A graph-based cell tracking algorithm with few manually tunable parameters and automated segmentation error correction

Katharina Löffler[1,2], Tim Scherr[1], Ralf Mikut[1]

1 Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Eggenstein-Leopoldshafen, Germany
2 Institute of Biological and Chemical Systems - Biological Information Processing, Karlsruhe Institute of Technology, Eggenstein-Leopoldshafen, Germany

* katharina.loeffler@kit.edu

Abstract

Automatic cell segmentation and tracking enables to gain quantitative insights into the processes driving cell migration. To investigate new data with minimal manual effort, cell tracking algorithms should be easy to apply and reduce manual curation time by providing automatic correction of segmentation errors. Current cell tracking algorithms, however, are either easy to apply to new data sets but lack automatic segmentation error correction, or have a vast set of parameters that needs either manual tuning or annotated data for parameter tuning. In this work, we propose a tracking algorithm with only few manually tunable parameters and automatic segmentation error correction. Moreover, no training data is needed. We investigate the performance of our approach by simulating erroneous segmentation data, including false negatives, over- and under-segmentation errors, on 2D and 3D cell data sets. We compare our approach against three well-performing tracking algorithms from the Cell Tracking Challenge. Our tracking algorithm can correct false negatives, over- and under-segmentation errors as well as a mixture of the aforementioned segmentation errors. Furthermore, in case of under-segmentation or a combination of segmentation errors our approach outperforms the other tracking approaches.

Introduction

The ability of cells to migrate is essential for many biological processes such as tissue formation, immune response, or wound healing [1]. Disruptions in cell migration can contribute to diseases such as malformation [2], autoimmune disease [3], and metastasis [4]. To better understand the mechanisms driving cell migration, the cell behavior can be analyzed quantitatively, for instance by tracking cells over time. However, tracking cells manually is tedious, even for small data sets, and becomes for large data sets infeasible. Therefore, automated cell tracking methods are needed which minimize manual curation effort and expert knowledge for parameter adjustments.

Recent cell tracking methods can be categorized into tracking by detection and tracking by model evolution approaches [5]. In tracking by model evolution approaches an initial segmentation is propagated over time [6], whereas tracking by detection approaches split segmentation and tracking in two steps. In this paper, we focus on tracking by detection approaches, due to promising improvements of cell segmentation algorithms [7–11].

At present a vast variety of tracking by detection approaches has been proposed. The most simplistic approaches use nearest neighbor methods [12, 13] or are based on overlap [14, 15]. Bayesian filters like the Kalman filter [16], particle filter [17–19] or Bernoulli filter [20, 21] have been adapted for cell tracking as well. Hybrid methods combine simplistic tracking methods, like nearest neighbors, with more sophisticated tracking methods [22–25]. Furthermore, deep learning based approaches have been proposed for cell tracking [26, 27]. Graph-based approaches offer the possibility to model cell behavior such as motion, mitosis or cell death explicitly [28–36].

Tracks are created by linking segmentation masks over time based on a "linking" measure. A simple linking measure is the Euclidean distance between the positions of the cell centroids. Other linking measures are based on handcrafted features, such as position and appearance [25, 30, 37, 38], features of the cell's neighborhood [39], features derived from a graph structure [40], or learned features [18, 19, 26, 27, 41]. The contribution of the extracted features in the measure is often learned for instance by using logistic regression [30], a structured support vector machine [42], a random forest [33] or training convolutional neural networks [18, 19]. Besides that, some approaches train additional detectors to detect mitosis [12, 14, 19]. However, to fit such linking measures to new data sets, annotated data are needed, which requires additional annotation effort. Using simple, position-based linking measures, in contrast, can be applied to a vast set of experiments without training.

As tracking by detection methods split tracking and segmentation, a reasonable segmentation quality of cells is needed for good tracking results. However, segmentation approaches need to handle challenging imaging conditions such as low signal-to-noise ratio or low contrast [6] as well as the wide range of appearance due to different imaging methods and cell types [5]. Therefore, an error-free segmentation is almost impossible. The resulting segmentation errors can be classified as False Positives (FP), False Negatives (FN), over-segmentation, under-segmentation, and wrong partitioning of touching cells [43].

To handle such segmentation errors in tracking by detection methods, two strategies exist: 1) Generating overlapping segmentation masks and selecting the final set of segmentation masks in the tracking step [31, 32, 34, 36, 42, 44]. 2) Using non overlapping segmentation masks and detecting and correcting segmentation errors [24, 28, 30, 33, 35, 39, 40, 45–47]. The first strategy is computationally expensive as several segmentation hypothesis are competing. For the second strategy, semi-automated methods with manual data curation [45, 46, 48] and automated methods [24, 28, 30, 33, 35, 39, 40, 47] have been proposed.

While semi-automated methods need manual effort for error correction, automated segmentation correction approaches often require a learning step. For instance, classifiers that estimate the number of objects per segmentation mask are trained [30, 33], where no objects correspond to FPs, and more than one object to an under-segmentation error. Another approach is to train a support vector machine to distinguish mitosis from over-segmentation [39]. While there are approaches resolving multiple over- or under-segmented cells, they assume no mitosis events are occurring [40, 47]. To detect FPs, prior knowledge on the length of mitosis cycle [35], or on the expected track length [24] is used. Also, a two stage tracking is proposed to first construct short tracks and then resolve segmentation errors in the second step to yield the final tracks [28]. Besides that, uncertainty information is propagated through the segmentation and tracking pipeline to improve results [49].

In summary, current tracking approaches are either simple to apply but need manual error correction, or are able to correct segmentation errors but have a vast set of parameters to tune or need additional training data. In this work, we propose a compromise between the two sides, a simple to apply tracking algorithm, which needs

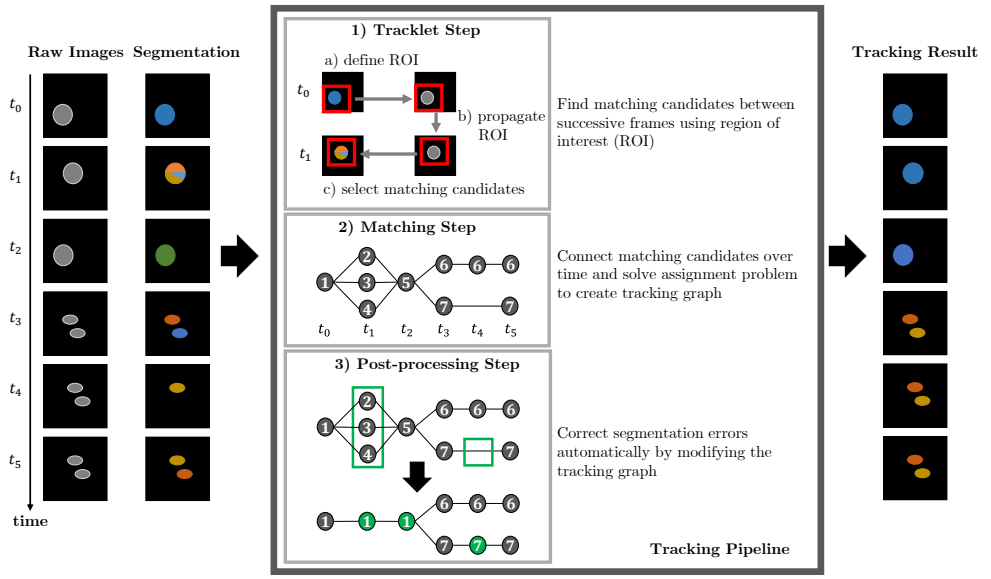


Fig 1. Tracking pipeline. Steps of our proposed tracking algorithm based on an input image sequence with erroneous segmentation data. After processing the image sequence through the tracking pipeline, the cells are tracked and segmentation errors are corrected. The node IDs in the tracking graph indicate the assigned track ID to the segmented objects.

no training and extensive parameter tuning, yet able to correct certain types of segmentation errors.

The main contributions are: a) We propose a tracking approach able to handle the segmentation errors under- and over-segmentation with more than two objects involved and FN. b) We make our Python code available as open source `https://git.scc.kit.edu/KIT-Sch-GE/2021-cell-tracking`. c) We compare our tracking approach to three other tracking approaches which performed competitive on the Cell Tracking Challenge (CTC) `http://celltrackingchallenge.net/` [5, 50] and investigate how robust the selected tracking approaches perform, when the segmentation quality decreases.

Materials and methods

We base our method on the following assumptions: The cell movement is small compared to the overall image size and the majority of segmentation masks are segmenting single cells correctly. The cell movement assumption is motivated by the need of a reasonable temporal resolution of the image sequence for a detailed analysis of cell lineage or cell behavior. The segmentation assumption is motivated by the availability of reasonably well-performing segmentation approaches [7–11]. Due to potentially occurring segmentation errors, we refer to segmentation masks as segmented objects and not as cells, as the segmentation masks can contain an as cell detected artifact, only parts of a cell, a single cell or several cells.

We split the task of cell tracking 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

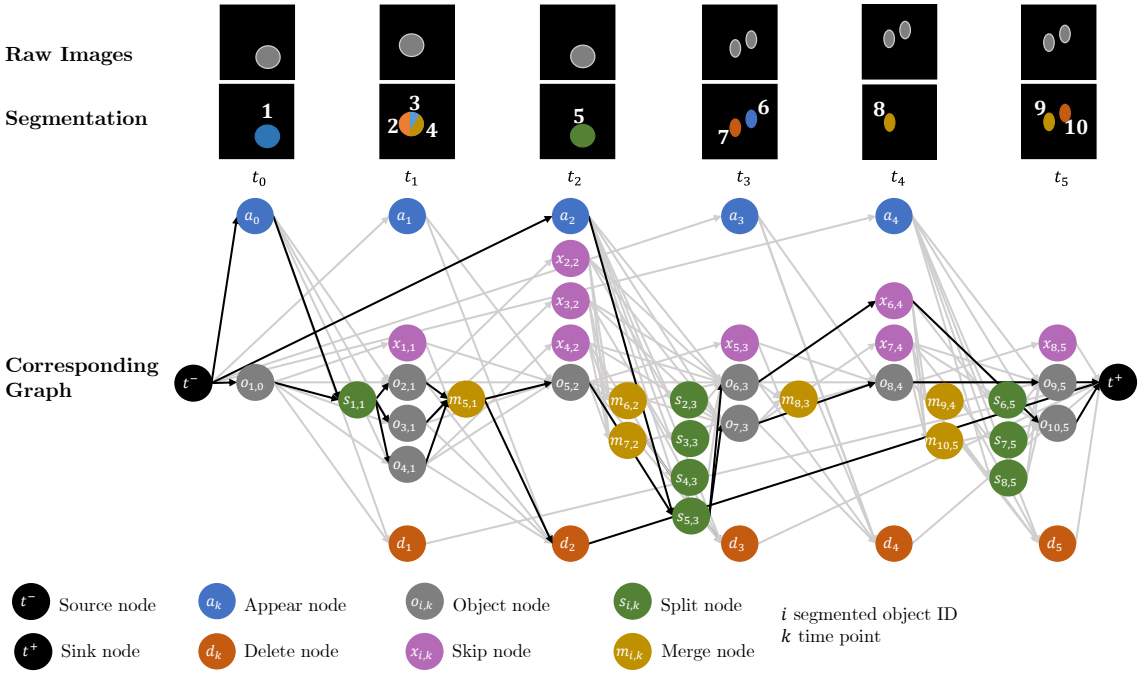


Fig 2. Matching step example graph. The graph in the matching step is constructed for an image sequence with erroneous segmentation. Each segmented object is assigned an unique ID i. Nodes corresponding to a segmented object share the same ID i, however, depending on the node type these nodes are assigned to different time points k in the graph. We link segmented objects over a maximum time span of $\Delta t = 2$ frames by adding for each object node $o_{i,k}$ a skip node $x_{i,k+1}$, which models a missing segmentation mask. The segmented objects are assigned to tracks by finding optimal paths through the graph by solving a coupled minimum cost flow problem [29]. The optimal paths through the graph are highlighted in black, where we omitted highlighting paths that connect appear and delete nodes only.

problem. The graph models cell behavior including appearance, disappearance, movement, and mitosis as well as the segmentation errors over- and under-segmentation and FN. Lastly, a post-processing step is applied to correct segmentation errors. An overview of the tracking pipeline based on an example is shown in Fig 1.

