S}_n\right|+2\left|\left\{\mathcal{P}_v \mid v\in\mathcal{S}_v\right\}\right|+1\,,
\end{aligned} \tag{3.30}$$

where r and q are indices that each reference a set of tracks in the sets of mergeable tracks $\mathcal{M}_n$ and $\mathcal{M}\left(S_n\right)$ respectively, and w and v are indices referring to a single track. The variables to be optimized are the merge tracks variables z^m_r and z^m_q, the split track variables z^s_w and z^s_v, and the edge remove variables z^e_{wv}, where z^m_r denotes merging the set of tracks r into a single track, denotes splitting the track ω_w, creating z^s_w additional tracks, and z^e_{wv} denotes removing the predecessor-successor link between the predecessor track ω_w and the successor track ω_v. The set $\mathcal{M}_n$ contains all sets of tracks that can be merged with the track ω_n, $S_{n,r}$ is the number of tracks of the set of mergeable tracks r that share the same successors as track ω_n, $P_{q,n}$ is the number of tracks of the set of mergeable tracks q that have track ω_n as a predecessor, and $\mathcal{W}_n$ is a set, which contains all tracks that can be merged with track ω_n including the track ω_n. $|\mathcal{S}_n|$ is the number of successors of track ω_n, whereas $\mathcal{S}_w$ is the set of successors of track ω_w. $P_{q,n}$ is the number of tracks of the set of mergeable tracks q that have track ω_n as a predecessor. The total number of predecessors of the successors of track ω_n is given by $\left|\left\{\mathcal{P}_v \;\middle|\; v \in \mathcal{S}_v\right\}\right|$.

In addition, a track can be merged with at most one set of tracks r, which is modeled by adding for each track a constraint

$$\sum_{r \in \mathcal{M}_r} z_r^m \leq 1\,. \tag{3.31}$$

Moreover, if a set of tracks is to be merged, their edge remove operations are coupled, such that for merged tracks either all edges are removed on the predecessor or successor side or none. To enforce this, for each set of mergeable tracks r all pairs of tracks are constructed that share a predecessor or successor, and two constraints are added

$$\begin{aligned} z_{pv}^e - z_{pn}^e &\leq -z_r^m + 1\,, \\ -z_{pv}^e + z_{pn}^e &\leq -z_r^m + 1\,, \end{aligned} \tag{3.32}$$

where $r = \{n, v, \dots\}$ and the tracks ω_n and ω_v share the predecessor track ω_p.

The merge tracks and edge remove variables are constrained to be binary variables, whereas the split variables are of integer type to provide the number of additionally created tracks.

Predecessor and Successor Inequality Constraints Example

In the following, the setup of the proposed inequality constraints from Equation 3.29 and Equation 3.30 is illustrated for the track with track ID 5 from the tracking graph shown in Figure 3.6. The track is connected to three predecessor tracks with the track IDs 2, 3, 4 and two successor tracks with the track IDs 6 and 7. As the track does not share its predecessors or successors with other tracks, there are no tracks the track can be merged with, therefore, the set containing all sets of mergeable tracks is $\mathcal{M}_5 = \{\}$ and the set of tracks which can be merged with track 5 is $\mathcal{W}_5 = \{5\}$. The set of predecessor tracks is $\mathcal{P}_5 = \{2, 3, 4\}$ and $|\mathcal{P}_5| = 3$, whereas the set of successor tracks is $\mathcal{S}_5 = \{5, 6\}$ and $|\mathcal{S}_5| = 2$.

The set containing all possible sets of mergeable tracks that contain predecessor tracks is $\mathcal{M}(\mathcal{P}_5) = \{\{2, 3\}, \{2, 4\}, \{3, 4\}, \{2, 3, 4\}\}$, whereas the set containing all possible sets of mergeable tracks that contain successor tracks is $\mathcal{M}(\mathcal{S}_5) = \{\{6, 7\}\}$. The predecessors of track 5 have only one successor, which is track 5, resulting in $|\left\{\mathcal{S}_p \;\middle|\; p \in \mathcal{P}_5\right\}| = 1$. The successor tracks of track 5 have only one predecessor, which is track 5, resulting in $|\left\{\mathcal{P}_v \;\middle|\; v \in \mathcal{S}_5\right\}| = 1$.

By merging predecessor tracks or successor tracks into a single track, the number of predecessors or successors of a track is connected to changes. The change in the number of predecessors or successors if sets of them are merged is represented by the terms $(P_{n,r} - 1)$ and $\max(0, S_{q,n} - 1)$ from Equation 3.29, and $(S_{n,r} - 1)$ and $\max(0, P_{q,n} - 1)$ in Equation 3.30. For example, by merging the tracks $\{2, 3, 4\}$ into a single track, which is modeled by $z_{\{2,3,4\}}^m$, two predecessor links of track 5 are removed, as now instead of three predecessor tracks only one predecessor track is connected to it. Therefore, $z_{\{2,3,4\}}^m$ is multiplied by a factor of 2.

After inserting the terms in the inequality constraints, the predecessor inequality constraint of track 5 is given by

$$\begin{aligned}&\overbrace{0}^{\text{merge tracks}} - \overbrace{z_5^s}^{\text{split tracks}} - \overbrace{\left(z_{2,5}^e + z_{3,5}^e + z_{4,5}^e\right)}^{\text{remove edges to predecessors}} \\ &+ \underbrace{z_2^s + z_3^s + z_4^s}_{\text{split predecessors}} \underbrace{-2z_{\{2,3,4\}}^m - z_{\{2,3\}}^m - z_{\{2,4\}}^m - z_{\{3,4\}}^m}_{\text{merge predecessors}} \leq -3 + 1 = -2\,,\end{aligned} \tag{3.33}$$

whereas the successor inequality constraint is given by

$$\begin{aligned}&\overbrace{0}^{\text{merge tracks}} - \overbrace{z_5^s}^{\text{split tracks}} - \overbrace{\left(z_{5,6}^e + z_{5,7}^e\right)}^{\text{remove edges to successors}} \\ &+ \underbrace{z_6^s + z_7^s}_{\text{split successors}} \underbrace{-z_{\{6,7\}}^m}_{\text{merge successors}} \leq -2 + 2 + 1 = 1\,.\end{aligned} \tag{3.34}$$

The successor inequality constraint Equation 3.34 is fulfilled without applying untangling operations, as the right-hand side of the inequality constraint is 1. However, untangling operations need to be applied so the predecessor inequality constraint Equation 3.29 holds, as the right-hand side of the inequality constraint is -2. This is valid, as track 5 has three predecessors and two successors and the aim of the untangling step is to transform the tracking graph such that each track has at most one predecessor and at most two successors.

Untangling Costs

The untangling costs can be chosen arbitrarily. For instance, features based on the visual appearance of the segmentation masks of the track, or spatial-temporal features could be extracted. Here, simple cost terms based on the temporal length and number of merged tracks are proposed:

$$\begin{aligned} c(z_{pn}^e) &= E\,, \\ c(z_r^m) &= \Delta\omega_r \left(M_r - 1\right)\,, \\ c(z_n^s) &= \Delta\omega_n\,, \end{aligned} \tag{3.35}$$

where $c(z_{pn}^e)$ is the cost of removing the edge between the tracks ω_p and ω_n, $c(z_r^m)$ is the cost of merging the set of tracks r, $c(z_n^s)$ is the cost of splitting track ω_n, E is a constant, M_r is the number of merged tracks, $\Delta\omega_n$ is the temporal length of the track ω_n and $\Delta\omega_r$ is the temporal length of the track after merging, respectively.

Splitting a track creates z_n^s additional tracks, whereas merging M_r tracks into a single track results in a reduction by $M_r - 1$ tracks. For the chosen cost functions, merging K tracks or splitting a track into K tracks over the same temporal length $\Delta\omega$, results in the same change of the value of the objective function, as z_r^m is constrained to be binary, whereas z_n^s is integer-valued

$$\begin{aligned}&\text{merging } K \text{ tracks into 1 track: } \Delta\omega\,(K-1)\cdot z_r^m = \Delta\omega\,(K-1)\ \text{, where } z_r^m = 1\,,\\&\text{splitting 1 track into } K \text{ tracks: } \Delta\omega\cdot z_n^s = \Delta\omega\,(K-1)\ \text{, where } z_n^s = K-1\,.\end{aligned} \tag{3.36}$$

In theory, over- and under-segmentation errors can be resolved by only applying merging and splitting of tracks. However, there can be constellations where removing edges provides better tracking results, for instance, due to a wrong link assigned in the matching step. Removing an edge should add more costs than correcting over- and under-segmentation errors through merging or splitting tracks. However, removing an edge should not be too costly to avoid splitting or merging tracks erroneously instead of removing the erroneously assigned link. Hence, the costs to remove an edge need to be chosen with respect to the costs of merging and splitting tracks, which depend on the number of time steps over which this operation is applied – the length of the erroneous tracks – and the number of tracks to merge or split, respectively. To find reasonable costs to remove edges, the number of time points tracks need to be corrected and the number of tracks that are merged or a track is split into are estimated from the tracking graph. To estimate the length of erroneous tracks, it is assumed that segmentation errors occur only for short time spans, leading to short erroneous tracks consisting of merged objects or several tracks due to over-segmentation. Based on this assumption, the shortest n% of tracks in a tracking graph are likely caused by segmentation errors. To estimate the number of tracks that need to be merged or a track is split into, tracks that have excessively many predecessors or successors are assumed to be caused by over- or under-segmentation errors. Based on these assumptions the cost to remove edges, E is set to $2\cdot\lceil\Delta T_{0.3}\Delta N_{0.99}\rceil$, where $\Delta T_{0.3}$ is the 0.3 quantile of the track length and $\Delta N_{0.99}$ the 0.99 quantile of the number of predecessors/successor links per track.

The set of untangling operations is selected by solving a mixed integer linear program using a standard optimization toolbox such as Gurobi [116]. After solving the optimization problem, the selected set of untangling operations is applied to modify the tracking graph. Each selected predecessor-successor link, indicated by z_{pn}^e, is removed by dropping the track ω_p from the set of predecessor tracks of track ω_n, and the track ω_n is dropped from the set of successor tracks of track ω_p. For each split variable $z_n^s > 0$, the corresponding track ω_n is split into $z_n^s + 1$ tracks, by splitting each segmentation mask of the track into $z_n^s + 1$ fragments. Therefore, seed points that are far away from the segmentation mask border are calculated by thresholding a distance transform for each segmentation mask. The seed points are then either matched with the centroids of the predecessor tracks or the successor tracks: If the number of predecessors is equal to the number of tracks to create via splitting, the last positions of the predecessor tracks are matched. Otherwise, the first positions of the successor tracks are matched to the seed points. The matching is conducted for each segmentation mask of the track ω_n such that each centroid of the predecessor or successor tracks is matched to a seed point and the overall sum of Euclidean distances between the seed points and their matched centroids is minimal. Next, each pixel of the segmentation mask is matched to its closest seed point to create the segmentation mask fragments. Then, the predecessor and successor tracks are matched with the newly created tracks based on their centroid distances, such that each track has at most one predecessor and two successors. Tracks are merged by creating a new track which consists at each time point of the concatenated segmentation masks of the tracks to merge at each time point. The tracks to merge are then replaced by the newly created track and the successor tracks of the predecessor track and the predecessor tracks of the successor track are updated accordingly.

FN Correction

FN errors are corrected by adding segmentation masks to tracks with missing segmentation masks. Therefore, the last available segmentation mask, before a FN error occurs, is placed at positions computed from a linear interpolation between the available segmentation masks. In image sequences with densely placed objects, adding masks can lead to conflicts, where an interpolated mask overlaps with another segmentation mask. As conflicts can appear not only between pairs of segmentation masks but several segmentation masks, the overlaps are resolved when all sets of conflicting segmentation masks are known. Therefore, the segmentation masks are placed sequentially in the image plane. If a segmentation mask cannot be placed in the image plane without causing an overlap with another segmentation mask, a new image plane is created and the segmentation mask is placed in the new image plane. After all masks have been placed, conflicting pixels can be extracted by finding pixels, which have several segmentation mask IDs over the image plane dimension. To assign a pixel to one of the several segmentation masks, the segmentation mask centroids of the conflicting segmentation masks are calculated and the pixel is assigned to the segmentation mask with the closest centroid. This process is visualized in Figure 3.8.

3.3. Experiment

The proposed tracking algorithm is applied to the degraded segmentation data introduced in Section 2.2. The influence of the parametrization and post-processing concerning correcting segmentation errors is investigated, and the tracking algorithm is compared against the three selected tracking algorithms of Section 2.2. To compare the segmentation quality before and after applying a tracking approach, the SEG score of the erroneous segmentation data without any tracking approach applied is reported and is referred to as "No Tracking". Experiments concerning the runtime of the proposed method compared to other approaches on 2D and 3D datasets are conducted in Section 5.3.3.

3.3.1. Experimental Setup

The same synthetically degraded segmentation data are used for evaluation in Section 2.2. All parameters of the three tracking methods: MU-Lux-CZ, KTH-SE, and KIT-Sch-GE (1) are chosen as in Section 2.2. For the proposed tracking algorithm two parameters need to be set manually: Δt, and the default ROI size. Δt is set to 3 and the default ROI size is set to twice the size of the median segmentation mask. All other parameters of the proposed tracking method are estimated automatically from the data or are based on these two parameters. The code of the proposed graph-based tracking approach is written in Python and made publicly available at `https://git.scc.kit.edu/KIT-Sch-GE/2021-cell-tracking`.

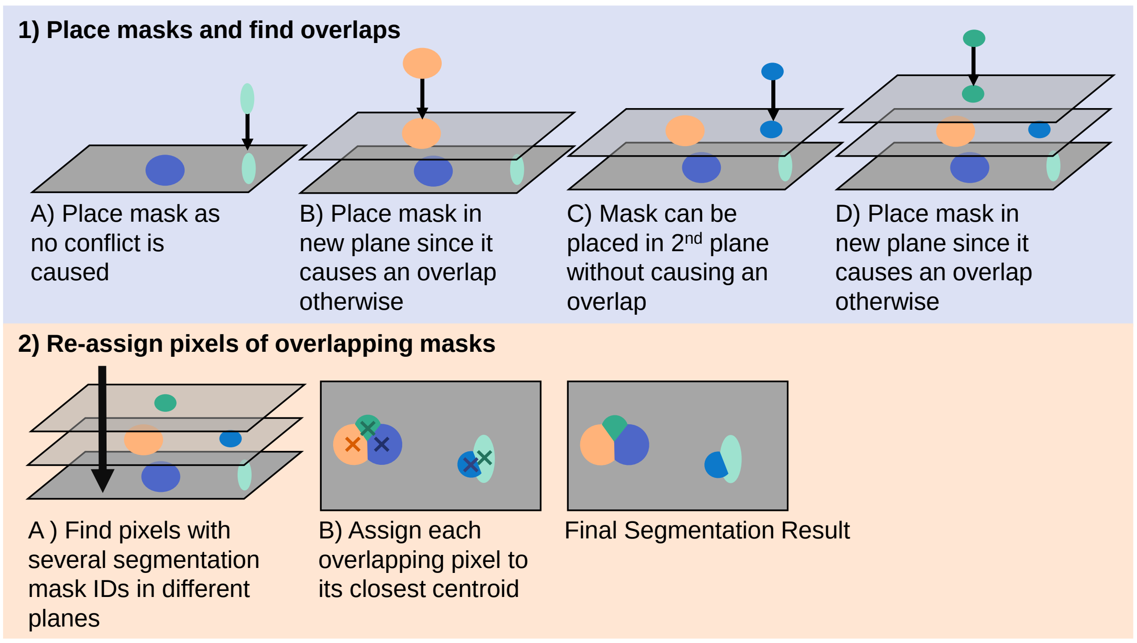

1) Place masks and find overlaps
A) Place mask as no conflict is caused
B) Place mask in new plane since it causes an overlap otherwise
C) Mask can be placed in 2nd plane without causing an overlap
D) Place mask in new plane since it causes an overlap otherwise
2) Re-assign pixels of overlapping masks
A) Find pixels with several segmentation mask IDs in different planes
B) Assign each overlapping pixel to its closest centroid
Final Segmentation Result

Figure 3.8.: Resolving overlapping segmentation masks. Interpolating missing segmentation masks can lead to masks with overlapping pixels. Therefore, the segmentation masks are placed sequentially and new image planes are added if a mask cannot be placed without overlapping with another mask. After all masks have been placed, overlapping pixels are extracted by finding pixels, which have several segmentation mask IDs over the image dimension plane. Each of these pixels is assigned to the segmentation mask with the closest centroid.

3.3.2. Evaluation

In the following, the influence of the two manually tunable parameters, Δt and the ROI size, and the post-processing steps is analyzed. In addition, the approach is compared to other tracking approaches.

Influence of the Manually Tunable Parameters Δt and ROI Size

First, the influence of the parameter Δt is examined. Therefore, the ROI size is kept fixed at twice the size of the median segmentation mask, while the parameter Δt is varied between $\{1, \ldots, 5\}$. The results are shown in Figure 3.9. The influence of the parameter Δt is strongest on image sequences with FNs, where increasing Δt improves the performance. This makes sense, as Δt models the length of the graph, and indirectly the maximum time span segmentation masks can be missing in a track to still reconstruct them. For $\Delta t = 1$ only two successive time points are linked in the graph and no skip nodes are added, hence, no FNs can be corrected. Therefore, the SEG score on image sequences with FNs is the same for tracking and no tracking. With an increasing fraction of FNs, it becomes more likely that segmentation masks are missing in a track at several successive time points. Hence, the performance difference between $\Delta t = 2$ and $\Delta t = 5$ becomes more distinct for large fractions of FNs. Since the mixed errors also contain FNs, the effect of Δt is also visible there, however, less distinctive as FNs cover just a third of the overall fraction of segmentation errors. For under- and over-segmentation errors, the performance remains roughly similar over different Δt, as these are modeled by the split and merge nodes, which are not affected by the temporal length of the graph.

Next, the influence of the ROI size is investigated. Therefore, the parameter Δt is kept fixed at 3, while the ROI size is varied between one up to four times the size of the median segmentation mask, which is computed for each image sequence individually. The results are shown in Figure 3.10. The influence of the parameter is strongest for over-segmentation errors and FNs, whereas for under-segmentation errors and mixed errors the tracking performance is roughly similar over different ROI sizes. For over-segmentation, setting the ROI size to the size of the median segmentation mask – ROI size 1 in the Figure – decreases performance for large fractions of over-segmentation errors. Each over-segmentation error yields two up to four segmentation mask fragments, each smaller than the original segmentation mask. Since the ROI depends on the size of the median segmentation mask, the increasing fraction of over-segmentation errors results in more and more small segmentation mask fragments, which yields to a decreasing size of the median segmentation mask and hence the ROI. If the ROI is too small, fewer potential matching candidates are selected in the tracklet step, resulting occasionally in missing links and no correction of the over-segmentation errors. In contrast, under-segmentation errors lead to an increasing size of the median segmentation mask. Consequently, the effects on the ROI size roughly cancel each other out, resulting in a similar tracking performance over the different ROI sizes for image sequences with mixed segmentation errors. For FNs, in contrast, a large ROI size can occasionally lead to linking an object to its closest neighbor in the next frame, which is then erroneously detected as an under-segmentation error.

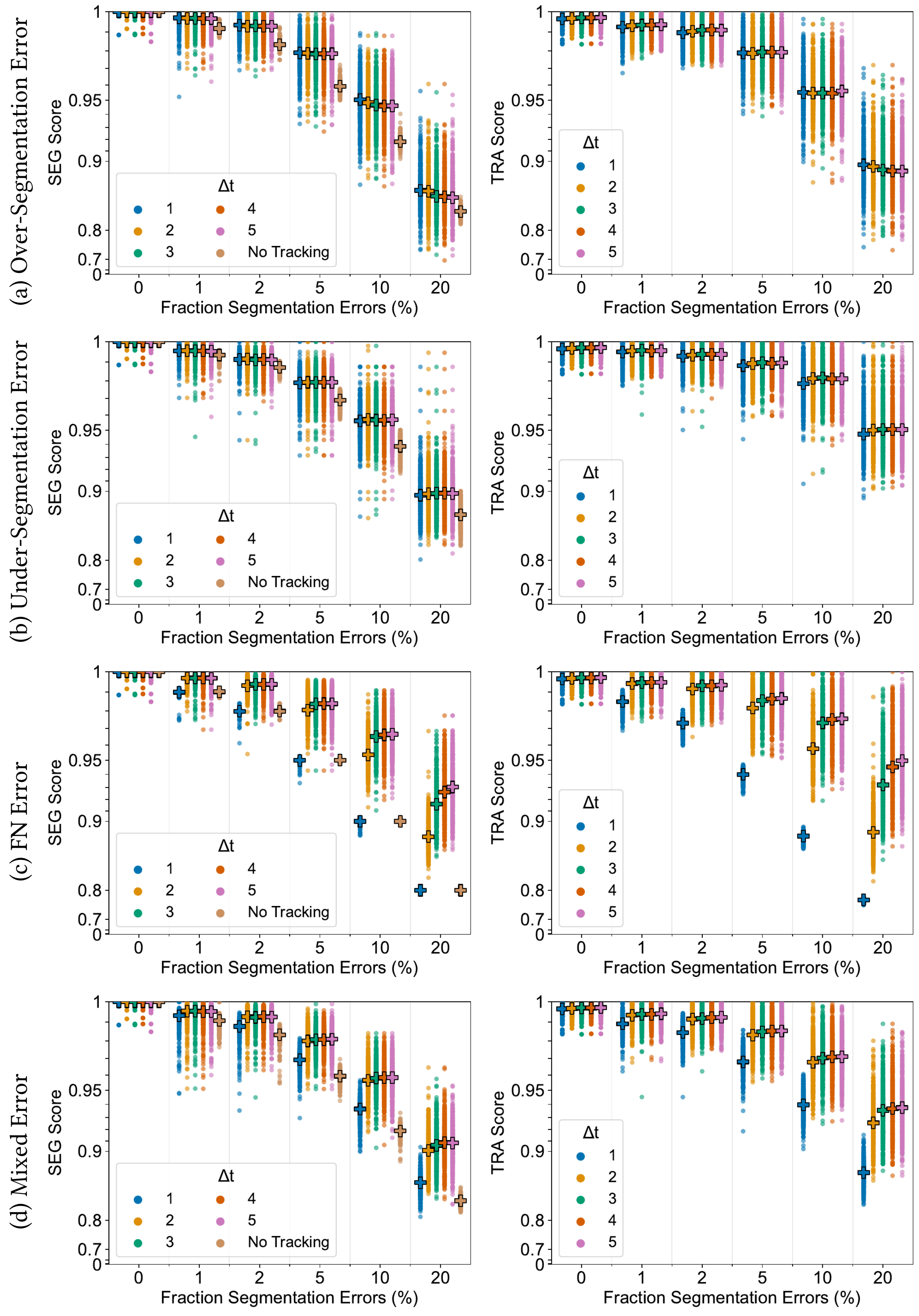


Figure 3.9.: SEG and TRA score of the proposed method with different Δt on degraded segmentation data of the DeepCell dataset. The score of a single dataset is shown as a circle, whereas + is the median calculated over $N = 405$ erroneous image sequences.

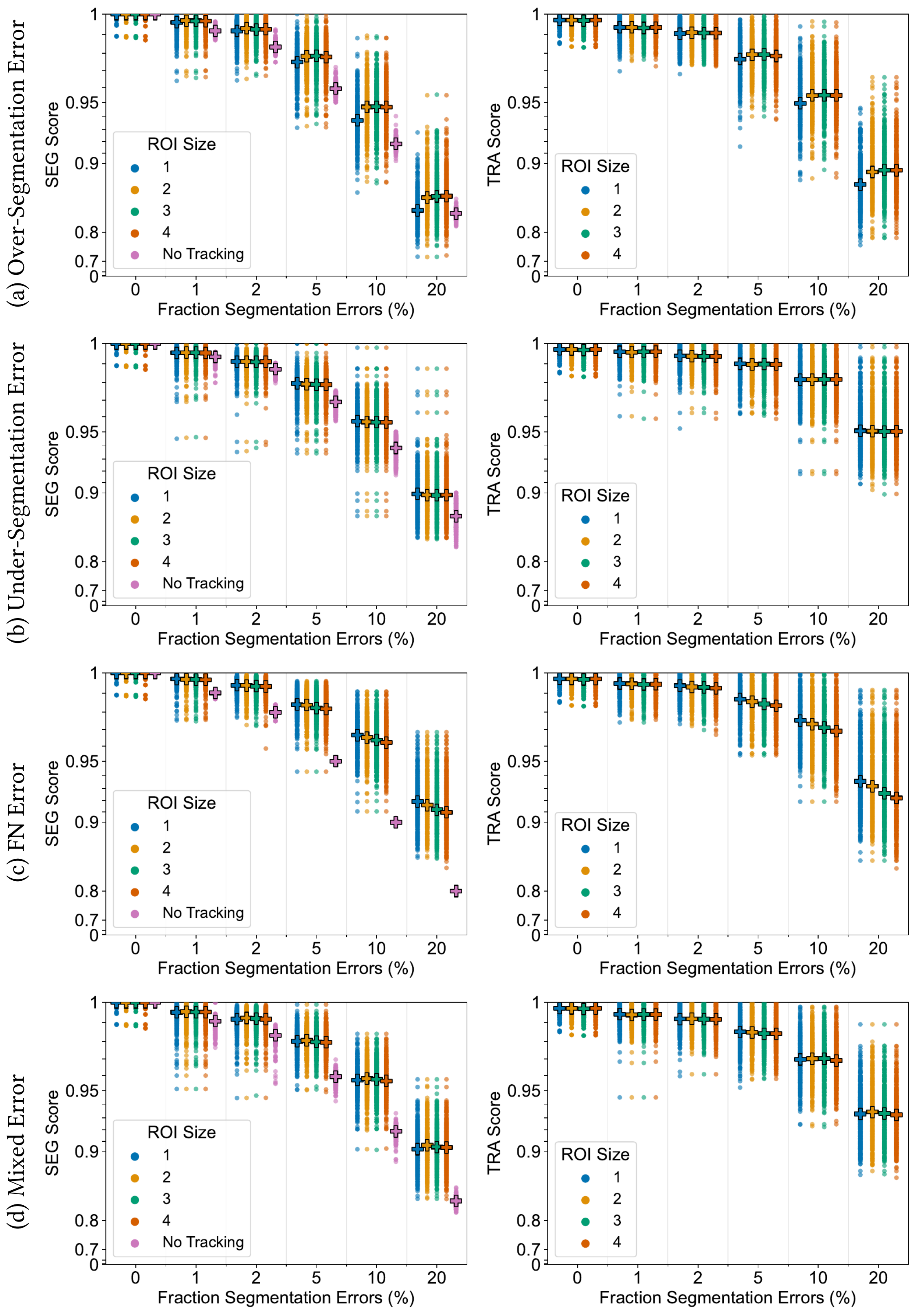


Figure 3.10.: SEG and TRA score of the proposed method with different ROI size, which is a multiple of the size of the median segmentation mask in an image sequence, on degraded segmentation data of the DeepCell dataset. The score of a single image sequence is shown as a circle, whereas + is the median calculated over $N = 405$ erroneous image sequences.

Post-Processing Analysis

The influence of the post-processing is analyzed by modifying the post-processing step while keeping all other steps the same. To investigate the influence of the FN correction and the untangling step, they are replaced as follows: Although the matching step provides the information that a track is fragmented, keeping this information by setting the previous track segment as the predecessor of the subsequent track was shown to lead to a worse score on the TRA metric [147] than setting no predecessor at all. Therefore, the FN correction step is replaced by creating short tracks without a predecessor for each track with missing masks, as the TRA measure yields for tracks with missing masks otherwise an error during TRA score computation. The untangling step is replaced by removing predecessor information of tracks with more than one predecessor and removing successor information of tracks with more than two successors. The untangling step is referred to as untangle and the FN correction step as masks, whereas a missing post-processing step is indicated by an over-line $\overline{(\ldots)}$.

Applying or removing the post-processing steps has on the SEG and TRA metric a similar influence, as shown in Figure 3.11. For under- and over-segmentation errors, tracking with the untangling step performs better than without untangling step, $\overline{\text{untangle}}$, as shown in Figure 3.11 a and Figure 3.11 b. On image sequences with FN errors, tracking with the FN correction step performs better than without the FN correction step, $\overline{\text{masks}}$, which is shown in Figure 3.11 c. On mixed segmentation errors, applying both post-processing steps performs best, which is shown in Figure 3.11 d.

In general, the segmentation measure SEG, Figure 3.11 left column, improves on image sequences including a specific segmentation error, when the corresponding error correction step of the post-processing is applied. However, while the median result improves when applying the corresponding post-processing step to correct a specific segmentation error type, the spread in the final metric score is larger, especially with an increasing fraction of segmentation errors. A potential explanation for this observation is that with an increasing number of tracking errors it becomes more difficult to identify segmentation errors correctly. For instance, a track that has more over-segmented masks than correctly segmented masks is going to be split into several tracks. Therefore, the remaining, correct masks are split, resulting in more segmentation errors.

Tracking Performance Comparison

Next, the performance of the proposed tracking approach including both post-processing steps is compared with the tracking approaches of KTH-SE, KIT-Sch-GE (1), and MU-Lux-CZ on erroneous segmentation data. The results are shown in Figure 3.12. On the TRA metric, right column in Figure 3.12, the proposed method outperforms the other tracking methods on image sequences with over-segmentation errors and mixed errors – Figure 3.12 a and Figure 3.12 d. On image sequences with under-segmentation errors and FNs – Figure 3.12 b and Figure 3.12 c – for small fractions of segmentation errors the proposed method and KIT-Sch-GE (1) perform roughly similar, whereas, on image sequences with high fractions of segmentation errors, the proposed method outperforms all other methods. However, on image sequences with perfect segmentation, the fraction

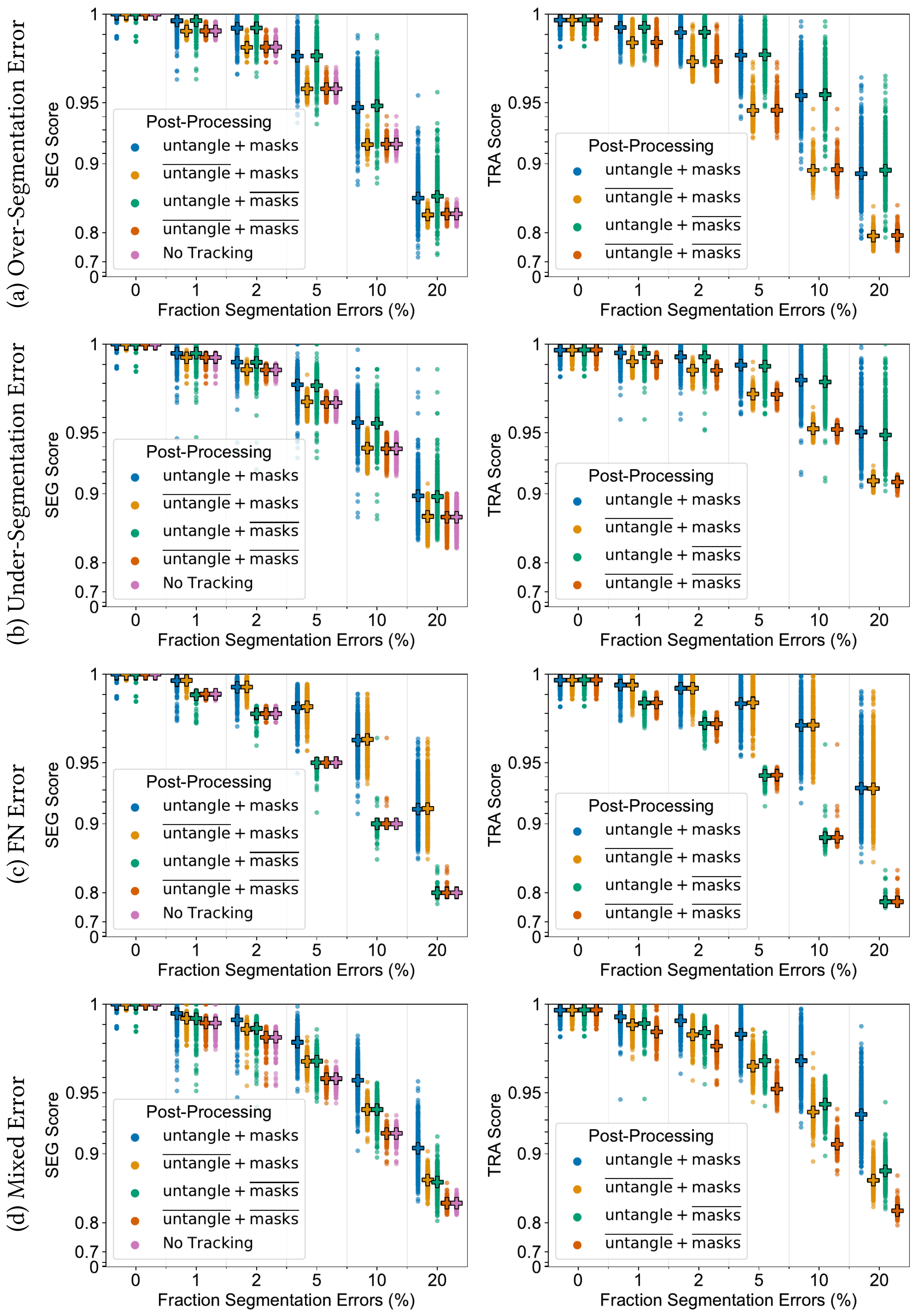


Figure 3.11.: SEG and TRA score of the proposed method with different post-processing – with untangling (“untangle"), FN correction (“masks"), $\overline{(\dots)}$ indicates a missing post-processing step – on degraded segmentation data of the DeepCell dataset. The score of a single image sequence is shown as a circle, whereas + is the median calculated over $N = 405$ erroneous image sequences.

of segmentation errors is 0, and the simple overlap-based tracking of MU-Lux-CZ performs slightly better than the proposed tracking and the KIT-Sch-GE (1) approach on the TRA metric. An analysis of the TRA measure shows that the proposed method and KIT-Sch-GE (1) have more "edge add" errors, which refer to missing links between segmentation masks that belong to the same track. For both methods, this error type occurs at the image border when only a small fraction of the object is in the field of view. In such cases, a disappearing object at t and an appearing object at $t+1$ results in smaller costs than linking the two segmentation masks. A remedy for this behavior would be an adaption of the appearance and disappearance costs in both approaches or adding a field of interest for the image sequence to filter out segmentation masks that extend only a few pixels into the image plane as done in other benchmark datasets [7].

On the SEG metric, left column in Figure 3.12, the proposed tracking approach outperforms the other approaches on mixed errors while performing roughly similar as KIT-Sch-GE (1) on image sequences with FNs errors. On over-segmentation errors, the proposed method outperforms all other methods concerning the median, however, shows a large spread in the SEG score between the synthetically degraded image sequences. Interestingly, on image sequences with a fraction of 20% over-segmentation errors, the simplistic, overlap-based tracker MU-Lux-CZ performs only slightly worse than the proposed tracker. On image sequences with under-segmentation errors – Figure 3.12 c – the proposed method and KIT-Sch-GE (1) perform roughly similar.

While the median of the proposed method is most of the time higher or at least equal to other tracking algorithms, the approach shows a large spread in the final metric scores over all types of segmentation errors. Since the proposed method has more error correction capabilities than KIT-Sch-GE (1), which only corrects FNs, and MU-Lux-CZ, which has no segmentation error correction at all, the proposed method has a wider range of potential outcomes.

As the "No Tracking" segmentation data indicate a spread in the scores of the initial segmentation data itself, further analysis is needed to distinguish the cases where the results overlap in the metric score. Therefore, the results of the different tracking approaches were ranked on each erroneous image sequence separately. Then, the average rank and the fraction of times the tracking approach reached the 1st rank calculated over the subset of image sequences containing a specific type and fraction of segmentation errors were computed. As several approaches can reach the same score on an image sequence, the fraction of 1st rank does not sum up to 100. The TRA results are shown in Tables 3.1, whereas DET and SEG score are provided in Tables A.2, and A.3. For perfect segmentation data, MU-Lux-CZ reaches on all image sequences rank 1, whereas the proposed method performs on par with MU-Lux-CZ in roughly 70% of all cases. On the erroneous image sequences, the simple overlap-based tracking of MU-Lux-CZ, however, is outrun by the proposed tracking approach. The proposed tracking method scores best on all tracking metrics, error fractions, and error types apart from the subset of FNs with fraction 1%, where KIT-Sch-GE (1) performs best.