Step 1: Tracklet step

Based on the cell movement assumption, segmented objects belonging to the same track should be spatially close between successive time points. Similar as in [9], we define for each segmented object a rectangular shaped region of interest (ROI), which size is derived from the average size of the segmentation masks, to find objects which could belong to the same track at successive time points. The ROI is propagated over time by estimating a displacement between successive frames using a phase correlation [51]. We consider segmented objects which overlap with the propagated ROI as matching candidates which will be linked in the matching step.

Step 2: Matching step

We model segmented objects and their matching candidates as nodes in a directed graph. The graph models the cell behavior: appearance, disappearance, movement, and

mitosis as well as the segmentation errors: FNs, under- and over-segmentation. By finding optimal paths through this graph, the segmented objects are linked over time.

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 We define node types to model cell behavior and segmentation errors:

- t^-: source node
- t^+: sink node
- o: object nodes modeling segmented objects
- s: split nodes modeling mitosis / over-segmentation errors
- m: merge nodes modeling under-segmentation errors
- x: skip nodes modeling FNs
- d: delete nodes modeling disappearing objects
- a: appear nodes modeling appearing objects

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

For each segmented object i at time point k a corresponding object node $o_{i,k}$ is added to the graph. To link tracks with missing segmentation masks over a maximum time span of Δt, we add skip nodes for each segmented object from time point k at $\Delta t - 1$ successive time points. For each time point k an appearance node a_k is added to model appearing objects at time point $k + 1$, whereas a delete node d_k is added for each time point k to model disappearing objects at time point $k - 1$. Mitosis and over-segmentation errors are modeled by adding for each object node and skip node at time point k a split node $s_{i,k+1}$ at time point $k + 1$. Under-segmentation errors are modeled by adding a merge node $m_{i,k-1}$ at time point $k - 1$ for each object node and a skip node at time point k. The source node t^- is added before the first time point and a sink node t^+ is added after the last time point of the considered set of time points $\mathcal{T}$.

Edges The nodes are connected by directed edges to model events, such as linking segmented objects between successive time points. We allow directed edges 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:

- $t^- : \{a, o\}$
- $t^+ : \{\}$
- $o : \{d, m, o, s, t^+, x\}$
- $s : \{o\}$
- $m : \{d, o\}$
- $x : \{m, o, s, t^+, x\}$
- $d : \{t^+\}$
- $a : \{d, o, s\}$

Fig 2 shows the constructed graph based on the image sequence with erroneous segmentation from Fig 1.

Connecting all object nodes and skip nodes at time point k naïvely to all other object nodes at time point $k+1$, would result in a quadratically growing number of edges. To reduce the number of edges in the graph, we use the matching candidates from the tracklet step and connect nodes only to the nodes corresponding with its matching candidates. This is applied to the split and merge nodes as well, by connecting a split node $s_{i,k+1}$ only to the object nodes at $k+1$, the object node $o_{i,k}$ or skip node $x_{i,k}$ is connected to. A merge node $m_{i,k}$ is only connected to the object nodes and skip nodes at k, the object node $o_{i,k+1}$ is connected to. A visualization how nodes are connected is shown in Fig 3 based on the graph example of Fig 2. The used costs functions are introduced in more detail in the following.

Formulation as coupled minimum cost flow problem In theory the graph could be spanned over the full time span of an image sequence, however, for data sets with many cells and time points this would lead to large optimization problems which need to be solved. Therefore, we solve smaller optimization problems by dividing the image sequence in smaller time spans and constructing graphs which overlap in time.

In the following, optimal paths through the graph are found by solving a coupled minimum cost flow problem. Our formulation is most similar to the coupled minimum cost flow problem [29], which we extend such that many to one and one to many links are possible as well as introducing skip nodes. Therefore, over- and under-segmentation of two or more objects as well as missing segmentation masks are modeled in the graph. To find optimal paths through the graph, a flow variable $f(u,v)$ is defined for each edge (u,v), where $f(u,v) \in \mathbb{N}_0$. The optimization problem is given as

$$\begin{aligned} \min_{f(u,v)} \quad & \sum_{(u,v)\in\mathcal{E}} c(u,v) f(u,v) \\ \text{subject to:} \quad & \\ & g_{i'}(f(u,v)) = 0,\ i' = 0,\dots,N \\ & h_{j'}(f(u,v)) \leq 0,\ j' = 0,\dots,M \end{aligned} \tag{1}$$

where $c(u,v)$ is a cost and $g_{i'}$ are equality constraints and $h_{j'}$ inequality constraints, which will be introduced in the following.

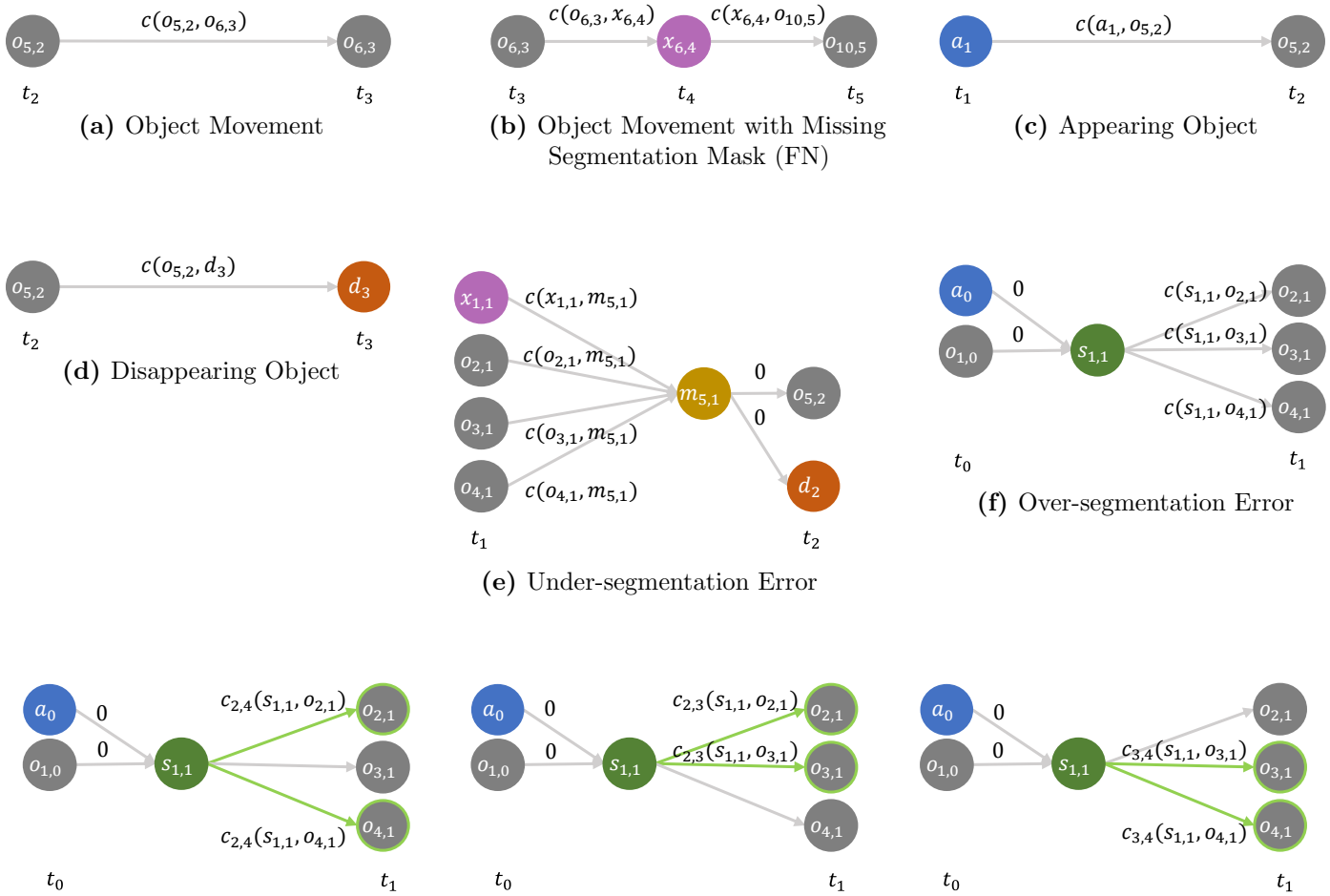
A flow conservation constraint is added for all nodes apart from source node and sink node:

$$\sum_{u\in\mathcal{V}} f(u,v) = \sum_{w\in\mathcal{V}} f(v,w). \tag{2}$$

Flow requirements enforce a flow of a fixed number of units through the graph. We ensure that each segmented object is assigned to a track, by enforcing a flow of one trough each object node $o_{i,k}$ and setting the flow from the source node t^- to the total number of segmented object nodes

$$\begin{aligned} \sum_{u\in\mathcal{V}} f(t^-,u) &= \sum_{k\in\mathcal{T}} |\mathcal{O}_k|, \\ \sum_{u\in\mathcal{V}} f(u,o_{i,k}) &= 1, \\ f(t^-,a_k) &= |\mathcal{O}_{k+1}|, \end{aligned} \tag{3}$$

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



(a) Object Movement

(b) Object Movement with Missing Segmentation Mask (FN)

(c) Appearing Object

(d) Disappearing Object

(e) Under-segmentation Error

(f) Over-segmentation Error

(g) Mitosis. Pairwise coupled edges highlighted in green.

Fig 3. Modeling cell behavior and segmentation errors. Parts of the graph example of Fig 2 are shown to visualize how cell behavior and segmentation errors are modeled in the graph. Each node $v_{i,k}$ is assigned a unique ID i, which corresponds to a segmented object, and a time point k. Annotations $c(\cdot,\cdot)$ on the edges are assigned edge costs. To model mitosis (Fig 3g), edges which are connected to pairs of "daughter" nodes are pairwise coupled.

The flow over an edge (u, v) is restricted by a maximum capacity constraint $b(u, v)$:

$$0 \leq f(u, v) \leq b(u, v). \tag{4}$$

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

$$b(u, o_{i,k}) = b(u, x_{i,k}) = b(o_{i,k}, v) = b(x_{i,k}, v) = 1. \tag{5}$$

To model over- and under-segmentation of more than two objects, the capacity of edges connecting merge nodes $m_{i,k-1}$ to delete nodes d_k and appear nodes a_k to split nodes $s_{i,k+1}$ depends on the number of edges ending in the merge node and edges starting from the split node, respectively:

$$\begin{aligned} b(m_{i,k-1}, d_k) &= |\{(v_{\cdot,k-1}, m_{i,k-1}) \mid v_{\cdot,k-1} \text{ connected to } m_{i,k-1}\}|, \\ b(a_k, s_{i,k+1}) &= |\{(s_{i,k+1}, v_{\cdot,k+1}) \mid v_{\cdot,k+1} \text{ connected to } s_{i,k+1}\}|. \end{aligned} \tag{6}$$

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

$$\begin{aligned} b(t^-, a_k) &= |\mathcal{O}_{k+1}|, \\ b(a_{k-1}, d_k) &= |\mathcal{O}_k|, \\ b(d_k, t^+) &= \sum_{t=k-\Delta t}^{k} |\mathcal{O}_t|, \end{aligned} \tag{7}$$

where $|\mathcal{O}_k|$ is the number of object nodes at time point k. The sum of the capacity constraint $b(d_k, t^+)$ 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 $b(d_k, t^+) = |\mathcal{O}_k| + |\mathcal{O}_{k-1}|$, providing a large upper bound. For $\Delta t = 2$ for each object node a skip node is added, allowing a flow from an object node at $k-2$ over its skip node to a merge node at $k-1$, which is connected to the delete node d_k. To provide a large enough upper bound, the number of object nodes from time point $k-2$ is added to the maximum capacity:

$$b(d_k, t^+) = \sum_{t=k-2}^{k} |\mathcal{O}_t| = |\mathcal{O}_k| + |\mathcal{O}_{k-1}| + |\mathcal{O}_{k-2}|.$$

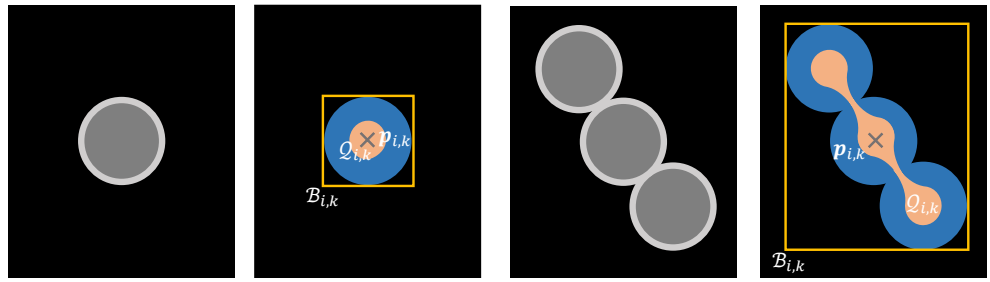
To model under-segmentation of two or more objects, for each merge node $m_{i,k-1}$ the following constraints are added:

$$\begin{aligned} f(m_{i,k-1}, o_{i,k}) - f(m_{i,k-1}, d_k) &\leq 0, \\ f(v_{\cdot,k-1}, m_{i,k-1}) - f(m_{i,k-1}, o_{i,k}) &\leq 0 \;\forall\; v_{\cdot,k-1} \text{ connected to } m_{i,k}. \end{aligned} \tag{8}$$

Combining Eq. 4 and Eq. 8, we derive

$$0 \leq f(v_{\cdot,k-1}, m_{i,k-1}) \leq f(m_{i,k-1}, o_{i,k}) \leq f(m_{i,k-1}, d_k).$$

For a flow $f(v_{\cdot,k-1}, m_{i,k-1})$ from a node $v_{\cdot,k-1}$ to the merge node $m_{i,k-1}$ larger than zero, the flow from the merge node to the object node $f(m_{i,k-1}, o_{i,k})$ and the flow from the merge node to the delete node $f(m_{i,k-1}, d_k)$ need to be at least as large. The flow conservation constraint Eq. 2 enforces the same flow into a node and from a node, resulting in a flow of at least two through the merge node $m_{i,k-1}$ or zero.



(a1) Raw Image **(a2)** Features **(b1)** Raw Image **(b2)** Features

(a) Single Object **(b)** Merged Objects

Fig 4. Features extracted from segmentation mask. From the segmentation mask i at time point k the mask centroid $\mathbf{p}_{i,k}$, a set of mask points $\mathcal{Q}_{i,k}$ and a bounding box $\mathcal{B}_{i,k}$ is extracted. The segmentation mask is shown in blue, the mask centroid as a gray cross, the set of mask points in orange, and the spanned bounding box in yellow.

To model over-segmentation into two or more objects, for each split node the following constraints are added:

$$\begin{aligned} -f(a_k, s_{i,k+1}) + f(o_{i,k}, s_{i,k+1}) &\leq 0, \\ -f(o_{i,k}, s_{i,k+1}) + f(s_{i,k+1}, v_{\cdot,k+1}) &\leq 0 \; \forall \; v_{\cdot,k+1} \text{ connected to } s_{i,k+1}. \end{aligned} \tag{9}$$

Similar to before, we derive by combining Eq. 4 and Eq. 9

$$0 \leq f(s_{i,k+1}, v_{\cdot,k+1}) \leq f(o_{i,k}, s_{i,k+1}) \leq f(a_k, s_{i,k+1}).$$

For a flow $f(s_{i,k+1}, v_{\cdot,k+1})$ from the split node $s_{i,k+1}$ to a node $v_{\cdot,k+1}$ larger than zero, the flow from the object node to the split node $f(o_{i,k}, s_{i,k+1})$ and the flow from the appear node to the split node $f(a_k, s_{i,k+1})$ need to be at least as large. The flow conservation constraint Eq. 2 enforces the same flow into a node and from a node, resulting in a flow of at least two through the split node $s_{i,k+1}$ or zero.

To distinguish an over-segmentation from a mitosis and assign different cost functions, we construct all pairs of "daughter" nodes $o_{j,k+1}$ & $o_{l,k+1}$ the "mother" node $s_{i,k+1}$ is connected to and add pairwise coupled flow variables. We refer to those pairwise coupled flow variables as $f_{jl}(\cdot,\cdot)$,where jl refers to the indices of the pair of coupled daughter nodes. From each mother cell at most one pair of daughter cells can emerge, which is modeled by connecting the split node $s_{i,k+1}$ to at most two daughter nodes

$$\begin{aligned} f_{jl}(s_{i,k+1}, o_{j,k+1}) &= f_{jl}(s_{i,k+1}, o_{l,k+1}). \\ \sum_{j} \sum_{\substack{l \\ l \neq j}} f_{jl}(s_{i,k+1}, o_{j,k+1}) &\leq 2. \end{aligned} \tag{10}$$

In addition, a split node $s_{i,k+1}$ can either model a mitosis or an over-segmentation. We enforce this by adding for all pairs of flow variables that correspond to edges starting from $s_{i,k+1}$ an inequality constraint:

$$f_{jl}(s_{i,k+1}, o_{j,k+1}) - f(s_{i,k+1}, o_{l,k+1}) \leq 1. \tag{11}$$

The number of daughter pairs grows quadratically with the number of potential daughter cells. To reduce the number of pairwise coupled flow variables, we prune the number of potential mitosis pairs to $N_{\max} = 10$ for each segmented object, based on the mitosis cost which is given in Eq. 15.

Cost functions Compared to other approaches, we choose costs $c(u, v)$ based on positional features only. We extract for each segmented object based on its segmentation mask the mask centroid $\mathbf{p}_{i,k}$, a bounding box, and a set of mask points. The bounding box is spanned by the top left and bottom right coordinates of the segmentation mask i at time point k and contains all points within the spanned rectangle, it will be referred to as $\mathcal{B}_{i,k}$. The set of mask points is derived by calculating a distance transformation on the segmentation mask and will be referred to as $\mathcal{Q}_{i,k}$, where a single point will be referred to as $\mathbf{q}_{i,k}$. A visualization of the extracted features is shown in Fig 4.

The features are propagated over time by updating their position-based features with the estimated displacement $\mathbf{d}_{i,\cdot,\cdot}$, from the tracklet step:

$$\begin{aligned}
\hat{\mathbf{p}}_{i,k+1} &= \mathbf{p}_{i,k} + \mathbf{d}_{i,k,k+1}, \\
\widehat{\mathcal{B}}_{i,k+1} &= \{\mathbf{b}_{i,k} + \mathbf{d}_{i,k,k+1} \mid \mathbf{b}_{i,k} \in \mathcal{B}_{i,k}\}, \\
\widehat{\mathcal{Q}}_{i,k+1} &= \{\mathbf{q}_{i,k} + \mathbf{d}_{i,k,k+1} \mid \mathbf{q}_{i,k} \in \mathcal{Q}_{i,k}\}.
\end{aligned} \tag{12}$$

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

$$c(o_{i,k}, o_{j,k+1}) = \|\hat{\mathbf{p}}_{i,k+1} - \mathbf{p}_{j,k+1}\|_2, \tag{13}$$

where $\hat{\mathbf{p}}_{i,k+1}$ is the estimated mask centroid of object i at time point $k+1$ and $\mathbf{p}_{j,k+1}$ the mask centroid of object j at time point $k+1$. The edge costs involving skip nodes are defined as

$$\begin{aligned}
c(o_{i,k}, x_{i,k+1}) &= \begin{cases} \|\mathbf{p}_{i,k} - \hat{\mathbf{p}}_{i,k+1}\|_2 = \|\mathbf{d}_{i,k,k+1}\|_2 & \text{if } \hat{\mathbf{p}}_{i,k+1} \notin \mathcal{B}_{j,k+1}\ \forall j \\ \theta & \text{else} \end{cases}, \\
c(x_{i,k+1}, x_{i,k+2}) &= \begin{cases} \|\hat{\mathbf{p}}_{i,k+1} - \hat{\mathbf{p}}_{i,k+2}\|_2 = \|\mathbf{d}_{i,k+1,k+2}\|_2 & \text{if } \hat{\mathbf{p}}_{i,k+2} \notin \mathcal{B}_{j,k+2}\ \forall j \\ \theta & \text{else} \end{cases}, \\
c(x_{i,k+1}, o_{j,k+2}) &= \|\hat{\mathbf{p}}_{i,k+2} - \mathbf{p}_{j,k+2}\|_2,
\end{aligned} \tag{14}$$

where θ is a large constant.

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

$$\begin{aligned}
c_1 &= \|\mathbf{p}_{i,k} - \frac{1}{2}(\mathbf{p}_{j,k+1} + \mathbf{p}_{l,k+1})\|_2, \\
c_2 &= \big|\|\mathbf{p}_{i,k} - \mathbf{p}_{j,k+1}\|_2 - \|\mathbf{p}_{i,k} - \mathbf{p}_{l,k+1}\|_2\big|, \\
c_3 &= \|\mathbf{p}_{j,k+1} - \mathbf{p}_{l,k+1}\|_2, \\
c_{jl}(s_{i,k+1}, o_{j,k+1}) = c_{jl}(s_{i,k+1}, o_{l,k+1}) &= \begin{cases} c_1 + c_2 & \text{if } c_3 \leq 1.5 b_{i,k} \\ \theta & \text{else} \end{cases},
\end{aligned} \tag{15}$$

where $b_{i,k}$ is the length of the diagonal spanned by the top left and bottom right coordinate of the bounding box. The cost enforce that daughter cells have a similar distance to the mother cell, their average position is close to the previous position of the mother cell and the distance between the daughter cells is small. An estimated position of the mother cell is not used, as the displacement estimation which is based on appearance of image crops is unreliable, when one image crop shows a single mother cell and the other shows two daughter cells.

In case of over- or under-segmentation, 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 Fig 5. To better link under- and over-segmented objects to their correctly

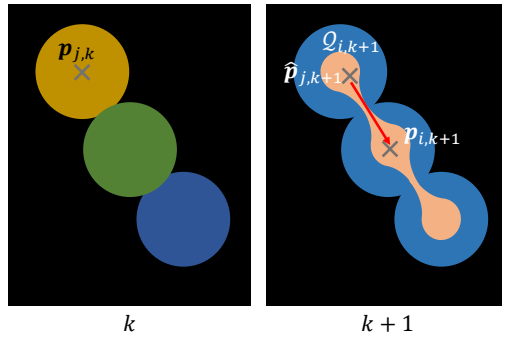


Fig 5. Linking correctly segmented objects to under-segmented objects. Shown are correctly segmented objects at time point k and a single segmented object due to an under-segmentation error at time point $k+1$. The mask centroids $\mathbf{p}_{j,k}$ and $\mathbf{p}_{i,k+1}$ and the propagated mask centroid $\hat{\mathbf{p}}_{j,k+1}$ are shown as gray crosses. The set of mask points $\mathcal{Q}_{i,k+1}$ is shown in orange. The Euclidean distance between the mask centroid $\mathbf{p}_{i,k+1}$ and the propagated mask centroid $\hat{\mathbf{p}}_{j,k+1}$ is large, which can result in wrong links. The minimal Euclidean distance between the propagated mask centroid $\hat{\mathbf{p}}_{j,k}$ and the set of mask points $\mathcal{Q}_{i,k+1}$, in contrast, is small.

segmented corresponding objects at successive time points, we use the set of mask points instead. For over-segmentation we define the costs

$$c(s_{i,k}, o_{j,k+1}) = \begin{cases} \min(\{\|\hat{\mathbf{q}}_{i,k+1} - \mathbf{p}_{j,k+1}\|_2 \mid \hat{\mathbf{q}}_{i,k+1} \in \widehat{\mathcal{Q}}_{i,k}\}) & \text{if } \mathbf{p}_{j,k+1} \in \widehat{\mathcal{B}}_{i,k+1} \\ \theta & \text{else} \end{cases}, \quad (16)$$

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

For under-segmentation, we define

$$c(o_{j,k}, m_{i,k}) = \begin{cases} \min(\{\|\mathbf{q}_{i,k+1} - \hat{\mathbf{p}}_{j,k+1}\|_2 \mid \mathbf{q}_{i,k+1} \in \mathcal{Q}_{i,k+1}\}) & \text{if } \hat{\mathbf{p}}_{j,k+1} \in \mathcal{B}_{i,k+1} \\ \theta & \text{else} \end{cases}, \quad (17)$$

where $\hat{\mathbf{p}}_{j,k+1}$ is the predicted position of the segmented object j at time point k and $\mathcal{Q}_{i,k+1}$ the set of mask points of the segmented object i at the next time point $k+1$. Appearance costs depend on a threshold α and the minimum distance of the mask centroid $\mathbf{p}_{i,k}$ to the image border

$$c(a_{k-1}, o_{i,k}) = \min\left(\alpha, \min\left(\min\left(\mathbf{a} - \mathbf{p}_{i,k}\right), \min\left(\mathbf{p}_{i,k}\right)\right)\right), \quad (18)$$

disappear costs are defined similar

$$c(o_{i,k}, d_{k+1}) = \min\left(\alpha, \min\left(\min\left(\mathbf{a} - \mathbf{p}_{i,k}\right), \min\left(\mathbf{p}_{i,k}\right)\right)\right), \quad (19)$$

where $\mathbf{a}$ is the image size and $\min\left(\min\left(\mathbf{a} - \mathbf{p}_{i,k}\right), \min\left(\mathbf{p}_{i,k}\right)\right)$ the minimal distance to the image border.