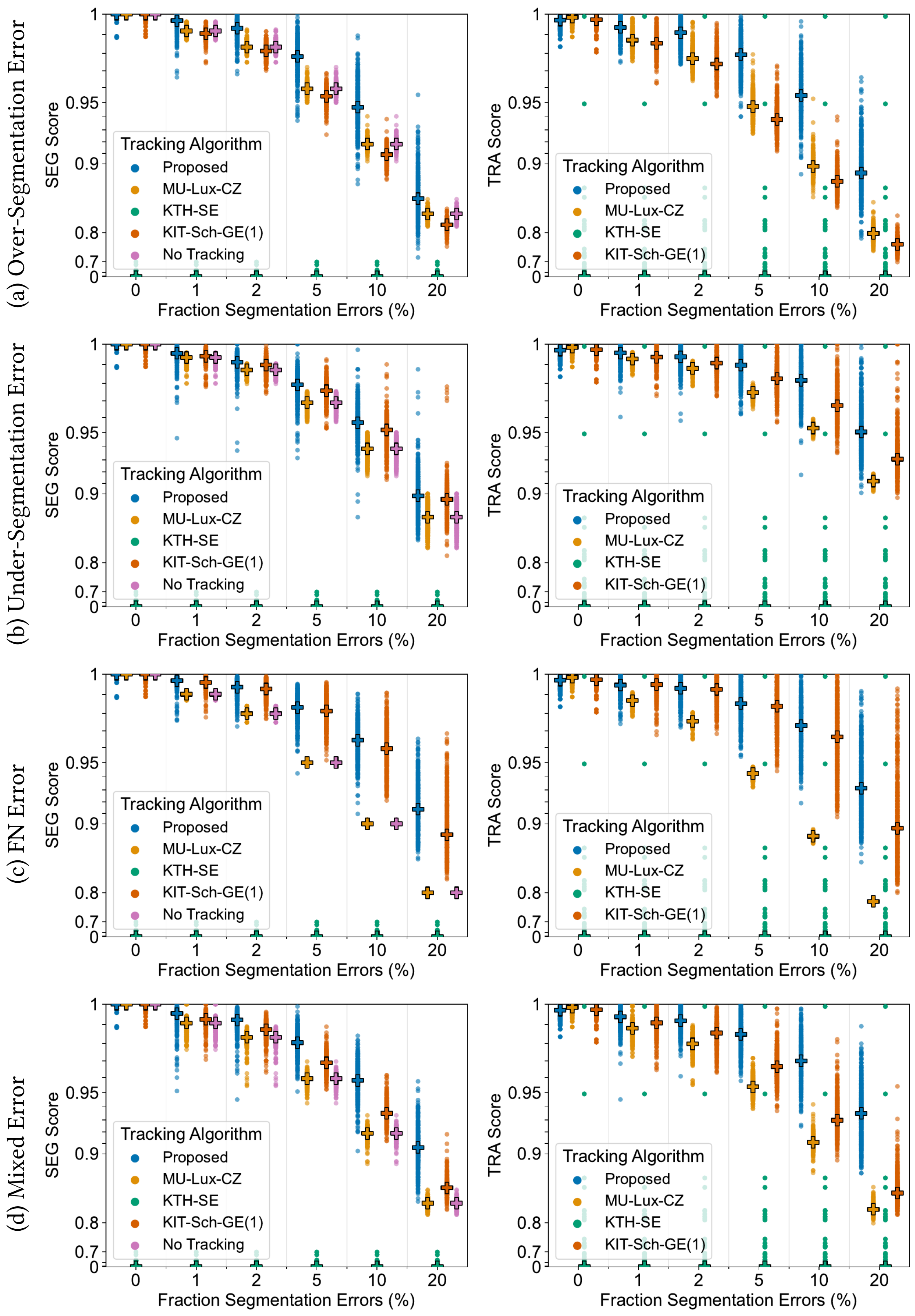


Figure 3.12.: SEG and TRA score of the proposed method and other tracking approaches evaluated on degraded segmentation data from the DeepCell dataset. The score of a single image sequence is shown as a circle, whereas + is the median calculated over $N = 405$ erroneous image sequences.

	Tracking Algorithm	**Mean Rank**				**1st Rank/%**			
		Proposed	MU-Lux-CZ	KTH-SE	KIT-Sch-GE (1)	Proposed	MU-Lux-CZ	KTH-SE	KIT-Sch-GE (1)
Segmentation Error	**Error Fraction/%**								
Ground Truth	0	2.22	**1.32**	4.00	1.98	25.9	**72.8**	0.0	33.3
Under-Segmentation	1	**1.58**	2.35	3.98	2.02	**58.0**	13.6	0.0	33.6
	2	**1.41**	2.60	3.98	1.97	**66.9**	6.2	0.0	30.6
	5	**1.22**	2.82	3.98	1.96	**81.2**	1.0	0.0	19.3
	10	**1.15**	2.93	3.97	1.95	**86.2**	0.0	0.2	14.3
	20	**1.15**	2.96	3.94	1.94	**85.7**	0.0	1.0	13.3
Over-Segmentation	1	**1.12**	2.15	3.97	2.70	**91.1**	6.7	0.2	3.2
	2	**1.05**	2.17	3.97	2.79	**96.3**	2.2	0.7	1.2
	5	**1.01**	2.06	3.95	2.97	**98.8**	0.0	1.2	0.0
	10	**1.01**	2.05	3.94	3.00	**98.5**	0.0	1.5	0.0
	20	**1.03**	2.11	3.73	3.13	**97.3**	0.0	2.7	0.0
False Negative	1	1.51	2.91	3.99	**1.46**	52.8	1.2	0.0	**59.3**
	2	**1.41**	3.00	3.98	1.47	**59.0**	0.0	0.2	53.6
	5	**1.35**	3.02	3.98	1.55	**64.9**	0.0	0.0	44.9
	10	**1.29**	3.02	3.97	1.65	**71.1**	0.0	0.5	34.8
	20	**1.15**	3.12	3.84	1.86	**84.9**	0.0	1.2	15.1
Mixed	1	**1.31**	2.62	4.00	1.96	**74.3**	8.9	0.0	24.0
	2	**1.14**	2.85	3.96	2.03	**87.9**	0.7	1.2	11.1
	5	**1.04**	2.97	3.97	2.02	**95.8**	0.2	0.7	3.2
	10	**1.02**	3.00	3.94	2.04	**97.5**	0.0	1.2	1.2
	20	**1.02**	3.04	3.88	2.06	**98.3**	0.0	1.5	0.2

Table 3.1.: TRA scores of different tracking approaches on synthetically degraded data from the DeepCell dataset. For each erroneous image sequence, all approaches are ranked based on their TRA metric, where approaches reaching the same score are assigned the same rank. The mean rank is calculated over the subset of image sequences containing a specific type and fraction of segmentation errors. In addition, the fraction of times the tracking approach reached rank 1 is reported, where the fractions accumulated over the tracking approaches can be larger than 100% as several approaches can reach the same TRA score.

3.4. Discussion

This chapter proposed a graph-based tracking approach with two manually tunable parameters and post-processing to correct segmentation errors automatically. The tracking approach can process 2D and 3D image sequences and can be combined with an arbitrary instance segmentation approach which yields instance segmentation masks to create a tracking-by-detection approach. The two parameters, Δt and the ROI size are easy to interpret for non-expert users. The graph-based tracking models object behavior and the segmentation errors over-segmentation, under-segmentation, and FNs. The information of many-to-one and one-to-many links of the derived tracking graph is then used in a post-processing step to detect and correct the segmentation errors: FNs, under-, and over-segmentation. To explore the potential of the approach, the influence of the parametrization and the post-processing steps was analyzed on degraded segmentation data that model different segmentation errors, which were proposed in Chapter 2. Moreover, the full tracking method was compared against other tracking methods using the same degraded segmentation data.

Choosing the two manually tunable parameters, Δt and the ROI size is a tradeoff between tracking performance and computation time. Increasing the ROI size allows to better link objects with larger movements, however, it also increases the number of potential matching candidates since more objects overlap with a larger ROI. With an increasing number of potential matching candidates, the number of edges in the matching graph increases and hence the size of the optimization problem, which can require more computation time to solve. Similarly, increasing Δt allows to model and correct tracks with missing segmentation masks over several successive time points, however, the size of the matching graph increases, resulting again in a larger optimization problem to solve. Hence, as a tradeoff, Δt was set to three whereas the ROI size was set to twice the size of the median segmentation mask in an image sequence.

The proposed post-processing steps can correct the segmentation errors FNs, over-, and under-segmentation as well as mixed segmentation errors. However, with decreasing segmentation quality the tracking quality decreases as well. While on the examined image sequences the SEG score improves through the proposed post-processing steps, the results can be only partly generalized to real-world erroneous segmentation data as the post-processing – splitting, merging, and interpolating missing segmentation masks – does not change the overall shape of the segmentation masks. Hence, penalties resulting from not well-matching shapes between ground truth masks and segmentation masks, e.g. the predicted segmentation masks are too large or too small, cannot be corrected through the proposed post-processing. Compared to other tracking algorithms the proposed tracking method performed well, especially on image sequences with mixed segmentation errors.

While the proposed method provides a step towards an easy-to-apply tracking approach for non-expert users, there are several limitations of the method that need to be taken into consideration. While applicable to 3D image sequences, the scalability of the approach to large-scale image sequences with thousands or hundreds of thousands of objects were not considered. Using the phase correlation for position estimation in 3D is slow. Hence replacing the position estimation with methods that for instance estimate the displacement of pixels on the whole image at once might be considered. Moreover, the coupling of the mitosis edges requires solving via mixed integer linear programming which is hard to solve. While there exist general purpose solvers with strong heuristics to speed up solving, remodeling this part should be considered. Also, these general-purpose solvers are usually not freely available and might require an extra license. Using specialized solvers which lead to an approximate solution fast such as the one of Haller *et al.* [119] should also be considered and investigated if the resulting approximate solutions differ strongly from the optimal solution. The proposed position-based cost functions should be adapted for the disappearance and appearance of objects which can not only happen due to leaving the field of view but also due to marker bleach or occlusion. Moreover, some cases can not be resolved by position-based features. For instance, if two objects at time point t have swapped their positions at $t + 1$.

While the post-processing can correct certain types of segmentation errors, more complex segmentation errors, such as an object being segmented as two at time point t and at $t+1$ as three parts, are not resolvable. Therefore, extending the post-processing formulation should be considered to be further extended.

A comparison of the proposed method concerning additional criteria such as runtime and the performance on a diverse set of 2D and 3D datasets is conducted in Chapter 5.

4. EmbedTrack – Simultaneous Object Segmentation and Tracking Through Learning Offsets and Clustering Bandwidths

The recent success of deep learning based instance segmentation in light microscopy images [54, 55, 56, 57], as well as the successful application of deep learning for MOT [171, 179, 180], give reason to apply deep learning for MOT in light microscopy images.

While there have been proposed several deep learning approaches for MOT in light microscopy images, they are accompanied by several shortcomings: (i) Some methods propose a cascade of neural networks [90, 122, 123, 124, 126, 127, 128, 129] which can be complicated to tune and need to be trained separately. In contrast, combining segmentation or detection with tracking in a single neural network [88, 89, 121, 125] can be trained at once. (ii) Approaches using recurrent neural network elements [89, 90, 121], can need careful re-initialization or updates in contrast to neural network architectures without recurrent elements [88, 122]. (iii) Some methods predict for all possible combinations of pairs of detected objects in t and $t + 1$ a similarity score [90, 124, 125, 126, 127] which can be computationally inefficient as each pair of objects is forwarded to the neural network. In contrast, estimating low-dimensional embeddings simultaneously for all objects in an image and calculating their pairwise distances afterwards [88, 89, 90, 92, 121, 122, 123] can be computationally more efficient. (iv) Often, the learned embeddings are not human comprehensible [89, 121, 125] making further post-processing or interpretation difficult. In contrast, learning embeddings that represent offsets in position are easy to interpret for humans [88, 92, 122, 123] allowing simple post-processing in successive steps.

To alleviate the aforementioned shortcomings, this chapter proposes a single neural network for instance segmentation and tracking (i), without any recurrent network elements (ii), which predicts human comprehensible embeddings (iii) jointly for all objects in a successive pair of frames (iv). The proposed method is based on the instance segmentation approach of Neven *et al.* [181], in which a branched ERFNet [60] is trained to learn the offset of pixels to its object center as well as a clustering bandwidth to cluster pixels into instance masks. This chapter proposes to extend the instance segmentation approach to

Section 4.1 and Section 4.2 are an adaption of the method description from K. Löffler and R. Mikut. "EmbedTrack — Simultaneous cell segmentation and tracking through learning offsets and clustering bandwidths". In: IEEE Access 10 (2022), pp. 77147–77157. doi: 10.1109/ACCESS.2022.3192880.

simultaneous instance segmentation and tracking by including an estimation of offsets for object pixels to their object center between successive frames.

The remainder of this chapter is organized as follows. Section 4.1 introduces the instance segmentation approach by Neven *et al.*. Next, Section 4.2 describes how the instance segmentation approach is extended to MOT. The proposed method is evaluated concerning the influence of training dataset size, augmentation strategies, and temporal resolution in Section 4.3. Finally, Section 4.4 concludes this chapter by providing a summary and discussion of the proposed method.

4.1. Instance Segmentation of Neven *et al.*

Instance segmentation allows the distinction between different instances of the same semantic class in an image. Therefore, foreground pixels need to be assigned to different instances, for instance by finding clusters of foreground pixels and assigning pixels belonging to the same cluster to the same instance mask. To cluster pixels, a clustering bandwidth, which is often defined manually, is needed. The clustering bandwidth determines in combination with a distance measure how far pixels can be apart from a cluster center to be still assigned to this cluster center. Neven *et al.* proposed the idea of training a CNN to predict offsets to shift pixels along their x- and y-dimension to their object centers, so they form compact clusters, as well as learning the clustering bandwidths s required for each object, instead of selecting them manually [181]. As distance measure, [181] use the Gaussian kernel

$$d(\mathbf{c}, \mathbf{p}) = \exp\left(-\frac{(c^x - p^x)^2}{s^x} - \frac{(c^y - p^y)^2}{s^y}\right), \tag{4.1}$$

where $\mathbf{c}$ and $\mathbf{p}$ are 2D position vectors, c^x and p^x the x-dimension of the vectors, c^y and p^y the y-dimension of the vectors, and s^x and s^y are the bandwidth in x- and y-dimension. Using such a distance measure provides the possibility to allow for more slack in estimating the object center of large objects and avoid over-segmentation by increasing the bandwidth, whereas a more precise estimation of the object center of small objects can be enforced by reducing the bandwidth. Figure 4.1 shows distance measures based on the Gaussian kernel with different bandwidths s.

A too large bandwidth leads to merging pixels belonging to different objects into one instance mask, which can result in merging of objects – under-segmentation error – whereas a too small bandwidth leads to not all shifted pixels being clustered to the same instance, which can result in splitting an object – over-segmentation error.

During inference the true object center is unknown. However, for each pixel that belongs to an object a clustering bandwidth and an offset to the object's center are estimated. Hence, shifting the pixels by their predicted offset results in an estimate of the object's center, whereas the clustering bandwidth defines how far other shifted pixels can be apart to be still assigned to the same instance. Therefore, any of the shifted pixels can be chosen to cluster the other pixels belonging to the same object into an instance segmentation mask.

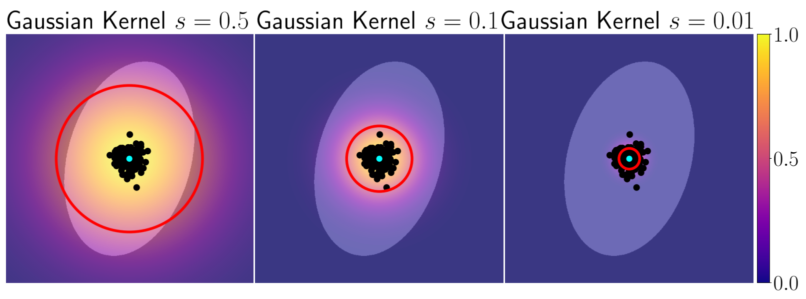


Figure 4.1.: Visualization of the clustering using different clustering bandwidths. The plots show the distance $d(\mathbf{c}, \mathbf{p})$ using Gaussian kernels with different bandwidths $s = s^x = s^y$ in the range $[0, 1]$ (heat maps from blue to yellow) of each pixel position in the plot to the object center $\mathbf{c}$ (cyan) of the ellipsoid object. The red circles are contours where the distance score between the center $\mathbf{c}$ and any pixel $\mathbf{p}$ is $d(\mathbf{c}, \mathbf{p}) = 0.5$. The points (black) visualize pixels shifted by their predicted offset to the center of the ellipsoid object. To cluster the pixels into instances, points (black) are assigned to the same instance if they lay within the red circle. Figure 1 by Löffler *et al.* [182] licensed under CC BY 4.0.

4.2. Method

In the following, it is shown how the instance segmentation approach of Neven *et al.* can be extended to object segmentation and tracking by adding an additional decoder path to the CNN for tracking and introducing an additional tracking step. An overview of the proposed method is shown in Figure 4.2. First, pairs of image crops of the successive time points t and $t - 1$ are forwarded to a CNN, which predicts for each pixel segmentation offsets and clustering bandwidths for segmentation, and tracking offsets of pixels belonging to an object at t to their object center at $t - 1$ for tracking. Next, the segmentation offsets and clustering bandwidth predictions are processed with a foreground-background prediction in a clustering step, to assign pixels predicted as an object (foreground) to instances. Finally, based on the predicted tracking offsets between the two time points and the instance masks retrieved from the clustering step, the instances are linked backwards in time.

In the following, the network architecture, the loss function, the clustering step, and the tracking step are explained in more detail.

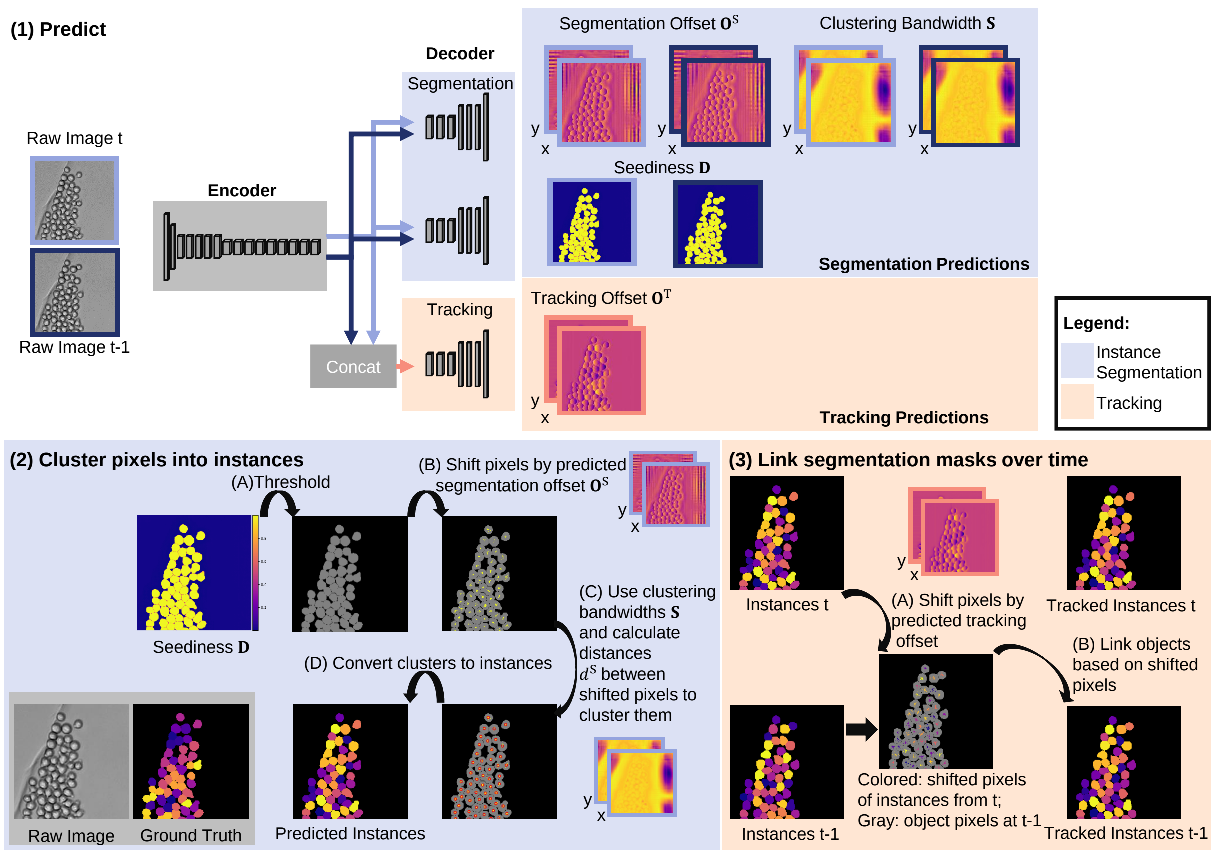

(1) Predict
Raw Image t
Raw Image t-1
Encoder
Decoder
Segmentation
Tracking
Concat
Segmentation Offset $\mathbf{O}^S$
Clustering Bandwidth $\mathbf{S}$
Seediness $\mathbf{D}$
Segmentation Predictions
Tracking Offset $\mathbf{O}^T$
Tracking Predictions
Legend:
Instance Segmentation
Tracking
(2) Cluster pixels into instances
(A)Threshold
(B) Shift pixels by predicted segmentation offset $\mathbf{O}^S$
Seediness $\mathbf{D}$
(C) Use clustering bandwidths $\mathbf{S}$ and calculate distances d^S between shifted pixels to cluster them
(D) Convert clusters to instances
Raw Image
Ground Truth
Predicted Instances
(3) Link segmentation masks over time
Instances t
(A) Shift pixels by predicted tracking offset
Tracked Instances t
(B) Link objects based on shifted pixels
Instances t-1
Colored: shifted pixels of instances from t; Gray: object pixels at t-1
Tracked Instances t-1

Figure 4.2.: Overview of the EmbedTrack method. (1) Prediction of offsets, clustering bandwidths, and seediness maps for pairs of raw images using a branched ERFNet. (2) Processing the predicted segmentation offsets and clustering bandwidths to retrieve an instance segmentation, where gray and black show the prediction of the network for objects and background and the yellow pixels show the object pixels after adding the predicted offset. The red circles indicate $d^S = 0.5$ using the predicted clustering bandwidths. (3) Linking the instance segmentation masks backwards in time using the predicted tracking offsets. Raw image crops BF-C2DL-HSC from the CTC [7, 66]. Derivative of Figure 2 by Löffler *et al.* [182] licensed under CC BY 4.0; rearranged plot and changed colors.

4.2.1. EmbedTrack Model

A branched ERFNet [60] with one shared encoder and three decoder paths is used – two decoders for segmentation and one decoder for tracking. A visualization of the model and the predicted outputs is shown in Figure 4.2. An image pair is fed into the shared encoder, where the images are processed individually to omit entangling their feature representations. For tracking, the resulting feature representations are concatenated and forwarded to the tracking branch, whereas for segmentation the feature representations are forwarded to the segmentation branches separately. For each time point, t and $t-1$, the network predicts a set of segmentation predictions, whereas for tracking one tracking offset prediction from t to $t-1$ is predicted to link the objects backward in time. Linking objects backward in time results in shifting pixels belonging to distinct objects at t to one object center at $t-1$ to link predecessor-successor objects. In contrast, linking objects forward in time would require to shift the pixels of a single object at t to several objects at $t+1$, which would either require the prediction of several offset vectors for each pixel, one offset prediction for each successor object, or to distribute the pixels of the single object at t to several objects at $t+1$.

For raw images of shape $[H, W]$, where H is the height and W the width, the first segmentation decoder predicts segmentation offsets $\mathbf{O}^{\mathrm{S}}$ of shape $[H, W, 2]$ between pixels belonging to an object and their corresponding object center, and clustering bandwidths $\mathbf{S}$ of shape $[H, W, 2]$. The second segmentation decoder predicts the seediness $\mathbf{D}$ of shape $[H, W]$, which is a score between 0 and 1 that serves as a foreground-background estimation as well as indicates if an object pixel estimates its object center correctly. The tracking decoder learns to predict tracking offsets $\mathbf{O}^{\mathrm{T}}$ of shape $[H, W, 2]$ between pixels belonging to an object at t and their corresponding object center at $t-1$. A tanh activation is applied on the segmentation offset and tracking offset prediction and a sigmoid activation on the clustering bandwidth prediction and the seediness prediction. After applying the activation functions, the segmentation and tracking offsets are in the range $[-1, 1]$, whereas the clustering bandwidth prediction and the seediness prediction are in the range $[0, 1]$. In summary, the network predicts one set of segmentation predictions $\{\mathbf{S}, \mathbf{O}^{\mathrm{S}}, \mathbf{D}\}$ for each image t and $t-1$, and one tensor of tracking offsets $\mathbf{O}^{\mathrm{T}}$ to link pixels belonging to an object at t to its corresponding object center at $t-1$.

4.2.2. Loss

The aim is to train the segmentation decoders of the CNN to predict a foreground-background prediction, segmentation offsets, and clustering bandwidths that are required for the subsequent clustering step, whereas the tracking decoder is trained to predict tracking offsets to link instance masks from t to $t-1$. In the loss, the centers of the instances and the ground truth instance masks are used. It is emphasized that the offsets and clustering bandwidths have no direct supervision during training.

Let $\mathcal{I} = \{(1, 1), (1, 2), \ldots, (H, W)\}$ be the set of all index tuples to access each element in a matrix and let accessing the element at index tuple i in a matrix $\mathbf{M}$ be denoted as $(\mathbf{M})_i$. In the context of 2D images, each index tuple references a pixel in the image. Moreover, let $\mathcal{I}_m$ be the set of index tuples referencing the pixels of an instance segmentation mask

m, where $\mathcal{I}_m$ is a subset of the set of index tuples of the image $\mathcal{I}$. Accessing elements in a tensor of shape $[H, W, Z]$ with such an index tuple, yields a vector of Z elements along the last dimension. For instance, accessing the tracking offset tensor at index tuple i yields a 2D vector with offset predictions in x- and y-direction, whereas accessing the seediness map at index tuple i results in a single seediness score.

Pixels belonging to the same instance should predict similar clustering bandwidths since in the clustering step any shifted pixel could be selected as the cluster center and its clustering bandwidth defines the range in which other pixels are assigned to the same cluster and therefore the same instance. To enforce similar clustering bandwidths, [181] use a loss component based on the variance between the clustering bandwidth vectors belonging to the same instance. It is emphasized that no traditional clustering algorithm is required to assign the pixels to instances afterwards, instead pixels are clustered iteratively which is described in Section 4.2.3 is used.

For each instance m the mean clustering bandwidth vector $\bar{\mathbf{s}}_m$, in x- and y-dimension, is calculated over the set of index tuples $\mathcal{I}_m$ referring the pixels of the instance mask m, where $\mathbf{s}_k$ is the clustering bandwidth vector at index tuple k of the clustering bandwidth prediction S

$$\bar{\mathbf{s}}_m = \frac{1}{|\mathcal{I}_m|} \sum_{k \in \mathcal{I}_m} \mathbf{s}_k \,. \tag{4.2}$$

The loss part enforcing similar clustering bandwidth predictions for pixels belonging to the same masks is then given as

$$\ell_{\text{var}} = \frac{1}{M_{\text{inst}}} \sum_{m=1}^{M_{\text{inst}}} \frac{1}{|\mathcal{I}_m|} \sum_{k \in \mathcal{I}_m} \left(\bar{\mathbf{s}}_m - \mathbf{s}_k\right)^2 , \tag{4.3}$$

where M_{inst} is the number of instance masks and $|\mathcal{I}_m|$ is the number of pixels of instance m.

To get the predictions of the object centers for segmentation and tracking, the predicted offsets $\mathbf{o}_i^{\text{S}}$ for segmentation and $\mathbf{o}_i^{\text{T}}$ for tracking are added to the position of each pixel

$$\begin{aligned} \mathbf{e}_i^{\text{S}} &= \mathbf{p}_i + \mathbf{o}_i^{\text{S}} , \\ \mathbf{e}_i^{\text{T}} &= \mathbf{p}_i + \mathbf{o}_i^{\text{T}} , \end{aligned} \tag{4.4}$$

where i refers to an index tuple of $\mathcal{I}$, $\mathbf{e}_i^{\text{S}}$ is the predicted object center of pixel i at the same time point, $\mathbf{e}_i^{\text{T}}$ is the predicted object center of pixel i from time point t for its object center at $t-1$, and $\mathbf{p}_i$ is the normalized pixel position – pixel coordinates from range $([0, H], [0, W])$ normalized to range $([0, 1], [0, 1])$. By using the already introduced distance measure from Equation 4.1, for each index tuple i a prediction how close its shifted position $\mathbf{e}_i^{\text{S}}$ is to the object center can be derived

$$\begin{aligned} \tilde{\mathbf{s}}_m &= \exp(w_{\text{s}} \cdot \bar{\mathbf{s}}_m) , \\ d^{\text{S}}\left(\mathbf{c}_m, \mathbf{e}_i^{\text{S}}\right) &= \exp\left(-\frac{\left(c_m^x - e_i^{\text{S},x}\right)^2}{\tilde{s}_m^x} - \frac{\left(c_m^y - e_i^{\text{S},y}\right)^2}{\tilde{s}_m^y} \right) , \end{aligned} \tag{4.5}$$

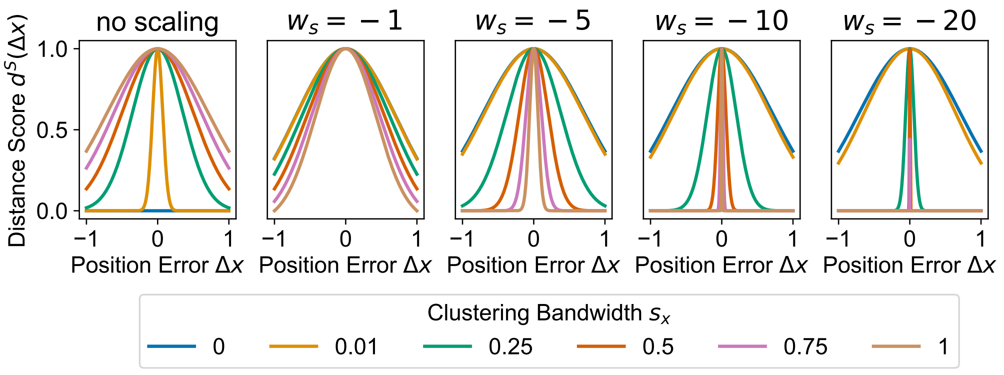


Figure 4.3.: Influence of scaling the clustering bandwidth on the distance score $d^{\mathrm{S}}(\Delta x)$. For visualization, the 1D scenario along the x-dimension is shown where $\Delta x = c^x - e^{\mathrm{S},x}$ is the difference between c^x, the position of the actual medoid in x-dimension, and $e^{\mathrm{S},x}$, the predicted position of the medoid in x-direction. Without scaling the clustering bandwidth s^x, the distance score is given as $d^{\mathrm{s}}(\Delta x) = \exp(\frac{-\Delta x^2}{s^x})$, shown in the left plot. For all other plots, a scaled clustering bandwidth $\tilde{s}^x = \exp(w_s \cdot s^x)$ is used and the distance score is given as $d^{\mathrm{s}}(\Delta x) = \exp(\frac{-\Delta x^2}{\tilde{s}^x})$. Usually, objects are small with respect to the overall size of the image crop. Hence, steeply decreasing distance scores are required so pixels of distinct, close objects are not assigned to the same instance segmentation mask. Without scaling the clustering bandwidth, however, very small values, below 0.01, are required for the clustering bandwidth to yield steep dropping distance scores $d^{\mathrm{S}}(\Delta x)$. By scaling the clustering bandwidth, a wider range of the initial clustering bandwidth can be used which yields steeply dropping distance scores.

where $d^{\mathrm{S}}(\mathbf{c}_m, \mathbf{e}_i^{\mathrm{S}})$ is the distance between the object center $\mathbf{c}_m$ of instance mask m and the predicted object center $\mathbf{e}_i^{\mathrm{S}}$, $\tilde{s}_m^x$ and $\tilde{s}_m^y$ are the x- and y-dimensions of a scaled mean clustering bandwidth vector $\tilde{\mathbf{s}}_m$, and w_s a scaling weight. A scaling of the clustering bandwidth $\mathbf{s}_m$, which is constrained to be in the range $[0, 1]$ in each dimension, is required as the object size is usually small compared to the overall size of the image crop. Hence, without scaling the clustering bandwidth, very small clustering bandwidth values are required to distinguish close small objects based on a quickly decreasing distance score d^{S}. A visualization of this issue is shown in Figure 4.3. The scaling weight w_s is set to -10 as in [181]. As in [62], the medoid is chosen as object center $\mathbf{c}_m$. The medoid is the element in a set with the overall smallest sum of distances to all other elements in the set. Hence, by choosing the medoid as object center, the object center always lays inside of the object.

In [181] the segmentation offsets and clustering bandwidths are learned jointly using an instance loss based on the Lovász hinge loss [183, 184]. The Lovász hinge loss is a convex surrogate for sub-modular losses which allows an efficient minimization of sub-modular loss functions such as the Jaccard loss [184]. The Jaccard loss is given by

$$\ell_{\mathrm{Jaccard}} = 1 - J(\mathcal{I}_{\mathrm{pred}}, \mathcal{I}_{\mathrm{GT}}) , \tag{4.6}$$

where $J(\mathcal{I}_{\text{pred}}, \mathcal{I}_{\text{GT}})$ is the Jaccard index between the set of pixels $\mathcal{I}_{\text{pred}}$ belonging to the predicted instance mask and the set of pixels $\mathcal{I}_{\text{GT}}$ belonging to the ground truth instance mask. By minimizing the Lovász hinge loss, the predicted segmentation offsets and clustering bandwidths can be jointly optimized such that the Jaccard index between predicted instance mask and ground truth instance mask is maximized

$$
\begin{aligned}
(\mathbf{B}_m)_i &= \begin{cases} 1 & \text{if } i \in \mathcal{I}_m \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in \mathcal{I}\,, \\
\left(\mathbf{D}^{\text{S}}_m\right)_i &= d^{\text{S}}(\mathbf{c}_m, \mathbf{e}^{\text{S}}_i) \quad \forall i \in \mathcal{I}\,, \\
\ell_{\text{instance}} &= \sum_{m=1}^{M_{\text{inst}}} \ell_{\text{Lovász}}\left(2 \cdot \mathbf{D}^{\text{S}}_m - \mathbf{1}, 2 \cdot \mathbf{B}_m - \mathbf{1}\right)\,,
\end{aligned} \tag{4.7}
$$

where $\mathbf{B}_m$ is a binary mask of shape $[H, W]$ which is 1 at indices belonging to the instance mask m and 0 otherwise, $\mathbf{D}^{\text{S}}_m$ is a distance matrix of shape $[H, W]$ containing the distance of each shifted pixel to the object center $\mathbf{c}_m$ of instance mask m, $\mathbf{1}$ is a matrix of ones the same size as $\mathbf{D}^{\text{S}}_m$, and $\ell_{\text{Lovász}}$ the Lovász hinge loss.

To cluster pixels into instances, first, the foreground pixels need to be selected, therefore, a foreground-background prediction is needed. The seediness map $\mathbf{D}$, which the second segmentation decoder is learning, predicts for each pixel if it belongs to the background or an object, and if the pixel belongs to an object how close the predictions of the object center $\mathbf{e}^{\text{S}}_i$ is to the actual object center. It therefore serves as a foreground-background estimation and as a prediction of how well pixels estimate their object center. To learn the seediness map, an additional seed loss part is used in [181]. For the seediness map, the CNN learns to predict for each pixel i the distance measure $d^{\text{S}}(\mathbf{c}_m, \mathbf{e}^{\text{S}}_i)$ for pixels belonging to an object and regressing to 0 for background pixels

$$
\begin{aligned}
\ell_{\text{seed}} = w_{\text{fg}} &\sum_{m=1}^{M_{\text{inst}}} \frac{1}{|\mathcal{I}_m|} \sum_{k \in \mathcal{I}_m} (d^{\text{S}}(\mathbf{c}_m, \mathbf{e}^{\text{S}}_k) - (\mathbf{D})_k)^2 \\
&+ \frac{1}{|\mathcal{I}_{\text{bg}}|} \sum_{j \in \mathcal{I}_{\text{bg}}} ((\mathbf{D})_j - 0)^2\,,
\end{aligned} \tag{4.8}
$$

where $d\,(\mathbf{D})_k$ is the predicted seediness score of a pixel belonging to the instance mask m which is referenced to by the index tuple k, $(\mathbf{D})_j$ is the predicted seediness score of a background pixel indexed by the index tuple j in the seediness map, $\mathcal{I}_{\text{bg}}$ the set of index tuples that refer to background pixels, and w_{fg} is a weight for foreground pixels.

The segmentation loss is then given as

$$
\ell_{\text{seg}} = w_{\text{instance}} \ell_{\text{instance}} + w_{\text{var}} \ell_{\text{var}} + w_{\text{seed}} \ell_{\text{seed}}\,, \tag{4.9}
$$

where the weights are set to $w_{\text{instance}} = w_{\text{seed}} = w_{\text{fg}} = 1$, and $w_{\text{var}} = 10$. Since the network provides segmentation predictions for pairs of images, the segmentation loss is calculated for both images of time points t and $t-1$ separately and accumulated.

For tracking, the aim is to shift pixels belonging to an object at t to their object center at $t-1$. Similar to the instance segmentation loss, it is proposed to use the Lovász hinge loss for tracking. Since the network predicts two sets of segmentation predictions, one for time point t and one for time point $t-1$, time indices are added in the following to highlight to which time point the predicted segmentation components belong. $d^{\mathrm{T}}(\mathbf{c}_m^{t-1}, \mathbf{e}_i^{\mathrm{T}})$ is the distance between $\mathbf{c}_m^{t-1}$, the object center of instance m at time point $t-1$, and $\mathbf{e}_i^{\mathrm{T}}$, which is the predicted object center of pixel i belonging to time point t shifted to its object center at $t-1$

$$\begin{aligned} d^{\mathrm{T}}\left(\mathbf{c}_{m,t-1}, \mathbf{e}_i^{\mathrm{T}}\right) &= \exp\left(-\frac{\left(c_{m,t-1}^x - e_i^{\mathrm{T},x}\right)^2}{\tilde{s}_{m,t}^x} - \frac{\left(c_{m,t-1}^y - e_i^{\mathrm{T},y}\right)^2}{\tilde{s}_{m,t}^y}\right), \\ \left(\mathbf{D}_m^{\mathrm{T}}\right)_i &= d^{\mathrm{T}}\left(\mathbf{c}_{m,t-1}, \mathbf{e}_i^{\mathrm{T}}\right) \quad \forall i \in \mathcal{I}\,, \\ \left(\mathbf{B}_{m,t}\right)_i &= \begin{cases} 1 & \text{if } i \in \mathcal{I}_{m,t} \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in \mathcal{I}\,, \\ \ell_{\text{track}} &= \sum_{m=1}^{M_{\text{inst},t}} \ell_{\text{Lovász}}\left(2 \cdot \mathbf{D}_m^{\mathrm{T}} - \mathbf{1}, 2 \cdot \mathbf{B}_{m,t} - \mathbf{1}\right)\,, \end{aligned} \tag{4.10}$$

where $\mathcal{I}_{m,t}$ is the set of index tuples referring to the pixels of instance segmentation mask m at time point t, $\mathbf{D}_m^{\mathrm{T}}$ is a distance matrix of shape $[H, W]$ containing the distance of each shifted pixel to the object center $\mathbf{c}_{m,t-1}$ of instance mask m at time point $t-1$, $\mathbf{1}$ is a matrix of ones the same size as $\mathbf{D}_m^{\mathrm{T}}$, and $\mathbf{B}_{m,t}$ is a binary mask of instance m at time point t, and $M_{\text{inst},t}$ is the number of instances at time point t.

The loss is finally given as the sum of segmentation and tracking loss

$$\ell = w_{\text{seg}}\ell_{\text{seg}} + w_{\text{track}}\ell_{\text{track}}\,, \tag{4.11}$$

where w_{seg} and w_{track} are weights both set to 1.

4.2.3. Pixel Clustering

To convert the pixel-wise predictions from the segmentation decoders to an instance segmentation, a clustering step is applied. The clustering proposed in [181] is modified as follows: First, the clustering bandwidth tensor is smoothed with a 3x3 kernel along x- and y- dimension and is referred to as $\mathrm{S}^{\text{smooth}}$. Since the seediness map $\mathbf{D}$ estimates how well pixels predict their object center, pixels with scores larger than 0.5 in the seediness map are selected and the shifted pixel positions $\mathbf{e}_i^{\mathrm{S}}$ are computed to find potential object centers. Each $\mathbf{e}_i^{\mathrm{S}}$, which is a normalized pixel position in range $([0,1],[0,1])$, is converted to an index tuple in range $([0,H],[0,W])$. The calculated index tuples are accumulated and potential object centers are found by selecting index tuples with more than five clustered pixels in their 3x3 neighborhood. Next, the index tuples j that refer to potential object centers are sorted by their seediness map score $\mathbf{d}_j$. Then, starting with the pixel with the highest seediness map score, the list of likely object centers is processed and pixels that

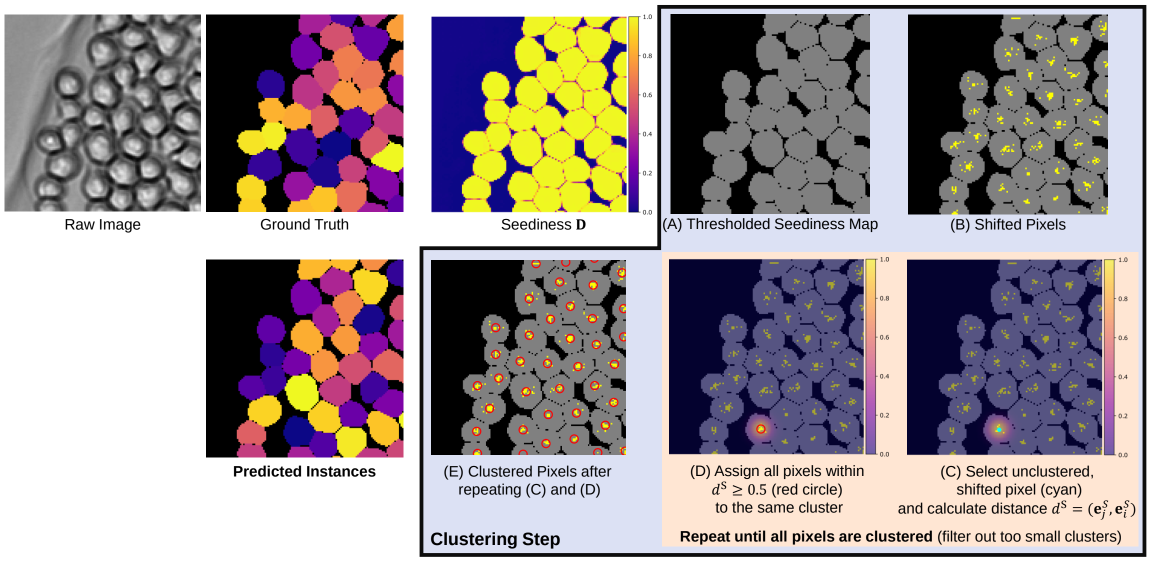

Raw Image
Ground Truth
Seediness **D**
(A) Thresholded Seediness Map
(B) Shifted Pixels
Predicted Instances
(E) Clustered Pixels after repeating (C) and (D)
(D) Assign all pixels within $d^S \geq 0.5$ (red circle) to the same cluster
(C) Select unclustered, shifted pixel (cyan) and calculate distance $d^S = (\mathbf{e}_j^S, \mathbf{e}_i^S)$
Clustering Step
Repeat until all pixels are clustered (filter out too small clusters)

Figure 4.4.: Clustering step. (A) the seediness map **D** is thresholded resulting in a foreground (gray) background (black) prediction. Next, the pixels predicted as foreground are shifted by the predicted segmentation offsets $\mathbf{O}^S$, where the shifted pixels are shown in yellow (B). (C) The shifted pixels are assigned to clusters by selecting a not yet clustered, shifted pixel – referenced by the index tuple j; shown in cyan – and calculating its distance $d^S(\mathbf{e}_j^S, \mathbf{e}_i^S)$ to all other pixels – distance map shown as a heat map. (D) The distance map is thresholded and pixels with a distance score higher than 0.5 (red circle) are assigned to the same cluster. Steps (C) and (D) are repeated until all pixels are clustered (E). Finally, the clustered pixels are converted into instance masks, where each cluster represents an instance. Raw image crop of the dataset BF-C2DL-HSC from the CTC [7, 66]. Derivative of Figure 3 by Löffler *et al.* [182] licensed under CC BY 4.0; rearranged plot and changed colors.

are referenced by their index tuple i are assigned to the object centers $\mathbf{e}_j^{\mathrm{S}}$ using a similar distance measure as in the loss

$$\tilde{\mathbf{s}}_j^{\mathrm{smooth}} = \exp\left(w_{\mathrm{s}} \cdot \mathbf{S}^{\mathrm{smooth}}[j]\right) ,$$
$$d^{\mathrm{S}}\left(\mathbf{e}_j^{\mathrm{S}}, \mathbf{e}_i^{\mathrm{S}}\right) = \exp\left(-\frac{\left(e_j^{\mathrm{S},x} - e_i^{\mathrm{S},x}\right)^2}{\tilde{s}_j^{\mathrm{smooth},x}} - \frac{\left(e_j^{\mathrm{S},y} - e_i^{\mathrm{S},y}\right)^2}{\tilde{s}_j^{\mathrm{smooth},y}}\right) , \tag{4.12}$$
$$\left(\hat{\mathbf{B}}_j\right)_i = \begin{cases} 1 & \text{if } d^{\mathrm{S}}(\mathbf{e}_j^{\mathrm{S}}, \mathbf{e}_i^{\mathrm{S}}) > 0.5 \quad \forall i \in \mathcal{I} , \\ 0 & \text{otherwise} , \end{cases} ,$$

where $\tilde{\mathbf{s}}_j^{\mathrm{smooth}}$ is as in the loss a scaled clustering bandwidth vector at pixel j with the same weight w_{s} as used in the loss, $\tilde{s}_j^{\mathrm{smooth},x}$ and $\tilde{s}_j^{\mathrm{smooth},y}$ the x- and y-dimension of the clustering bandwidth vector, and $\hat{\mathbf{B}}_j$ the resulting instance mask. Index tuples referring to potential object centers that are assigned to an instance mask are removed from the list of likely object centers. To filter out false positives, the minimum size of an instance mask is set to half the size of the 1% percentile of all mask sizes in the training dataset. While for the benchmarks used in Section 4.3 and Section 5.3.2 the chosen threshold to filter out FPs proved to be sufficient, depending on the dataset a different threshold might be required, for instance, when the training dataset has many small annotated artifacts. An elaborate analysis of the influence of the annotation quality in the training data on the final model performance is however outside of the scope of this thesis.

On object boundaries, pixels can have high distance scores for several object centers. To assign the pixel to the best object center, for each pixel its highest distance score d^{S} is kept to allow to reassign a pixel to a subsequent object center. Therefore, three conditions need to be fulfilled: the pixel receives a higher score d^{S} if it is assigned to the new object center, the number of pixels clustered into the new instance have at least the minimum size, and the fraction of pixels that are already assigned to another mask is less than half of the final instance mask. A visualization of the clustering step is shown in Figure 4.4.

4.2.4. Tracking

After the clustering step, the instance segmentation masks are linked over time. Therefore, the shifted positions $\mathbf{e}_i^{\mathrm{T}}$ are calculated for all pixels i that belong to an instance mask at time point t to find their corresponding object centers at $t-1$. Each instance mask at t is marked as a potential matching candidate for the instance mask at $t-1$ that contains the most shifted pixels of it. If a mask at $t-1$ has exactly one matching candidate, the two instance masks at t and $t-1$ are assigned to the same track, whereas if an instance mask at $t-1$ has two potential matching candidates at t, the instance masks at t are set as successors of the instance mask at $t-1$. In all other cases, shifted pixels not overlapping with any mask, more than two matching candidates - the mask at t is marked as a new track starting at t.

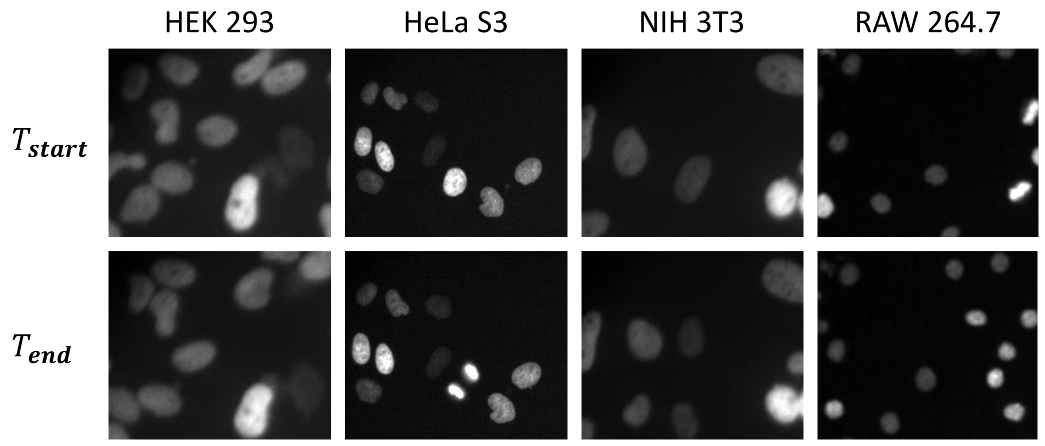


Figure 4.5.: Exemplary frames of training sequences of the DeepCell dataset. To show the change in the appearance of the training data image sequences, for each cell line the first and last frame of an image sequence from the training dataset is shown.

4.3. Experiment

In the following, the proposed method is compared concerning its model size with other neural network architectures, which are used for biomedical image processing tasks. Moreover, the influence of different augmentation strategies during training and inference, and the influence of the amount and kind of training data used are investigated. Finally, the performance of the proposed simultaneous instance segmentation and tracking approach is compared by combining the predicted instance segmentation masks with three other tracking-by-detection approaches.

4.3.1. Experimental Setup

For evaluation, two different datasets are used: the DeepCell dataset, which was used in Section 2.2 and the BF-C2DL-HSC dataset [7]. While the DeepCell dataset provides many image sequences, it is limited concerning the number of frames per image sequence and the number of objects per frame. In contrast, the BF-C2DL-HSC dataset has more frames per image sequence and more object movement, however, only two image sequences are available. In the following, the training and inference procedure for the two different datasets are described.

All models were trained and evaluated on a system with Ubuntu 18.04, an Intel i9 99000k, 32 GB RAM, and two Titan RTX with 24 GB VRAM each. The approach was implemented in Python with PyTorch as deep learning framework. The code of EmbedTrack was made publicly available at `https://git.scc.kit.edu/kit-loe-ge/embedtrack`.

DeepCell Dataset

The original DeepCell dataset [113] consists of 803 image sequences for training and validation and 81 image sequences for benchmarking. In Chapter 2 and Chapter 3 the 81 image sequences, which were held out for benchmarking, were modified with synthetic segmentation errors to compare different tracking-by-detection approaches. Now, the benchmarking image sequences are used to test the segmentation and tracking performance of the proposed approach, while the training is conducted on sub-datasets created from the image sequences hold out for training and validation. An overview of image sequences from different cell lines used for training is given in Figure 4.5.

Training In the following, the setup of the training process for the DeepCell dataset is described.

- **Training Data Creation** To investigate the influence of the amount and kind of training data on the performance of the proposed method, several training datasets – mixed and specialized – are constructed from the original DeepCell training and validation dataset. Mixed training datasets are constructed by using image sequences from all four cell lines: HEK 293, HeLa S3, NIH 3T3, and RAW 264.7 by selecting $N/4$ image sequences from each cell line. For example, the training dataset Mixed 8 consists of two image sequences from each of the four cell lines, whereas the dataset Mixed 80 consists of 20 image sequences from each of the four cell lines. Specialized datasets consist of image sequences from a single cell line. Specialized 20, for instance, refers to using 20 image sequences of a single cell line for training. An overview of the constructed training datasets is given in Table 4.1.

- **Augmentation** To investigate the influence of different types of augmentations, several augmentation strategies are implemented and combined. Therefore, as traditional augmentation strategies during training flipping, rotation by multiple of 90 degrees, contrast adaption, and blur are applied. The contrast adaption is done using CLAHE with a probability of 0.3, whereas the image is blurred with a probability of 0.3 and is applied to each image crop individually. Blurring and contrast adaption are summarized as a single augmentation strategy and referred to as image augmentation. Flipping and rotations are always applied to both image crops where the probability for flipping is set to 0.5 and a rotation by 0, 90, 180, or 270 degrees is drawn uniformly. In addition, an offset augmentation is implemented to simulate larger shifts in object position, by translating the image crop of time point $t-1$ by a small shift in xy-direction compared to the initial position. Therefore, a small random shift is added to the initial position of the crop in image $t-1$, and a new crop from image $t-1$ is created at this new crop position. The offset augmentation is applied with probability 0.3 and the shift is sampled uniformly in the range $[-0.1 \cdot \text{crop\_size}, 0.1 \cdot \text{crop\_size}]$, where crop_size is the size of the created image crops.

- **Image Preprocessing** Each image crop is normalized to range $[0, 1]$ using the 1% and 99% percentiles of the image crop.

			N Seq.	**N Frames**	**N Tracks**	**Overlap Fastest N% of Objects**		
	Dataset	**Data Split**				10	25	50
Mixed	Mixed 8	train	8	24.0	15.0	0.873	0.927	0.955
		val	8	6.0	13.0	0.905	0.933	0.958
	Mixed 20	train	20	24.0	16.0	0.867	0.924	0.954
		val	20	6.0	13.0	0.877	0.923	0.953
	Mixed 40	train	40	24.0	18.5	0.861	0.917	0.952
		val	40	6.0	15.0	0.873	0.916	0.952
	Mixed 80	train	80	24.0	17.0	0.859	0.914	0.950
		val	80	6.0	14.0	0.867	0.915	0.951
Specialized 20	HEK 293 20	train	20	24.0	27.0	0.856	0.907	0.951
		val	20	6.0	19.5	0.868	0.912	0.952
	HeLa S3 20	train	20	32.0	10.0	0.904	0.933	0.957
		val	20	8.0	8.0	0.906	0.929	0.955
	NIH 3T3 20	train	20	24.0	13.0	0.880	0.925	0.961
		val	20	6.0	11.5	0.882	0.925	0.962
	RAW 264.7 20	train	20	24.0	25.5	0.673	0.779	0.870
		val	20	6.0	17.5	0.667	0.799	0.884
Specialized 40	HEK 293 40	train	40	24.0	25.5	0.862	0.907	0.950
		val	40	6.0	19.5	0.872	0.914	0.953
	HeLa S3 40	train	40	32.0	9.5	0.904	0.933	0.957
		val	40	8.0	8.0	0.906	0.928	0.955
	NIH 3T3 40	train	40	24.0	14.0	0.888	0.932	0.966
		val	40	6.0	12.0	0.889	0.932	0.963
	RAW 264.7 40	train	40	24.0	22.0	0.679	0.791	0.882
		val	40	6.0	15.5	0.696	0.807	0.889

Table 4.1.: Training and validation datasets constructed from subsets of the original DeepCell training and validation dataset. The number of frames, the number of tracks, and the object motility statistics – overlap fastest $N\%$ of objects – are the median over the sequences selected for training and validation, respectively. The number of tracks and the object motility are calculated based on the fully annotated ground truth in the respective image sequences.

- **Training and Validation Split** From the original DeepCell training data several training and validation datasets are constructed, by splitting each image sequence selected from the initial training dataset into a training and validation sequence. Therefore, the first 80% of frames of each image sequence are used for training whereas the last 20% of frames of each image sequence are used for validation. The statistics of the constructed training and validation datasets are given in Table 4.1.
- **Optimizer and Early Stopping** As optimizer Adam [185] with learning rate $5 \cdot 10^{-4}$ is trained for a maximum of 150 epochs. In addition, an early stopping criterion is used based on which the training is stopped if the loss on the validation dataset decreased by less than 0.03 in the last 10 epochs of training.
- **Crop Size** The original images are cropped into image crops of size 128x128 and pairs of image crops referring to the same part of the image at t and $t-1$ are forwarded to the model during training and validation.
- **Runs** Each combination of training dataset and augmentation strategies, selecting a single augmentation up to using all augmentation strategies, is trained 10 times.

Testing The models trained on different subsets of the DeepCell training data are tested on the hold out benchmark dataset as follows.

- **Test Datasets** From the DeepCell benchmark dataset test datasets are generated by selecting all frames, every second, or every fifth from each image sequence. Decreasing the temporal resolution through subsampling leads to an increase in the object motility between frames. An overview of the created test datasets is given in Table 4.2. It is emphasized that this data is unseen during the whole training process and only used for final comparison of the differently trained models.
- **Model Selection** From each training run, the last model checkpoint is used for evaluating the performance on the test datasets.
- **Image Preprocessing** Overlapping image crops of shape 128x128 are generated from each image and pairs of image crops referring to the same part of the image at t and $t-1$ are forwarded to the model for inference. Each image crop is normalized to range $[0, 1]$ using the 1% and 99% percentiles of the image crop.
- **Test Time Augmentation** As test time augmentations flipping and rotation by multiple of 90 degrees are applied to the image pair, resulting in eight pairs of augmented image crops. Each pair of augmented image crops is forwarded to the model. Next, the predictions are flipped and rotated again to undo the augmentation. Then, the average over the respective eight predictions is calculated for time point t and $t-1$ respectively, which yields the final predictions for the two image crops.
- **Post-Processing** The predicted crops are stitched to a prediction covering the full image followed by the clustering step to retrieve the instance segmentation masks. Then the instance segmentation masks are linked using the tracking offset predictions.

		N Seq.	N Frames	N Tracks	Overlap Fastest N% of Objects		
Dataset	Δt				10	25	50
Mixed	1	81	30.0	15.0	0.875	0.922	0.951
	2	81	15.0	14.0	0.804	0.874	0.923
	5	81	6.0	13.0	0.644	0.756	0.868
HEK 293	1	26	30.0	23.5	0.868	0.911	0.945
	2	26	15.0	21.0	0.774	0.849	0.911
	5	26	6.0	19.5	0.578	0.724	0.829
HeLa S3	1	18	40.0	9.5	0.903	0.933	0.955
	2	18	20.0	8.5	0.874	0.911	0.948
	5	18	8.0	8.0	0.803	0.876	0.919
NIH 3T3	1	24	30.0	12.5	0.897	0.942	0.967
	2	24	15.0	11.5	0.830	0.897	0.945
	5	24	6.0	10.5	0.695	0.811	0.894
RAW 264.7	1	13	30.0	15.0	0.663	0.779	0.869
	2	13	15.0	15.0	0.551	0.725	0.826
	5	13	6.0	12.0	0.378	0.535	0.718

Table 4.2.: Test datasets constructed from the DeepCell benchmark dataset. Δt refers to subsampling the initial benchmark sequences by selecting each ($\Delta t = 1$), every second ($\Delta t = 2$), or every fifth ($\Delta t = 5$) frame of an image sequence to create subsampled datasets with more object motility. The number of frames, the number of tracks, and the object motility statistics – overlap fastest $N\%$ of objects – are the median over the sequences selected for the different test datasets. The number of tracks and the object motility are calculated based on the fully annotated ground truth in the respective image sequences. The mixed test dataset includes all image sequences of the initial DeepCell benchmark dataset, whereas the other test datasets consist of the image sequences of a single cell line.

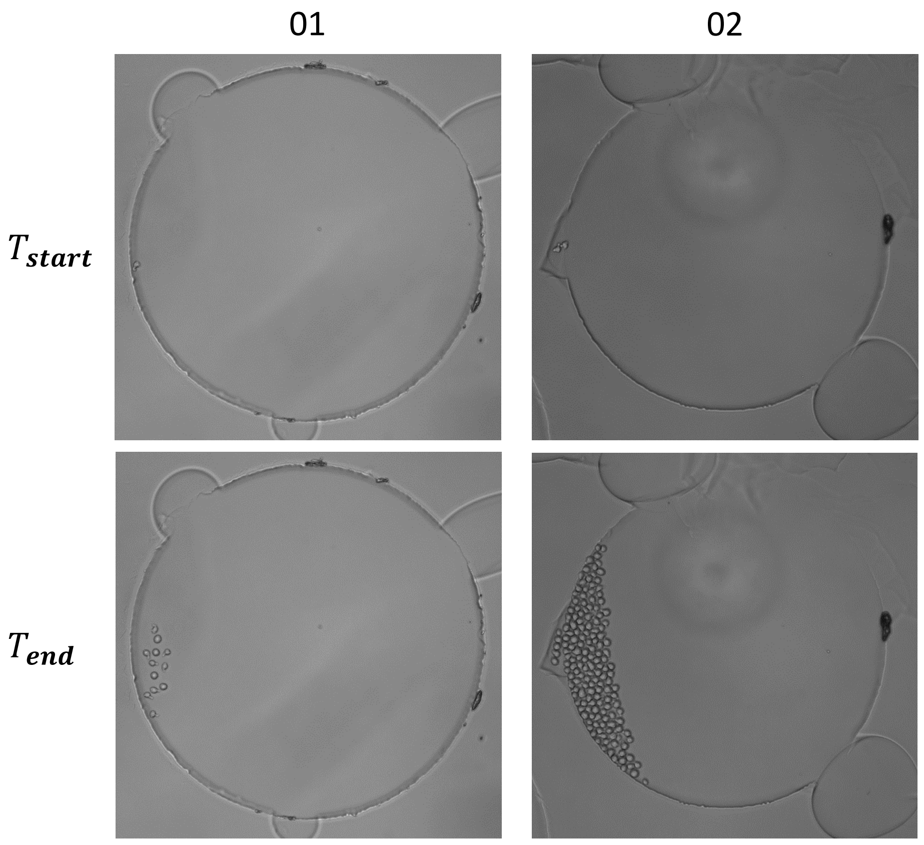


Figure 4.6.: BF-C2DL-HSC dataset. To show the change in appearance in the training and test dataset, the first and last frames of the two image sequences of BF-C2DL-HSC are shown, where image sequence 01 is used for training and image sequence 02 for testing.

Finally, to remove tiny segmentation masks at the image border Field of Interest (FOI) correction is applied. Therefore, a ten pixel thick frame is defined at the image borders and instance segmentation masks are removed that only overlap with this frame and not with the remaining image.

BF-C2DL-HSC Dataset

The BF-C2DL-HSC dataset consists of two image sequences with publicly available annotations covering point-wise tracking GT annotations and few frames with fully annotated instance segmentation masks. In addition, instance segmentation predictions averaged over several instance segmentation approaches are available. While both image sequences have the same number of frames, the amount of objects differs greatly with 12 objects in the last frame of image sequence 01 and 159 objects in the last frame of image sequence 02 a visualization of the two image sequences is given in Figure 4.6.

Training In the following, the setup of the training process on the BF-C2DL-HSC dataset is described.

		N Seq.	N Frames	N Tracks	Overlap Fastest N% of Objects		
Dataset	Data Split				10	25	50
BF-C2DL-HSC 01	train	1	1588.0	17.0	0.383	0.513	0.642
	val	1	176.0	15.0	0.408	0.537	0.674

Table 4.3.: BF-C2DL-HSC training and validation datasets constructed from the BF-C2DL-HSC 01 image sequence. The number of tracks is extracted from the point-wise tracking GT annotations, whereas the object motility is calculated by combining the point-wise tracking GT with the ST instance segmentation masks.

- **Training Data Creation** To create a fully labeled dataset with reasonable annotation quality, the point-wise tracking GT annotations are combined with the publicly available instance segmentation mask predictions, referred to as Silver Truth (ST), by assigning the instance segmentation masks which overlap with exactly one tracking GT marker the tracking ID of the GT tracking marker. An overview of the constructed training dataset is given in Table 4.3.

- **Augmentation** Similar as introduced for the training on the DeepCell dataset, the augmentations flipping, rotation by multiple of 90 degrees, offsets, contrast adaption, and blur are applied using the same parametrization as defined above.

- **Image Preprocessing** Each image crop is normalized to range $[0, 1]$ using the 1% and 99% percentiles per image crop.

- **Training and Validation Split** Since the number of objects is increasing over time in the BF-C2DL-HSC 01 image sequence, the first 90% of frames of the image sequence are used for training whereas the last 10% of frames of the image sequence are used for validation, to include examples of touching objects in the training data split. Statistics of the resulting training and validation datasets are given in Table 4.3.

- **Optimizer and Early Stopping** As optimizer Adam [185] with learning rate $5 \cdot 10^{-4}$ is trained for a maximum of 50 epochs. In addition, an early stopping criterion is used based on which the training is stopped if the loss on the validation dataset decreased by less than 0.03 in the last 5 epochs of training.

- **Crop Size** The original images are cropped into image crops of size 256x256 and pairs of image crops referring to the same part of the image at t and $t - 1$ are forwarded to the model during training and validation.

- **Runs** The model is trained 10 times using a combination of all augmentation strategies together.

Testing The models trained on the BF-C2DL-HSC 01 sequence are tested on the image sequence 02 as follows.

- **Test Datasets** From the BF-C2DL-HSC 02 image sequence three test datasets are generated by selecting all frames, every second, or every fifth the image sequence.

		N Seq.	N Frames	N Tracks	Overlap Fastest N% of Objects		
Dataset	Δt				10	25	50
BF-C2DL-HSC 02	1	1	1764.0	317.0	0.553	0.676	0.787
	2	1	882.0	317.0	0.376	0.547	0.709
	5	1	353.0	317.0	0.046	0.257	0.530

Table 4.4.: Test datasets constructed from the BF-C2DL-HSC 02 image sequence. Δt refers to subsampling the image sequence by selecting each ($\Delta t = 1$), every second ($\Delta t = 2$) or every fifth ($\Delta t = 5$) frame to create subsampled datasets with more object motility. The number of tracks is extracted from the point-wise tracking GT annotations, whereas the object motility is calculated by combining the point-wise tracking GT with the ST instance segmentation masks

Decreasing the temporal resolution through subsampling leads to an increase in the object motility between frames. An overview of the created test datasets is given in Table 4.4. It is emphasized that this data is unseen during the whole training process and only used for final comparison of the differently trained models.

- **Model Selection** From each training run, the last model checkpoint is used for evaluating the performance on the test datasets.
- **Image Preprocessing** Overlapping image crops of shape 256x256 are generated from each image and pairs of image crops referring to the same part of the image at t and $t-1$ are forwarded to the model for inference. Each image crop is normalized to range $[0, 1]$ using the 1% and 99% percentiles of the image crop.
- **Test Time Augmentation** As test time augmentation the same processing is used as described for the DeepCell dataset.
- **Post-Processing** The predicted crops are stitched to a prediction covering the full image followed by the clustering step to retrieve the instance segmentation masks. Then the instance segmentation masks are linked using the tracking offset predictions. Finally, to remove tiny segmentation masks at the image border Field of Interest (FOI) correction is applied. Therefore, a 25 pixel thick frame is defined at the image borders and instance segmentation masks are removed that only overlap with this frame and not with the remaining image.

4.3.2. Evaluation

In the following, the proposed method is compared to other neural network architectures concerning its model size. Then, the performance of the EmbedTrack approach is evaluated concerning the influence of training dataset size, augmentation strategies, and decreasing temporal resolution using the DeepCell test datasets. Moreover, the performance of the tracking part of EmbedTrack is compared with other tracking methods. Finally, the approach is trained and evaluated on the BF-C2DL-HSC dataset. A comparison of the proposed method concerning additional criteria such as runtime and the performance on a diverse set of 2D datasets is conducted in Chapter 5.

Task	Architecture	Number of Parameters
Semantic Segmentation	U-Net [186]	$31.0 \cdot 10^6$
Instance Segmentation	Mask R-CNN [59]	$44.2 \cdot 10^6$
	DeepLab v3 [187]	$39.6 \cdot 10^6$
	FCN [188]	$33.0 \cdot 10^6$
Point-wise Detection and Tracking	Hayashida *et al.* [88]	$17.3 \cdot 10^6$
Instance Segmentation and Tracking	Payer *et al.* [89]	$2.1 \cdot 10^6$
	Chen *et al.* [125]	Mask R-CNN based; no implementation available
	EmbedTrack	$2.5 \cdot 10^6$

Table 4.5.: Number of parameters of different neural network architectures used in biomedical segmentation or tracking tasks in comparison with the proposed EmbedTrack approach. The instance segmentation models, Mask R-CNN, DeepLab v3, and FCN use a ResNet-50 as the backbone. For each architecture a standard parametrization was used, hence, based on the chosen parametrization the number of parameters varies.

Model Size of EmbedTrack in Comparison with other Neural Network Architectures

First, the proposed neural network architecture, based on a branched ERFNet [60], is compared concerning its size with other neural network architectures used in biomedical image processing tasks which is shown in Table 4.5.

For the point-wise detection and tracking approach as well as the instance segmentation and tracking approaches implementations are used which were made available with the according publications. For the semantic segmentation approach and instance segmentation approaches implementations from the torchvision package [1] are used, whereas for U-Net the implementation of Aman Arora is used[2]. All instance segmentation approaches, Mask R-CNN, DeepLab v3, and FCN, use a ResNet-50 [189] as the backbone.

The neural network architectures proposed for instance segmentation of natural images – Mask R-CNN, DeepLab v3, and FCN – which have been applied to biomedical images as well, are two up to 20-times as big as the neural network architectures of Hayashida *et al.*, Payer *et al.*, and the proposed neural network architecture, which were specifically proposed for biomedical image processing tasks. Interestingly, U-Net which was proposed for biomedical image processing tasks is similar in size to Mask R-CNN, DeepLab v3, and FCN. The stacked hourglass neural network of Payer *et al.* and the proposed EmbedTrack approach are in comparison to the other neural network architectures the smallest concerning the number of trainable parameters and are hence less prone to overfit on small datasets.

Segmentation Quality and Field of Interest Correction

In Section 2.2 and Section 3.3 a fraction of the perfect GT instance segmentation masks were altered to simulate segmentation errors while all other perfect GT instance segmentation masks were kept unchanged. Now, the perfect GT instance segmentation masks are

[1] `https://pytorch.org/vision/stable/index.html`

[2] `https://github.com/amaarora/amaarora.github.io/blob/master/nbs/Training.ipynb`

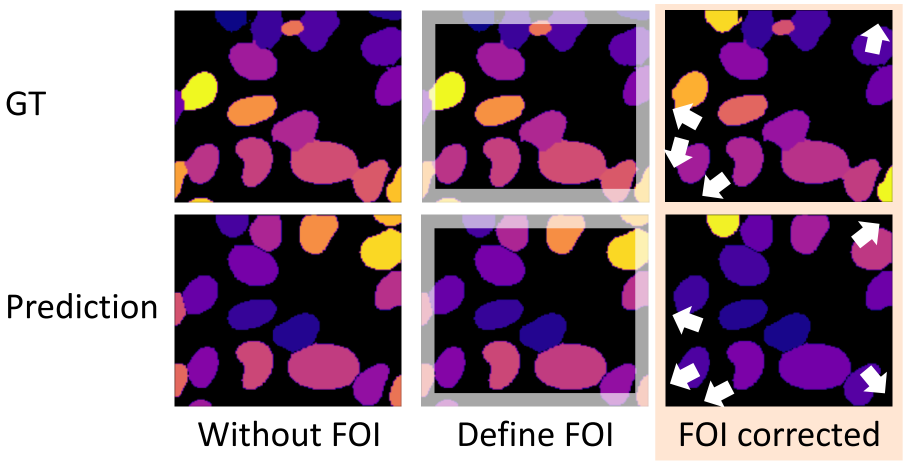


Figure 4.7.: Defining a Field of Interest (FOI) for DeepCell datasets. Tiny segmentation mask fragments at the image border can have a strong impact on the metric score. Therefore, a FOI is defined which is used to adapt the GT and the prediction such that segmentation masks not overlapping with the FOI – segmentation masks only overlapping with the gray frame – are removed. The removed instance segmentation masks in the GT and prediction are marked with white arrows.

compared against the instance segmentation masks predicted by the proposed method. In the current scenario, the tiny instance segmentation mask fragments of the GT at the image borders strongly affect the final metric score, as not segmenting a fragment sized a few pixels at the image border is penalized as a FN in the evaluation metrics.