We set the parameter $\theta = 1000\alpha$, where α is derived from the largest edge of the default size of the ROI, which is provided in the section parameter selection. All other edges are assigned 0 cost. An overview of the calculated costs is shown in Fig 3 for parts of the example graph from Fig 2.

To reduce the number of flow variables even further, edges with large costs are pruned. The formulated problem can be solved using integer linear programming with a standard optimization toolbox such as Gurobi [52].

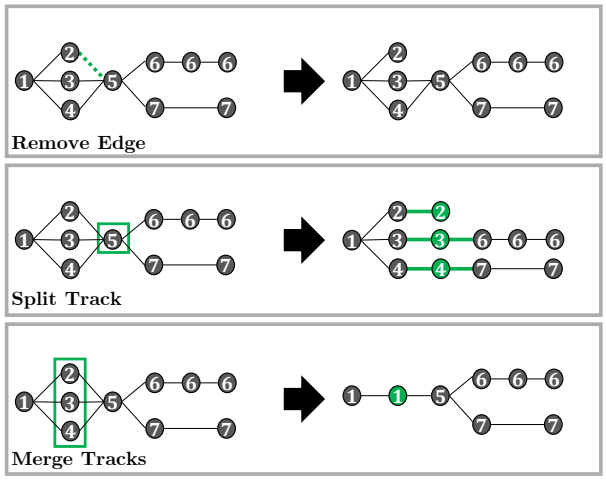


Fig 6. Untangling operations to modify the tracking graph. The tracking graph is modified by applying a set of untangling operations such that each track has at most one predecessor and at most two successors to model mitosis. The in green highlighted parts show how an untangling operation modifies the tracking graph.

The tracking graph is constructed by following the optimal 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.

Step 3: Post-processing step

In the post-processing, over- and under-segmentation errors are resolved and missing segmentation masks are added to resolve FNs.

Untangling problem After the matching step, tracks can be assigned to more than one predecessor and/or more than two successors as shown in Fig 1. 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", we will refer to this step as untangling step. We transform the tracking graph by applying a set of modifications on the tracking graph which will be 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, whereas the merge tracks operation merges several tracks resulting in a single track. A visualization of the untangling operations is shown in Fig 6.

Different combinations of untangling operations lead to valid tracking graphs, which is shown in Fig 7. The problem is to select a combination of untangling operations, which we model as an optimization problem

$$\min_x \sum_i c_i x_i, \tag{20}$$

where c_i are costs and x_i variables referring to untangling operations on the graph.

The untangling operations x_i are denoted as follows: an edge remove operation will be denoted as e_{ij}, where the predecessor track is ω_i and the successor track is ω_j. Splitting a track ω_i into several tracks will be denoted as s_i, whereas merging a set of tracks will be denoted as m_l, where l is a multi index that indicates a set of tracks.

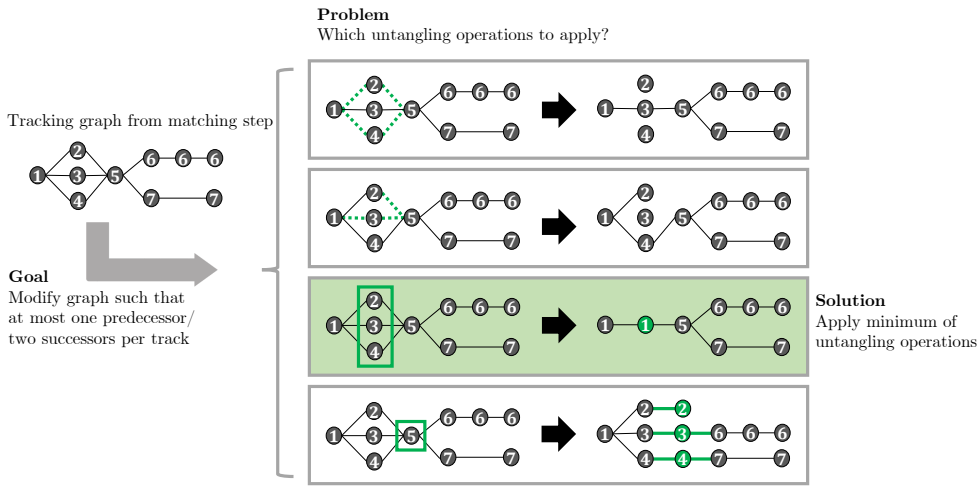


Fig 7. Selecting a set of untangling operations. The tracking graph shall be modified, however, different combinations of untangling operations lead to several valid tracking graphs. We model the problem of selecting a set of untangling operations as an optimization problem.

To merge tracks, the tracks need to: a) share the same predecessors and successors, b) share the same successors and some tracks have no predecessor but begin after the track with a predecessor starts, or c) share the same predecessors and some tracks have no successors but end before the track with successors end. Based on the aforementioned conditions, we construct all possible sets of mergeable tracks.

Moreover, each track should have at maximum one predecessor and at maximum two successors. This is modeled by two constraints, one for the predecessor side and one for the successor side for each track. The number of predecessors of a track ω_i is referred to as $|\mathcal{P}_i|$ whereas the number of successors of a track is referred to as $|\mathcal{S}_i|$. For each set of tracks l that can be merged, indicated by m_l, the number of tracks sharing the same set of predecessors as track ω_i is denoted as $P_{i,l}$ and the number of tracks sharing the same set of successors as track ω_i as $S_{i,l}$. Furthermore, for each predecessor track of track ω_i all sets of tracks the predecessor track can be merged with are computed, where the set $\mathcal{M}(\mathcal{P}_i)$ consists of all possible sets of mergeable tracks that contain predecessor tracks of track ω_i. Analogously for each successor track of track ω_i all sets of tracks the successor track can be merged with are computed, where the set $\mathcal{M}(\mathcal{S}_i)$ consists of all possible sets of mergeable tracks that contain successor tracks of track ω_i.

For each track we add one inequality constraint to enforce at most one predecessor and one inequality constraint to enforce at most two successors. As the modification of one track influences also its predecessors and successors, the predecessor inequality also includes the untangling operations on the predecessor tracks, whereas the successor inequality includes the untangling operations on the successor tracks. Furthermore, as tracks can be linked to more than one predecessor and more than two successors, the tracks which share the same predecessors or successors need to be considered in the inequality constraints as well. The predecessor inequality constraint for track ω_i is given

as:

$$\overbrace{\sum_{l\in\mathcal{M}_i}(P_{i,l}-1)m_l}^{\text{merge tracks}} - \overbrace{\sum_{k\in\mathcal{W}_i} s_k}^{\text{split tracks}} - \overbrace{\sum_{k\in\mathcal{W}_i}\sum_{v\in\mathcal{P}_k} e_{vk}}^{\text{remove edges to predecessors}}$$
$$+ \underbrace{\sum_{v\in\mathcal{P}_i} s_v}_{\text{split predecessors}} + \underbrace{\sum_{q\in\mathcal{M}(\mathcal{P}_i)} \min(0, -S_{q,i}+1)m_q}_{\text{merge predecessors}} \leq -|\mathcal{P}_i| + \max(1, |\{\mathcal{S}_v \mid v\in P_i\}|), \tag{21}$$

where l and q are multi indices referring to sets of mergeable tracks and k and v are indices referring to a single track. The variables to be optimized are the merge track variables m_l and m_q, the split track variables s_k and s_v, and the edge remove variables e_{vk}, where m_l denotes merging the set of tracks l into a single track, s_k denotes splitting the track ω_k into several tracks, and e_{vk} denotes removing the predecessor-successor link between the predecessor track ω_v and the successor track ω_k. The set $\mathcal{M}_i$ contains all sets of tracks that can be merged with the track ω_i, $P_{i,l}$ is the number of tracks of the set of mergeable tracks l that share the same predecessors as ω_i, $\mathcal{W}_i$ is a set which contains all tracks, including ω_i, that can be merged with track ω_i. $|\mathcal{P}_i|$ is the number of predecessors of track ω_i, whereas $\mathcal{P}_k$ is the set of predecessors of track ω_k. $S_{q,i}$ is the number of tracks of the set of mergeable tracks q that have track ω_i as a successor. The total number of successors of the predecessors of track ω_i is given by $|\{\mathcal{S}_v \mid v\in P_i\}|$.

The successor inequality constraint is given as:

$$\overbrace{\sum_{l\in\mathcal{M}_i}(S_{i,l}-1)m_l}^{\text{merge tracks}} - \overbrace{\sum_{k\in\mathcal{W}_i} s_k}^{\text{split tracks}} - \overbrace{\sum_{k\in\mathcal{W}_i}\sum_{t\in\mathcal{S}_k} e_{kt}}^{\text{remove edges to successors}}$$
$$+ \underbrace{\sum_{t\in\mathcal{S}_i} s_t}_{\text{split successors}} + \underbrace{\sum_{r\in\mathcal{M}(\mathcal{S}_i)} \min(0, -P_{r,i}+1)m_r}_{\text{merge successors}} \leq -|\mathcal{S}_i| + 2|\{\mathcal{P}_t \mid t\in S_t\}| + 1, \tag{22}$$

where l and r are multi indices referring to sets of mergeable tracks and k and t are indices referring to a single track. The variables to be optimized are the merge tracks variables m_l and m_r, the split track variables s_k and s_t, and the edge remove variables e_{kt}, where m_l denotes merging the set of tracks l into a single track, s_k denotes splitting the track ω_k into several tracks, and e_{kt} denotes removing the predecessor-successor link between the predecessor track ω_k and the successor track ω_t. The set $\mathcal{M}_i$ contains all sets of tracks that can be merged with the track ω_i, $S_{i,l}$ is the number of tracks of the set of mergeable tracks l that share the same successors as track ω_i, $P_{r,i}$ is number of tracks of the set of mergeable tracks r that have track ω_i as a predecessor, and $\mathcal{W}_i$ is a set which contains all tracks, including ω_i, that can be merged with track ω_i. $|\mathcal{S}_i|$ is the number of successors of track ω_i, whereas $\mathcal{S}_k$ is the set of successors of track ω_k. $P_{r,i}$ is the number of tracks of the set of mergeable tracks r that have track ω_i as a predecessor. The total number of predecessors of the successors of track ω_i is given by $|\{\mathcal{P}_t \mid t\in S_t\}|$.

A track can be merged with at most one set of tracks l, which we model by adding for each track a constraint

$$\sum_{l\in\mathcal{M}_i} m_l \leq 1. \tag{23}$$

In addition, 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, we construct from each set of mergeable tracks l all pairs

of tracks which share a predecessor or successor and add two constraints

$$\begin{aligned} e_{kj} - e_{ki} &\leq -m_l + 1, \\ -e_{kj} + e_{ki} &\leq -m_l + 1, \end{aligned} \tag{24}$$

where $l = \{i, j, \dots\}$ and the tracks ω_i and ω_j share the predecessor track ω_k. The merge tracks and edge remove variables are constraint to be binary variables, whereas the split variables are of integer type to provide the number of tracks a track will be split into.

Predecessor and successor inequality constraints example We illustrate the setup of the proposed inequality constraints from Eq. 21 and Eq. 22 for the track with track ID 5 from the tracking graph shown in Fig 7. 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 $\mathcal{W}_5 = \{5\}$. 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 $|\{\mathcal{S}_v \mid v \in P_5\}| = 1$. The successor tracks of track 5 have only one predecessor, which is track 5, resulting in $|\{\mathcal{P}_t \mid t \in S_t\}| = 1$.

By merging predecessor tracks or successor tracks into a single track, the number of predecessors or successors 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 $(P_{i,l} - 1)$ and $\min(0, -S_{q,i} + 1)$ from Eq. 21, and $(S_{i,l} - 1)$ and $\min(0, -P_{r,i} + 1)$ in Eq. 22. For example, by merging the tracks $\{2, 3, 4\}$ into a single track, which is modeled by $m_{\{2,3,4\}}$, two predecessor links of track 5 are removed, as now instead of three predecessor tracks only one predecessor track is connected to it. Therefore, $m_{\{2,3,4\}}$ is multiplied by a factor of 2.