An approach to reduce the effect of the segmentation mask fragments at the image border is to define a Field of Interest (FOI) as done for other datasets [7]. The idea is to define a frame at the image border which is a few pixels wide and remove instance segmentation masks that only overlap with this frame but not the remaining image. A visualization of the FOI is shown in Figure 4.7.

To investigate the influence of the FOI correction on the evaluation, ten models are trained on the Mixed 40 dataset using a combination of flipping, rotation by multiple of 90 degrees, image augmentations, and offsets as training augmentation. All trained models are evaluated on the test dataset with full temporal resolution, $\Delta t = 1$, of the four cell lines of the DeepCell dataset. Next, the SEG score is calculated for each predicted frame of instance segmentation masks, once considering FOI correction for GT and prediction and once without FOI correction for GT and prediction. The FOI correction removes all instance segmentation masks from GT and prediction that only overlap with a ten pixel wide frame at the image border. To quantify bad, median, and good segmentation performance of the trained models the 1%, 50%, and 90% percentiles of the SEG score per frame are calculated which are given in Table 4.6. For all datasets, the SEG score per frame is higher, especially concerning the 1% percentile of the SEG performance, which demonstrates the strong effect of small mask fragments at the image borders on the final metric score.

		Segmentation Quality by SEG Score Frame-wise		
		Worst Case	Median	Best Case
Dataset	**FOI**	1% Percentile	50% Percentile	90% Percentile
HEK 293	FOI	0.712	0.888	0.920
	No FOI	0.602	0.824	0.891
HeLa S3	FOI	0.606	0.862	0.911
	No FOI	0.469	0.846	0.896
NIH 3T3	FOI	0.637	0.870	0.928
	No FOI	0.631	0.847	0.906
RAW 264.7	FOI	0.787	0.901	0.919
	No FOI	0.682	0.847	0.899

Table 4.6.: Segmentation quality based on the SEG score of EmbedTrack with and without FOI correction on DeepCell test datasets with full temporal resolution, $\Delta t = 1$. The SEG score was calculated per frame and the 1%, 50%, and 90% percentiles of the SEG score per frame per test dataset are shown.

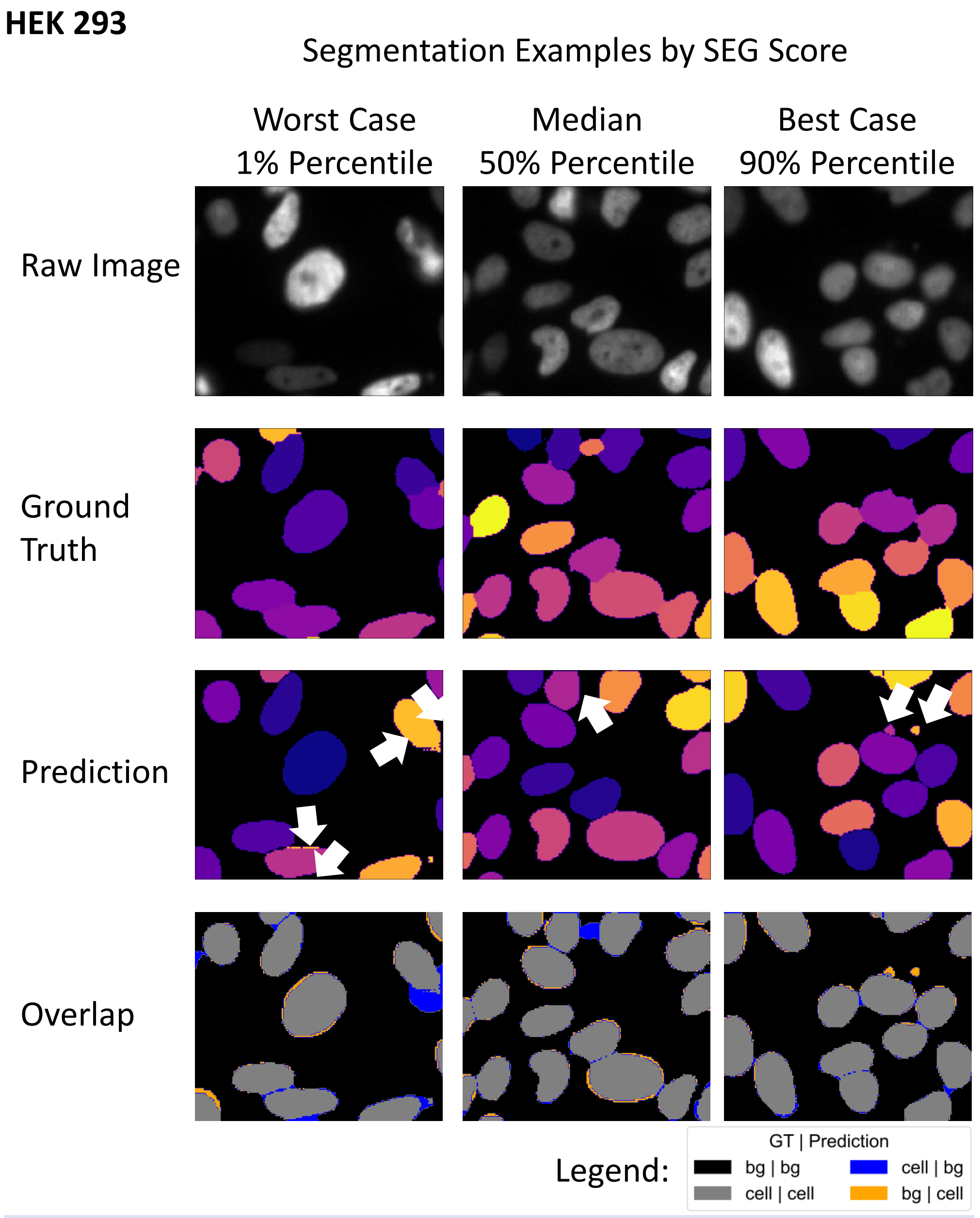
To show examples of segmented frames belonging to the 1%, 50%, and 90% percentile concerning segmentation quality measured based on the SEG metric, Figure 4.8 provides example frames for the cell line HEK 293, whereas Figure 4.9 provides example frames for the cell line HeLa S3. Further visualizations of RAW 264.7 and NIH 3T3 are provided in the appendix in Figure A.1 and Figure A.2, respectively. The bad segmentation example of HeLa S3, Figure 4.9 left column, demonstrates how a small GT instance segmentation mask fragment strongly influences the SEG score of this frame. By applying FOI correction, the SEG score of this frame improves from 0.47 to 0.94. Moreover, the GT of HEK 293 tends to segment the objects too big, see Figure 4.8 right column, touching objects in GT compared to non-touching objects in the raw image. Hence, for high SEG scores – above 0.8 – it remains questionable whether a higher SEG score indicates a better segmentation quality or just more similarity with the bias of the human annotator.

While the effect of tiny instance segmentation mask fragments affects the SEG score on a frame-wise level also all metrics are affected when evaluated per image sequence on the DeepCell test dataset. Table 4.7 displays the 1%, 50%, and 90% percentiles of all metric scores after evaluating the ten trained models on the test datasets with full temporal resolution, $\Delta t = 1$, of the four cell lines of the DeepCell dataset. Again, the predictions are evaluated with FOI correction applied to GT and prediction and without applying FOI correction. Besides the improvement of the SEG score, also the DET and TRA score improve fundamentally by applying FOI correction, as FNs are penalized ten times as much as FPs in DET and TRA metric.

In the following experiments, FOI correction is applied to all test datasets.

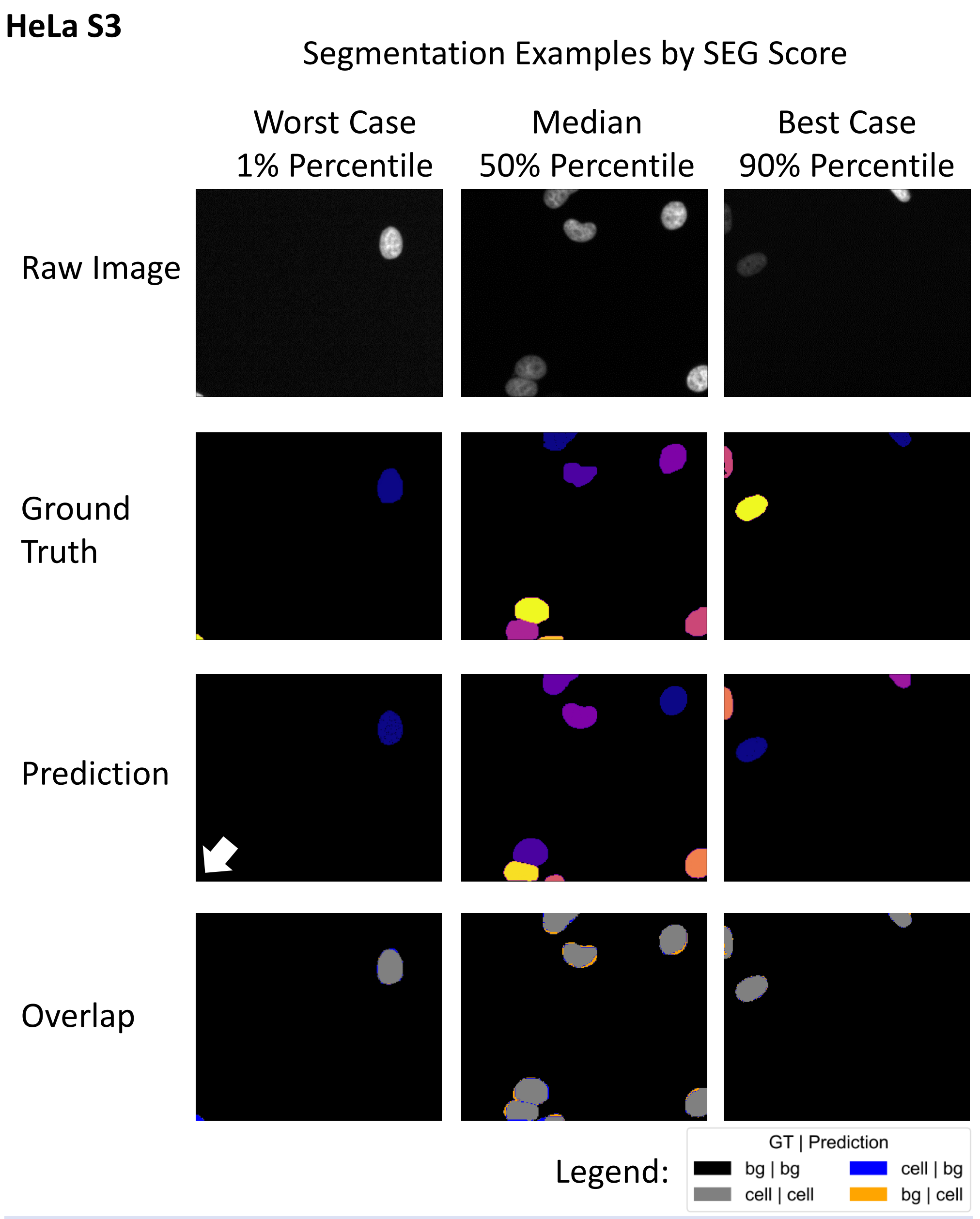
Influence of Augmentations and Training Dataset Size

Next, the influence of augmentations during training and the size of the training dataset on the model performance is analyzed. Therefore, two training datasets, Mixed 8 and Mixed 80 are used for training combined with different combinations of augmentations. Per combination of training dataset and augmentation ten models are trained which are evaluated on all the DeepCell Mixed test dataset with full temporal resolution, $\Delta t = 1$.



SEG Score	0.60	0.82	0.92
SEG Score (FOI corrected)	0.69	0.79	0.92

Figure 4.8.: Segmentation quality based on SEG score of EmbedTrack on HEK 293 test dataset. The SEG score is calculated for each frame and the frames closest to the 1%, 50%, and 90% percentile without FOI correction are shown. In addition, the SEG score after applying FOI correction to the frame is shown. The segmentation errors FN, FP, over-, and under-segmentation are marked with white arrows.



SEG Score	0.47	0.85	0.92
SEG Score (FOI corrected)	0.94	0.89	0.94

Figure 4.9.: Segmentation quality based on SEG score of EmbedTrack on HeLa S3 test dataset. The SEG score is calculated for each frame and the frames closest to the 1%, 50%, and 90% percentile without FOI correction are shown. In addition, the SEG score after applying FOI correction to the frame is shown. The segmentation errors FN, FP, over-, and under-segmentation are marked with white arrows.

			Tracking Results over Image Sequences		
			Worst Case	Median	Best Case
Dataset	**Metric**	**FOI**	1% Percentile	50% Percentile	90% Percentile
HEK 293	DET	FOI	0.884	0.974	0.993
		No FOI	0.774	0.924	0.974
	SEG	FOI	0.719	0.869	0.909
		No FOI	0.643	0.807	0.870
	TRA	FOI	0.858	0.967	0.989
		No FOI	0.761	0.917	0.971
HeLa S3	DET	FOI	0.857	0.990	1.000
		No FOI	0.784	0.986	0.997
	SEG	FOI	0.756	0.848	0.886
		No FOI	0.704	0.832	0.876
	TRA	FOI	0.869	0.987	1.000
		No FOI	0.781	0.986	0.996
NIH 3T3	DET	FOI	0.876	0.985	0.998
		No FOI	0.885	0.978	0.995
	SEG	FOI	0.706	0.854	0.918
		No FOI	0.688	0.830	0.882
	TRA	FOI	0.876	0.983	0.996
		No FOI	0.881	0.976	0.994
RAW 264.7	DET	FOI	0.957	0.993	0.999
		No FOI	0.902	0.963	0.985
	SEG	FOI	0.833	0.897	0.913
		No FOI	0.758	0.840	0.865
	TRA	FOI	0.947	0.988	0.997
		No FOI	0.897	0.958	0.984

Table 4.7.: Influence of FOI correction on the DeepCell test datasets with full temporal resolution, $\Delta t = 1$. Ten models trained on the Mixed 40 dataset are evaluated on the DeepCell $\Delta t = 1$ test datasets with and without using FOI correction on prediction and GT.

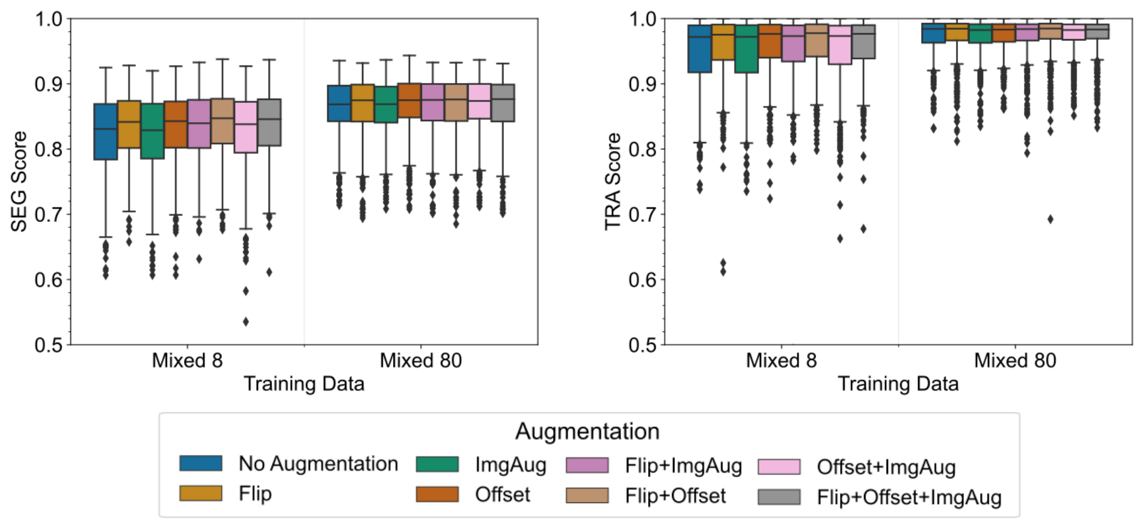


Figure 4.10.: SEG and TRA on DeepCell Mixed test dataset with full temporal resolution, $\Delta t = 1$, with different augmentation strategies during training. '+' refers to combining several augmentations during training, 'ImgAug' refers to using CLAHE and blur, whereas 'No Augmentation' refers to not using any augmentation. Per box plot, the median over $N = 810$ image sequences – ten models evaluated on 81 image sequences in the test dataset – is shown.

During inference, test time augmentation is used for all models. The results of using the different training setups concerning SEG and TRA scores are shown in Figure 4.10.

Overall, models trained on the larger training dataset Mixed 80 reach higher median scores on the SEG and TRA metric on the DeepCell Mixed test dataset with full temporal resolution, $\Delta t = 1$, independent of the used augmentation. The influence of augmentations during training on the performance of the test dataset is more distinct for the models trained on the smaller Mixed 8 training dataset than for models trained on the larger Mixed 80 training dataset. Using flipping or offset augmentation in combination with the Mixed 8 training dataset has the largest improvement on the SEG score, whereas adding the image augmentations CLAHE and blur, referred to as ImgAug in the plot, performs similarly on the test dataset to adding no augmentation during training. The effect of augmentation during training on the performance on the test dataset is more distinct on the SEG metric compared to the TRA metric. A potential explanation for this is the ability of the SEG metric to quantify changes in the Jaccard index for each segmentation mask in the range $[0, 1]$, whereas penalties in the TRA metric are discrete – a link between two instance segmentation masks can either be set or missing which causes a fixed penalty.

Training General versus Specialized Models

To investigate whether it is beneficial to include related data or train a model only on data similar to the test data, models are trained on datasets consisting of all four cell lines versus just image sequences showing a single cell line. All models are trained with the augmentations flipping, rotation by multiple of 90 degrees, offsets, CLAHE, and blur. Per training dataset, ten models are trained which are all evaluated on the test datasets with

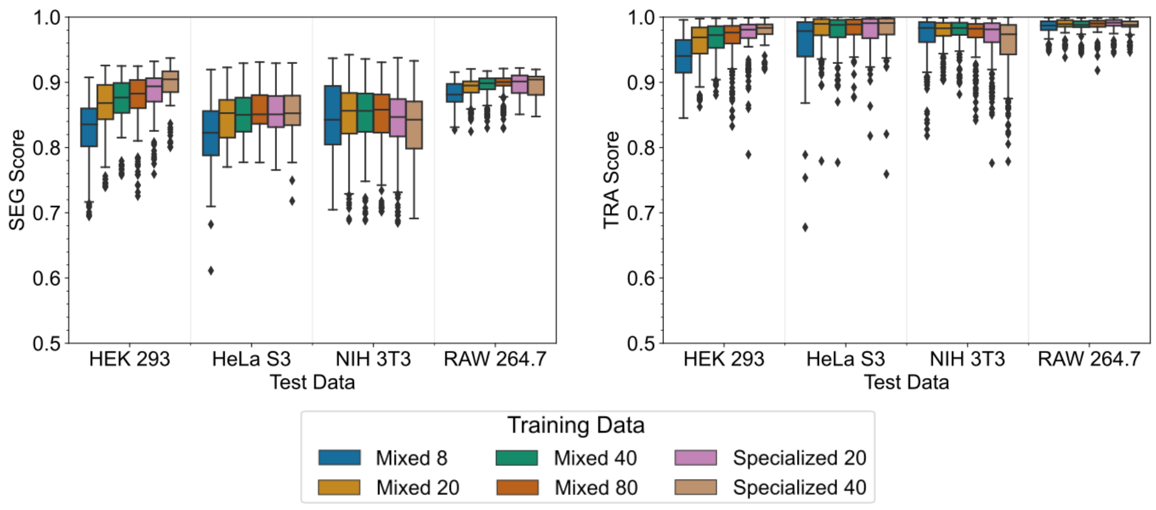


Figure 4.11.: SEG and TRA on DeepCell test datasets with full temporal resolution, $\Delta t = 1$, after training models with mixed and specialized data. For Specialized 20 and Specialized 40, the cell line of the training dataset and test dataset are equal. Per box plot, ten models are evaluated on the respective test dataset and the median is calculated over $N = 260$ for HEK 293, $N = 180$ for HeLa S3, $N = 240$ for NIH 3T3, and $N = 130$ for RAW 264.7 predicted image sequences.

full temporal resolution, $\Delta t = 1$, of the four cell lines of the DeepCell dataset. The results of the different training setups concerning SEG and TRA scores are shown in Figure 4.11. For brevity, the training datasets consisting of image sequences of only one cell line are referred to as Specialized 20 and Specialized 40 – referring to the respective single cell line training datasets HEK 293 20, HeLa S3 20, NIH 3T3 20, RAW 264.7 20 and HEK 293 40, HeLa S3 40, NIH 3T3 40, RAW 264.7 40, respectively. For Specialized 20 and Specialized 40, the cell line of the training dataset is equal to the cell line used in the test dataset.

On all test datasets increasing the training dataset from Mixed 8 to Mixed 20 leads to a distinct improvement in the SEG metric. On the TRA metric, only HEK 293 and HeLa S3 show a distinct improvement by increasing the training dataset from Mixed 8 to Mixed 20. Only for HEK 293, the performance of trained models concerning the SEG and TRA score improves when more image sequences of HEK 293 are included in the training data. For HeLa S3 and RAW 264.7, the SEG and TRA scores show no distinct improvement by increasing the training dataset beyond Mixed 20. Interestingly, on the NIH 3T3 test dataset, the median SEG and TRA score decrease slightly when during training only image sequences from NIH 3T3 are used.

An explanation for the different variances of the four test datasets is given by the different sizes of the test sets. RAW 264.7, which shows the smallest variance in SEG and TRA scores, consists of only 13 image sequences for testing, whereas HEK 293 consists of 26 image sequences.

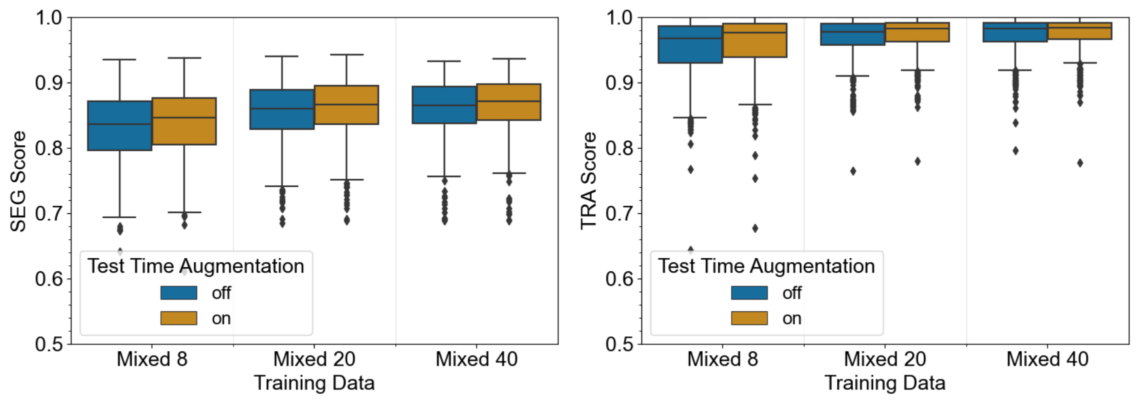


Figure 4.12.: SEG and TRA on DeepCell Mixed test dataset with full temporal resolution, $\Delta t = 1$, with and without test time augmentation. Per box plot, the median over $N = 810$ image sequences – ten models evaluated on 81 image sequences in the test dataset – is shown.

Influence of Test Time Augmentation

Next, the influence of using test time augmentation is investigated. Therefore, models are trained using the Mixed 8, Mixed 20, and Mixed 40 training datasets each combined with all augmentations. On each training dataset, ten models are trained. During inference, each trained model is once evaluated without test time augmentation and once with test time augmentation on the DeepCell Mixed test dataset with full temporal resolution, $\Delta t = 1$. The SEG and TRA score on the test dataset are shown in Figure 4.12.

Applying test time augmentation improves the performance of the trained models on the test datasets concerning SEG and TRA independent from the training dataset. However, the improvement using test time augmentation is smaller for models trained on the larger Mixed 40 dataset compared to applying test time augmentation on models trained on the smaller Mixed 8 dataset.

Influence of Temporal Resolution

To investigate how the proposed approach handles larger changes in object appearance and position between successive frames, the DeepCell test datasets with full temporal resolution, $\Delta t = 1$, are subsampled in time by selecting every second and fifth frame, respectively. The subsampled DeepCell test datasets are referred to as $\Delta t = 2$ and $\Delta t = 5$ and are compared to the original, not subsampled test datasets which are referred to as $\Delta t = 1$. Ten models are trained on the Mixed 40 dataset with all augmentations during training and test time augmentation. No subsampling in time is applied during training. The SEG and TRA scores on the subsampled test datasets of the trained models are shown in Figure 4.13.

The segmentation quality quantified by the SEG metric does not decrease with decreased temporal resolution. This is explained by the architecture of the proposed method – the instance segmentation masks are predicted for the two forwarded frames individually without any coupling between them. However, the spread in the SEG score increases with

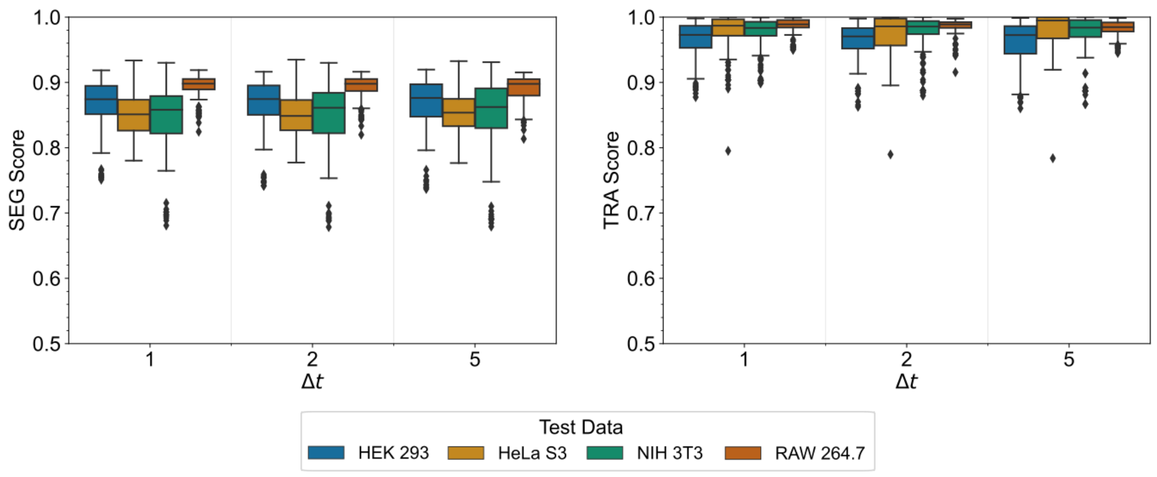


Figure 4.13.: SEG and TRA of EmbedTrack on subsampled DeepCell test datasets. Δt refers to selecting each ($\Delta t = 1$), every second ($\Delta t = 2$), or every fifth ($\Delta t = 5$) frame of an image sequence in the test dataset to create subsampled test datasets with more object motility. Per box plot, ten models are evaluated on the respective test dataset and the median is calculated over $N = 260$ for HEK 293, $N = 180$ for HeLa S3, $N = 240$ for NIH 3T3, and $N = 130$ for RAW 264.7 predicted image sequences.

the subsampling as the corresponding GT and predicted segmentation mask sequences are now very short, for $\Delta t = 5$ only six frames per sequence, see Table 4.2, hence, the influence of one less well predicted instance segmentation mask has a stronger effect on the SEG score. The TRA score also remains unchanged with decreased temporal resolution. A potential explanation is the small object motility even for larger subsampling rates, which is given in Table 4.2.

Comparison of Tracking Performance

Next, the tracking of EmbedTrack, linking through predicting offsets from pixels at t to their object center at $t - 1$, is compared with other tracking-by-detection methods. For comparison, the proposed graph-based tracking of Chapter 3 and the in Section 2.2 introduced tracking methods MU-Lux-CZ and KIT-Sch-GE (1) are selected. KTH-SE is not selected due to its low performance on the DeepCell benchmark dataset in Section 2.2. All tracking approaches receive as segmentation the instance segmentation masks predicted by the EmbedTrack approach. Therefore, EmbedTrack models are trained on the Mixed 40 training dataset with all augmentations during training. No subsampling in time is applied during training. During inference, test time augmentation is applied. Ten EmbedTrack models are trained and instance segmentation masks are predicted for all Mixed test datasets. The resulting instance segmentation mask sequences and corresponding raw image sequences are then forwarded to the three tracking approaches. The results concerning SEG and TRA score on the subsampled DeepCell Mixed test datasets are given in Figure 4.14.

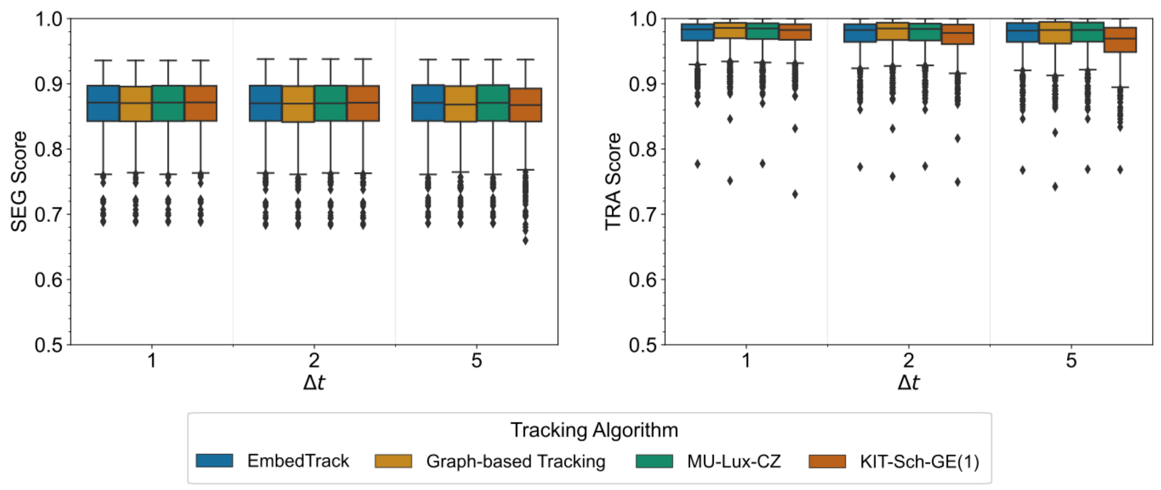


Figure 4.14.: SEG and TRA of different tracking approaches on subsampled DeepCell Mixed test datasets. Δt refers to selecting each ($\Delta t = 1$), every second ($\Delta t = 2$), or every fifth ($\Delta t = 5$) frame of an image sequence in the test dataset to create subsampled test datasets with more object motility. Per box plot, the median over $N = 810$ image sequences – ten models evaluated on 81 image sequences in each test dataset – is shown.

On the SEG score all methods perform on par, apart from KIT-Sch-GE (1) which is slightly decreasing in the SEG score on the subsampled test dataset $\Delta t = 5$. EmbedTrack and MU-Lux-CZ do not alter the instance segmentation masks, whereas the graph-based tracking from Chapter 3 and KIT-Sch-GE (1) can alter the instance segmentation masks leading to slight changes for $\Delta t = 5$. An explanation for the slight drop in performance of these two methods is their capability to interpolate missing segmentation masks, causing for the subsampled dataset however FPs.

All methods show a high median TRA score on all test datasets. The graph-based tracking from Chapter 3 performs slightly better concerning the median TRA score on $\Delta t = 1$ and $\Delta t = 2$, whereas KIT-Sch-GE (1) decreases in performance for increased subsampling. The worse performance of KIT-Sch-GE (1) is due to the increased number of erroneous links which are penalized by the TRA metric.

Evaluation on BF-C2DL-HSC 02

Finally, the performance of EmbedTrack is evaluated on the BF-C2DL-HSC dataset, which consists of only a single image sequence with few objects for training and a single image sequence for testing with many touching objects. The setup for training and testing on the BF-C2DL-HSC dataset is described in Section 4.3.1.

A qualitative demonstration of the segmentation performance of EmbedTrack is shown in Figure 4.15. Although trained on a dataset with few touching objects, the approach is able to segment touching objects well.

Next, the performance of the tracking part of EmbedTrack is compared with three other tracking approaches: the graph-based tracking from Chapter 3, MU-Lux-CZ, and

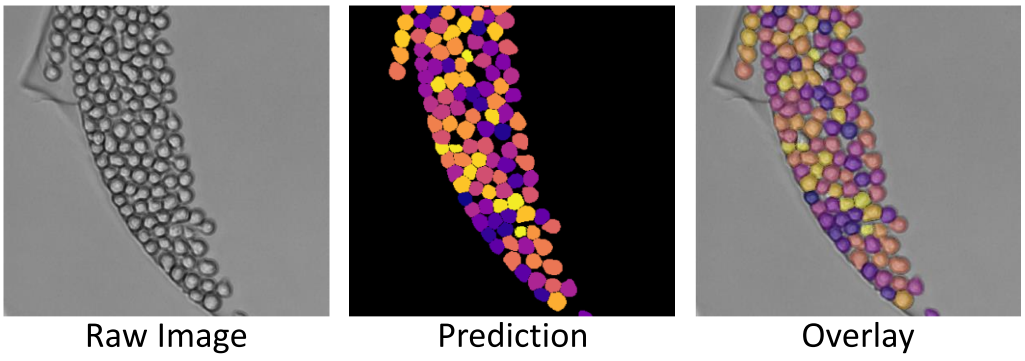


Figure 4.15.: Qualitative Segmentation Results on BF-C2DL-HSC 02. Shown are a crop of the raw image, the predicted instance segmentation masks through the proposed approach, and an overlay of raw image and predicted instance segmentation masks.

KIT-Sch-GE (1). Therefore, ten EmbedTrack models are trained on the image sequence 01 using all augmentations during training. During inference, test time augmentation is used. The predicted instance segmentation masks of the EmbedTrack approach are forwarded together with the raw image sequence as input to the other tracking approaches. The SEG and TRA scores of the different methods on the subsampled BF-C2DL-HSC 02 dataset are given in Figure 4.16.

Concerning the SEG scores, it needs to be highlighted that the amount of fully labeled GT is limited. For the $\Delta t = 1$ test dataset only eight fully annotated frames of instance segmentation masks are available, whereas for the $\Delta t = 5$ test dataset only a single GT frame is available based on which the SEG score is calculated. For the $\Delta t = 1$ test dataset, all approaches perform similarly, however for the $\Delta t = 2$ and the $\Delta t = 5$ test datasets the graph-based tracking approach performs worse. The increased subsampling leads to large changes in the position of some objects, which makes them difficult to link, even for humans. For these objects, the graph-based tracking erroneously detects an under-segmentation error and selects a nearby object which is split. The split segmentation masks cause a drop in the SEG score as the Jaccard index between the instance segmentation mask fragments and the GT instance segmentation mask is small.

On the TRA metric, all models decrease in performance with increased subsampling, as the test dataset becomes more difficult due to increased object mobility and mitosis events become more difficult to assign to the correct predecessor. KIT-Sch-GE (1) performs worse than the other three approaches, due to many incorrect links which are likely caused by the big ROI, which is used to estimate movements between frames that seems to fail for crowded scenarios. The other three tracking approaches, EmbedTrack, graph-based tracking, and MU-Lux-CZ, perform similarly. While the object movement increases with increased subsampling, the objects in the center of the crowd, see Figure 4.15, are constrained in their movement by the surrounding objects. Due to their constrained movement, most objects still overlap by more than 0.5 with their predecessor at $t - 1$ even for $\Delta t = 5$, see Table 4.4. Hence, while overlap-based tracking MU-Lux-CZ shows worse performance for the more motile objects at the border of the crowd, the final TRA score is

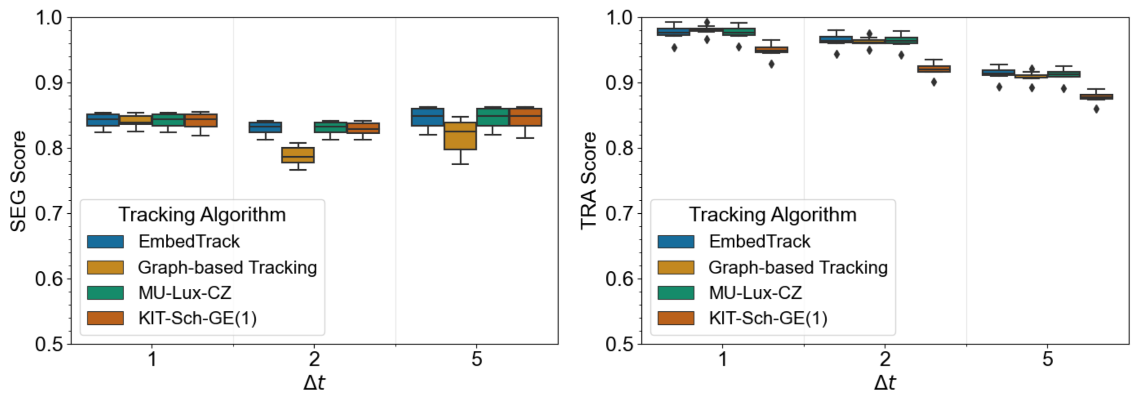


Figure 4.16.: SEG and TRA scores of different tracking approaches on subsampled BF-C2DL-HSC 02 test datasets. Δt refers to selecting each ($\Delta t = 1$), every second ($\Delta t = 2$) or every fifth ($\Delta t = 5$) frame of the image sequence 02 to create subsampled test datasets with more object motility. Per box plot, the median over $N = 10$ image sequences – ten models evaluated on one image sequence in each test dataset – is shown.

still dominated by the majority of objects which are constrained in their movement that can be easily linked using overlap-based tracking, resulting in a similar performance as EmbedTrack and the graph-based tracking.

4.4. Discussion

This chapter proposed learning instance segmentation and tracking jointly in a single neural network by learning offsets of pixels belonging to an object to their object centers at t and $t-1$ and a clustering bandwidth. The proposed network architecture does not require any recurrent neural network parts and the predicted embeddings – offsets and clustering bandwidth – are human comprehensible.

To investigate the performance of the proposed approach, several experiments were conducted using the DeepCell dataset as well as the BF-C2DL-HSC dataset. Compared to other neural network architectures the branched ERFNet, the EmbedTrack approach is based on, is small, which makes it less susceptible to overfit on small datasets. The advantage of the small model architecture was validated by the conducted experiments concerning the required size of the training dataset: already a small training dataset consisting of only eight image sequences performed competitively compared to using 20 or 40 image sequences for training. Concerning augmentation during training, flipping and offset augmentations had the strongest effect on the performance during testing, however, their effect was small. Also, using test time augmentation lead to a small improvement in results. Moreover, on the BF-C2DL-HSC dataset, the approach showed its potential to learn on a training dataset with few objects and performing well on a crowded image sequence with many touching objects.

When the temporal resolution was decreased by subsampling the test image sequences, the segmentation quality of EmbedTrack did not decrease because the instance segmentation mask predictions at time points t and $t-1$ are independent of each other. In addition, when comparing the tracking performance of EmbedTrack with three other tracking approaches, the graph-based tracking from Chapter 3, MU-Lux-CZ, and EmbedTrack performed similarly on the subsampled DeepCell and BF-C2DL-HSC test datasets. However, while for the subsampled BF-C2DL-HSC test datasets the object motility between successive frames increased, linking the objects becomes more difficult for all tracking approaches as well as for humans, due to the crowded scenery and the visual similarity of objects. Hence, the approach needs to be tested on other datasets that have more object motility in combination with more distinguishable objects.

In its current formulation, EmbedTrack requires dense annotations of segmentation masks and links over time. To reduce the burden of annotation, directions of future work are training the approach on simulated data, for instance by using GANs as in [131], or adapting the training to learn on sparsely labeled data. In addition, the approach is only applicable to 2D datasets, thus, extending the approach to 3D should be investigated.

5. Application and Comparison of Proposed MOT Methods

In Chapter 3 and Chapter 4 two MOT methods were proposed with an initial analysis on small benchmark datasets from the DeepCell dataset. In this chapter, the proposed MOT approaches are compared on a diverse benchmark dataset covering different image modalities, varying numbers of objects as well as image sequences in 2D and 3D. Moreover, the proposed MOT approaches are compared against other algorithms submitted to the benchmark. While tracking accuracy is an important measure to compare different tracking approaches, additional comparison criteria are defined to allow for a comprehensive comparison between MOT approaches.

The remainder of this chapter is organized as follows: Section 5.1 proposes criteria to compare different MOT approaches. In Section 5.2 the benchmark dataset is introduced including an overview of approaches that were already submitted to the benchmark. Next, in Section 5.3 the two in this thesis proposed MOT methods are compared based on the in Section 5.1 defined criteria against other to the benchmark submitted methods. Finally, Section 5.4 concludes this chapter by providing a summary and discussion of the applicability of the two MOT approaches.

5.1. Criteria for Comparison

In the following, criteria to compare MOT approaches are introduced.

Tracking Quality As MOT provides the basis for subsequent analysis, a high tracking quality is needed for subsequent tasks such as motion analysis [190, 191], behavior analysis [192], scene understanding [193], or human-machine interaction [194]. Hence, most MOT benchmarks use several evaluation measures to compare the tracking quality [66, 195, 196, 197, 198]. An overview of often used tracking measures is given in Section 1.1.5.

Scalability Usually there are minimum requirements on the MOT algorithm concerning runtime and computing resources. Some application domains require real-time capabilities [180] or require the MOT algorithm to run on devices with limited computing power [199]. Besides these strict requirements, applying the MOT approach from several

Parts of this chapter are adapted from K. Löffler and R. Mikut. "EmbedTrack — Simultaneous cell segmentation and tracking through learning offsets and clustering bandwidths". In: IEEE Access 10 (2022), pp. 77147–77157. doi: 10.1109/ACCESS.2022.3192880.

dozen to several hundreds of objects per image or applying an approach in 2D or 3D scenes should only require a reasonable increase in runtime and computing resources.

Additional Criteria for Evaluation Since MOT is just one building block in the image processing or automation pipeline, also time, financial resources, and expert knowledge required to apply the MOT approach to the task at hand need to be considered. Hence, the following criteria are proposed:

- Manual tuning required?: Tuning a large set of manually tunable parameters requires expert knowledge or is time intensive and hence expensive.
- Labeled data required?: Collecting labeled data to train or fine-tune supervised or semi-supervised approaches can be time expensive and can require expert knowledge to label objects for the task at hand.
- Robustness: Small changes in the input data and the parametrization of the algorithm should not lead to large drops in performance. Also, robustness towards segmentation errors should be considered.
- Applicability to 2D and 3D datasets: Some methods can be applied to 2D and 3D datasets, whereas other approaches are only applicable to 2D datasets
- Additional Requirements: High hardware requirements or software licenses might make an approach infeasible to use in practice due to high financial costs.

Hence, several criteria need to be considered when selecting a MOT approach for an application. Depending on the application, these criteria can have different importance, for instance, scalability might be more important than choosing the best approach concerning tracking accuracy.

5.2. Cell Tracking Challenge Benchmark

In the following, the MOT approaches proposed in Chapter 3 and Chapter 4 are compared against other MOT approaches on the Cell Tracking Challenge (CTC) [7].

5.2.1. The Benchmark

The CTC provides two benchmarks based on the same cell datasets: the Cell Segmentation Benchmark (CSB) for segmentation and the Cell Tracking Benchmark (CTB) for tracking. Both benchmarks are open for submission and the leaderboards are updated each month. As the present thesis proposed MOT approaches, the CTB is used for evaluation.

Datasets The CTC data consists of two labeled image sequences for training, which are referred to as training sequences 01 and 02, and two unlabeled image sequences for testing, which are referred to as challenge sequences 01 and 02. For the training sequences, manually generated, Gold Truth (GT), annotations are provided for some segmentation masks, whereas the tracking information is fully provided as point-wise annotations for the selected datasets. Moreover, for some datasets also a Silver Truth (ST) is provided, which are segmentation masks obtained from averaging predictions of previous submissions to the CTC. The CTC provides a total of 20 different datasets – ten in 2D and ten in 3D. From these 20 datasets, 16 require full tracking, whereas the remaining four datasets only provide partly tracked data. Thus, for the present evaluation the only the datasets that require full tracking are used. Each dataset name consists of the imaging modality, annotation style, image dimensionality, image resolution, and cell line, where Figure 5.1 provides an overview of the 16 selected datasets for evaluation. Moreover, Table 5.1 provides an overview of the cell statistics which are computed from the training sequences.

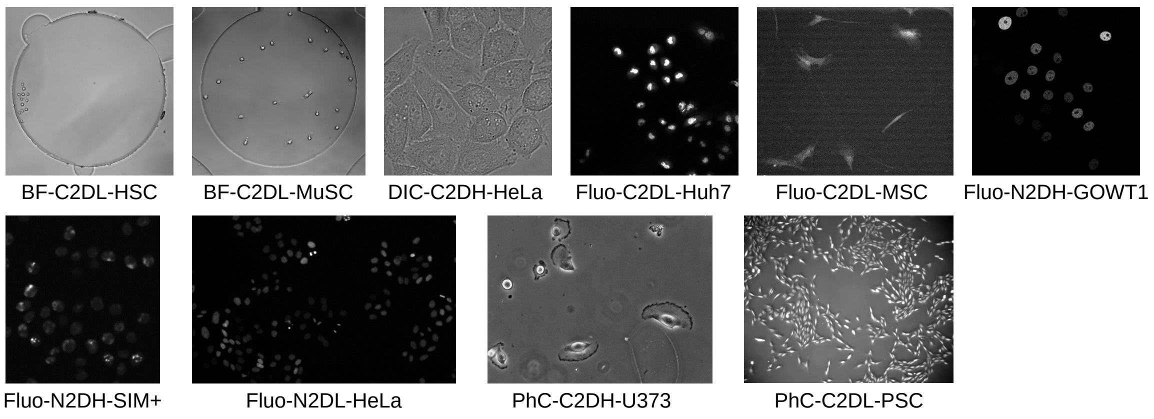
As a measure for cell motility, the ST annotations are used to calculate the overlap of cells between successive frames as their intersection of the two masks divided by the smallest size of the two masks and the overlap percentiles of the 10%, 25%, and 50% most motile cells are reported. For instance, the most motile 10% of the cells in the 01 training sequence of BF-C2DL-HSC have an overlap smaller than 0.388. Based on the cell statistics, challenging datasets are identified based on their cell motility and their cell count. In the following, datasets with more than 10 cells in the last frame as well as an overlap of the fastest 10% of cells of less than 0.5 are defined as datasets with high cell motility. Datasets with more than 100 cells in the last frame are defined as datasets with high cell counts. Based on these definitions datasets with high cell motility are the 2D datasets BF-C2DL-HSC 01 and BF-C2DL-MuSC and the 3D dataset FLuo-C3DL-MDA231. Datasets with high cell count are the 2D datasets BF-C2DL-HSC 02, Fluo-N2DH-HeLa, and PhC-C2DL-PSC and the 3D dataset Fluo-N3DH-CE.

Evaluation Metrics To compare the submitted cell segmentation and tracking algorithms, the CTC uses the metrics DET, SEG, and TRA, which all lay in the range $[0, 1]$, where a higher score corresponds with a better performance. All metrics were introduced in Section 1.1.5. The segmentation benchmark CSB uses the metrics DET and SEG, whereas the tracking benchmark CTB uses the metrics SEG, TRA, and the average between SEG and TRA score, referred to as $\mathrm{OP}_{\mathrm{CTB}}$.

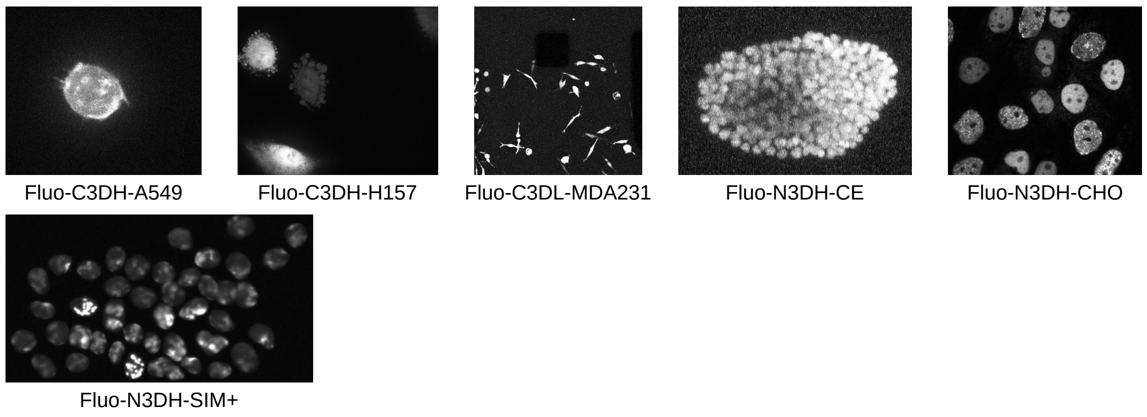
5.2.2. Overview of Participating Methods

By 2022, 50 teams have participated on the CTC[1]. Each team can select the datasets on which their submitted approach is evaluated. Moreover, each team can submit various approaches, which allows for submitting segmentation-only methods to the segmentation benchmark CSB as well as submitting MOT methods to the segmentation and tracking benchmarks CSB and the CTB. Table 5.2 provides a summary of all MOT approaches

[1] `http://celltrackingchallenge.net/participants/`



(a) 2D datasets from the CTC.



(b) 3D datasets from the CTC.

Figure 5.1.: Sample images of the benchmark data from the CTC. The contrast of the raw images has been adapted using min-max scaling to the percentiles 1 and 99 for visualization purposes. For 3D datasets their maximum projection of the last frame in an image sequence is shown. All datasets can be retrieved from `http://celltrackingchallenge.net/`.

			N Frames	**N Tracks**	**N Cells**		**Overlap Fastest N% of Cells**		
	Dataset	**Seq.**			T_{start}	T_{end}	10	25	50
2D Datasets	**BF-C2DL-HSC**	01	1764	23	1	12	0.388	0.519	0.647
		02	1764	317	2	159	0.553	0.676	0.787
	BF-C2DL-MuSC	01	1376	71	1	23	0.098	0.321	0.588
		02	1376	48	1	22	0.275	0.511	0.694
	DIC-C2DH-HeLa	01	84	38	10	18	0.774	0.861	0.920
		02	84	32	10	17	0.848	0.899	0.933
	Fluo-C2DL-Huh7†	01	30	34	31	32	-	-	-
		02	30	69	52	58	-	-	-
	Fluo-C2DL-MSC	01	48	15	9	8	0.709	0.814	0.870
		02	48	10	5	3	0.031	0.293	0.594
	Fluo-N2DH-GOWT1	01	92	28	23	20	0.927	0.947	0.966
		02	92	58	25	28	0.909	0.939	0.961
	Fluo-N2DH-SIM+	01	65	95	30	45	0.863	0.892	0.922
		02	150	107	8	54	0.841	0.874	0.909
	Fluo-N2DL-HeLa	01	92	265	43	137	0.721	0.834	0.906
		02	92	674	125	363	0.759	0.853	0.911
	PhC-C2DH-U373	01	115	8	6	7	0.875	0.914	0.949
		02	115	12	6	5	0.881	0.915	0.945
	PhC-C2DL-PSC	01	300	1370	74	661	0.862	0.908	0.943
		02	300	1025	66	498	0.875	0.916	0.946
3D Datasets	Fluo-C3DH-A549	01	30	1	1	1	0.933	0.939	0.949
		02	30	1	1	1	0.810	0.871	0.896
	Fluo-C3DH-H157	01	60	4	4	4	0.855	0.890	0.938
		02	60	17	3	4	0.331	0.660	0.918
	Fluo-C3DL-MDA231	01	12	33	31	29	0.482	0.629	0.734
		02	12	61	50	44	0.526	0.667	0.761
	Fluo-N3DH-CE	01	195	720	4	362	0.624	0.718	0.799
		02	190	724	2	360	0.571	0.675	0.763
	Fluo-N3DH-CHO	01	92	27	10	14	0.837	0.880	0.919
		02	92	24	4	11	0.851	0.892	0.924
	Fluo-N3DH-SIM+	01	150	81	6	43	0.826	0.853	0.877
		02	80	117	30	55	0.828	0.853	0.880

Table 5.1.: Cell statistics of the selected CTC datasets. Datasets marked with † have no ST provided, and hence no overlap percentiles between cells were calculated as the annotations are sparse. Datasets in bold have high cell count – more than 100 cells in the last frame, whereas underlined datasets have high cell motility – datasets with more than 10 cells in the last frame and an overlap of the fastest 10% of cells of less than 0.5.

that perform on at least one of the 16 selected datasets, shown in Figure 5.1, in the top 3 participants on the CTB.

The majority of top 3 performing MOT methods use tracking-by-detection approaches that combine a deep learning based segmentation approach with a subsequent, not deep learning based tracking step. Besides the graph-based tracking proposed in Chapter 3 that is combined with a deep learning based segmentation – approach KIT-GE (3), also DESU-US, KIT-GE (2) and UCSB-US use a graph-based tracking step. However, many methods use a simple overlap-based or nearest neighbors based tracking step – BGU-UL, CUNI-CZ, FR-GE(2), FR-GE(3), HIT-CN, and MU-CZ. The approaches CAS-CN, TUG-AT, USYD-AU, and UVA-NL use deep learning in the tracking step. However, only TUG-AT and USYD-AU predict segmentation and tracking in a single neural network, similar to the deep learning based instance segmentation and tracking method which was proposed in Chapter 4 – named KIT-GE (4) in the CTC.

Algorithm	Segmentation	Tracking
KIT-GE (3) [177, 200]	U-Net with two decoders predicting cell and neighbor distances; watershed-based post-processing	**graph-based tracking from Chapter 3**
KIT-GE (4) [182, 201]	**deep learning based instance segmentation and tracking method of Chapter 4**	
AC	anonymous contribution; no description available	
BGU-IL [202, 203]	U-Net with convolutional long short term memory (LSTM) blocks	overlap-based tracking
CAS-CN [204]	U-Net to predict center regions and masks; watershed-based post-processing	Siamese network for cell tracking
CUNI-CZ [205]	traditional image processing using k-means threshold selection algorithm	nearest neighbors
CVUT-CZ [206]	J-Net predicting distance to cell boundary and foreground-background prediction; partitioning algorithm to assign foreground pixels to instances	Markov chain Monte Carlo
DESU-US [83, 207]	segmenting cells in spatio-temporal domain using a sequence of traditional image processing methods	model tracking as graph-partitioning problem
DREX-US [208]	unsupervised segmentation	multi-temporal association tracking
FR-GE (2) [209]	U-Net	overlap-based tracking
FR-GE (3) [210]	U-Net	overlap-based tracking
HIT-CN [211]	two U-Nets one predicting cell masks, one predicting cell centroids	overlap-based tracking
KIT-GE (2) [57, 212]	dual U-Net predicting cell and neighbor distances, watershed-based post-processing	graph-based tracking
KTH-SE [85, 213]	distinct traditional image segmentation methods for each dataset	tracking based on the Viterbi algorithm

Algorithm	Segmentation	Tracking
LEID-NL [214, 215]	joint cell segmentation and tracking using level-sets	
MU-CZ [96]	modified U-Net predicting markers and cell boundaries, watershed-based post-processing	overlap-based tracking
MU-US (2) [216, 217]	HRNet predicting centroids and cell segmentation masks, split cells in post-processing using morphological operators	linear assignment based on IoU score, additional gating and Kalman filter modules to improve robustness
MU-US (3) [217, 218]	modified U-Net predicting markers and binary cell masks	linear assignment based on IoU score, additional gating and Kalman filter modules to improve robustness
ND-US [219]	Fully Convolutional Network (FCN) model	tracking based on earth mover's distance
PURD-US [220, 221]	U-Net; apply connected component analysis to generate instance segmentation	volumetric particle tracking velocimetry
RWTH-GE [222]	U-Net for cell centroid detection; watershed-based post-processing	nearest neighbors tracking based on detected seeds; with additional heuristics to handle redundant detections
TUG-AT [89, 223]	stacked hourglass neural network with recurrent units; predict unique pixel-wise embeddings for each instance for joint segmentation and tracking	
UCSB-US [224]	centroid prediction with watershed-based post-processing; additional boundary correction using Simple Linear Iterative Clustering (SLIC)	graph-based tracking
USYD-AU [225]	joint cell segmentation and tracking by extending a Mask R-CNN with tracking branch based on spatial and visual features predicting pairwise similarity scores	
UVA-NL [226]	same segmentation as MU-CZ	Siamese tracking using SiamFC tracker, re-segmentation if a collision of multiple cells detected

Table 5.2.: Overview of submitted cell segmentation and tracking approaches. Short description of the submitted cell segmentation and tracking algorithms that reached at least one top 3 performance on one of the 16 selected datasets.

5.3. Comparison

In the following, the proposed MOT approaches from Chapter 3 and Chapter 4 are compared based on the defined criteria of Section 5.1.

5.3.1. Setup of the Proposed Methods for Evaluation on the CTB

First, the setup of the proposed methods for submission to the CTB is introduced.

Graph-based Tracking

The graph-based tracking approach from Chapter 3 is combined with the deep learning based instance segmentation of Scherr *et al.* [57, 227]. The instance segmentation approach uses a U-Net with two decoder paths and predicts distances to the cell border – cell distance maps – and to the borders of neighboring cells –neighbor distance maps. A detailed description of the setup of the instance segmentation is given in [200, 227].

Selected Datasets All 16 datasets from Figure 5.1 are selected for evaluation.

Training For each dataset, an instance segmentation model is trained. For training, image crops of size 320x320 are generated from the GT and ST annotations, where 80% are used for training and 20% for validation. As augmentations during training flipping, rotation, scaling, contrast adaption, blurring, and noise are applied. The maximum number of epochs ranges between 200 and 560 and depends on the number of training image crops – more crops result in a higher maximum number of training epochs. In addition, early stopping and a learning rate scheduler are used. Depending on the dataset the Ranger optimizer [228, 229] or Adam optimizer [185] are used. Several models are trained, where the model with the highest $\mathrm{OP}_{\mathrm{CSB}}$, the average between SEG and DET of the GT annotated data, is selected for submission to the CTB. Instance segmentation masks for 3D datasets are not predicted by processing image volumes but instead by processing the 3D image volume slice-wise, which results in a similar processing as for 2D datasets.

For the tracking approach, no training is required.

Inference To predict instance segmentation masks, first images are normalized to range $[-1, 1]$. In addition, an optional downsampling and contrast adaption with CLAHE can be applied. Then, the pre-processed images are forwarded to the CNN, where 3D images are predicted slice-wise. To generate the instance segmentation masks, the predicted cell distance map and neighbor distance map are processed using a watershed-based post-processing step. For the post-processing step, the thresholds for the cell distance and neighbor distance map can be defined manually.

Then, the instance segmentation masks and raw images are forwarded to the graph-based tracking to link the segmentation masks over time and correct segmentation errors. Finally, for datasets that have specified a field of interest, objects outside the specified Field of Interest (FOI) are removed.

Parametrization The segmentation approach of Scherr *et al.* uses a dataset-specific selection of the number of training samples. Moreover, two thresholds for post-processing of the segmentation are defined which are set to the same values for all datasets.

For tracking the same parametrization of the two manually tunable parameters is used for all datasets: Δt is set to 3 and the default ROI size is set to twice the size of the average segmentation mask, which is computed from the predicted segmentation masks.

EmbedTrack

In the following, the training and inference procedure of the joint object segmentation and tracking approach from Chapter 4 for evaluation on the CTB is described.

Selected Datasets The proposed method requires fully annotated datasets concerning segmentation masks and links over time. However, only Fluo-N2DH-SIM+ has a fully annotated GT as it is a synthetic dataset. Thus, all 2D datasets that provide a ST or fully annotated GT are selected, resulting in a total of nine datasets.

Training For the eight 2D datasets with ST annotations, the segmentation masks from the ST are merged with the point-wise annotations of the tracking GT to create fully labeled training data with reasonable annotation quality. To train EmbedTrack, the training sequences 01 and 02 are split, where the first 90% of each image sequence are used for training and the last 10% of each image sequence are used for evaluation. For submission to the CTB, the model with the best Intersection over Union (IoU) score on the evaluation dataset is selected. During training, overlapping crops of size 256x256 (512x512 for Fluo-C2DL-MSC) are generated. The following augmentations are used during training: CLAHE, blur, rotation by multiple of 90 degrees, flipping, and shifts between successive crops to simulate larger cell movement. The augmentation is conducted similar as described in Section 4.3.1 for the BF-C2DL-HSC dataset. Moreover, each image crop is normalized to range $[0, 1]$ using the 1% and 99% percentiles per image. Each model is trained for 15 epochs using the Adam optimizer [185] with a learning rate $5 \cdot 10^{-4}$ and a one-cycle learning rate scheduler.

Inference For inference, overlapping crops of size 256x256 (512x512 for Fluo-C2DL-MSC) are generated from each image and pairs of image crops referring to the same part of the image at t and $t - 1$ are forwarded to the model for inference. Each image crop is normalized to range $[0, 1]$ using the 1% and 99% percentiles of the image crop. As test time augmentation, rotation by multiple of 90 degrees, flipping, and min-max normalization of each crop to range $[0, 1]$ using the 1% and 99% percentiles per image is applied, resulting in eight pairs of augmented image crops. Each pair of augmented image crops is forwarded to the model. Then, the predictions are flipped and rotated to undo the augmentation. Next, the average over the respective eight predictions is calculated for time point t and $t - 1$ respectively, which yields the final predictions for the two image crops. The predicted crops are stitched to a prediction covering the entire image. Next, the clustering step, which was introduced in Section 4.2.3, generates the instance segmentation on the segmentation predictions of the entire image. Finally, instances are linked to tracks by processing the tracking offsets and the instance segmentation masks as explained in the tracking step of EmbedTrack in Section 4.2.4. For the datasets that have a FOI specified, objects outside the FOI are removed.

Parametrization Apart from the different crop size used for Fluo-C2DL-MSC, for all datasets, the same parametrization is chosen concerning learning rates, the weighting of loss parts, etc. All other parameters are calculated automatically for each dataset based on the training data properties – for instance the minimum mask size which is used in the clustering step to filter out FPs.

5.3.2. Tracking Quality

First, the two proposed MOT approaches are evaluated concerning their tracking quality compared to the other competing methods.

Graph-based Tracking

We participated as team KIT-GE (3) on the CTB in 2021, the results are shown in Table 5.3. The approach ranks on 11 out of 16 datasets within the top 3 performing algorithms on at least one evaluation metric. Moreover, on five 2D datasets – BF-C2DL-HSC, BF-C2DL-MuSC, Fluo-C2DL-Huh7, Fluo-N2DL-HeLa, and PhC-C2DL-PSC – as well as on two 3D datasets – Fluo-C3DH-A549 and Fluo-C3DL-MDA231 – the approach ranked first on at least one evaluation metric. Concerning TRA performance the approach performs well on datasets with high cell motility – BF-C2DL-HSC, BF-C2DL-MuSC, and Fluo-C3DL-MDA231 as well as on datasets with high cell count Fluo-N2DL-HeLa, PhC-C2DL-PSC, and Fluo-C3DH-CE.

However, on the datasets DIC-C2DH-HeLa, Fluo-N2DH-GOWT1, PhC-C2DH-U373, Fluo-N2DH-SIM+, and Fluo-N3DH-CHO the approach performs worse. Potential reasons are over-segmentation, for instance on DIC-C2DH-HeLa, and cells with very low contrast resulting in FNs during instance segmentation, for instance on Fluo-N2DH-GOWT1.

Concerning overall top 3 performances on the examined datasets, KTH-SE, a traditional image segmentation that is adapted for each dataset individually combined with tracking based on the Viterbi Algorithm, performs best – 13 out of 16 datasets. The proposed method ranks second best concerning the number of top 3 performances – 11 out of 16 datasets. MU-CZ, a modified U-Net for segmentation combined with overlap-based tracking, ranks third on the examined datasets – 5 out of 16 datasets with a top 3 performance. While the approach of KTH-SE performs well on the CTB, applying the tracking method to other datasets, as done in Section 2.2, performed poorly – even when provided with perfect segmentation data. A potential explanation for the very distinctive performance is the possibility to tune the parametrization heavily for the CTB using expert knowledge. In contrast, using the same parametrization of the two manually tunable parameters of the graph-based tracking performed well when combined with different segmentation approaches as shown in Section 3.3 and Section 4.3, as well as when combined with a deep learning based instance segmentation for the CTB. Thus, the proposed graph-based tracking algorithm is a competitive baseline tracker, which can perform well on different types of data. Moreover, the combination of deep learning based instance segmentation and graph-based tracking performs well on a diverse set of data covering different imaging modalities and cell lines and performs well in 2D and 3D.

Dataset	Rank	OP_{CTB}	SEG	TRA
BF-C2DL-HSC	1st	**0.901 (1st)**	**0.818 (1st)**	**0.984 (1st)**
	2nd	0.868	0.757	0.978
	3rd	0.843	0.750	0.964
BF-C2DL-MuSC	1st	**0.870 (1st)**	**0.774 (1st)**	0.971
	2nd	0.849	0.742	0.967
	3rd	0.835	0.703	**0.966 (3rd)**
DIC-C2DH-HeLa	1st	0.909	0.863	0.955
	2nd	0.904	0.853	0.954
	3rd	0.898	0.852	0.954
		0.848 (9th)	**0.778 (10th)**	**0.918 (8th)**
Fluo-C2DL-Huh7	1st	**0.875 (1st)**	**0.791 (1st)**	**0.960 (1st)**
	2nd	0.843	0.751	0.934
	3rd	0.772	0.690	0.865
Fluo-C2DL-MSC	1st	0.759	0.645	0.873
	2nd	0.740	0.641	0.839
	3rd	**0.683 (3rd)**	0.619	0.788
			0.617 (4 th)	**0.749 (4th)**
Fluo-C3DH-A549	1st	**0.925 (1st)**	**0.849 (1st)**	
	2nd	0.916	0.832	1.000
	3rd	0.913	0.829	
Fluo-C3DH-H157	1st	0.938	0.888	0.987
	2nd	**0.929 (2nd)**	0.884	**0.980 (2nd)**
	3rd	0.898	**0.878 (3rd)**	0.976
Fluo-C3DL-MDA231	1st	**0.797 (1st)**	**0.710 (1st)**	**0.884 (1st)**
	2nd	0.761	0.642	0.882
	3rd	0.757	0.632	0.880
Fluo-N2DH-GOWT1	1st	0.951	0.931	0.979
	2nd	0.939	0.927	0.976
	3rd	0.934	0.927	0.967
		0.894 (14th)	**0.850 (18th)**	**0.938 (10th)**
Fluo-N2DL-HeLa	1st	0.956	0.923	**0.993 (1st)**
	2nd	0.953	0.922	0.991
	3rd	0.953	0.919	0.991
		0.938 (12th)	0.883 (12th)	
Fluo-N3DH-CE	1st	0.808	0.729	0.975
	2nd	0.803	0.705	0.945
	3rd	0.803	0.662	**0.901 (3rd)**
		0.772 (6th)	**0.642 (4th)**	
Fluo-N3DH-CHO	1st	0.926	0.917	0.953
	2nd	0.912	0.902	0.948
	3rd	0.911	0.899	0.935
		0.869 (7th)	**0.833 (10th)**	**0.906 (10th)**

Dataset	**Rank**	**OP_{CTB}**	**SEG**	**TRA**
PhC-C2DH-U373	1st	0.954	0.924	0.985
	2nd	0.951	0.923	0.982
	3rd	0.948	0.923	0.981
		0.925 (12th)	**0.876 (13th)**	**0.975 (11th)**
PhC-C2DL-PSC	1st	**0.855 (1st)**	**0.743 (1st)**	**0.967 (1st)**
	2nd	0.843	0.720	0.966
	3rd	0.836	0.715	0.959
Fluo-N2DH-SIM+	1st	0.896	0.825	0.978
	2nd	0.896	0.822	0.975
	3rd	0.894	0.821	0.973
		0.881 (7th)	**0.801 (9th)**	**0.962 (10th)**
Fluo-N3DH-SIM+	1st	0.897	0.820	0.974
	2nd	**0.865 (2nd)**	**0.759 (2nd)**	**0.972 (2nd)**
	3rd	0.848	0.746	0.967

Table 5.3.: Results of the Cell Tracking Benchmark (CTB) – status 22.10.2021. We participated as team KIT-GE (3) on the CTB `http://celltrackingchallenge.net/latest-ctb-results/`. Per dataset, we compare our performance including achieved rank on the benchmark in brackets – marked in blue and bold, against the top 3 performing CTB submissions – color-coded by participating team see Table 5.2. Datasets in bold highlight a top 3 performance on at least one metric of the proposed method.

EmbedTrack

We participate as team KIT-GE (4) on the CTB in 2022, the results are shown in Table 5.4. The approach ranked on 7 out of 9 datasets within the top 3 performing algorithms on at least one evaluation metric. Moreover, on four 2D datasets – BF-C2DL-HSC, BF-C2DL-MuSC, Fluo-N2DH-SIM+, and PhC-C2DL-PSC – the approach ranked first on at least one evaluation metric.

Dataset	**Rank**	**OP_{CTB}**	**SEG**	**TRA**
BF-C2DL-HSC	1st	**0.906 (1st)**	**0.826 (1st)**	**0.985 (1st)**
	2nd	0.901	0.818	0.984
	3rd	0.868	0.757	0.978
BF-C2DL-MuSC	1st	**0.878 (1st)**	**0.782 (1st)**	**0.974 (1st)**
	2nd	0.870	0.774	0.971
	3rd	0.849	0.742	0.967
DIC-C2DH-HeLa	1st	0.909	0.863	0.955
	2nd	0.904	0.853	0.954
	3rd	0.898	0.852	0.954
		0.879 (6th)	**0.824 (6th)**	**0.934 (6th)**
Fluo-C2DL-MSC	1st	0.759	0.645	0.873
	2nd	0.740	0.645	0.839
	3rd	0.701	0.641	0.788
		0.636 (9th)	**0.579 (9th)**	**0.693 (11th)**

Dataset	Rank	OP_{CTB}	SEG	TRA
Fluo-N2DH-GOWT1	1st	0.951	0.931	0.979
	2nd	**0.940 (2nd)**	**0.929 (2nd)**	0.976
	3rd	0.939	0.927	0.967
				0.951 (4th)
Fluo-N2DL-HeLa	1st	0.956	0.923	0.993
	2nd	0.953	0.922	**0.992 (2nd)**
	3rd	0.953	0.919	0.991
		0.949 (4th)	**0.906 (4th)**	
PhC-C2DH-U373	1st	0.954	0.924	0.985
	2nd	**0.951 (2nd)**	0.923	**0.982 (2nd)**
	3rd	0.951	0.923	0.982
			0.920 (8th)	
PhC-C2DL-PSC	1st	0.855	0.743	**0.986 (1st)**
	2nd	**0.854 (2nd)**	**0.740 (2nd)**	0.967
	3rd	0.843	0.720	0.966
Fluo-N2DH-SIM+	1st	**0.905 (1st)**	**0.830 (1st)**	**0.979 (1st)**
	2nd	0.896	0.825	0.978
	3rd	0.896	0.822	0.975

Table 5.4.: Results of the Cell Tracking Benchmark (CTB) – status 14.03.2022. We participated as team KIT-GE (4) on the CTB `http://celltrackingchallenge.net/latest-ctb-results/`. Per dataset, we compare our performance including achieved rank on the benchmark in brackets – marked in purple and bold, against the top 3 performing CTB submissions – color-coded by participating team see Table 5.2. Datasets in bold highlight a top 3 performance on at least one metric of the proposed method.

The approach performs well on datasets with high cell count – BF-C2DL-HSC, Fluo-N2DL-HeLa, and PhC-C2DL-PSC – as well as on datasets with high cell motility – BF-C2DL-HSC and BF-C2DL-MuSC. However, on the datasets DIC-C2DH-HeLa and Fluo-C2DL-MSC the approach performs worse. The dataset Fluo-C2DL-MSC contains very elongated cells with long extensions, which the approach fails to correctly segment, resulting in missing segmentation of the extensions and sometimes over-segmentation, whereas on the DIC-C2DH-HeLa dataset also over-segmentation occurs.