After inserting the terms in the inequality constraints, we derive for the predecessor inequality constraint of track 5

$$\begin{aligned} &\overbrace{0}^{\text{merge tracks}} - \overbrace{s_5}^{\text{split tracks}} - \overbrace{(e_{2,5} + e_{3,5} + e_{4,5})}^{\text{remove edges to predecessors}} \\ &+ \underbrace{s_2 + s_3 + s_4}_{\text{split predecessors}} \underbrace{-2m_{\{2,3,4\}} - m_{\{2,3\}} - m_{\{2,4\}} - m_{\{3,4\}}}_{\text{merge predecessors}} \leq -3 + 1 = -2, \end{aligned} \tag{25}$$

and for the successor inequality constraint

$$\begin{aligned} &\overbrace{0}^{\text{merge tracks}} - \overbrace{s_5}^{\text{split tracks}} - \overbrace{(e_{5,6} + e_{5,7})}^{\text{remove edges to successors}} \\ &+ \underbrace{s_6 + s_7}_{\text{split successors}} \underbrace{-m_{\{6,7\}}}_{\text{merge successors}} \leq -2 + 2 + 1 = 1. \end{aligned} \tag{26}$$

The successor inequality constraint Eq. 26 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 Eq. 21 holds, as the right hand side of the inequality constraint is -2. This makes sense, 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. Here we propose simple cost terms based on the temporal length and number of merged tracks:

$$\begin{aligned} c(e_{ij}) &= \gamma, \\ c(m_l) &= \Delta\omega_l(N_l - 1), \\ c(s_i) &= \Delta\omega_i, \end{aligned} \tag{27}$$

where γ is a constant, N_l is the number of merged tracks, $\Delta\omega_i$ is the temporal length of the track ω_i and $\Delta\omega_l$ is the temporal length of the track after merging, respectively. For the chosen cost functions, merging n tracks or splitting a track in n parts over the same time span, results in the same change of the value of the objective function. 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. To define a cut off when removing edges is more beneficial than modifying tracks, we set γ to $2\lceil \Delta T_{0.3} \Delta N_{0.99} \rceil$, 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 an integer linear program using a standard optimization toolbox such as Gurobi [52]. After solving the optimization problem, the untangling operations are applied to the selected tracks. Tracks are split by computing for each mask s_i seed points, where s_i is the value of the split variable from the optimization problem. Based on the seed points, a nearest neighbor approach is applied to the mask and each mask pixel is assigned to the closest seed point, resulting in s_i segmentation masks. To merge tracks, their segmentation masks are concatenated for each time point.

FN correction Finally, we correct FN errors by adding segmentation masks to tracks with missing segmentation masks. We place the last available segmentation mask, before a FN error occurs, at positions computed from a linear interpolation between the available segmentation masks. In image sequences with touching cells, adding masks can lead to conflicts, where an interpolated mask overlaps with another segmentation mask. We resolve those mask conflicts by assigning conflicting pixels to the segmentation mask with the closest centroid.

Data sets

We select the cell data sets Fluo-N2DH-SIM+ and Fluo-N3DH-SIM+ from the CTC [5, 50] for evaluation, as they are publicly available and provide a fully annotated ground truth, i.e. segmentation masks are given for all cells as well as the cell lineage. Both cell data sets show simulated human leukemia cells, where Fluo-N2DH-SIM+ is a 2D data set and Fluo-N3DH-SIM+ is a 3D data set. Per data set two image sequences are available which will be referred to as sequence 01 and 02. Statistics of the chosen data sets are shown in Table 1.

Simulation of segmentation errors We modify a fixed fraction of $n\%$ of the ground truth segmentation masks, to simulate data sets with an erroneous segmentation. We model the segmentation errors FNs, under- and over segmentation, and the combination of the aforementioned segmentation errors. FNs are simulated by removing segmentation masks randomly, the resulting data sets are referred to as "FN error". Over-segmentation is simulated by splitting segmentation masks randomly in two parts and is referred to as "over-segmentation error". Under-segmentation is simulated by selecting neighboring segmentation masks randomly and merging them to a single mask

Table 1. Statistics of cell data sets. Information about the number of frames, tracks and cells of the CTC data sets.

Data Set	**Fluo-N2DH-SIM+**		**Fluo-N3DH-SIM+**	
Sequence	01	02	01	02
N frames	65	150	150	80
N tracks	95	107	81	117
avg. number of cells/frame	40	22	19	43
min number of cells/frame	30	8	6	30
max number of cells/frame	47	54	43	55

by applying a morphological closing operation. Data sets showing this error type will be referred to as "under-segmentation error". Furthermore, the error types are mixed by combining FN, under- and over-segmentation errors equally so in total $n\%$ of the segmentation masks are modified, which is referred to as "mixed error".

FN and over-segmentation errors are simulated by drawing uniformly from the set of segmentation masks until a fraction of $n\%$ of the ground truth masks is modified. Under-segmentation errors, in contrast, are sampled by constructing neighbor pairs of segmentation masks and assigning them a sampling weight proportional to their distance. As a result, cells with a smaller distance have a higher probability to be merged, which is also the case in real segmentation data. Segmentation masks are merged iteratively until a fraction of $n\%$ of the ground truth masks is merged, also allowing more than two cells to be merged.

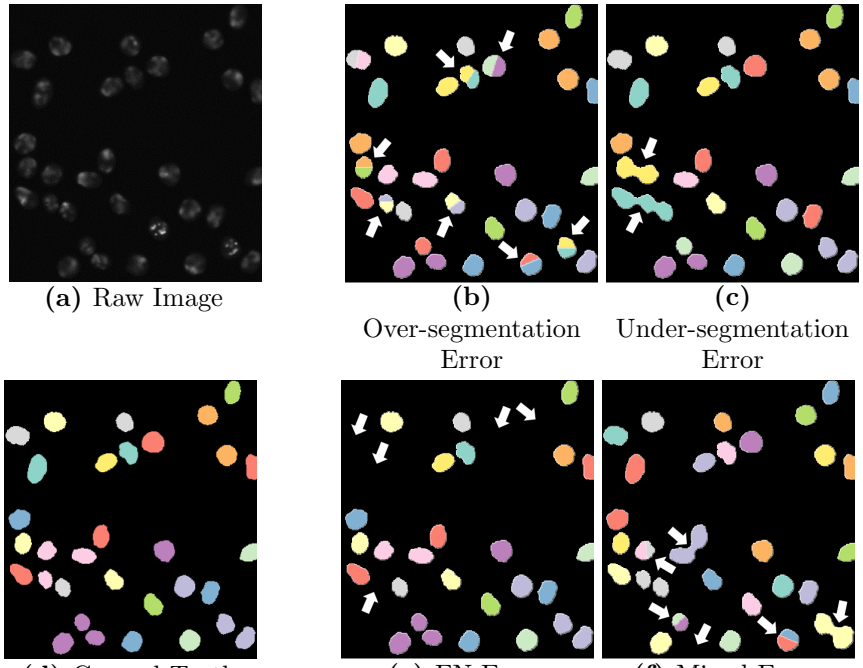
For each cell data set and image sequence we modify $n = 1, 2, 5, 10, 20$ % of the ground truth masks and generate $N = 5$ runs for each defined segmentation error which results in a total of 400 data sets for evaluation. A visualization of a raw image with corresponding ground truth and simulated segmentation errors is shown in Fig 8.

Evaluation measure

We use the TRA measure [5] from the CTC, to evaluate the tracking performance. In the TRA measure, graphs are constructed from the ground truth and the tracking data. Nodes in these graphs represent segmented objects, whereas edges represent links between the segmented objects over time. The tracking data graph is transformed into the ground truth graph by applying untangling operations: add/remove/split node, add/remove an edge and edit the edge semantic. Each graph operation results in a penalty, where adding nodes (FN) is penalized the most. The final measure is normalized between 0 and 1, where 1 means that ground truth graph and tracking data graph match perfectly.

Compared tracking algorithms

We select three tracking by detection approaches from former CTC participants to compare our approach with: Mu-Lux-CZ, KIT-Sch-GE, and KTH-SE (`http://celltrackingchallenge.net/participants/`). All approaches provide an implementation which we used for comparison. The MU-Lux-CZ algorithm [15] is an overlap-based approach for 2D, which we extended to 3D. The tracking algorithm links segmentation masks between successive frames if their overlap is larger than a fixed threshold. Due to the simplicity of the algorithm, no automatic segmentation error correction is available. The KIT-Sch-GE algorithm [9, 53] implements a coupled minimum cost flow algorithm which is capable to detect mitosis and handles FNs for short time spans. The KTH-SE algorithm [30, 54] uses the Viterbi algorithm to link cells. It includes segmentation error correction for FP, FN, over- and under-segmentation.



(a) Raw Image **(b)** Over-segmentation Error **(c)** Under-segmentation Error

(d) Ground Truth **(e)** FN Error **(f)** Mixed Error

Fig 8. Simulated segmentation errors. Shown is a raw image of the Fluo-N2DH-SIM+ 01 data set with corresponding ground truth segmentation masks and modified segmentation masks with simulated segmentation errors, highlighted with white arrows.

Compared to the tracking algorithm of Scherr et al. (Team KIT-Sch-GE in CTC) [9, 53], our proposed tracking models FNs as skip nodes as well as over- and under-segmentation of two or more objects. Moreover, we propose different costs and the untangling post-processing step to correct segmentation errors automatically.

Parameter selection

For the tracking approaches MU-Lux-CZ, KIT-Sch-GE, and KTH-SE we kept the same parameters as provided by their CTC submission, and only modified the algorithms such that they use the provided erroneous segmentation masks instead of using their own segmentation.

For our tracking algorithm we manually set two parameters: Δt, and the default ROI size. We set $\Delta t = 3$ and the default ROI size to twice of the average segmentation mask size. All other parameters are estimated automatically from the data or are based on these two parameters.

Results

Post-processing analysis We investigate the influence of the post-processing steps, untangling tracks and FN correction, by modifying the post-processing step, while keeping all other steps the same. 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 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. In the following, we will refer to the untangling step as untangle and the FN correction step as masks.

For under- and over-segmentation errors, tracking approaches without the untangling step, indicated by untangle, perform worse as shown in Fig 9a and Fig 9b for 2D and in Fig 10a and Fig 10b for 3D data sets. On data sets with FN errors, tracking approaches without the FN correction step, indicated by masks, perform worse which is shown for 2D data sets in Fig 9c and for 3D data sets in Fig 10c. When combining different segmentation error types, applying both post-processing steps performs best, which is shown in Fig 9d and Fig 10d.

Tracking performance comparison We compare the performance of our proposed approach including the proposed post-processing to the tracking approaches of KTH-SE, KIT-Sch-GE, and MU-Lux-Sch on erroneous segmentation data. The results are shown in Fig 11 for 2D data sets and in Fig 12 for 3D data sets.

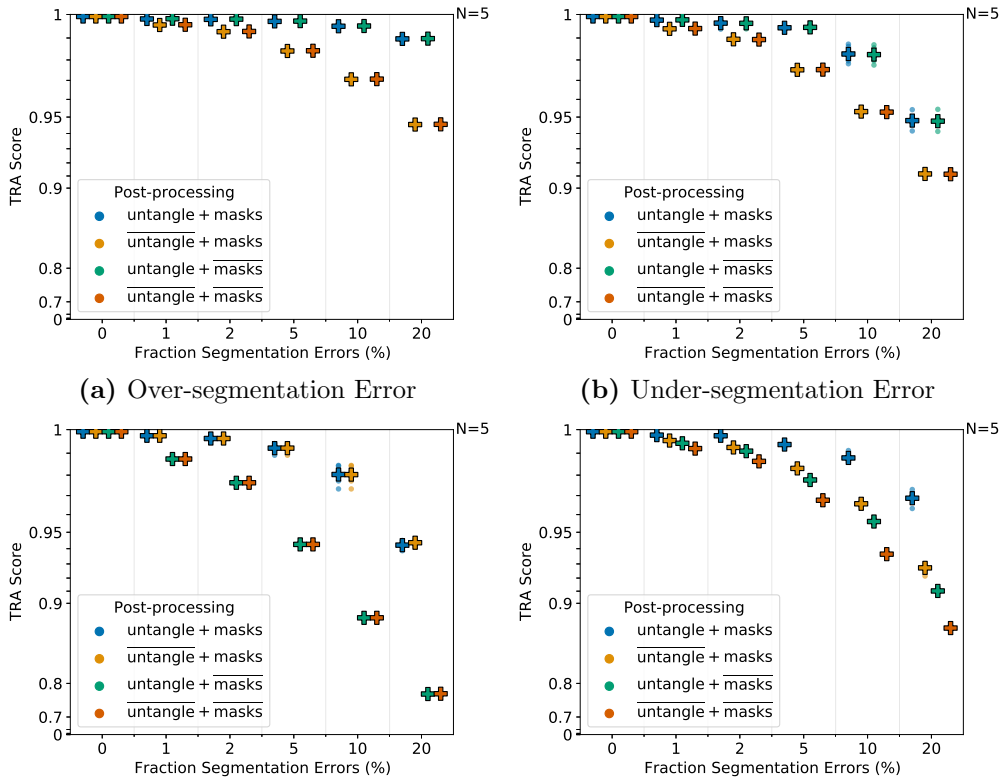
For under-segmentation errors, our approach performs best, as shown in Fig 11b and Fig 12b. On data sets with over-segmentation errors, our approach and the KTH-SE approach perform similarly, which is shown in Fig 11a and Fig 12a. In case of FN errors, our approach and the KIT-Sch-GE approach perform similarly, as shown in Fig 11c and Fig 12c. For data sets with a combination of segmentation error types, our approach outperforms all other approaches, as shown in Fig 11d and Fig 12d.