On the examined datasets, the proposed method performs best concerning top 3 performances – 7 out of 9 datasets. KIT-GE (3), KTH-SE, and MU-US rank second best, all three ranking on 4 out of 9 datasets within the top 3 in at least one metric. Compared to KIT-GE (3), the combination of deep learning based segmentation and the proposed graph-based tracking, the proposed simultaneous instance segmentation and tracking approach performs considerably better on Fluo-N2DH-GOWT1, PhC-C2DH-U373, and Fluo-N2DH-SIM+. However, KIT-GE (4) performs worse on Fluo-C2DL-MSC. Interestingly both approaches, KIT-GE (3) and KIT-GE (4) perform less well on DIC-C2DH-HeLa which contains low contrast, touching cells with high heterogeneity within the cells.

Although the same manual parametrization for training and inference was used for all datasets – apart from the crop size for Fluo-C2DL-MSC – the approach performed well on a diverse set of data outperforming other approaches, which use substantial fine-tuning per dataset.

5.3.3. Scalability

Next, the two proposed methods are analyzed concerning their scalability. Therefore, four datasets from 2D and 3D with few and many cells per image frame are selected. As 2D datasets, Fluo-N2DH-SIM+ and PhC-C2DL-PSC are selected and as 3D datasets, Fluo-N3DH-CE and Fluo-N3DH-SIM+ are chosen. PhC-C2DL-PSC and Fluo-N3DH-CE are selected as datasets with high cell counts, whereas Fluo-N2DH-SIM+ and Fluo-N3DH-SIM+ are selected as datasets with few cell counts. Since the approaches are evaluated on a challenge sequence, no ground truth is publicly available. Hence, the number of cells was estimated manually. PhC-C2DL-PSC contains over 70 cells in the first frame up to a few hundred cells in the last image frame, whereas Fluo-N3DH-CE has few cells in the first frame and due to frequent mitosis events over 100 cells in the last image frame. In contrast, Fluo-N2DH-SIM+ and Fluo-N3DH-SIM+ contain between 30 and 60 cells per image frame. The respective challenge sequences 01 contain 110 frames for Fluo-N2DH-SIM+, 150 frames for Fluo-N3DH-SIM+, 190 frames for Fluo-N3DH-CE, and 300 frames for PhC-C2DL-PSC. Per dataset, each approach is evaluated five times on the respective challenge sequence 01 and the average runtime is calculated over the runs. To compare the graph-based tracker to other tracking-only approaches, tracking methods of other CTB submissions[2], which were introduced in Section 2.2, are chosen. The approach of KTH-SE is excluded from runtime comparison as it only runs under Windows. All tracking-only methods are combined with the instance segmentation approach of Scherr *et al.* [57, 212], where the combination of the proposed graph-based tracking from Chapter 3 with the instance segmentation approach of Scherr *et al.* is the KIT-GE (3) approach.

For evaluation, a system with Ubuntu 18.04, an Intel i9 99000k, 32 GB RAM, and two Titan RTX with 24 GB VRAM each is used. All approaches are implemented in Python, where the proposed method EmbedTrack, as well as the instance segmentation of Scherr *et al.* [57, 212], use PyTorch as deep learning framework.

Table 5.5 shows the average runtimes of evaluating the different approaches on the challenge sequence 01 of each dataset. The runtimes include all steps from loading the images from disk to saving the predicted and tracked instance segmentation masks to disk. An executable is used for the instance segmentation of Scherr *et al.*, hence the segmentation runtime also includes any overhead of running the executable. For the tracking-only approaches, the runtime of segmentation and tracking is provided separately, whereas for EmbedTrack the runtime includes segmentation and tracking as the model predicts segmentation and tracking simultaneously.

For a few dozen cells in 2D datasets – Fluo-N2DH-SIM+ – the approach of KIT-Sch-GE (1) is the fastest. However, the approach does not scale well to tracking a few hundred cells in 2D – PhC-C2DL-PSC. While the overlap-based tracking of MU-Lux-CZ is fast in 2D it has issues tracking cells in 3D requiring three to four times longer than KIT-Sch-GE (1)

[2] The naming convention for the submissions to the Cell Tracking Challenge changed. The mapping of old and new submission names is provided in Table A.1

and the proposed graph-based tracking. The proposed EmbedTrack approach requires more time to segment and track a few dozen cells in 2D, however, for a few hundred cells the runtime of EmbedTrack is comparable to combining the segmentation approach of Scherr *et al.* with overlap-based tracking. However, since EmbedTrack is a 2D method, no runtimes on 3D datasets are reported.

Task		2D Datasets		3D Datasets	
		Fluo-N2DH-SIM+	PhC-C2DL-PSC	Fluo-N3DH-CE	Fluo-N3DH-SIM+
Segment	Scherr *et al.* [57, 212]	51.4 s	78.9 s	1098.2 s	3039.1 s
Track	KIT-Sch-GE (1)	26.0 s (77.4 s)	7642.4 s (7721.3 s)	656.1 s (1754.3 s)	597.4 s (3633.5 s)
Track	MU-Lux-CZ	33.6 s (85.0 s)	353.9 s (432.8 s)	1965.4 s (3063.6 s)	2466.9 s (5506.0 s)
Track	Graph-based Tracking (Chapter 3)	37.9 s (89.3 s)	462.5 s (505.4 s)	482.4 s (1580.6 s)	640.9 s (3680.0 s)
Segment & Track	EmbedTrack (Chapter 4)	147.2 s	451.6 s	–	–

Table 5.5.: Runtimes of different MOT approaches on the CTC. Per dataset, each approach was run five times on the challenge sequence 01 and the average runtime in seconds was calculated. For the approaches graph-based tracking, KIT-Sch-GE (1), and MU-Lux-CZ the total runtime, adding segmentation and tracking times, is shown in brackets (·).

5.3.4. Additional Criteria for Evaluation

Finally, the two in this thesis proposed MOT methods are compared concerning additional comparison criteria, which were introduced in Section 5.3.4, with the approaches MU-Lux-CZ, KTH-SE, and KIT-Sch-GE (1). For the methods MU-Lux-CZ, KTH-SE, KIT-Sch-GE (1), and the graph-based tracking only the tracking is considered, which can be combined with a segmentation approach, whereas for EmbedTrack segmentation and tracking are considered jointly.

Manual Tuning Required

- **MU-Lux-CZ**: One tunable parameter; good performance with the same parametrization when combined with high-quality segmentation data on new dataset (Section 2.2)
- **KTH-SE**: Many parameters that require careful tuning to the specific dataset and segmentation method; poor performance with the same parametrization when combined with high-quality segmentation data on new dataset (Section 2.2)
- **KIT-Sch-GE (1)**: Two manually tunable parameters; good performance with same parametrization when combined with high-quality segmentation data on new dataset (Section 2.2)

- **Graph-based Tracking**: Two manually tunable parameters; using the same parametrization performs well on a wide set of 2D and 3D datasets (Table 5.3) as well as when combined with synthetically degraded segmentation data (Figure 3.12)
- **EmbedTrack**: Same parametrization, apart from crop size for Fluo-C2DL-MSC, that performs well on a wide set of 2D datasets (Table 5.4, Section 4.3)

Labeled Data Required

- **MU-Lux-CZ, KTH-SE, KIT-Sch-GE (1), Graph-based Tracking**: Tracking itself requires no training step with labeled data, however, instance segmentation might require labeled data; depending on the domain pre-trained instance segmentation approaches [55, 56] available
- **EmbedTrack**: Full segmentation and tracking annotations are required for training; at present however very few datasets in the microscopy domain that contain segmentation and tracking labels

Robustness

- **MU-Lux-CZ**: No segmentation error correction capabilities, hence decreasing performance with degraded segmentation quality (Figure 3.12)
- **KTH-SE**: Although in theory segmentation error correction capabilities, poor performance when applied to new dataset and degraded segmentation quality (Figure 3.12)
- **KIT-Sch-GE (1)**: FN correction capabilities, better performance than MU-Lux-CZ when combined with degraded segmentation quality (Figure 3.12)
- **Graph-based Tracking**: Automatic correction of segmentation errors FN, under- and over-segmentation, outperforming other methods on degraded segmentation data (Figure 3.12); negligible changes in performance when slightly increasing or decreasing the two manually tunable parameters (Figure 3.10, Figure 3.9)
- **EmbedTrack**: No segmentation error correction capabilities; same training and inference setup, apart from crop size selection performs well for a wide set of 2D datasets (Table 5.4, Section 4.3); handling of lower temporal resolution possible without retraining (Section 4.3); however issues when segmenting very elongated cells with long extensions, such as Fluo-C2DL-MSC (Figure 4.3)

Applicability to 2D and 3D Datasets

- **MU-Lux-CZ, KTH-SE, KIT-Sch-GE (1), Graph-based Tracking**: Applicable to 2D and 3D datasets given a respective 2D or 3D instance segmentation of the dataset
- **EmbedTrack**: Applicable to 2D datasets, however, extension of the method to 3D possible

Additional Requirements

- **MU-Lux-CZ**: No additional requirements; while the tracking does not require a GPU predicting instance segmentation masks with a deep learning based approach requires a GPU
- **KTH-SE**: Matlab required; while the tracking does not require a GPU predicting instance segmentation masks with a deep learning based approach requires a GPU
- **KIT-Sch-GE (1)**: Solver for mixed integer linear programs required such as GLPK [118]; while the tracking does not require a GPU predicting instance segmentation masks with a deep learning based approach requires a GPU
- **Graph-based Tracking**: Solver for mixed integer linear programs required, here Gurobi [116] used due to fast runtime which requires a software license; while the tracking does not require a GPU predicting instance segmentation masks with a deep learning based approach requires a GPU
- **EmbedTrack**: No additional licenses needed; however part of the code is licensed as non-commercial use; GPU for training and prediction required

5.4. Discussion

In this chapter, the proposed graph-based tracking approach from Chapter 3 and the deep learning based instance segmentation and tracking, EmbedTrack, from Chapter 4 were compared with other MOT approaches. Therefore, comparison criteria were defined and the two approaches were submitted to the CTB to evaluate their performance on a wide variety of datasets.

The graph-based tracking was submitted with the same parametrization to all datasets, whereas for EmbedTrack only the crop size was adapted for Fluo-C2DL-MSC. Although both approaches were not fine-tuned to the datasets, both proposed methods performed well on a wide set of datasets. The graph-based tracking approach was combined with a deep learning based instance segmentation of Scherr *et al.* and evaluated on 16 2D and 3D datasets, where the approach performed on 11 out of 16 datasets on at least one metric within the top 3 performing submissions. The EmbedTrack was submitted on nine 2D datasets and performed on 7 out of 9 datasets within the top 3 performing submissions.

However, when segmenting very elongated cells with long extensions EmbedTrack tends to over-segment cells by segmenting extensions as additional cells. In contrast, the instance segmentation approach of Scherr *et al.* handles this cell type better. Interestingly, on datasets where the approach of Scherr *et al.* performed worse – Fluo-N2DH-GOWT1, Fluo-N2DL-HeLa, PhC-C2DL-U373, and Fluo-N2DH-SIM+ – EmbedTrack showed better segmentation quality.

In comparison with two other tracking approaches from the CTB, both proposed methods performed in a reasonable runtime. On 3D datasets, the graph-based tracking was faster than naive overlap-based tracking, whereas on 2D data with many cells EmbedTrack was faster than the graph-based tracking but slower than overlap-based tracking.

Both approaches were compared on a variety of datasets with up to several hundreds of cells per frame, however, an analysis of both approaches on large-scale datasets covering several thousands of objects per frame was not conducted. Future work could evaluate how both approaches scale on large datasets and propose adaptions to make both approaches more computationally efficient.

While annotated data for the segmentation of microscopy images become more and more available, currently datasets providing comprehensive segmentation and tracking annotations are rare, which is the main limitation for applying EmbedTrack to other image sequences. Future work could utilize the segmentation and tracking information of other approaches, as done for the ST annotations of the CTC, to train EmbedTrack, using simulated data for training, or adapting the training to sparse annotations. Moreover, training on sparse annotations would also facilitate extending EmbedTrack, which is currently limited to 2D datasets, to 3D datasets. For the graph-based tracking, in contrast, the main limitation is the need for a general-purpose solver, which can require as in the case of Gurobi, additional software licenses.

6. Conclusions and Outlook

MOT approaches allow the efficient extraction of information about the movement of objects. The extracted information provides the basis for further evaluation such as behavior analysis, path planning, or movement prediction. While MOT has numerous applications such as in autonomous driving, surveillance, or robotics, also there is a high demand for MOT approaches in biomedical applications. For instance, the processing of light microscopy image data that captures experiments with hundreds or even thousands of objects over hours or even days requires automated tracking to make analysis feasible. While MOT in light microscopy data is challenging due to many similar objects, in addition, a high tracking quality is required to discover meaningful insights. Hence, towards the long-term vision of virtually error-free object segmentation and tracking, this thesis proposed two new tracking methods as well as approaches to compare the performance of tracking methods.

Chapter 2 proposed a concept to compare different tracking methods by replacing the segmentation method in a tracking-by-detection approach with a method that yields synthetically degraded segmentation data. By using synthetically degraded segmentation data as input for the tracking it is possible to investigate how different tracking methods handle varying segmentation quality and their abilities to correct segmentation errors. In Chapter 3 a competitive baseline tracker was proposed which is applicable to 2D and 3D data when combined with an arbitrary instance segmentation approach. The tracking method models object behavior and segmentation errors in a graph, whereas prior knowledge of the shape of the tracking graph is used to correct segmentation errors. Chapter 4 proposed a neural network that learns instance segmentation and tracking simultaneously and predicts representations that are comprehensible to humans. Finally, in Chapter 5 the two proposed tracking methods were applied to a diverse benchmark dataset called the Cell Tracking Challenge. Moreover, besides tracking quality additional comparison criteria for MOT approaches were proposed which were used to compare the two proposed tracking methods.

Thus, the major contributions presented in this thesis can be summarized as:

1. A new method to benchmark tracking methods of tracking-by-detection approaches concerning their robustness towards different types of segmentation errors and their capabilities to segmentation errors (Section 2.1).

2. Application of the proposed benchmark approach to analyze tracking methods and evaluation metrics (Section 2.2).

3. A new graph-based tracking method modeling object behavior and the segmentation errors FNs as well as over- and under-segmentation of two or more objects using simple, position-based cost functions (Section 3.2.2).

4. A new post-processing method harnessing prior knowledge on the shape of the tracking graph to correct segmentation errors automatically (Section 3.2.3).
5. A new approach to resolve conflicting pixels of overlapping instance segmentation masks (Section 3.2.3).
6. The thorough validation of the functionality of the proposed graph-based tracking concerning the robustness of the parametrization, the influence of the post-processing operations, and the performance compared to other tracking methods (Section 3.3).
7. A novel deep learning approach named EmbedTrack which learns instance segmentation and tracking simultaneously in a single neural network without any recurrent parts that predicts as embeddings offsets and clustering bandwidths, which are comprehensible for humans (Section 4.2).
8. The thorough validation of EmbedTrack concerning the influence augmentations during training and inference, and the influence of the size of the training dataset and kind of training data used on the performance during inference and well as the comparison with other tracking methods (Section 4.3).
9. Application of the graph-based tracking approach to the Cell Tracking Challenge benchmark on 16 datasets in 2D and 3D using the same parametrization for all datasets resulting in 11 out 16 top 3 performances including 7 first rank performances (Section 5.3.2).
10. Application of the deep learning based tracking approach to the Cell Tracking Challenge benchmark on 9 datasets in 2D using the same training setup for all datasets resulting in 7 out of 9 top 3 performances including 4 first rank performances (Section 5.3.2).
11. Definition of additional criteria based on which tracking algorithms should be evaluated including applying the defined criteria to compare the two proposed tracking approaches (Section 5.1, Section 5.3.3, Section 5.3.4).
12. To enable further development and usability of the proposed methods, the code of all proposed methods has been made publicly available (Section 2.2.1, Section 3.3.1, Section 4.3.1).

While the benchmark proposed in Chapter 2 models real-world segmentation errors, approaches that make the appearance of these segmentation errors more realistic for instance by simulating low-contrast objects in the corresponding raw image could be explored. Also, the synthetic degradation of segmentation data could be combined with synthetic or semi-synthetic benchmark datasets [3, 168] which model different object behaviors to create more challenging benchmarks to compare tracking-only methods. Moreover, the potential of the approach for finding a good parametrization of a tracking method could be explored.

Concerning the graph-based tracking method proposed in Chapter 3, all three tracking steps can be investigated further. For the tracklet step, a linear phase correlation is used currently to estimate offsets coarsely based on image crops. This coarse movement estimation could be replaced for instance by predicting offsets over the whole image simultaneously using deep learning approaches [182]. Moreover, in the matching step very simple, position-based cost functions are used, which could be replaced by more sophisticated cost functions that for example also include the neighborhood of an object in the cost. In addition, for the post-processing step an adaption of the costs to modify the tracking graph should be investigated. Also worth exploring is the combination of domain-specific solvers [119, 120] that yield approximate solutions with the proposed method for application to large-scale tracking tasks.

The main limitation of the deep learning based instance segmentation and tracking approach is the need for fully annotated data – fully annotated instance segmentation masks and their linking over time. As the generation of fully labeled datasets is time intensive, the burden of annotation can be alleviated by adapting the training process to utilize sparsely labeled data or training on simulated data, for instance by using GANs [131, 169]. Moreover, to forward information between the encoder and decoder branches, skip connections could be explored similarly to the U-Net architecture [186]. Also, the predicted tracking offsets could be processed further to improve tracking quality for instance by using the offset predictions as inputs for a graph-based tracking method. Currently, due to the need for fully annotated data, EmbedTrack was only applied to 2D datasets as 3D datasets are expensive to annotate. However, a direction of future work is the extension to 3D data. During evaluation, the influence of the temporal resolution on the tracking quality was investigated. Decreasing the temporal resolution on the selected datasets led to an increased object movement, which however was accompanied by a large change in the appearance of the examined objects. As a result, linking corresponding objects became challenging, even for humans. Therefore, further experiments on datasets with more object movement and less change in the object appearance between image frames should be conducted.

Reliable instance segmentation and tracking pave the way to derive new insights in numerous applications. For instance, in the scope of biomedical applications virtually error-free tracks offer the possibility to investigate movements and discover movement patterns or deviations from them. This information can be the key to deepening our understanding of the processes causing illness and might provide new angles to developing cures in the future.

A. Appendix

A.1. Naming of Participating Teams on the CTC

The hosts of the CTC changed the naming of the participating teams and their submitted approaches to the CTC in June 2022.

Old Submission Name	**New Submission Name**
KIT-Sch-GE (1)	KIT-GE (2)
KTH-SE	KTH-SE
MU-Lux-CZ	MU-CZ
KIT-Sch-GE (2)	KIT-GE (3)
KIT-Loe-GE	KIT-GE (4)

Table A.1.: Team names before and after the naming convention was changed.

A.2. Comparison of Tracking Results

Table A.2 and Table A.3 provide the SEG and DET scores of the evaluation from Section 3.3.2.

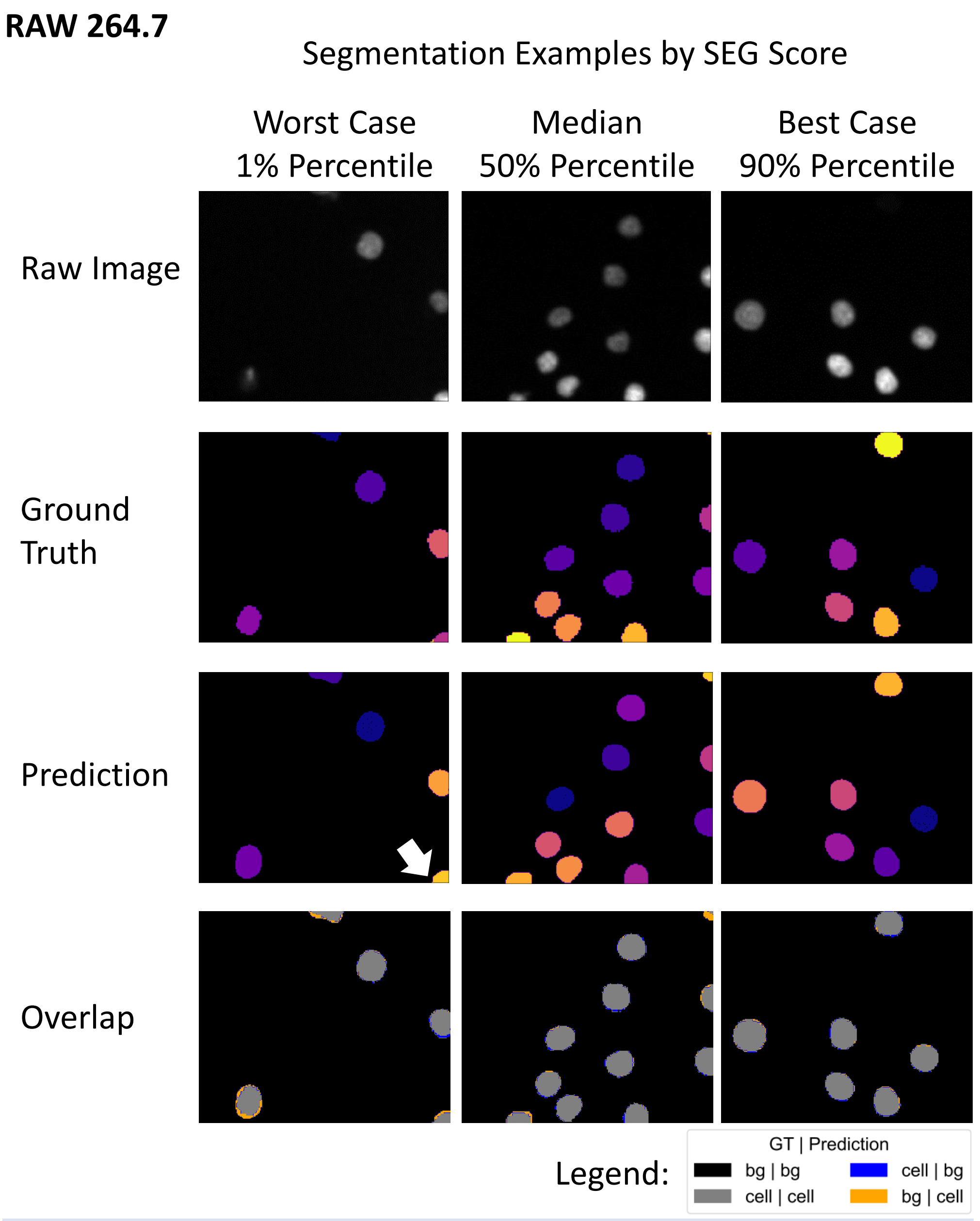
	Tracking Algorithm	**Mean Rank**					**1st Rank/%**				
		Proposed	MU-Lux-CZ	KTH-SE	KIT-Sch-GE (1)	No Tracking	Proposed	MU-Lux-CZ	KTH-SE	KIT-Sch-GE (1)	No Tracking
Segmentation Error	**Error Fraction/%**										
Ground Truth	0	1.56	**1.00**	5.00	2.20	**1.00**	80.2	**100.0**	0.0	55.6	**100.0**
Under-Segmentation	1	**1.59**	2.36	5.00	2.25	2.36	**67.2**	8.1	0.0	30.1	8.1
	2	**1.60**	2.58	5.00	2.00	2.58	**65.7**	5.9	0.0	31.4	5.9
	5	**1.49**	2.79	5.00	1.87	2.79	**69.6**	1.5	0.0	29.4	1.5
	10	**1.34**	2.92	5.00	1.80	2.92	**71.6**	0.0	0.0	28.4	0.0
	20	**1.46**	2.92	5.00	1.69	2.92	**65.2**	0.0	0.0	34.8	0.0
Over-Segmentation	1	**1.29**	1.87	5.00	3.29	1.87	**88.9**	13.3	0.0	4.4	13.3
	2	**1.14**	1.95	5.00	3.58	1.95	**94.6**	5.7	0.0	1.7	5.7
	5	**1.14**	1.95	5.00	3.88	1.95	**94.3**	5.7	0.0	0.5	5.7
	10	**1.11**	1.96	5.00	3.96	1.96	**95.1**	4.9	0.0	0.2	4.9
	20	**1.40**	1.83	5.00	3.95	1.83	**83.0**	17.0	0.0	0.0	17.0
False Negative	1	**1.45**	2.89	5.00	1.54	2.89	**62.7**	0.5	0.0	58.5	0.5
	2	**1.46**	2.97	5.00	1.49	2.97	**58.3**	0.2	0.0	53.8	0.2
	5	**1.43**	3.00	5.00	1.54	3.00	**57.0**	0.0	0.0	46.4	0.0
	10	**1.35**	3.00	5.00	1.63	3.00	**64.7**	0.0	0.0	36.5	0.0
	20	**1.17**	3.00	5.00	1.83	3.00	**83.2**	0.0	0.0	17.3	0.0
Mixed	1	**1.43**	2.55	5.00	2.18	2.55	**78.5**	9.1	0.0	22.7	9.1
	2	**1.25**	2.81	5.00	2.10	2.81	**86.7**	0.5	0.0	12.8	0.5
	5	**1.07**	2.97	5.00	1.98	2.97	**94.1**	0.2	0.0	5.7	0.2
	10	**1.04**	3.00	5.00	1.97	3.00	**96.8**	0.0	0.0	3.2	0.0
	20	**1.02**	3.00	5.00	1.99	3.00	**97.8**	0.0	0.0	2.2	0.0

Table A.2.: SEG scores of different tracking approaches on erroneous segmentation data. For each erroneous dataset all approaches are ranked based on their SEG metric, where approaches reaching the same score are assigned the same rank. The mean rank is calculated over the subset of image sequences containing a specific type and fraction of segmentation errors. In addition the fraction of times the tracking approach reached rank 1 is reported, where the fractions accumulated over the tracking approaches can be larger than 100% as several approaches can reach the same SEG score.

	Tracking Algorithm	Mean Rank					1^{st} Rank/%				
		Proposed	MU-Lux-CZ	KTH-SE	KIT-Sch-GE (1)	No Tracking	Proposed	MU-Lux-CZ	KTH-SE	KIT-Sch-GE (1)	No Tracking
Segmentation Error	**Error Fraction/%**										
Ground Truth	0	1.68	**1.00**	5.00	2.93	**1.00**	69.1	**100.0**	0.0	33.3	**100.0**
Under-Segmentation	1	**1.34**	2.30	4.96	2.79	2.30	**83.0**	10.4	0.0	22.2	10.4
	2	**1.32**	2.48	4.97	2.53	2.48	**82.0**	6.9	0.0	22.5	6.9
	5	**1.22**	2.74	4.97	2.24	2.74	**86.4**	1.7	0.0	15.6	1.7
	10	**1.13**	2.90	4.96	2.07	2.90	**89.4**	0.0	0.2	12.1	0.0
	20	**1.20**	2.93	4.93	1.96	2.93	**83.2**	0.0	1.0	16.3	0.0
Over-Segmentation	1	**1.08**	2.14	4.96	3.38	2.14	**95.8**	4.2	0.2	3.2	4.2
	2	**1.04**	2.18	4.96	3.51	2.18	**98.3**	1.7	0.7	1.0	1.7
	5	**1.01**	2.07	4.94	3.85	2.07	**98.8**	0.0	1.2	0.0	0.0
	10	**1.01**	2.05	4.91	3.98	2.05	**98.8**	0.0	1.2	0.0	0.0
	20	**1.03**	2.11	4.63	4.12	2.11	**97.5**	0.0	2.5	0.0	0.0
False Negative	1	**1.21**	2.93	4.98	1.63	2.93	**82.2**	0.5	0.0	45.2	0.5
	2	**1.21**	3.00	4.97	1.54	3.00	**78.5**	0.0	0.2	47.7	0.0
	5	**1.27**	3.01	4.95	1.54	3.01	**73.1**	0.0	0.0	45.9	0.0
	10	**1.25**	3.02	4.94	1.65	3.02	**75.1**	0.0	0.5	34.8	0.0
	20	**1.15**	3.12	4.72	1.85	3.12	**85.2**	0.0	1.2	15.3	0.0
Mixed	1	**1.17**	2.60	5.00	2.33	2.60	**89.6**	6.7	0.0	14.8	6.7
	2	**1.10**	2.87	4.95	2.18	2.87	**92.6**	0.2	1.2	7.2	0.2
	5	**1.03**	2.99	4.96	2.02	2.99	**97.0**	0.2	0.7	2.7	0.2
	10	**1.02**	3.02	4.92	2.01	3.02	**98.3**	0.0	1.2	0.7	0.0
	20	**1.01**	3.05	4.85	2.03	3.05	**98.5**	0.0	1.2	0.2	0.0

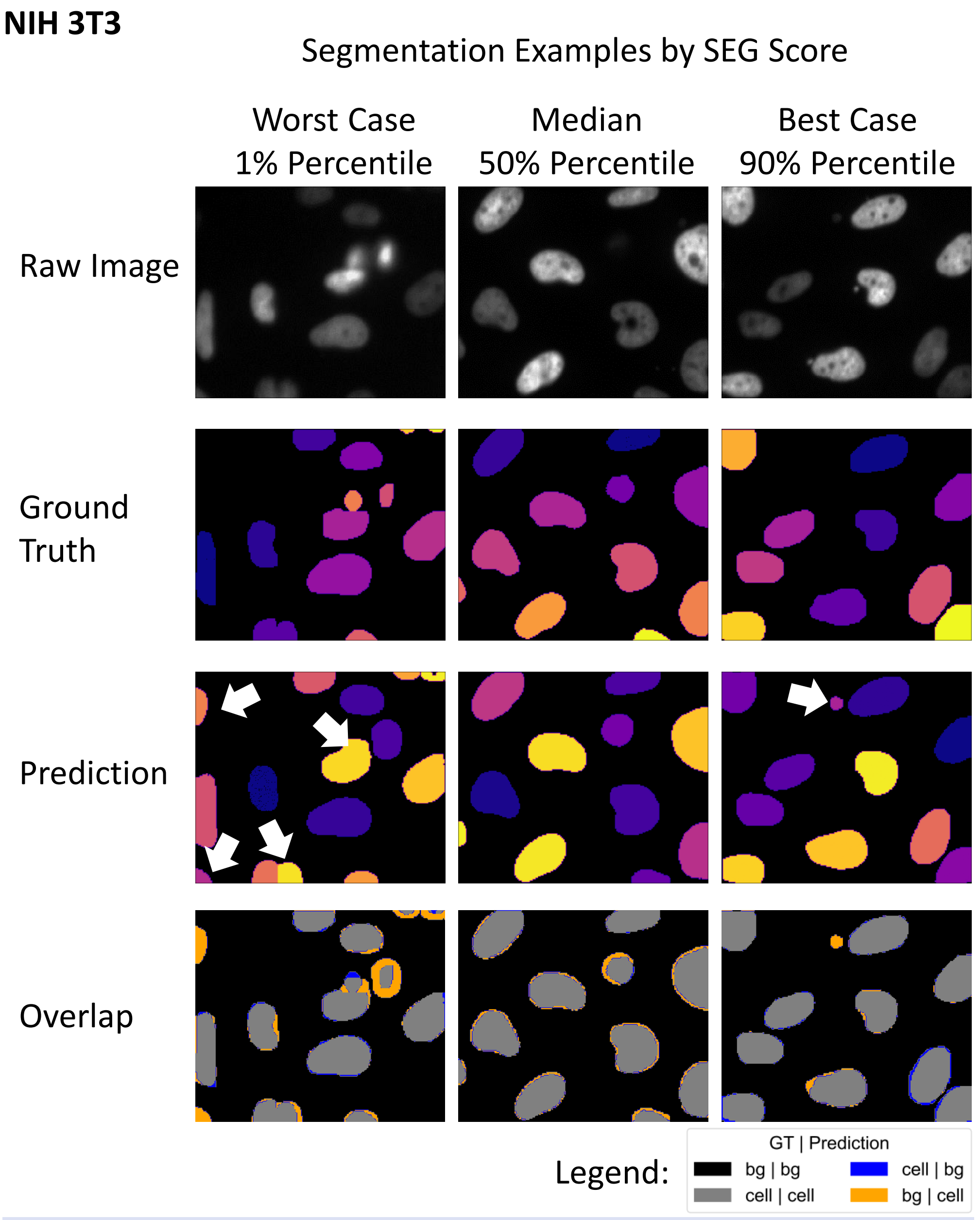
Table A.3.: DET scores of different tracking approaches on erroneous segmentation data. For each erroneous dataset all approaches are ranked based on their DET metric, where approaches reaching the same score are assigned the same rank. The mean rank is calculated over the subset of image sequences containing a specific type and fraction of segmentation errors. In addition the fraction of times the tracking approach reached rank 1 is reported, where the fractions accumulated over the tracking approaches can be larger than 100% as several approaches can reach the same DET score.

A.3. Segmentation Quality of EmbedTrack on DeepCell Benchmark Dataset



SEG Score	0.68	0.85	0.92
SEG Score (FOI corrected)	0.85	0.92	0.92

Figure A.1.: Segmentation quality based on SEG score of EmbedTrack on RAW 264.7 test dataset. The SEG score is calculated for each frame and the frames closest to the 1%, 50%, and 90% percentile without FOI correction are shown. In addition, the SEG score after applying FOI correction to the frame is shown. The segmentation errors FN, FP, over- and under-segmentation are marked with white arrows.



SEG Score	0.63	0.85	0.93
SEG Score (FOI corrected)	0.70	0.87	0.93

Figure A.2.: Segmentation quality based on SEG score of EmbedTrack on NIH 3T3 test dataset. The SEG score is calculated for each frame and the frames closest to the 1%, 50%, and 90% percentile without FOI correction are shown. In addition, the SEG score after applying FOI correction to the frame is shown. The segmentation errors FN, FP, over- and under-segmentation are marked with white arrows.

B. Nomenclature and Symbols

This section summarizes the notation used throughout this thesis, which is partly derived from [230]. First general rules of the nomenclature are listed, then the subsequent table introduces all symbols used throughout this thesis

- Lower case letters denote indexing variables (for example i, j, l), scalars (for example w), variables (for example x, y), and function names (for instance $f(\cdot), g(\cdot), c(\cdot)$)
- Bold, lower case letters denote vectors (for example $\mathbf{p}$, $\mathbf{c}$)
- Upper case letters denote constants (for example H, W, Z)
- Bold, upper case letters denote matrices and tensors (for example $\mathbf{B}$)
- Script, upper case letters denote sets (for example $\mathcal{A}$), where the size of a set is denoted as $|\cdot|$ (for example the size of $\mathcal{A}$ is $|\mathcal{A}|$)
- $||\cdot||_2$ is the Euclidean norm of a vector
- The absolute value is denoted as $|\cdot|$ (for instance $|x|$ is the absolute value of x)
- Estimates are denoted as $\hat{\cdot}$ (for example $\hat{\mathbf{B}}, \hat{\mathbf{p}}$)
- Tuples are denoted as $(\cdot, \cdot)$ (for example (u, v))
- Loss functions are denoted as a small script ℓ (for example ℓ_{seed})
- Elements in a 2D vector are referred to by their x- and y-position which is denoted as $\cdot^x$ and $\cdot^y$ (for example $\mathbf{c}^\top = [c^x, c^y]$)
- A bar over an entity denotes a mean value (for example $\bar{\mathbf{s}}_m$)
- A tilde denotes a modified entity (for example $\tilde{s}^x_m$)
- Accessing an element in a matrix or tensor at an index tuple i is denoted as $(\cdot)_i$ (for example $(\mathbf{B})_i$)

Symbol	Description
$\mathbf{1}$	Matrix containing only ones
(u, v)	Directed edge from node u to node v
$2D$	Two dimensional
$3D$	Three dimensional
a_t	Appearance node at time point t in the graph

Symbol	**Description**
$\mathbf{a}$	Image size as a vector
$\mathcal{A}_{\text{IDFN}}$	Set of Identity False Negative (IDFN) associations
$\lvert\mathcal{A}_{\text{IDFN}}\rvert$	Number of Identity False Negative (IDFN) associations
$\mathcal{A}_{\text{IDFP}}$	Set of Identity False Positive (IDFP) associations
$\lvert\mathcal{A}_{\text{IDFP}}\rvert$	Number of Identity False Positive (IDFP) associations
$\mathcal{A}_{\text{IDTP}}$	Set of Identity True Positive (IDTP) associations
$\lvert\mathcal{A}_{\text{IDTP}}\rvert$	Number of Identity True Positive (IDTP) associations
$\mathcal{A}_{\text{FNA}}$	Set of False Negative Associations (FNA)s
$\lvert\mathcal{A}_{\text{FNA}}\rvert$	Number of False Negative Associations (FNA)s
$\mathcal{A}_{\text{FPA}}$	Set of False Positive Associations (FPA)s
$\lvert\mathcal{A}_{\text{FPA}}\rvert$	Number of False Positive Associations (FPA)s
$\mathcal{A}_{\text{TPA}}$	Set of True Positive Associations (TPA)s
$\lvert\mathcal{A}_{\text{TPA}}\rvert$	Number of True Positive Associations (TPA)s
AOGM	Acyclic Oriented Graphs Matching
AOGM_0	AOGM cost to create the ground truth tracking graph
$\text{AOGM}_{\text{D},0}$	AOGM cost to create the ground truth detection graph
AOGM_{D}	AOGM metric including only detection errors
$b(\cdot)$	Balance of a node
$b_{i,t}$	Length of the diagonal of the bounding box of the segmented object m at time point t
$\mathbf{b}_{i,t}$	A bounding box point of the segmented object m at time point t
$\mathbf{B}_m$	Binary mask which is one at indices belonging to an instance segmentation mask m and 0 otherwise
$\mathbf{B}_{m,t}$	Binary mask which is one at indices belonging to an instance segmentation mask m at time point t and 0 otherwise
$\hat{\mathbf{b}}_{i,t+1}$	Propagated bounding box point of segmented object i at time point $t+1$
$\mathcal{B}_{i,t}$	Bounding box of segmented object i at time point t
$\hat{\mathbf{B}}_j$	Predicted instance segmentation mask given the pixel j
$\hat{\mathcal{B}}_{i,t+1}$	Propagated bounding box of segmented object i at time point $t+1$
$c(\cdot,\cdot)$	Edge cost function in a graph
$c(\cdot)$	Cost function in an optimization problem
$\mathbf{c}_{i,t}$	Center of instance segmentation mask i at time point t
$\hat{\mathbf{c}}_{i,t+1}$	Estimated center of instance segmentation mask i at time point $t+1$
c^x	x-dimension of the center position of an instance segmentation mask
c^y	y-dimension of the center position of an instance segmentation mask
c^x_m	x-dimension of the center position of the instance segmentation mask m
c^y_m	y-dimension of the center position of the instance segmentation mask m

Symbol	Description
CLAHE	Contrast Limited Adaptive Histogram Equalization
CNN	Convolutional Neural Network
CTB	Cell Tracking Benchmark
CSB	Cell Segmentation Benchmark
CTC	Cell Tracking Challenge
d_t	Disappearance node at time point t in the graph
d_{TP}	A True Positive (TP) detection
d^*_{TP}	A True Positive (TP) detection
$\hat{\mathbf{d}}_{i,t,t+1}$	Estimated displacement of the instance segmentation mask i between time points t and $t-1$
$d(\cdot,\cdot)$	Distance measure based on the Gaussian kernel
$\mathbf{D}$	Seediness matrix
$d^{\mathrm{S}}(\cdot,\cdot)$	Distance measure based on the Gaussian kernel between to pixel positions at the same time point
$d^{\mathrm{T}}(\cdot,\cdot)$	Distance measure based on the Gaussian kernel between an object center at the previous time point and a pixel position at the current time point
$\mathbf{D}^{\mathrm{S}}$	Distance matrix containing the distance of each shifted pixel to an object center at the same time point
$\mathbf{D}^{\mathrm{T}}$	Distance matrix containing the distance of each shifted pixel at time point t to an object center at time point $t-1$
$\mathcal{D}_{\mathrm{FN}}$	Set of False Negative (FN) detections
$\lvert\mathcal{D}_{\mathrm{FN}}\rvert$	Number of False Negative (FN) detections
$\mathcal{D}_{\mathrm{FP}}$	Set of False Positive (FP) detections
$\lvert\mathcal{D}_{\mathrm{FP}}\rvert$	Number of False Positive (FP) detections
$\mathcal{D}_{\mathrm{GT}}$	Set of detections in the Ground Truth (GT)
$\lvert\mathcal{D}_{\mathrm{GT}}\rvert$	Number of detections in the Ground Truth (GT)
$\mathcal{D}_{\mathrm{NS}}$	Set of under-segmented objects
$\lvert\mathcal{D}_{\mathrm{NS}}\rvert$	Number of under-segmented objects
$\mathcal{D}_{\mathrm{TP}}$	Set of True Positive (TP) detections
$\lvert\mathcal{D}_{\mathrm{TP}}\rvert$	Number of True Positive (TP) detections
DET	Detection metric in the Cell Tracking Challenge
$\mathbf{e}^{\mathrm{S}}_i$	Object center prediction of a pixel referenced by the index tuple i at the same time point
$e^{\mathrm{S},x}_i$	x-dimension of object center prediction of a pixel referenced by the index tuple i at the same time point
$e^{\mathrm{S,y}}_i$	y-dimension of object center prediction of a pixel referenced by the index tuple i at the same time point
$e^{\mathrm{S,x}}_j$	x-dimension of object center prediction of a pixel referenced by the index tuple j at the same time point
$e^{\mathrm{S,y}}_j$	y-dimension of object center prediction of a pixel referenced by the index tuple j at the same time point

Symbol	Description
$\mathbf{e}_i^{\mathrm{T}}$	Object center prediction of a pixel referenced by the index tuple i at time point $t-1$ for its object center at time point t
$e_i^{\mathrm{T},x}$	x-dimension of object center prediction of a pixel referenced by the index tuple i at time point $t-1$ for its object center at time point t
$e_i^{\mathrm{T},y}$	y-dimension of object center prediction of a pixel referenced by the index tuple i at time point $t-1$ for its object center at time point t
$\mathcal{E}$	Set of edges in a graph
E	Edge split cost constant
$f(\cdot,\cdot)$	flow over a directed edge
$f_{jl}(\cdot,\cdot)$	Pairwise coupled flow variable where the objects with IDs j and l are coupled
FN	False Negative
FNA	False Negative Association
FOI	Field Of Interest
FP	False Positive
FPA	False Positive Association
$g(\cdot)$	A linear function
$\mathcal{G}$	A graph
GT	Ground Truth
$h(\cdot)$	A linear function
H	Height of an image
HOTA	Higher Order Tracking Accuracy
HOTA_o	HOTA metric at localization threshold o
i	1. Run index 2. Index tuple
i'	Run index
$\mathcal{I}$	Set of all index tuples that reference all elements in a matrix or in a 2D image
$\mathcal{I}_m$	Set of index tuples referencing all pixels that belong to an instance segmentation mask m
$\mathcal{I}_m^t$	Set of index tuples referencing all pixels that belong to an instance segmentation mask m at time point t
$\lvert\mathcal{I}_m\rvert$	Number of pixels belonging to a instance segmentation mask m
$\mathcal{I}_{\mathrm{bg}}$	Set of index tuples referencing all pixels that to the background of an image
$\lvert\mathcal{I}_{\mathrm{bg}}\rvert$	Number of background pixels in an image
$\mathcal{I}_{\mathrm{pred}}$	Set of index tuples referencing all pixels that belong to the predicted instance segmentation mask
$\mathcal{I}_{\mathrm{GT}}$	Set of index tuples referencing all pixels that belong to the ground truth instance segmentation mask
IDF_1	Identification F_1 score

Symbol	Description
IDSW	Identity Switch
j'	Run index
j	1. Run index 2. Index tuple
$J(\cdot,\cdot)$	Jaccard index
k	Run index
K	A constant
$k_l(\cdot,\cdot)$	Lower capacity of an edge
$k_u(\cdot,\cdot)$	Upper capacity of an edge
l	Run index
ℓ	Loss
ℓ_{instance}	Instance loss
ℓ_{Jaccard}	Jaccard loss
$\ell_{\text{Lovász}}$	Lovász loss
ℓ_{seed}	Seed loss
ℓ_{seg}	Segmentation loss
ℓ_{track}	Tracking loss
ℓ_{var}	Variance loss
$\mathcal{L}_{\text{EA}}$	Set of missing links
$\lvert\mathcal{L}_{\text{EA}}\rvert$	Number of missing links
$\mathcal{L}_{\text{EC}}$	Set of links with wrong semantic
$\lvert\mathcal{L}_{\text{EC}}\rvert$	Number of links with wrong semantic
$\mathcal{L}_{\text{ED}}$	Set of erroneous links that need to be removed
$\lvert\mathcal{L}_{\text{ED}}\rvert$	Number of erroneous links that need to be removed
$\mathcal{L}_{\text{IDSW}}$	Set of Identity Switch (IDSW) linking errors
$\lvert\mathcal{L}_{\text{IDSW}}\rvert$	Number of Identity Switch (IDSW) linking errors
m	Instance segmentation mask index
$m_{i,t-1}$	Merge node at time point $t-1$ in the graph referring to segmented object i
M_r	Number of merged tracks where r is the set of tracks that will be merged
M_{inst}	Number of instance segmentation masks in an image
M_{inst}^t	Number of instance segmentation masks in the image at time point t
$\mathcal{M}_n$	All subsets of tracks that can be merged with the track ω_n
$\mathcal{M}(\mathcal{S}_n)$	Set containing all subsets of successor tracks of track ω_n that can be merged
$\mathcal{M}(\mathcal{P}_n)$	Set containing all subsets of predecessor tracks of track ω_n that can be merged
MOT	Multiple Object Tracking
MOTA	Multiple Object Tracking Accuracy
n	Run index
$\Delta N_{0.99}$	0.99 quantile of the number of predecessors / successor links per track in an image sequence

Symbol	Description
o	Intersection over Union (IoU) threshold
$o_{i,t}$	Object node at time point t in the graph referencing segmented object i
$\mathbf{o}_i^{\mathrm{S}}$	Segmentation offset vector of a pixel referenced by the index tuple i
$\mathbf{O}^{\mathrm{S}}$	Segmentation offset tensor
$\mathbf{o}_i^{\mathrm{T}}$	Tracking offset vector of a pixel referenced by the index tuple i
$\mathbf{O}^{\mathrm{T}}$	Tracking offset tensor
$\mathcal{O}_t$	Set of object nodes at time point t
$\|\mathcal{O}_t\|$	Number of object nodes at time point t
p	1. Predecessor track index 2. Run index
p^x	x-dimension of a position vector
p^y	y-dimension of a position vector
$\mathbf{p}$	Position vector
$\mathbf{p}_i$	Position vector of a pixel referenced by the index tuple i
$P_{n,r}$	Number of tracks of the set of mergeable tracks, index by r, that share the same predecessors as track ω_n
$\mathcal{P}_n$	Set of predecessor tracks of the track ω_n
$\|\mathcal{P}_n\|$	Number of predecessor tracks of the track ω_n
q	Index referencing a set of tracks in a set containing several sets of mergeable tracks
q^-	Source node
q^+	Sink node
$\mathbf{q}_{i,t}$	Mask point of segmented object i at time point t
$\mathcal{Q}_{i,t}$	Set of mask points of segmented object i at time point t
$\hat{\mathbf{q}}_{i,t+1}$	Propagated mask point of segmented object i at time point $t+1$
$\hat{\mathcal{Q}}_{i,t+1}$	Propagated set of mask points of segmented object i at time point $t+1$
r	Index referencing a set of tracks in a set containing several sets of mergeable tracks
ROI	Region Of Interest
s	1. Successor track index 2. Run index
$s_{i,t+1}$	Split node at time point $t+1$ in the graph referring to segmented object i
$S_{q,n}$	Number of tracks of the set of mergeable tracks, index by q, that share the same successors as track ω_n
S	Clustering bandwidths tensor
$\mathbf{s}_i$	Clustering bandwidth vector of the pixel referenced by the index tuple i
$\mathcal{S}_n$	Set of successor tracks of the track ω_n
$\|\mathcal{S}_n\|$	Number of successor tracks of the track ω_n

Symbol	Description
$\bar{\mathbf{s}}_m$	Mean clustering bandwidth vector of instance segmentation mask m
$\tilde{\mathbf{s}}_m$	Scaled clustering bandwidth vector of instance segmentation mask m
$\tilde{\mathbf{s}}_j^{\text{smooth}}$	Scaled, smoothed clustering bandwidth vector of a pixel referenced by the index tuple j
s^x	x-dimension of bandwidth vector
s^y	y-dimension of bandwidth vector
$\tilde{s}_m^x$	x-dimension of scaled clustering bandwidth vector of instance segmentation mask m
$\tilde{s}_m^y$	y-dimension of scaled clustering bandwidth vector of instance segmentation mask m
$\tilde{s}_{m,t}^x$	x-dimension of scaled clustering bandwidth vector of instance segmentation mask m at time point t
$\tilde{s}_{m,t}^y$	y-dimension of scaled clustering bandwidth vector of instance segmentation mask m at time point t
$\tilde{s}_j^{\text{smooth,x}}$	x-dimension of the scaled, smoothed clustering bandwidth vector of a pixel referenced by the index tuple j
$\tilde{s}_j^{\text{smooth,y}}$	y-dimension of the scaled, smoothed clustering bandwidth vector of a pixel referenced by the index tuple j
SEG	Segmentation metric in the Cell Tracking Challenge
$\mathsf{S}^{\text{smooth}}$	Smoothed clustering bandwidths tensor
ST	Silver Truth
SNR	Signal to Noise Ratio
t	1. Time point 2. Run index for time points
t'	Run index
$\mathcal{T}$	Set of time points
T_{max}	Last time point in an image sequence
T	Time point in an image sequence
TPA	True Positive Association
Δt	Time span
$\Delta T_{0.3}$	0.3 quantile of the track lengths in an image sequence
TRA	Tracking metric in the Cell Tracking Challenge
u	Node in a graph
v	1. Node in a graph 2. Run index
$v_{i,t}$	Node of type v at time point t in the graph corresponding to the segmented object i
$\mathcal{V}$	Set of nodes in a graph
w	1.Node in a graph 2. Run index
w_{EA}	Weight factor for adding an edge in the AOGM metric

Symbol	Description
w_{EC}	Weight factor to correct the semantic of an edge in the AOGM metric
w_{ED}	Weight factor for removing an edge in the AOGM metric
w_{FN}	Weight factor for FN detections in the AOGM metric
w_{FP}	Weight factor for FP detections in the AOGM metric
w_{NS}	Weight factor for under-segmented detections in the AOGM metric
W	Width of an image
w_{fg}	Foreground weighting factor
w_{instance}	Instance loss weighting factor
w_{var}	Variance loss weighting factor
w_{seed}	Seed loss weighting factor
w_s	Scaling weight
w_{seg}	Segmentation loss weighting factor
w_{track}	Tracking loss weighting factor
$\mathcal{W}_n$	Set of tracks that can be merged with track ω_n
x	x-dimension of a vector, matrix or tensor
$x_{i,t+1}$	Skip node at time point $t+1$ in the graph referring to segmented object i
y	y-dimension of a vector, matrix or tensor
$z_.$	Variable
$z^e_{p,n}$	Variable modeling an edge remove operation between predecessor track ω_p and track ω_n in the untangling problem
z^m_r	Variable modeling the merging of a set of tracks indexed by r in the untangling problem
z^s_n	Variable modeling the splitting of the track ω_n in the untangling problem
Z	Depth of a 3D image
α	A constant
ω_n	Track with the ID n
$\Delta\omega_n$	Temporal length of track ω_n

List of Own Publications

[57] T. Scherr et al. "Cell segmentation and tracking using CNN-based distance predictions and a graph-based matching strategy". In: *PLOS ONE* 15.12 (2020), e0243219. DOI: `10.1371/journal.pone.0243219`.

[137] B. Schott et al. "EmbryoMiner: A new framework for interactive knowledge discovery in large-scale cell tracking data of developing embryos". In: *PLOS Computational Biology* 14.4 (2018), pp. 1–18. DOI: `10.1371/journal.pcbi.1006128`.

[147] M. Hartmann, K. Löffler, and R. Mikut. "Simulation of synthetically degraded tracking data to benchmark MOT metrics". In: *Proceedings 32nd Workshop Computational Intelligence, Berlin, Germany, December 1-2, 2022.* Ed. by H. Schulte. Karlsruhe, Germany: KIT Scientific Publishing, 2022, pp. 163–180.

[160] S. Graham et al. *CoNIC Challenge: Pushing the frontiers of nuclear detection, segmentation, classification and counting.* arXiv Preprint. arXiv:2303.06274. Mar. 2023. URL: `https://arxiv.org/abs/2303.06274` (visited on 05/22/2023).

[163] M. Maška et al. "The Cell Tracking Challenge: 10 years of objective benchmarking". In: *Nature Methods* (2023). DOI: `10.1038/s41592-023-01879-y`.

[177] K. Löffler, T. Scherr, and R. Mikut. "A graph-based cell tracking algorithm with few manually tunable parameters and automated segmentation error correction". In: *PLOS ONE* 16.9 (2021), e0249257. DOI: `10.1371/journal.pone.0249257`.

[182] K. Löffler and R. Mikut. "EmbedTrack — Simultaneous cell segmentation and tracking through learning offsets and clustering bandwidths". In: *IEEE Access* 10 (2022), pp. 77147–77157. DOI: `10.1109/ACCESS.2022.3192880`.

[200] T. Scherr and K. Löffler. *KIT-GE (3).* Sept. 2021. URL: `http://celltrackingchallenge.net/participants/KIT-GE/` (visited on 09/07/2022).

[201] K. Löffler. *KIT-GE (4).* Feb. 2022. URL: `http://celltrackingchallenge.net/participants/KIT-GE/` (visited on 09/07/2022).

[212] T. Scherr and K. Löffler. *KIT-GE (2).* Mar. 2020. URL: `http://celltrackingchallenge.net/participants/KIT-GE/` (visited on 09/07/2022).

[227] T. Scherr et al. *On improving an already competitive segmentation algorithm for the Cell Tracking Challenge - Lessons learned.* bioRxiv Preprint. bioRxiv 2021.06.26.450019. June 2021. URL: `https://www.biorxiv.org/content/early/2021/06/28/2021.06.26.450019` (visited on 10/01/2023).

Bibliography

[1] T. Aspelmeier, A. Egner, and A. Munk. “Modern statistical challenges in high-resolution fluorescence microscopy”. In: *Annual Review of Statistics and Its Application* 2.1 (2015), pp. 163–202. DOI: 10.1146/annurev-statistics-010814-020343.

[2] S. Weisenburger and V. Sandoghdar. “Light microscopy: An ongoing contemporary revolution”. In: *Contemporary Physics* 56.2 (2015), pp. 123–143. DOI: 10.1080/00107514.2015.1026557.

[3] A. Y. Kobitski et al. “An ensemble-averaged, cell density-based digital model of zebrafish embryo development derived from light-sheet microscopy data with single-cell resolution”. In: *Scientific Reports* 5.1 (2015), p. 8601. DOI: 10.1038/srep08601.

[4] P. J. Keller et al. “Reconstruction of zebrafish early embryonic development by scanned light sheet microscopy”. In: *Science* 322.5904 (2008), pp. 1065–1069. DOI: 10.1126/science.1162493.

[5] R. Tomer et al. “Quantitative high-speed imaging of entire developing embryos with simultaneous multiview light-sheet microscopy”. In: *Nature Methods* 9.7 (2012), pp. 755–763. DOI: 10.1038/nmeth.2062.

[6] J. Stegmaier et al. “Real-time three-dimensional cell segmentation in large-scale microscopy data of developing embryos”. In: *Developmental Cell* 36.2 (2016), pp. 225–240. DOI: 10.1016/j.devcel.2015.12.028.

[7] M. Maška et al. “A benchmark for comparison of cell tracking algorithms”. In: *Bioinformatics* 30.11 (2014), pp. 1609–1617. DOI: 10.1093/bioinformatics/btu080.

[8] D. A. Lauffenburger and A. F. Horwitz. “Cell migration: A physically integrated molecular process”. In: *Cell* 84.3 (1996), pp. 359–369. DOI: 10.1016/S0092-8674(00)81280-5.

[9] A. Ortega-Carrion, L. Feo-Lucas, and M. Vicente-Manzanares. “Cell migration”. In: *Encyclopedia of Cell Biology*. Ed. by R. A. Bradshaw and P. D. Stahl. Waltham, MA, USA: Academic Press, 2016, pp. 720–730. DOI: 10.1016/B978-0-12-394447-4.20070-9.

[10] R. Keller. “Cell migration during gastrulation”. In: *Current Opinion in Cell Biology* 17.5 (2005), pp. 533–541. DOI: 10.1016/j.ceb.2005.08.006.

[11] C. J. Weijer. “Collective cell migration in development”. In: *Journal of Cell Science* 122.18 (2009), pp. 3215–3223. DOI: 10.1242/jcs.036517.

[12] A. Tremel et al. “Cell migration and proliferation during monolayer formation and wound healing”. In: *Chemical Engineering Science* 64.2 (2009), pp. 247–253. DOI: 10.1016/j.ces.2008.10.008.

[13] A. Ratheesh, V. Belyaeva, and D. E. Siekhaus. "Drosophila immune cell migration and adhesion during embryonic development and larval immune responses". In: *Current Opinion in Cell Biology* 36 (2015), pp. 71–79. DOI: https://doi.org/10.1016/j.ceb.2015.07.003.

[14] B. Laviña et al. "Defective endothelial cell migration in the absence of Cdc42 leads to capillary-venous malformations". In: *Development* 145.13 (2018), dev161182. DOI: 10.1242/dev.161182.

[15] A. A. Mosabbir, A. Qudrat, and K. Truong. "Engineered cell migration to lesions linked to autoimmune disease". In: *Biotechnology and Bioengineering* 115.4 (2018), pp. 1028–1036. DOI: 10.1002/bit.26523.

[16] F. Entschladen et al. "Tumour-cell migration, invasion, and metastasis: navigation by neurotransmitters". In: *The Lancet Oncology* 5.4 (2004), pp. 254–258. DOI: 10.1016/S1470-2045(04)01431-7.

[17] C.-M. Svensson et al. "Untangling cell tracks: Quantifying cell migration by time lapse image data analysis". In: *Cytometry Part A* 93.3 (2018), pp. 357–370. DOI: 10.1002/cyto.a.23249.

[18] S. Sato et al. "Single-cell lineage tracking analysis reveals that an established cell line comprises putative cancer stem cells and their heterogeneous progeny". In: *Scientific Reports* 6.1 (2016), p. 23328. DOI: 10.1038/srep23328.

[19] M.-L. Tremblay et al. "Using MRI cell tracking to monitor immune cell recruitment in response to a peptide-based cancer vaccine". In: *Magnetic Resonance in Medicine* 80.1 (2018), pp. 304–316. DOI: 10.1002/mrm.27018.

[20] J. B. Beltman, A. F. M. Marée, and R. J. de Boer. "Analysing immune cell migration". In: *Nature Reviews Immunology* 9.11 (2009), pp. 789–798. DOI: 10.1038/nri2638.

[21] K. W. Eliceiri et al. "Biological imaging software tools". In: *Nature Methods* 9.7 (2012), pp. 697–710. DOI: 10.1038/nmeth.2084.

[22] P. M. A. Antony et al. "Light microscopy applications in systems biology: Opportunities and challenges". In: *Cell Communication and Signaling* 11.1 (2013), p. 24. DOI: 10.1186/1478-811X-11-24.

[23] I. Arganda-Carreras and P. Andrey. "Designing image analysis pipelines in light microscopy: A rational approach". In: *Light Microscopy*. Ed. by Y. Markaki and H. Harz. Vol. 1563. New York, NY, USA: Springer, 2017, pp. 185–207. DOI: 10.1007/978-1-4939-6810-7_13.

[24] T. Ogama. "A beginner's guide to improving image acquisition in fluorescence microscopy". In: *The Biochemist* 42.6 (2020), pp. 22–27. DOI: 10.1042/BIO20200075.

[25] L. Fritzky and D. Lagunoff. "Advanced methods in fluorescence microscopy". In: *Analytical Cellular Pathology* 36.1-2 (2013), pp. 5–17. DOI: 10.1155/2013/569326.

[26] D. J. Stephens and V. J. Allan. "Light microscopy techniques for live cell imaging". In: *Science* 300.5616 (2003), pp. 82–86. DOI: 10.1126/science.1082160.

[27] E. Ward and R. Pal. “Image scanning microscopy: An overview”. In: *Journal of Microscopy* 266.2 (2017), pp. 221–228. DOI: 10.1111/jmi.12534.

[28] A. P. Cuny et al. “Live cell microscopy: From image to insight”. In: *Biophysics Reviews* 3.2 (2022), p. 021302. DOI: 10.1063/5.0082799.

[29] A. Kiepas et al. “Optimizing live-cell fluorescence imaging conditions to minimize phototoxicity”. In: *Journal of Cell Science* 133.4 (2020), jcs.242834. DOI: 10.1242/jcs.242834.

[30] F. Mubaid and C. M. Brown. “Less is more: Longer exposure times with low light intensity is less photo-toxic”. In: *Microscopy Today* 25.6 (2017), pp. 26–35. DOI: 10.1017/S1551929517000980.

[31] H. Schneckenburger and V. Richter. “Challenges in 3D live cell imaging”. In: *Photonics* 8.7 (2021), p. 275. DOI: 10.3390/photonics8070275.

[32] J. Icha et al. “Phototoxicity in live fluorescence microscopy, and how to avoid it”. In: *BioEssays* 39.8 (2017), p. 1700003. DOI: 10.1002/bies.201700003.

[33] M. A. Reiche et al. “When light meets biology – how the specimen affects quantitative microscopy”. In: *Journal of Cell Science* 135.6 (2022), jcs259656. DOI: 10.1242/jcs.259656.

[34] S. M. Pizer et al. “Adaptive histogram equalization and its variations”. In: *Computer Vision, Graphics, and Image Processing* 39.3 (1987), pp. 355–368. DOI: https://doi.org/10.1016/S0734-189X(87)80186-X.

[35] T. K. Agarwal, M. Tiwari, and S. S. Lamba. “Modified histogram based contrast enhancement using homomorphic filtering for medical images”. In: *2014 IEEE International Advance Computing Conference, IACC 2014, Gurgaon, India, February 21-22, 2014.* IEEE, 2014, pp. 964–968. DOI: 10.1109/IAdCC.2014.6779453.

[36] S. Cakir et al. “Contrast enhancement of microscopy images using image phase information”. In: *IEEE Access* 6 (2018), pp. 3839–3850. DOI: 10.1109/ACCESS.2018.2796646.

[37] F. Chen et al. “An accurate and universal approach for short-exposure-time microscopy image enhancement”. In: *Computerized Medical Imaging and Graphics* 83 (2020), p. 101743. DOI: 10.1016/j.compmedimag.2020.101743.

[38] M. Weigert et al. “Content-aware image restoration: Pushing the limits of fluorescence microscopy”. In: *Nature Methods* 15.12 (2018), pp. 1090–1097. DOI: 10.1038/s41592-018-0216-7.

[39] K. de Haan et al. “Deep-learning-based image reconstruction and enhancement in optical microscopy”. In: *Proceedings of the IEEE* 108.1 (2020), pp. 30–50. DOI: 10.1109/JPROC.2019.2949575.

[40] P. Stępień, W. Krauze, and M. Kujawińska. “Preprocessing methods for quantitative phase image stitching”. In: *Biomedical Optics Express* 13.1 (2022), pp. 1–13. DOI: 10.1364/BOE.439045.

[41] A. Bria et al. “Exploiting multi-level parallelism for stitching very large microscopy images”. In: *Frontiers in Neuroinformatics* 13 (2019), p. 41. DOI: `10.3389/fninf.2019.00041`.

[42] T. Du and M. Wasser. “3D image stack reconstruction in live cell microscopy of Drosophila muscles and its validation”. In: *Cytometry Part A* 75A.4 (2009), pp. 329–343. DOI: `https://doi.org/10.1002/cyto.a.20701`.

[43] X. Shi et al. “Deformed alignment of super-resolution images for semi-flexible structures”. In: *PLOS ONE* 14.3 (2019), e0212735. DOI: `10.1371/journal.pone.0212735`.

[44] S. Yang et al. “Nonrigid registration of 3-D multichannel microscopy images of cell nuclei”. In: *IEEE Transactions on Image Processing* 17.4 (2008), pp. 493–499. DOI: `10.1109/TIP.2008.918017`.

[45] M. Yigitsoy and N. Navab. “Structure propagation for image registration”. In: *IEEE Transactions on Medical Imaging* 32.9 (2013), pp. 1657–1670. DOI: `10.1109/TMI.2013.2263151`.

[46] A. Zaritsky et al. “Benchmark for multi-cellular segmentation of bright field microscopy images”. In: *BMC Bioinformatics* 14.1 (2013), p. 319. DOI: `10.1186/1471-2105-14-319`.

[47] P. Ma et al. “A state-of-the-art survey of object detection techniques in microorganism image analysis: From classical methods to deep learning approaches”. In: *Artificial Intelligence Review* (2022). DOI: `10.1007/s10462-022-10209-1`.

[48] E. Meijering et al. “Tracking in cell and developmental biology”. In: *Seminars in Cell & Developmental Biology* 20.8 (2009), pp. 894–902. DOI: `10.1016/j.semcdb.2009.07.004`.

[49] E. Meijering. “Cell segmentation: 50 years down the road [life sciences]”. In: *IEEE Signal Processing Magazine* 29.5 (2012), pp. 140–145. DOI: `10.1109/MSP.2012.2204190`.

[50] T. A. Nketia et al. “Analysis of live cell images: Methods, tools and opportunities”. In: *Methods* 115 (2017), pp. 65–79. DOI: `10.1016/j.ymeth.2017.02.007`.

[51] E. Moen et al. “Deep learning for cellular image analysis”. In: *Nature Methods* 16.12 (2019), pp. 1233–1246. DOI: `10.1038/s41592-019-0403-1`.

[52] Z. Liu et al. “A survey on applications of deep learning in microscopy image analysis”. In: *Computers in Biology and Medicine* 134 (2021), p. 104523. DOI: `10.1016/j.compbiomed.2021.104523`.

[53] T. Falk et al. “U-Net: Deep learning for cell counting, detection, and morphometry”. In: *Nature Methods* 16.1 (2019), pp. 67–70. DOI: `10.1038/s41592-018-0261-2`.

[54] M. Weigert et al. “Star-convex polyhedra for 3D object detection and segmentation in microscopy”. In: *IEEE Winter Conference on Applications of Computer Vision, WACV 2020, Snowmass Village, CO, USA, March 1-5, 2020.* IEEE, 2020, pp. 3655–3662. DOI: `10.1109/WACV45572.2020.9093435`.

[55] C. Stringer et al. “Cellpose: A generalist algorithm for cellular segmentation”. In: *Nature Methods* 18.1 (2021), pp. 100–106. DOI: 10.1038/s41592-020-01018-x.

[56] K. J. Cutler et al. “Omnipose: A high-precision morphology-independent solution for bacterial cell segmentation”. In: *Nature Methods* 19.11 (2022), pp. 1438–1448. DOI: 10.1038/s41592-022-01639-4.

[57] T. Scherr et al. “Cell segmentation and tracking using CNN-based distance predictions and a graph-based matching strategy”. In: *PLOS ONE* 15.12 (2020), e0243219. DOI: 10.1371/journal.pone.0243219.

[58] A. Arbelle, S. Cohen, and T. R. Raviv. “Dual-Task ConvLSTM-UNet for instance segmentation of weakly annotated microscopy videos”. In: *IEEE Transactions on Medical Imaging* 41.8 (2022), pp. 1948–1960. DOI: 10.1109/TMI.2022.3152927.

[59] K. He et al. “Mask R-CNN”. In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017.* IEEE Computer Society, 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[60] E. Romera et al. “ERFNet: Efficient residual factorized ConvNet for real-time semantic segmentation”. In: *IEEE Transactions on Intelligent Transportation Systems* 19.1 (2018), pp. 263–272. DOI: 10.1109/TITS.2017.2750080.

[61] S. Fujita and X. Han. “Cell detection and segmentation in microscopy images with improved Mask R-CNN”. In: *Computer Vision - ACCV 2020 Workshops - 15th Asian Conference on Computer Vision, Kyoto, Japan, November 30 - December 4, 2020, Revised Selected Papers.* Ed. by I. Sato and B. Han. Vol. 12628. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2020, pp. 58–70. DOI: 10.1007/978-3-030-69756-3_5.

[62] M. Lalit, P. Tomancak, and F. Jug. “Embedding-based instance segmentation in microscopy”. In: *Medical Imaging with Deep Learning, Lübeck, Germany, July 7-9, 2021.* Ed. by M. P. Heinrich et al. Vol. 143. Proceedings of Machine Learning Research. PMLR, 2021, pp. 399–415.

[63] L. Maddalena et al. “Artificial intelligence for cell segmentation, event detection, and tracking for label-free microscopy imaging”. In: *Algorithms* 15.9 (2022), p. 313. DOI: 10.3390/a15090313.

[64] A. M. Lucas et al. “Open-source deep-learning software for bioimage segmentation”. In: *Molecular Biology of the Cell* 32.9 (2021), pp. 823–829. DOI: 10.1091/mbc.E20-10-0660.

[65] L. von Chamier et al. “Democratising deep learning for microscopy with ZeroCostDL4Mic”. In: *Nature Communications* 12.1 (2021), p. 2276. DOI: 10.1038/s41467-021-22518-0.

[66] V. Ulman et al. “An objective comparison of cell-tracking algorithms”. In: *Nature Methods* 14.12 (2017), pp. 1141–1152. DOI: 10.1038/nmeth.4473.

[67] K. Rohr et al. “Tracking and quantitative analysis of dynamic movements of cells and particles”. In: *Cold Spring Harbor Protocols* 2010.6 (2010), pdb.top80. DOI: 10.1101/pdb.top80.

[68] E. Türetken et al. “Network flow integer programming to track elliptical cells in time-lapse sequences”. In: *IEEE Transactions on Medical Imaging* 36.4 (2017), pp. 942–951. DOI: 10.1109/TMI.2016.2640859.

[69] J. Wan, C. Xu, and Z. Xianhang. “Cell tracking via structured prediction and learning”. In: *Machine Vision and Applications* 28.8 (2017), pp. 859–874. DOI: 10.1007/s00138-017-0872-0.

[70] S. U. Akram et al. “Joint cell segmentation and tracking using cell proposals”. In: *13th IEEE International Symposium on Biomedical Imaging, ISBI 2016, Prague, Czech Republic, April 13-16, 2016.* IEEE, 2016, pp. 920–924. DOI: 10.1109/ISBI.2016.7493415.

[71] M. Schiegg et al. “Graphical model for joint segmentation and tracking of multiple dividing cells”. In: *Bioinformatics* 31.6 (2015), pp. 948–956. DOI: 10.1093/bioinformatics/btu764.

[72] M. Rempfler et al. “Efficient algorithms for moral lineage tracing”. In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017.* IEEE Computer Society, 2017, pp. 4705–4714. DOI: 10.1109/ICCV.2017.503.

[73] M. Rempfler et al. “Tracing cell lineages in videos of lens-free microscopy”. In: *Medical Image Analysis* 48 (2018), pp. 147–161. DOI: 10.1016/j.media.2018.05.009.

[74] A. Dufour et al. “Segmenting and tracking fluorescent cells in dynamic 3-D microscopy with coupled active surfaces”. In: *IEEE Transactions on Image Processing* 14.9 (2005), pp. 1396–1410. DOI: 10.1109/TIP.2005.852790.

[75] S. K. Nath, K. Palaniappan, and F. Bunyak. “Cell segmentation using coupled level sets and graph-vertex coloring”. In: *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2006, 9th International Conference, Copenhagen, Denmark, October 1-6, 2006, Proceedings, Part I.* Ed. by R. Larsen, M. Nielsen, and J. Sporring. Vol. 4190. Lecture Notes in Computer Science. Berlin, Heidelberg, Germany: Springer, 2006, pp. 101–108. DOI: 10.1007/11866565_13.

[76] O. Debeir et al. “Tracking of migrating cells under phase-contrast video microscopy with combined mean-shift processes”. In: *IEEE Transactions on Medical Imaging* 24.6 (2005), pp. 697–711. DOI: 10.1109/TMI.2005.846851.

[77] A. Arbelle et al. “A probabilistic approach to joint cell tracking and segmentation in high-throughput microscopy videos”. In: *Medical Image Analysis* 47 (2018), pp. 140–152. DOI: 10.1016/j.media.2018.04.006.