Run-time comparison We compare the run-time of the tracking algorithms when provided with perfect, ground truth segmentation data and when provided with erroneous segmentation data. As erroneous segmentation we choose the 20% mixed error segmentation data. We evaluated all tracking algorithms on a desktop computer with an Intel Core i7-6700 processor and 64GB of RAM running Python 3.7 and MATLAB 2018b in Windows 10.

For the KTH-SE algorithm, which is implemented in MATLAB, we used the `tic/toc` functionality of MATLAB for benchmarking, whereas for all other algorithms, which are implemented in Python, we used the `default_timer` of the `timeit` package. The run-times are shown in Table 2. The results show that the proposed method can track 2D and 3D data sets in reasonable times.

Table 2. Run-times of tracking algorithms. Run times of the tracking algorithms on 2D and 3D data sets when provided with perfect ground truth (GT) segmentation as well as when provided with erroneous segmentation data.

Tracking Algorithm	**Data Sets**			
	Fluo-N2DH-SIM+ 01		Fluo-N3DH-SIM+ 01	
	GT	Erroneous	GT	Erroneous
Proposed	21.82 s	22.94 s	1238.38 s	1179.69 s
MU-Lux-CZ	25.79 s	26.65 s	1055.12 s	1047.86 s
KTH-SE	39.73 s	44.22 s	523.84 s	404.34 s
KIT-Sch-GE	15.11 s	19.56 s	1004.63 s	1585.43 s



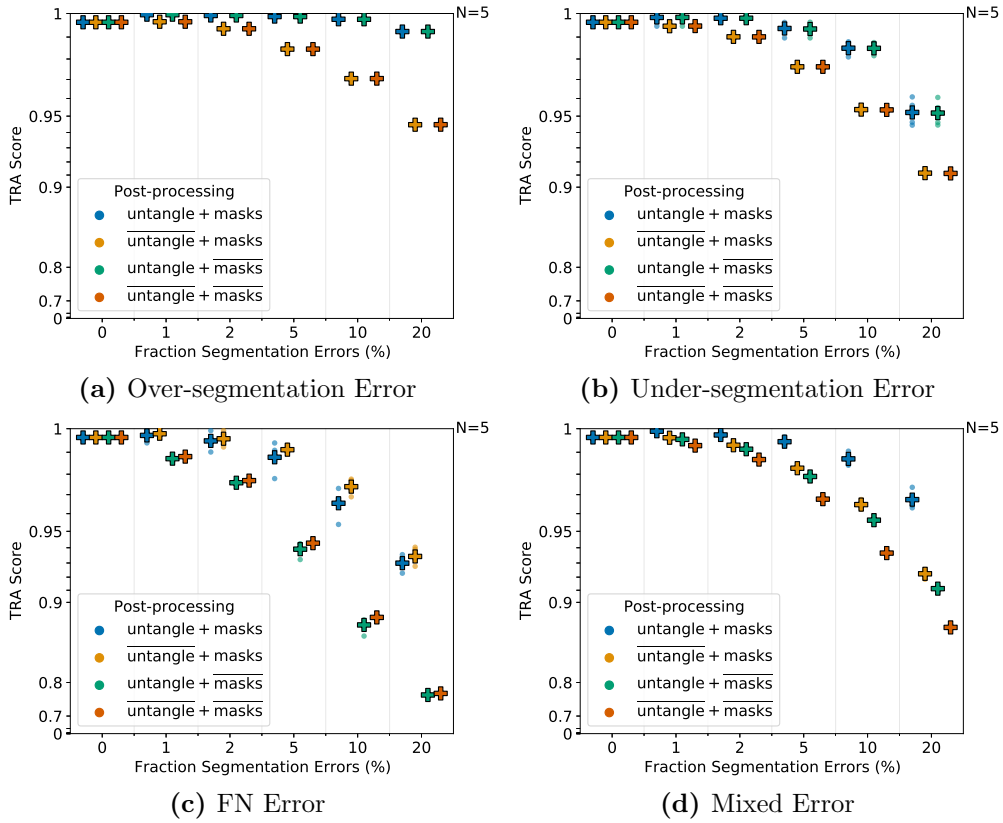
(a) Over-segmentation Error

(b) Under-segmentation Error

(c) FN Error

(d) Mixed Error

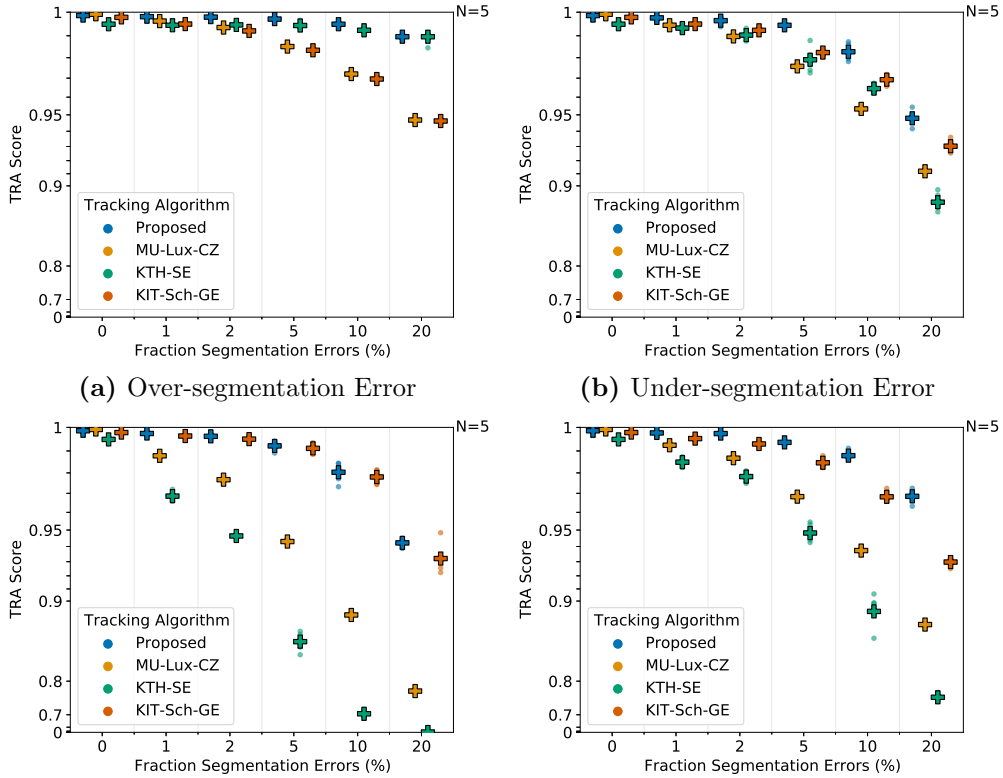
Fig 9. Influence of the post-processing on Fluo-N2DH-SIM+. Influence of our proposed post-processing on the TRA score for the 2D data set Fluo-N2DH-SIM+ 01. Scores of a single run are shown as circles, while + shows the TRA score averaged over $N = 5$ runs. Per run a fixed fraction of ground truth segmentation masks is modified randomly to simulate segmentation errors. "untangle" refers to the untangling step, which transforms the tracking graph such that each track has at most one predecessor and two successors, whereas "masks" refers to adding missing segmentation masks. Over lined post-processing steps $\overline{(\dots)}$ indicate that the post-processing step is missing.



(a) Over-segmentation Error **(b)** Under-segmentation Error

(c) FN Error **(d)** Mixed Error

Fig 10. Influence of the post-processing on Fluo-N3DH-SIM+. Influence of our proposed post-processing on the TRA score for the 3D data set Fluo-N3DH-SIM+ 01. Scores of a single run are shown as circles, while + shows the TRA score averaged over $N = 5$ runs. Per run a fixed fraction of ground truth segmentation masks is modified randomly to simulate segmentation errors. "untangle" refers to the untangling step, which transforms the tracking graph such that each track has at most one predecessor and two successors, whereas "masks" refers to adding missing segmentation masks. Over lined post-processing steps $\overline{(\ldots)}$ indicate that the post-processing step is missing.



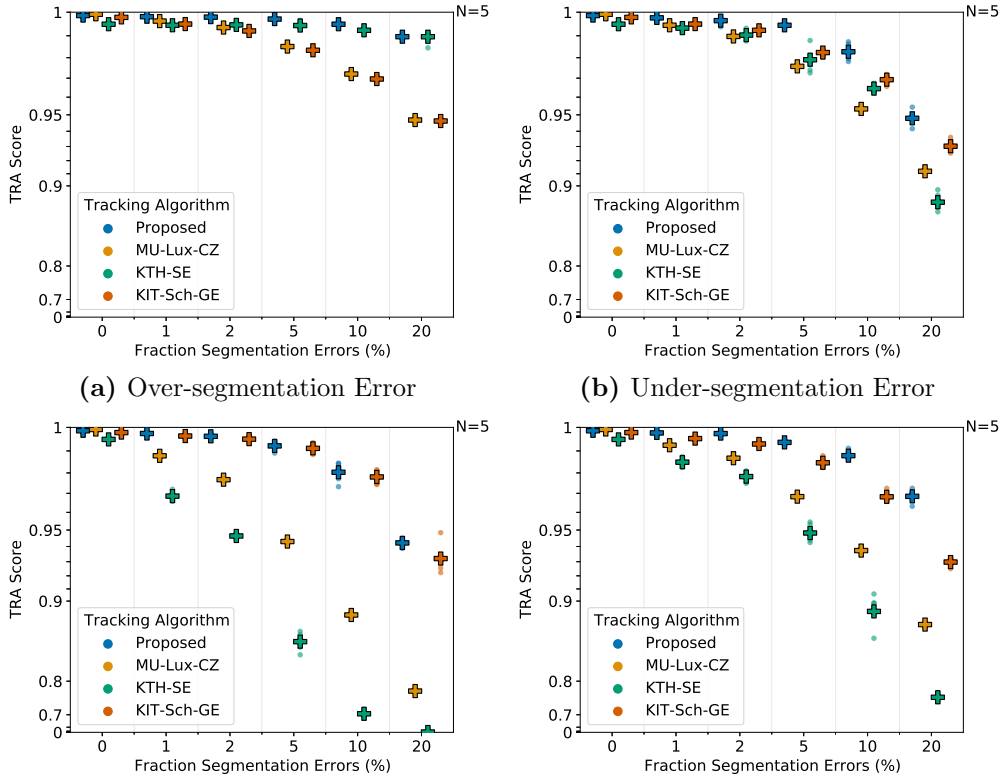
(a) Over-segmentation Error

(b) Under-segmentation Error

(c) FN Error

(d) Mixed Error

Fig 11. Comparing tracking algorithms on Fluo-N2DH-SIM+. TRA scores of tracking algorithms on 2D data set Fluo-N2DH-SIM+ 01 when provided with the same erroneous segmentation data. Scores of a single run are shown as circles, while + shows the TRA score averaged over $N = 5$ runs.



(a) Over-segmentation Error

(b) Under-segmentation Error

(c) FN Error

(d) Mixed Error

Fig 12. Comparing tracking algorithms on Fluo-N3DH-SIM+. TRA scores of tracking algorithms on 3D data set Fluo-N3DH-SIM+ 01 when provided with the same erroneous segmentation data. Scores of a single run are shown as circles, while + shows the TRA score averaged over $N = 5$ runs.

Discussion

Our proposed tracking method can correct the segmentation errors FN, under- and over-segmentation and yields a good tracking quality while having a reasonable run-time. The proposed post-processing with untangling step and FN correction improves in most cases the TRA score. However, when correcting FN errors on the 3D data set, shown in Fig 10c, we observe that applying both post-processing steps, referred to as untangle + masks, performs worse compared to just applying the mask interpolation step, referred to as untangle + masks. We examined the TRA score in more detail and found that the scores of the untangle + masks post-processing had more FPs. In some cases if a segmentation mask is missing and another segmentation mask is spatially close, the track with missing segmentation masks is linked to the spatially close track in the matching step, which is then linked to two segmentation masks a few time points later when the object is segmented again. As a result the spatially close track has two predecessors assigned, which is resolved in the untangling step of the post-processing by splitting the track. In the TRA score this behavior is reported as FPs. An approach to resolve this, is using more complex cost functions in the untangling and the matching step, which for instance include information on the cell appearance.

All tracking approaches do not reach a perfect TRA score of one on data sets with reduced segmentation quality. However, our approach is able to correct different types of segmentation errors without needing training data, a large set of parameters or extensive parameter tuning. Especially on data sets with a mixture of segmentation errors, our proposed method showed its potential as an "allrounder" method. To further improve tracking results, a manual correction step can be applied.

The similar performance of our approach and the KIT-Sch-GE approach, shown in Fig 11c and Fig 12c, is due to the capability of both approaches to link tracks with missing masks over a maximum time span Δt, where both approaches set $\Delta t = 3$.

To our surprise the in the CTC [50] well-performing approach of KTH-SE drops in performance when provided with segmentation data which include under-segmentation and/or FN errors. We examined the predicted tracking masks and observed that the approach removes some segmentation masks. Besides removing merged segmentation masks, sometimes also masks without added segmentation errors are removed by the tracking algorithm. In the TRA measure, FN errors are penalized twice as much as not resolving an under-segmentation error. Hence, the MU-Lux-CZ approach, which applies no modification on the segmentation masks at all, performs better than the KTH-SE approach on all data sets which add FN errors and/or under-segmentation errors.

Conclusion

We proposed a graph-based tracking approach with automatic correction of FN and under-/over-segmentation errors of two or more segmented objects. Our approach neither needs training data to learn cost functions nor has a vast set of parameters that need manual tuning. We investigated the performance of our approach on a 2D and a 3D cell data set with synthetically degraded segmentation masks simulating FN, under-segmentation, over-segmentation and a combination of the aforementioned segmentation errors. We evaluated the tracking performance using the TRA measure. For a fair comparison, we compared the performance of our tracking approach against three other tracking methods on the same erroneous segmentation data. Our proposed tracking algorithm is capable to correct certain types of segmentation errors without requiring additional training steps or parameter tuning automatically. Furthermore, on data sets with under-segmentation or a combination of different segmentation errors, our approach outperformed all other approaches, especially a parameter heavy tracking

algorithm with automated segmentation error correction. Directions of future work could be incorporation of more complex cost functions for the matching and the untangling step, using more enhanced methods for the position estimation step, and application to other data sets.

Supporting information

S1 Fig. Influence post-processing on Fluo-N2DH-SIM+ 02.

S2 Fig. Influence post-processing on Fluo-N3DH-SIM+ 02.

S3 Fig. Comparing tracking algorithms on Fluo-N2DH-SIM+ 02.

S4 Fig. Comparing tracking algorithms on Fluo-N3DH-SIM+ 02.

S1 Table Run-times of tracking algorithms on image sequences 02.

References

1. Ortega-Carrion A, Feo-Lucas L, Vicente-Manzanares M. Cell migration. In: Bradshaw RA, Stahl PD, editors. Encyclopedia of Cell Biology. Waltham: Academic Press; 2016. p. 720–730. doi:10.1016/B978-0-12-394447-4.20070-9.

2. Laviña B, Castro M, Niaudet C, Cruys B, Álvarez-Aznar A, Carmeliet P, et al. Defective endothelial cell migration in the absence of Cdc42 leads to capillary-venous malformations. Development. 2018;145(13). doi:10.1242/dev.161182.

3. Mosabbir AA, Qudrat A, Truong K. Engineered cell migration to lesions linked to autoimmune disease. Biotechnology and Bioengineering. 2018;115(4):1028–1036. doi:10.1002/bit.26523.

4. Entschladen F, Drell TL IV, Lang K, Joseph J, Zaenker KS. Tumour-cell migration, invasion, and metastasis: navigation by neurotransmitters. The Lancet Oncology. 2004;5(4):254–258. doi:10.1016/S1470-2045(04)01431-7.

5. Maška M, Ulman V, Svoboda D, Matula P, Matula P, Ederra C, et al. A benchmark for comparison of cell tracking algorithms. Bioinformatics. 2014;30(11):1609–1617. doi:10.1093/bioinformatics/btu080.

6. Meijering E, Dzyubachyk O, Smal I, van Cappellen WA. Tracking in cell and developmental biology. Seminars in Cell & Developmental Biology. 2009;20(8):894–902. doi:10.1016/j.semcdb.2009.07.004.

7. Stringer C, Wang T, Michaelos M, Pachitariu M. Cellpose: a generalist algorithm for cellular segmentation. Nature Methods. 2021;18(1):100–106. doi:10.1038/s41592-020-01018-x.

8. Falk T, Mai D, Bensch R, Çiçek O, Abdulkadir A, Marrakchi Y, et al. U-Net: deep learning for cell counting, detection, and morphometry. Nature Methods. 2019;16(1):67–70. doi:10.1038/s41592-018-0261-2.

9. Scherr T, Löffler K, Böhland M, Mikut R. Cell segmentation and tracking using CNN-based distance predictions and a graph-based matching strategy. PLOS ONE. 2020;15(12):1–22. doi:10.1371/journal.pone.0243219.

10. Weigert M, Schmidt U, Haase R, Sugawara K, Myers G. Star-convex polyhedra for 3D object detection and segmentation in microscopy. In: 2020 IEEE Winter Conference on Applications of Computer Vision (WACV); 2020. p. 3655–3662. doi:10.1109/WACV45572.2020.9093435.

11. Guerrero Peña FA, Marrero Fernandez PD, Tarr PT, Ren TI, Meyerowitz EM, Cunha A. J regularization improves imbalanced multiclass segmentation. In: 2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI); 2020. p. 1–5. doi:10.1109/ISBI45749.2020.9098550.

12. Kondratiev AY, Yaginuma H, Okada Y, Krylov AS, Sorokin DV. A method for automatic tracking of cell nuclei with weakly-supervised mitosis detection in 2D microscopy image sequences. In: Proceedings of the 2020 5th International Conference on Biomedical Signal and Image Processing. New York, NY, USA: Association for Computing Machinery; 2020. p. 67–73. doi:10.1145/3417519.3417558.

13. Wang Z, Yin L, Wang Z. A new approach for cell detection and tracking. IEEE Access. 2019;7:99889–99899. doi:10.1109/ACCESS.2019.2930539.

14. Zhou Z, Wang F, Xi W, Chen H, Gao P, He C. Joint multi-frame detection and segmentation for multi-cell tracking. In: Zhao Y, Barnes N, Chen B, Westermann R, Kong X, Lin C, editors. Image and Graphics. Cham: Springer International Publishing; 2019. p. 435–446. doi:10.1007/978-3-030-34110-7_36.

15. Lux F, Matula P. MU-Lux-CZ; 2020. Available from: `http://celltrackingchallenge.net/participants/MU-Lux-CZ/`.

16. Chang YH, Yokota H, Abe K, Tang CT, Tasi MD. Automated detection and tracking of cell clusters in time-lapse fluorescence microscopy images. Journal of Medical and Biological Engineering. 2017;37(1):18–25. doi:10.1007/s40846-016-0216-y.

17. Hirose O, Kawaguchi S, Tokunaga T, Toyoshima Y, Teramoto T, Kuge S, et al. SPF-CellTracker: Tracking multiple cells with strongly-correlated moves using a spatial particle filter. IEEE/ACM Transactions on Computational Biology and Bioinformatics. 2018;15(6):1822–1831. doi:10.1109/TCBB.2017.2782255.

18. He T, Mao H, Guo J, Yi Z. Cell tracking using deep neural networks with multi-task learning. Image and Vision Computing. 2017;60:142–153. doi:10.1016/j.imavis.2016.11.010.

19. Wang Y, Mao H, Yi Z. Stem cell motion-tracking by using deep neural networks with multi-output. Neural Computing and Applications. 2019;31(8):3455–3467. doi:10.1007/s00521-017-3291-2.

20. Xu B, Shi J, Lu M, Cong J, Wang L, Nener B. An automated cell tracking approach with multi-Bernoulli filtering and ant colony labor division. IEEE/ACM transactions on computational biology and bioinformatics. 2019;PP. doi:10.1109/tcbb.2019.2954502.

21. Hossain MI, Gostar AK, Bab-Hadiashar A, Hoseinnezhad R. 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. p. 1–5. doi:10.23919/ICIF.2018.8455486.

22. Santella A, Du Z, Bao Z. A semi-local neighborhood-based framework for probabilistic cell lineage tracing. BMC Bioinformatics. 2014;15(1):217. doi:10.1186/1471-2105-15-217.

23. Meng S, Shen H. A robust cell tracking framework by fusing global and local optimization algorithms. In: 2016 Chinese Control and Decision Conference (CCDC); 2016. p. 2079–2084. doi:10.1109/CCDC.2016.7531327.

24. Bise R, Yin Z, Kanade T. Reliable cell tracking by global data association. In: 2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro; 2011. p. 1004–1010. doi:10.1109/ISBI.2011.5872571.

25. Boukari F, Makrogiannis S. Automated cell tracking using motion prediction-based matching and event handling. IEEE/ACM Transactions on Computational Biology and Bioinformatics. 2020;17(3):959–971. doi:10.1109/TCBB.2018.2875684.

26. Payer C, Štern D, Feiner M, Bischof H, Urschler M. Segmenting and tracking cell instances with cosine embeddings and recurrent hourglass networks. Medical Image Analysis. 2019;57:106–119. doi:10.1016/j.media.2019.06.015.

27. Nishimura K, Hayashida J, Wang C, Ker DFE, Bise R. Weakly-supervised cell tracking via backward-and-forward propagation. In: Vedaldi A, Bischof H, Brox T, Frahm JM, editors. Computer Vision – ECCV 2020. Cham: Springer International Publishing; 2020. p. 104–121. doi:10.1007/978-3-030-58610-2_7.

28. Jaqaman K, Loerke D, Mettlen M, Kuwata H, Grinstein S, Schmid SL, et al. Robust single-particle tracking in live-cell time-lapse sequences. Nature Methods. 2008;5(8):695–702. doi:10.1038/nmeth.1237.

29. Padfield D, Rittscher J, Roysam B. Coupled minimum-cost flow cell tracking for high-throughput quantitative analysis. Medical Image Analysis. 2011;15(4):650–668. doi:10.1016/j.media.2010.07.006.

30. Magnusson KEG, Jaldén J, Gilbert PM, Blau HM. Global linking of cell tracks using the Viterbi algorithm. IEEE Transactions on Medical Imaging. 2015;34(4):911–929. doi:10.1109/TMI.2014.2370951.

31. Arbelle A, Reyes J, Chen JY, Lahav G, Riklin Raviv T. A probabilistic approach to joint cell tracking and segmentation in high-throughput microscopy videos. Medical Image Analysis. 2018;47:140–152. doi:10.1016/j.media.2018.04.006.

32. Türetken E, Wang X, Becker CJ, Haubold C, Fua P. Network flow integer programming to track elliptical cells in time-lapse sequences. IEEE Transactions on Medical Imaging. 2017;36(4):942–951. doi:10.1109/TMI.2016.2640859.

33. Schiegg M, Hanslovsky P, Kausler BX, Hufnagel L, Hamprecht FA. Conservation tracking. In: 2013 IEEE International Conference on Computer Vision; 2013. p. 2928–2935.

34. Schiegg M, Hanslovsky P, Haubold C, Koethe U, Hufnagel L, Hamprecht FA. Graphical model for joint segmentation and tracking of multiple dividing cells. Bioinformatics. 2015;31(6):948–956. doi:10.1093/bioinformatics/btu764.

35. Kausler BX, Schiegg M, Andres B, Lindner M, Koethe U, Leitte H, et al. A discrete chain graph model for 3d+t cell tracking with high misdetection robustness. In: Fitzgibbon A, Lazebnik S, Perona P, Sato Y, Schmid C, editors. Computer Vision – ECCV 2012. Berlin, Heidelberg: Springer Berlin Heidelberg; 2012. p. 144–157. doi:10.1007/978-3-642-33712-3_11.