[78] R. S. Zou and C. Tomasi. “Deformable graph model for tracking epithelial cell sheets in fluorescence microscopy”. In: *IEEE Transactions on Medical Imaging* 35.7 (2016), pp. 1625–1635. DOI: 10.1109/TMI.2016.2521653.

[79] C. Zimmer et al. “On the digital trail of mobile cells”. In: *IEEE Signal Processing Magazine* 23.3 (2006), pp. 54–62. DOI: 10.1109/MSP.2006.1628878.

[80] N. Chenouard et al. “Objective comparison of particle tracking methods”. In: *Nature Methods* 11.3 (2014), pp. 281–289. DOI: 10.1038/nmeth.2808.

[81] A. J. Pretorius, I. A. Khan, and R. J. Errington. “A survey of visualization for live cell imaging: Visualization for live cell imaging”. In: *Computer Graphics Forum* 36.1 (2017), pp. 46–63. DOI: 10.1111/cgf.12784.

[82] C. Versari et al. “Long-term tracking of budding yeast cells in brightfield microscopy: CellStar and the evaluation platform”. In: *Journal of The Royal Society Interface* 14.127 (2017), p. 20160705. DOI: 10.1098/rsif.2016.0705.

[83] F. Boukari and S. Makrogiannis. “Automated cell tracking using motion prediction-based matching and event handling”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 17.3 (2020), pp. 959–971. DOI: 10.1109/TCBB.2018.2875684.

[84] M. A. A. Dewan, M. O. Ahmad, and M. N. S. Swamy. “Tracking biological cells in time-lapse microscopy: An adaptive technique combining motion and topological features”. In: *IEEE Transactions on Biomedical Engineering* 58.6 (2011), pp. 1637–1647. DOI: 10.1109/TBME.2011.2109001.

[85] K. E. G. Magnusson et al. “Global linking of cell tracks using the Viterbi algorithm”. In: *IEEE Transactions on Medical Imaging* 34.4 (2015), pp. 911–929. DOI: 10.1109/TMI.2014.2370951.

[86] F. Li et al. “Multiple nuclei tracking using integer programming for quantitative cancer cell cycle analysis”. In: *IEEE Transactions on Medical Imaging* 29.1 (2010), pp. 96–105. DOI: 10.1109/TMI.2009.2027813.

[87] A. Narayanaswamy et al. “Multi-temporal globally-optimal dense 3-D cell segmentation and tracking from multi-photon time-lapse movies of live tissue microenvironments”. In: *Spatio-temporal Image Analysis for Longitudinal and Time-Series Image Data - Second International Workshop, STIA 2012, Held in Conjunction with MICCAI 2012, Nice, France, October 1, 2012. Proceedings.* Ed. by S. Durrleman et al. Vol. 7570. Lecture Notes in Computer Science. Berlin, Heidelberg, Germany: Springer, 2012, pp. 147–162. DOI: 10.1007/978-3-642-33555-6_13.

[88] J. Hayashida, K. Nishimura, and R. Bise. “MPM: Joint representation of motion and position map for cell tracking”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020.* Computer Vision Foundation / IEEE, 2020, pp. 3822–3831. DOI: 10.1109/CVPR42600.2020.00388.

[89] C. Payer et al. “Segmenting and tracking cell instances with cosine embeddings and recurrent hourglass networks”. In: *Medical Image Analysis* 57 (2019), pp. 106–119. DOI: 10.1016/j.media.2019.06.015.

[90] T. He et al. “Cell tracking using deep neural networks with multi-task learning”. In: *Image and Vision Computing* 60 (2017), pp. 142–153. DOI: 10.1016/j.imavis.2016.11.010.

[91] Y. Wang, H. Mao, and Z. Yi. “Stem cell motion-tracking by using deep neural networks with multi-output”. In: *Neural Computing and Applications* 31.8 (2019), pp. 3455–3467. DOI: 10.1007/s00521-017-3291-2.

[92] K. Nishimura et al. “Weakly-supervised cell tracking via backward-and-forward propagation”. In: *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XII*. Ed. by A. Vedaldi et al. Vol. 12357. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2020, pp. 104–121. DOI: 10.1007/978-3-030-58610-2_7.

[93] M. Schiegg et al. “Conservation tracking”. In: *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*. IEEE Computer Society, 2013, pp. 2928–2935. DOI: 10.1109/ICCV.2013.364.

[94] Z. Zhou et al. “Joint multi-frame detection and segmentation for multi-cell tracking”. In: *Image and Graphics - 10th International Conference, ICIG 2019, Beijing, China, August 23-25, 2019, Proceedings, Part II*. Ed. by Y. Zhao et al. Vol. 11902. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2019, pp. 435–446. DOI: 10.1007/978-3-030-34110-7_36.

[95] A. Y. Kondratiev et al. “A method for automatic tracking of cell nuclei with weakly-supervised mitosis detection in 2D microscopy image sequences”. In: *ICBIP 2020: 5th International Conference on Biomedical Signal and Image Processing, Suzhou, China, August 21-23, 2020*. New York, NY, USA: Association for Computing Machinery (ACM), 2020, pp. 67–73. DOI: 10.1145/3417519.3417558.

[96] F. Lux and P. Matula. *MU-Lux-CZ*. Mar. 2020. URL: http://celltrackingchallenge.net/participants/MU-Lux-CZ/ (visited on 09/30/2020).

[97] Z. Wang, L. Yin, and Z. Wang. “A new approach for cell detection and tracking”. In: *IEEE Access* 7 (2019), pp. 99889–99899. DOI: 10.1109/ACCESS.2019.2930539.

[98] J. Stegmaier and R. Mikut. “Fuzzy-based propagation of prior knowledge to improve large-scale image analysis pipelines”. In: *PLOS ONE* 12.11 (2017), e0187535. DOI: 10.1371/journal.pone.0187535.

[99] F. W. Yang et al. “Investigating optimal time step intervals of imaging for data quality through a novel fully-automated cell tracking approach”. In: *Journal of Imaging* 6.7 (2020), p. 66. DOI: 10.3390/jimaging6070066.

[100] A. Santella, Z. Du, and Z. Bao. “A semi-local neighborhood-based framework for probabilistic cell lineage tracing”. In: *BMC Bioinformatics* 15 (2014), p. 217. DOI: 10.1186/1471-2105-15-217.

[101] S. Meng and H. Shen. “A robust cell tracking framework by fusing global and local optimization algorithms”. In: *2016 Chinese Control and Decision Conference, CCDC 2016, Yinchuan, China, May 28-30, 2016*. IEEE, 2016, pp. 2079–2084. DOI: 10.1109/CCDC.2016.7531327.

[102] R. Bise, Z. Yin, and T. Kanade. “Reliable cell tracking by global data association”. In: *Proceedings of the 8th IEEE International Symposium on Biomedical Imaging: From Nano to Macro, ISBI 2011, Chicago, Illinois, USA, March 30 - April 2, 2011*. IEEE, 2011, pp. 1004–1010. DOI: 10.1109/ISBI.2011.5872571.

[103] B.-n. Vo et al. “Multitarget tracking”. In: *Wiley Encyclopedia of Electrical and Electronics Engineering*. Hoboken, NJ, USA: John Wiley & Sons, Ltd, 2015, pp. 1–15. DOI: 10.1002/047134608X.W8275.

[104] S. Särkkä. *Bayesian filtering and smoothing*. 1st ed. Vol. 3. Institute of Mathematical Statistics textbooks. Cambridge University Press, 2013. DOI: 10.1017/CBO9781139344203.

[105] Y.-H. Chang et al. “Automated detection and tracking of cell clusters in time-lapse fluorescence microscopy images”. In: *Journal of Medical and Biological Engineering* 37.1 (2017), pp. 18–25. DOI: 10.1007/s40846-016-0216-y.

[106] M. Kaakinen et al. “Automatic detection and analysis of cell motility in phase-contrast time-lapse images using a combination of maximally stable extremal regions and Kalman filter approaches”. In: *Journal of Microscopy* 253.1 (2014), pp. 65–78. DOI: 10.1111/jmi.12098.

[107] M. Liu et al. “Plant cell tracking using Kalman filter based local graph matching”. In: *Image and Vision Computing* 60 (Apr. 2017), pp. 154–161. DOI: 10.1016/j.imavis.2016.08.005.

[108] O. Hirose et al. “SPF-CellTracker: Tracking multiple cells with strongly-correlated moves using a spatial particle filter”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 15.6 (2018), pp. 1822–1831. DOI: 10.1109/TCBB.2017.2782255.

[109] B. Xu et al. “An automated cell tracking approach with multi-Bernoulli filtering and ant colony labor division”. In: *IEEE/ACM transactions on computational biology and bioinformatics* 18.5 (2021), pp. 1850–1863. DOI: 10.1109/tcbb.2019.2954502.

[110] M. I. Hossain et al. “Visual mitosis detection and cell tracking using labeled Multi-Bernoulli filter”. In: *21st International Conference on Information Fusion, FUSION 2018, Cambridge, UK, July 10-13, 2018*. IEEE, 2018, pp. 1–5. DOI: 10.23919/ICIF.2018.8455486.

[111] K. Jaqaman et al. “Robust single-particle tracking in live-cell time-lapse sequences”. In: *Nature Methods* 5.8 (2008), pp. 695–702. DOI: 10.1038/nmeth.1237.

[112] D. Padfield, J. Rittscher, and B. Roysam. “Coupled minimum-cost flow cell tracking for high-throughput quantitative analysis”. In: *Medical Image Analysis* 15.4 (2011), pp. 650–668. DOI: 10.1016/j.media.2010.07.006.

[113] E. Moen et al. *Accurate cell tracking and lineage construction in live-cell imaging experiments with deep learning*. bioRxiv Preprint. bioRxiv 803205. Oct. 2019. URL: https://www.biorxiv.org/content/early/2019/10/14/803205 (visited on 07/27/2022).

[114] B. X. Kausler et al. “A discrete chain graph model for 3d+t cell tracking with high misdetection robustness”. In: *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part III.* Ed. by A. W. Fitzgibbon et al. Vol. 7574. Lecture Notes in Computer Science. Berlin, Heidelberg, Germany: Springer, 2012, pp. 144–157. DOI: `10.1007/978-3-642-33712-3_11`.

[115] S. U. Akram et al. *Cell tracking via proposal generation and selection.* arXiv Preprint. arXiv:1705.03386. May 2017. URL: `https://arxiv.org/abs/1705.03386` (visited on 10/01/2023).

[116] Gurobi Optimization, LLC. *Gurobi optimizer reference manual.* 2021. URL: `http://www.gurobi.com` (visited on 01/10/2023).

[117] K. Bestuzheva et al. *The SCIP optimization suite 8.0.* arXiv Preprint. arXiv:2112.08872. Dec. 2021. URL: `https://arxiv.org/abs/2112.08872` (visited on 10/01/2023).

[118] A. Makhorin. *GLPK (GNU Linear Programming Kit) package.* 2000. URL: `https://www.gnu.org/software/glpk/` (visited on 12/08/2022).

[119] S. Haller et al. “A primal-dual solver for large-scale tracking-by-assignment”. In: *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, August 26-28, 2020, Online [Palermo, Sicily, Italy].* Ed. by S. Chiappa and R. Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, 2020, pp. 2539–2549.

[120] C. Haubold et al. “A generalized successive shortest paths solver for tracking dividing targets”. In: *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VII.* Ed. by B. Leibe et al. Vol. 9911. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2016, pp. 566–582. DOI: `10.1007/978-3-319-46478-7_35`.

[121] M. Zhao et al. “VoxelEmbed: 3D instance segmentation and tracking with voxel embedding based deep learning”. In: *Machine Learning in Medical Imaging - 12th International Workshop, MLMI 2021, Held in Conjunction with MICCAI 2021, Strasbourg, France, September 27, 2021, Proceedings.* Ed. by C. Lian et al. Vol. 12966. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2021, pp. 437–446. DOI: `10.1007/978-3-030-87589-3_45`.

[122] J. Hayashida and R. Bise. “Cell tracking with deep learning for cell detection and motion estimation in low-frame-rate”. In: *Medical Image Computing and Computer Assisted Intervention - MICCAI 2019 - 22nd International Conference, Shenzhen, China, October 13-17, 2019, Proceedings, Part I.* Ed. by D. Shen et al. Vol. 11764. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2019, pp. 397–405. DOI: `10.1007/978-3-030-32239-7_44`.

[123] K. Sugawara, Ç. Çevrim, and M. Averof. “Tracking cell lineages in 3D by incremental deep learning”. In: *eLife* 11 (2022). Ed. by M. W. Mathis et al., e69380. DOI: `10.7554/eLife.69380`.

[124] C. Wen et al. “3DeeCellTracker, a deep learning-based pipeline for segmenting and tracking cells in 3D time lapse images”. In: *eLife* 10 (2021), e59187. DOI: `10.7554/eLife.59187`.

[125] Y. Chen et al. *CellTrack R-CNN: A novel end-to-end deep neural network for cell segmentation and tracking in microscopy images.* arXiv Preprint. arXiv:2102.10377. Feb. 2021. URL: `https://arxiv.org/abs/2102.10377` (visited on 10/01/2023).

[126] Y. Xie et al. "A deep local patch matching network for cell tracking in microscopy image sequences without registration". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 19.6 (2022), pp. 3202–3212. DOI: `10.1109/TCBB.2021.3113129`.

[127] J.-B. Lugagne, H. Lin, and M. J. Dunlop. "DeLTA: Automated cell segmentation, tracking, and lineage reconstruction using deep learning". In: *PLOS Computational Biology* 16.4 (2020), e1007673. DOI: `10.1371/journal.pcbi.1007673`.

[128] J. Wang et al. "Deep reinforcement learning for data association in cell tracking". In: *Frontiers in Bioengineering and Biotechnology* 8.298 (2020). DOI: `10.3389/fbioe.2020.00298`.

[129] T. Ben-Haim and T. R. Raviv. "Graph neural network for cell tracking in microscopy videos". In: *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXI.* Ed. by S. Avidan et al. Vol. 13681. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2022, pp. 610–626. DOI: `10.1007/978-3-031-19803-8_36`.

[130] J. Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017.* IEEE Computer Society, 2017, pp. 2242–2251. DOI: `10.1109/ICCV.2017.244`.

[131] Q. Liu et al. "ASIST: Annotation-free synthetic instance segmentation and tracking by adversarial simulations". In: *Computers in Biology and Medicine* 134 (2021), p. 104501. DOI: `10.1016/j.compbiomed.2021.104501`.

[132] B. Berthet and A. Maizel. "Light sheet microscopy and live imaging of plants". In: *Journal of Microscopy* 263.2 (2016), pp. 158–164. DOI: `10.1111/jmi.12393`.

[133] A. Mosig et al. "Tracking cells in life cell imaging videos using topological alignments". In: *Algorithms for Molecular Biology* 4 (2009), p. 10. DOI: `10.1186/1748-7188-4-10`.

[134] P. Matula et al. "Cell tracking accuracy measurement based on comparison of acyclic oriented graphs". In: *PLOS ONE* 10.12 (2015), e0144959. DOI: `10.1371/journal.pone.0144959`.

[135] J. M. Graham and B. P. Ayati. "A unified term for directed and undirected motility in collective cell invasion". In: *Applied Mathematics Letters* 25.12 (2012), pp. 2267–2271. DOI: `10.1016/j.aml.2012.06.015`.

[136] V. Foe and B. Alberts. "Studies of nuclear and cytoplasmic behaviour during the five mitotic cycles that precede gastrulation in Drosophila embryogenesis". In: *Journal of Cell Science* 61.1 (1983), pp. 31–70. DOI: `10.1242/jcs.61.1.31`.

[137] B. Schott et al. "EmbryoMiner: A new framework for interactive knowledge discovery in large-scale cell tracking data of developing embryos". In: *PLOS Computational Biology* 14.4 (2018), pp. 1–18. DOI: 10.1371/journal.pcbi.1006128.

[138] M. Winter et al. "Vertebrate neural stem cell segmentation, tracking and lineaging with validation and editing". In: *Nature Protocols* 6.12 (2011), pp. 1942–1952. DOI: 10.1038/nprot.2011.422.

[139] R. N. U. Kok et al. "OrganoidTracker: Efficient cell tracking using machine learning and manual error correction". In: *PLOS ONE* 15.10 (2020), pp. 1–18. DOI: 10.1371/journal.pone.0240802.

[140] J. Jeong et al. "Accurately tracking single-cell movement trajectories in microfluidic cell sorting devices". In: *PLOS ONE* 13.2 (2018), pp. 1–16. DOI: 10.1371/journal.pone.0192463.

[141] A. Kan et al. "Automated and semi-automated cell tracking: Addressing portability challenges". In: *Journal of Microscopy* 244.2 (2011), pp. 194–213. DOI: https://doi.org/10.1111/j.1365-2818.2011.03529.x.

[142] A. Kan et al. "Measures for ranking cell trackers without manual validation". In: *Pattern Recognition* 46.11 (2013), pp. 2849–2859. DOI: 10.1016/j.patcog.2013.04.007.

[143] A. Kan et al. "Ranking cell tracking systems without manual validation". In: *Pattern Recognition Letters* 53 (2015), pp. 38–43. DOI: 10.1016/j.patrec.2014.11.005.

[144] E. Ristani et al. "Performance measures and a data set for multi-target, multi-camera tracking". In: *Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II.* Ed. by G. Hua and H. Jégou. Vol. 9914. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2016, pp. 17–35. DOI: 10.1007/978-3-319-48881-3_2.

[145] J. Luiten et al. "HOTA: A higher order metric for evaluating multi-object tracking". In: *International Journal of Computer Vision* 129.2 (2021), pp. 548–578. DOI: 10.1007/s11263-020-01375-2.

[146] Y. Chen and Y. Huo. *Limitation of acyclic oriented graphs matching as cell tracking accuracy measure when evaluating mitosis.* arXiv Preprint. arXiv:2012.12084. Dec. 2020. URL: http://arxiv.org/abs/2012.12084 (visited on 01/12/2021).

[147] M. Hartmann, K. Löffler, and R. Mikut. "Simulation of synthetically degraded tracking data to benchmark MOT metrics". In: *Proceedings 32nd Workshop Computational Intelligence, Berlin, Germany, December 1-2, 2022.* Ed. by H. Schulte. Karlsruhe, Germany: KIT Scientific Publishing, 2022, pp. 163–180.

[148] P. J. Thul et al. "A subcellular map of the human proteome". In: *Science* 356.6340 (2017), eaal3321. DOI: 10.1126/science.aal3321.

[149] S. Graham et al. "Hover-Net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images". In: *Medical Image Analysis* 58 (2019), p. 101563. DOI: 10.1016/j.media.2019.101563.

[150] V. Ljosa, K. L. Sokolnicki, and A. E. Carpenter. “Annotated high-throughput microscopy image sets for validation”. In: *Nature Methods* 9.7 (2012), pp. 637–637. DOI: 10.1038/nmeth.2083.

[151] E. M. Christiansen et al. “In silico labeling: Predicting fluorescent labels in unlabeled images”. In: *Cell* 173.3 (2018), 792–803.e19. DOI: 10.1016/j.cell.2018.03.040.

[152] C. Ounkomol et al. “Label-free prediction of three-dimensional fluorescence images from transmitted-light microscopy”. In: *Nature Methods* 15.11 (2018), pp. 917–920. DOI: 10.1038/s41592-018-0111-2.

[153] H.-F. Tsai et al. “Usiigaci: Instance-aware cell tracking in stain-free phase contrast microscopy enabled by machine learning”. In: *SoftwareX* 9 (2019), pp. 230–237. DOI: 10.1016/j.softx.2019.02.007.

[154] C. Edlund et al. “LIVECell — A large-scale dataset for label-free live cell segmentation”. In: *Nature Methods* 18.9 (2021), pp. 1038–1045. DOI: 10.1038/s41592-021-01249-6.

[155] T. Vicar et al. “Cell segmentation methods for label-free contrast microscopy: Review and comprehensive comparison”. In: *BMC Bioinformatics* 20.1 (2019), p. 360. DOI: 10.1186/s12859-019-2880-8.

[156] J. C. Caicedo et al. “Evaluation of deep learning strategies for nucleus segmentation in fluorescence images”. In: *Cytometry Part A* 95.9 (2019), pp. 952–965. DOI: https://doi.org/10.1002/cyto.a.23863.

[157] N. Kumar et al. “A dataset and a technique for generalized nuclear segmentation for computational pathology”. In: *IEEE Transactions on Medical Imaging* 36.7 (2017), pp. 1550–1560. DOI: 10.1109/TMI.2017.2677499.

[158] S. Anjum and D. Gurari. “CTMC: Cell tracking with mitosis detection dataset challenge”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 4228–4237. DOI: 10.1109/CVPRW50498.2020.00499.

[159] Y.-T. Su et al. “Spatio-temporal mitosis detection in time-lapse phase-contrast microscopy image sequences: A benchmark”. In: *IEEE Transactions on Medical Imaging* 40.5 (2021), pp. 1319–1328. DOI: 10.1109/TMI.2021.3052854.

[160] S. Graham et al. *CoNIC Challenge: Pushing the frontiers of nuclear detection, segmentation, classification and counting*. arXiv Preprint. arXiv:2303.06274. Mar. 2023. URL: https://arxiv.org/abs/2303.06274 (visited on 05/22/2023).

[161] S. Javadi and S. A. Mirroshandel. “A novel deep learning method for automatic assessment of human sperm images”. In: *Computers in Biology and Medicine* 109 (2019), pp. 182–194. DOI: 10.1016/j.compbiomed.2019.04.030.

[162] H. S. Demir, A. E. Cetin, and R. Cetin Atalay. “A visual object tracking benchmark for cell motility in time-lapse imaging”. In: *Signal, Image and Video Processing* 13.6 (2019), pp. 1063–1070. DOI: 10.1007/s11760-019-01443-2.

[163] M. Maška et al. “The Cell Tracking Challenge: 10 years of objective benchmarking”. In: *Nature Methods* (2023). DOI: 10.1038/s41592-023-01879-y.

[164] D. F. E. Ker et al. "Phase contrast time-lapse microscopy datasets with automated and manual cell tracking annotations". In: *Scientific Data* 5.1 (2018), p. 180237. DOI: `10.1038/sdata.2018.237`.

[165] T. Scherr et al. "microbeSEG: A deep learning software tool with OMERO data management for efficient and accurate cell segmentation". In: *PLOS ONE* 17.11 (2022), e0277601. DOI: `10.1371/journal.pone.0277601`.

[166] T. Scherr et al. "BeadNet: Deep learning-based bead detection and counting in low-resolution microscopy images". In: *Bioinformatics* 36.17 (2020), pp. 4668–4670. DOI: `10.1093/bioinformatics/btaa594`.

[167] M. Weigert et al. "Biobeam — Multiplexed wave-optical simulations of light-sheet microscopy". In: *PLOS Computational Biology* 14.4 (2018), e1006079. DOI: `10.1371/journal.pcbi.1006079`.

[168] P. Malm, A. Brun, and E. Bengtsson. "Simulation of bright-field microscopy images depicting pap-smear specimen". In: *Cytometry Part A* 87.3 (2015), pp. 212–226. DOI: `https://doi.org/10.1002/cyto.a.22624`.

[169] D. Eschweiler et al. "3D fluorescence microscopy data synthesis for segmentation and benchmarking". In: *PLOS ONE* 16.12 (2021), e0260509. DOI: `10.1371/journal.pone.0260509`.

[170] L. Maier-Hein et al. "Why rankings of biomedical image analysis competitions should be interpreted with care". In: *Nature Communications* 9.1 (2018), p. 5217. DOI: `10.1038/s41467-018-07619-7`.

[171] G. Ciaparrone et al. "Deep learning in video multi-object tracking: A survey". In: *Neurocomputing* 381 (2020), pp. 61–88. DOI: `10.1016/j.neucom.2019.11.023`.

[172] W. Luo et al. "Multiple object tracking: A literature review". In: *Artificial Intelligence* 293 (2021), p. 103448. DOI: `10.1016/j.artint.2020.103448`.

[173] K. E. G. Magnusson. *Baxter Algorithms*. Feb. 2019. URL: `https://github.com/klasma/BaxterAlgorithms` (visited on 09/30/2020).

[174] J. Bang-Jensen and G. Z. Gutin. *Digraphs - theory, algorithms and applications*. 2nd ed. Springer Monographs in Mathematics. London, UK: Springer, 2009. DOI: `10.1007/978-1-84800-998-1`.

[175] B. Korte and J. Vygen. *Combinatorial optimization - theory and algorithms*. 6th ed. Algorithms and Combinatorics. Berlin, Heidelberg, Germany: Springer, 2018. DOI: `10.1007/978-3-662-56039-6`.

[176] F. A. Guerrero Peña et al. "J regularization improves imbalanced multiclass segmentation". In: *17th IEEE International Symposium on Biomedical Imaging, ISBI 2020, Iowa City, IA, USA, April 3-7, 2020*. IEEE, 2020, pp. 1–5. DOI: `10.1109/ISBI45749.2020.9098550`.

[177] K. Löffler, T. Scherr, and R. Mikut. "A graph-based cell tracking algorithm with few manually tunable parameters and automated segmentation error correction". In: *PLOS ONE* 16.9 (2021), e0249257. DOI: `10.1371/journal.pone.0249257`.

[178] C. Kuglin and D. Hines. "The phase correlation image alignment method". In: *Proceedings of the IEEE International Conference on Cybernetics and Society.* 1975, pp. 163–165.

[179] S. K. Pal et al. "Deep learning in multi-object detection and tracking: State of the art". In: *Applied Intelligence* 51.9 (2021), pp. 6400–6429. DOI: 10.1007/s10489-021-02293-7.

[180] L. Kalake, W. Wan, and L. Hou. "Analysis based on recent deep learning approaches applied in real-time multi-object tracking: A review". In: *IEEE Access* 9 (2021), pp. 32650–32671. DOI: 10.1109/ACCESS.2021.3060821.

[181] D. Neven et al. "Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019.* Computer Vision Foundation / IEEE, 2019, pp. 8837–8845. DOI: 10.1109/CVPR.2019.00904.

[182] K. Löffler and R. Mikut. "EmbedTrack — Simultaneous cell segmentation and tracking through learning offsets and clustering bandwidths". In: *IEEE Access* 10 (2022), pp. 77147–77157. DOI: 10.1109/ACCESS.2022.3192880.

[183] M. Berman, A. R. Triki, and M. B. Blaschko. "The Lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018.* Computer Vision Foundation / IEEE Computer Society, 2018, pp. 4413–4421.

[184] J. Yu and M. B. Blaschko. "Learning submodular losses with the Lovász hinge". In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, July 6-11, 2015.* Ed. by F. R. Bach and D. M. Blei. Vol. 37. JMLR Workshop and Conference Proceedings. JMLR.org, 2015, pp. 1623–1631.

[185] D. P. Kingma and J. Ba. *Adam: A method for stochastic optimization.* arXiv Preprint. arXiv:1412.6980. Dec. 2014. URL: https://arxiv.org/abs/1412.6980 (visited on 10/01/2023).

[186] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional networks for biomedical image segmentation". In: *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III.* Ed. by N. Navab et al. Vol. 9351. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

[187] L.-C. Chen et al. *Rethinking atrous convolution for semantic image segmentation.* arXiv Preprint. arXiv:1706.05587. Dec. 2017. URL: https://arxiv.org/abs/1706.05587 (visited on 10/28/2022).

[188] J. Long, E. Shelhamer, and T. Darrell. "Fully convolutional networks for semantic segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015.* IEEE Computer Society, 2015, pp. 3431–3440. DOI: 10.1109/CVPR.2015.7298965.

[189] K. He et al. “Deep residual learning for image recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016.* IEEE Computer Society, 2016, pp. 770–778. DOI: `10.1109/CVPR.2016.90`.

[190] J. Pineda et al. *Geometric deep learning reveals the spatiotemporal fingerprint of microscopic motion.* arXiv Preprint. arXiv:2202.06355. Feb. 2022. URL: `http://arxiv.org/abs/2202.06355` (visited on 05/31/2022).

[191] K. Sheppard et al. “Stride-level analysis of mouse open field behavior using deep-learning-based pose estimation”. In: *Cell Reports* 38.2 (2022), p. 110231. DOI: `10.1016/j.celrep.2021.110231`.

[192] M. Chang, N. Krahnstoever, and W. Ge. “Probabilistic group-level motion analysis and scenario recognition”. In: *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011.* Ed. by D. N. Metaxas et al. IEEE Computer Society, 2011, pp. 747–754. DOI: `10.1109/ICCV.2011.6126312`.

[193] M. Cordts et al. “The cityscapes dataset for semantic urban scene understanding”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016.* IEEE Computer Society, 2016, pp. 3213–3223. DOI: `10.1109/CVPR.2016.350`.

[194] H. Liu and L. Wang. “Gesture recognition for human-robot collaboration: A review”. In: *International Journal of Industrial Ergonomics* 68 (2018), pp. 355–367. DOI: `10.1016/j.ergon.2017.02.004`.

[195] A. Geiger et al. “Vision meets robotics: The KITTI dataset”. In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237. DOI: `10.1177/0278364913491297`.

[196] P. Dendorfer et al. *CVPR19 tracking and detection challenge: How crowded can it get?* arXiv Preprint. arXiv:1906.04567. June 2019. URL: `http://arxiv.org/abs/1906.04567` (visited on 04/01/2020).

[197] L. Leal-Taixé et al. *MOTChallenge 2015: Towards a benchmark for multi-target tracking.* arXiv Preprint. arXiv:1504.01942. Apr. 2015. URL: `https://arxiv.org/abs/1504.01942` (visited on 10/10/2022).

[198] P. Sun et al. “DanceTrack: Multi-object tracking in uniform appearance and diverse motion”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022.* IEEE, 2022, pp. 20961–20970. DOI: `10.1109/CVPR52688.2022.02032`.

[199] P. Emami, L. Elefteriadou, and S. Ranka. “Long-range multi-object tracking at traffic intersections on low-power devices”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.3 (2022), pp. 2482–2493. DOI: `10.1109/TITS.2021.3115513`.

[200] T. Scherr and K. Löffler. *KIT-GE (3).* Sept. 2021. URL: `http://celltrackingchallenge.net/participants/KIT-GE/` (visited on 09/07/2022).

[201] K. Löffler. *KIT-GE (4).* Feb. 2022. URL: `http://celltrackingchallenge.net/participants/KIT-GE/` (visited on 09/07/2022).

[202] A. Arbelle, S. Cohen, and T. Riklin Raviv. *BGU-IL*. June 2018. URL: http://celltrackingchallenge.net/participants/BGU-IL/ (visited on 06/22/2022).

[203] A. Arbelle and T. R. Raviv. "Microscopy cell segmentation via convolutional LSTM networks". In: *16th IEEE International Symposium on Biomedical Imaging, ISBI 2019, Venice, Italy, April 8-11, 2019*. IEEE, 2019, pp. 1008–1012. DOI: 10.1109/ISBI.2019.8759447.

[204] X. Zhao et al. *CAS-CN*. Mar. 2021. URL: http://celltrackingchallenge.net/participants/CAS-CN/ (visited on 09/06/2022).

[205] P. Křížek and G. Hagen. *CUNI-CZ*. Mar. 2013. URL: http://celltrackingchallenge.net/participants/CUNI-CZ/ (visited on 09/06/2022).

[206] T. Sixta et al. *CVUT-CZ*. Aug. 2018. URL: http://celltrackingchallenge.net/participants/CVUT-CZ/ (visited on 09/06/2022).

[207] Y. Wang et al. *DESU-US*. Nov. 2021. URL: http://celltrackingchallenge.net/participants/DESU-US (visited on 06/22/2022).

[208] L. Aho, R. Yung, and A. Cohen. *DREX-CN*. July 2021. URL: http://celltrackingchallenge.net/participants/DREX-US/ (visited on 09/06/2022).

[209] O. Ronneberger et al. *FR-GE-(2)*. Mar. 2015. URL: http://celltrackingchallenge.net/participants/FR-GE/ (visited on 09/07/2022).

[210] T. Falk et al. *FR-GE-(3)*. Aug. 2018. URL: http://celltrackingchallenge.net/participants/FR-GE/ (visited on 09/07/2022).

[211] Z. Zhou and F. Wang. *HIT-CN*. Apr. 2019. URL: http://celltrackingchallenge.net/participants/HIT-CN/ (visited on 06/22/2022).

[212] T. Scherr and K. Löffler. *KIT-GE (2)*. Mar. 2020. URL: http://celltrackingchallenge.net/participants/KIT-GE/ (visited on 09/07/2022).

[213] K. Magnusson, J. Jaldén, and H. M. Blau. *KTH-SE*. Mar. 2021. URL: http://celltrackingchallenge.net/participants/KTH-SE/ (visited on 06/22/2022).

[214] O. Dzyubachyk et al. "Advanced level-set-based cell tracking in time-lapse fluorescence microscopy". In: *IEEE Transactions on Medical Imaging* 29.3 (2010), pp. 852–867. DOI: 10.1109/TMI.2009.2038693.

[215] O. Dzyubachyk and E. Meijering. *LEID-NL*. Mar. 2021. URL: http://celltrackingchallenge.net/participants/LEID-NL/ (visited on 09/06/2022).

[216] N. M. Al-Shakarji et al. *MU-US-(2)*. Mar. 2021. URL: http://celltrackingchallenge.net/participants/MU-US/ (visited on 06/22/2022).

[217] N. M. Al-Shakarji et al. "Multi-object tracking cascade with multi-step data association and occlusion handling". In: *15th IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS 2018, Auckland, New Zealand, November 27-30, 2018*. IEEE, 2018, pp. 1–6. DOI: 10.1109/AVSS.2018.8639321.

[218] R. Bao et al. *MU-US-(3)*. Mar. 2021. URL: http://celltrackingchallenge.net/participants/MU-US/ (visited on 06/22/2022).

[219] P. Liang. *ND-US*. Mar. 2019. URL: http://celltrackingchallenge.net/participants/ND-US/ (visited on 06/22/2022).

[220] T. Guo and Y. Wang. *PURD-US*. Dec. 2020. URL: http://celltrackingchallenge.net/participants/PURD-US/ (visited on 06/22/2022).

[221] T. Guo, A. M. Ardekani, and P. P. Vlachos. "Microscale, scanning defocusing volumetric particle-tracking velocimetry". In: *Experiments in Fluids* 60.6 (2019), p. 89. DOI: 10.1007/s00348-019-2731-4.

[222] D. Eschweiler and J. Stegmaier. *RWTH-GE*. Mar. 2019. URL: http://celltrackingchallenge.net/participants/RWTH-GE/ (visited on 09/06/2022).

[223] C. Payer et al. *TUG-AT*. Feb. 2019. URL: http://celltrackingchallenge.net/participants/TUG-AT/ (visited on 06/22/2022).

[224] S. Shailja, J. Jiang, and B. S. Manjunath. *UCSB-US*. Oct. 2020. URL: http://celltrackingchallenge.net/participants/UCSB-US/ (visited on 09/06/2022).

[225] Y. Chen and C. Zhang. *USYD-AU*. Oct. 2020. URL: http://celltrackingchallenge.net/participants/USYD-AU/ (visited on 09/06/2022).

[226] A. Panteli et al. *UVA-NL*. Feb. 2020. URL: http://celltrackingchallenge.net/participants/UVA-NL/ (visited on 06/22/2022).

[227] T. Scherr et al. *On improving an already competitive segmentation algorithm for the Cell Tracking Challenge - Lessons learned.* bioRxiv Preprint. bioRxiv 2021.06.26.450019. June 2021. URL: https://www.biorxiv.org/content/early/2021/06/28/2021.06.26.450019 (visited on 10/01/2023).

[228] L. Wright and N. Demeure. *Ranger21: A synergistic deep learning optimizer.* arXiv Preprint. arXiv:2106.13731. June 2021. URL: https://arxiv.org/abs/2106.13731 (visited on 09/27/2022).

[229] L. Wright. *Ranger - A synergistic optimizer.* Aug. 2019. URL: https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer (visited on 09/27/2022).

[230] R. Mikut. *Data Mining in der Medizin und Medizintechnik.* Vol. 22. Schriftenreihe des Instituts für Angewandte Informatik - Automatisierungstechnik, Universität Karlsruhe (TH). Karlsruhe, Germany: Universitätsverlag Karlsruhe, 2008. DOI: 10.5445/KSP/1000008476.