36. Akram SU, Kannala J, Eklund L, Heikkilä J. Joint cell segmentation and tracking using cell proposals. In: 2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI); 2016. p. 920–924. doi:10.1109/ISBI.2016.7493415.

37. Versari C, Stoma S, Batmanov K, Llamosi A, Mroz F, Kaczmarek A, et al. Long-term tracking of budding yeast cells in brightfield microscopy: CellStar and the evaluation platform. Journal of The Royal Society Interface. 2017;14(127):20160705. doi:10.1098/rsif.2016.0705.

38. Dewan MAA, Ahmad MO, Swamy MNS. Tracking biological cells in time-lapse microscopy: An adaptive technique combining motion and topological features. IEEE Transactions on Biomedical Engineering. 2011;58(6):1637–1647. doi:10.1109/TBME.2011.2109001.

39. Li F, Zhou X, Ma J, Wong STC. Multiple nuclei tracking using integer programming for quantitative cancer cell cycle analysis. IEEE Transactions on Medical Imaging. 2010;29(1):96–105. doi:10.1109/TMI.2009.2027813.

40. Narayanaswamy A, Merouane A, Peixoto A, Ladi E, Herzmark P, Von Andrian U, et al. Multi-temporal globally-optimal dense 3-D cell segmentation and tracking from multi-photon time-lapse movies of live tissue microenvironments. In: Durrleman S, Fletcher T, Gerig G, Niethammer M, editors. Spatio-temporal Image Analysis for Longitudinal and Time-Series Image Data. Berlin, Heidelberg: Springer Berlin Heidelberg; 2012. p. 147–162. doi:10.1007/978-3-642-33555-6_13.

41. Hayashida J, Nishimura K, Bise R. MPM: Joint representation of motion and position map for cell tracking. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2020. p. 3822–3831. doi:10.1109/CVPR42600.2020.00388.

42. Jiuqing W, Xu C, Xianhang Z. Cell tracking via structured prediction and learning. Mach Vision Appl. 2017;28(8):859–874. doi:10.1007/s00138-017-0872-0.

43. Mosig A, Jäger S, Wang C, Nath S, Ersoy I, Palaniappan Kp, et al. Tracking cells in life cell imaging videos using topological alignments. Algorithms for Molecular Biology. 2009;4(1):10. doi:10.1186/1748-7188-4-10.

44. Rempfler M, Lange J, Jug F, Blasse C, Myers EW, Menze BH, et al. Efficient algorithms for moral lineage tracing. In: 2017 IEEE International Conference on Computer Vision (ICCV); 2017. p. 4705–4714. doi:10.1109/ICCV.2017.503.

45. Winter M, Wait E, Roysam B, Goderie SK, Ali RAN, Kokovay E, et al. Vertebrate neural stem cell segmentation, tracking and lineaging with validation and editing. Nature Protocols. 2011;6(12):1942–1952. doi:10.1038/nprot.2011.422.

46. Kok RNU, Hebert L, Huelsz-Prince G, Goos YJ, Zheng X, Bozek K, et al. OrganoidTracker: Efficient cell tracking using machine learning and manual error correction. PLOS ONE. 2020;15(10):1–18. doi:10.1371/journal.pone.0240802.

47. Jeong J, Frohberg NJ, Zhou E, Sulchek T, Qiu P. Accurately tracking single-cell movement trajectories in microfluidic cell sorting devices. PLOS ONE. 2018;13(2):1–16. doi:10.1371/journal.pone.0192463.

48. Schott B, Traub M, Schlagenhauf C, Takamiya M, Antritter T, Bartschat A, et al. EmbryoMiner: A new framework for interactive knowledge discovery in large-scale cell tracking data of developing embryos. PLOS Computational Biology. 2018;14(4):1–18. doi:10.1371/journal.pcbi.1006128.

49. Stegmaier J, Mikut R. Fuzzy-based propagation of prior knowledge to improve large-scale image analysis pipelines. PLOS ONE. 2017;12(11):1–26. doi:10.1371/journal.pone.0187535.

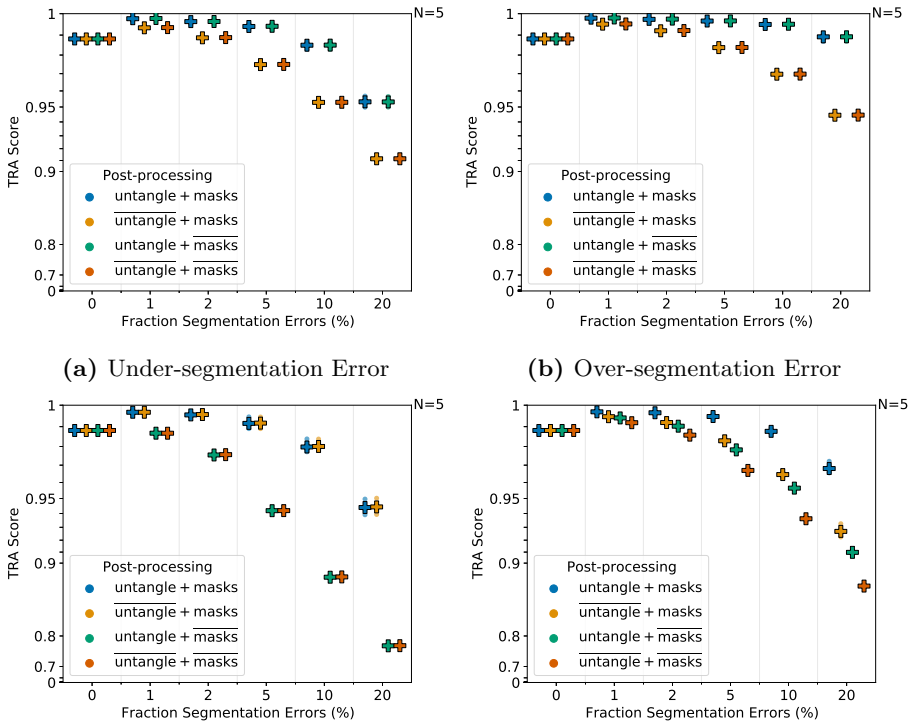
50. Ulman V, Maška M, Magnusson KEG, Ronneberger O, Haubold C, Harder N, et al. An objective comparison of cell-tracking algorithms. Nature Methods. 2017;14(12):1141–1152. doi:10.1038/nmeth.4473.

51. Kuglin CD, Hines DC. The phase correlation image alignment method. In: Proceedings of the IEEE International Conference on Cybernetics and Society; 1975. p. 163–165.

52. Gurobi Optimization L. Gurobi Optimizer Reference Manual; 2021. Available from: `http://www.gurobi.com`.

53. Scherr T, Löffler K. KIT-Sch-GE; 2020. Available from: `https://bitbucket.org/t_scherr/cell-segmentation-and-tracking/src/master/`.

54. Magnusson KEG. BaxterAlgorithms; 2019. Available from: `https://github.com/klasma/BaxterAlgorithms`.



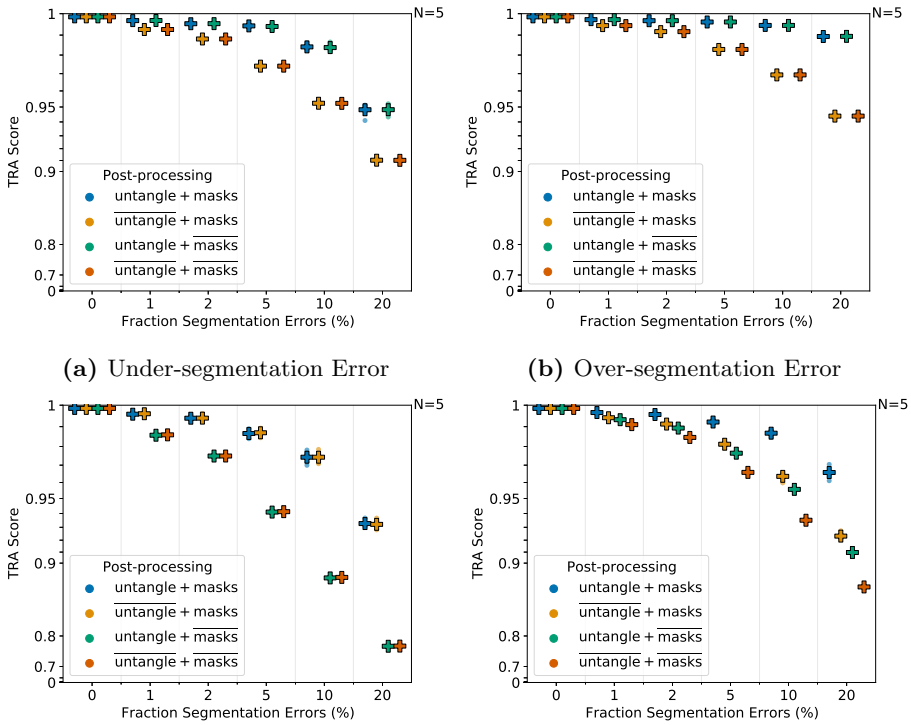
(a) Under-segmentation Error

(b) Over-segmentation Error

(c) FN Error

(d) Mixed Error

Fig. S1. Influence post-processing on Fluo-N2DH-SIM+ 02 Influence of our proposed post-processing on the TRA score for the 2D data set Fluo-N2DH-SIM+ 02. Scores of a single run are shown as circles, while + shows the TRA score averaged over $N = 5$ runs. Per run a fixed fraction of ground truth segmentation masks is modified randomly to simulate segmentation errors. 'untangle' refers to the untangling step, which transforms the tracking graph such that each track has at most one predecessor and two successors, whereas 'masks' refers to adding missing segmentation masks. Over lined post-processing steps $\overline{(\dots)}$ indicate that the post-processing step is missing.



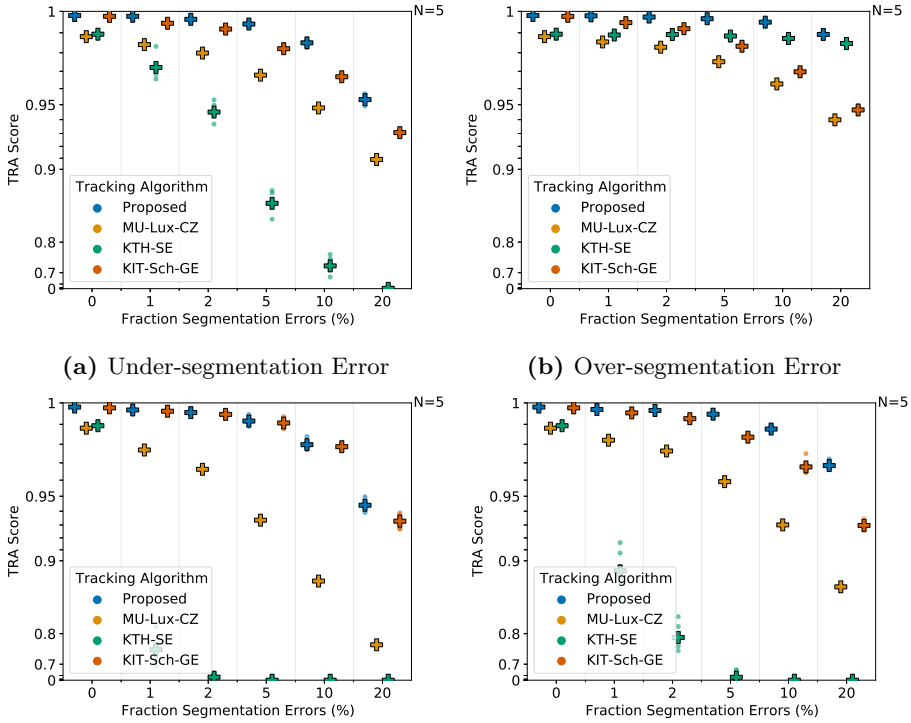
(a) Under-segmentation Error

(b) Over-segmentation Error

(c) FN Error

(d) Mixed Error

Fig. S2. Influence post-processing on Fluo-N3DH-SIM+ 02 Influence of our proposed post-processing on the TRA score for the 3D data set Fluo-N3DH-SIM+ 02. Scores of a single run are shown as circles, while + shows the TRA score averaged over $N = 5$ runs. Per run a fixed fraction of ground truth segmentation masks is modified randomly to simulate segmentation errors. 'untangle' refers to the untangling step, which transforms the tracking graph such that each track has at most one predecessor and two successors, whereas 'masks' refers to adding missing segmentation masks. Over lined post-processing steps $\overline{(\dots)}$ indicate that the post-processing step is missing.



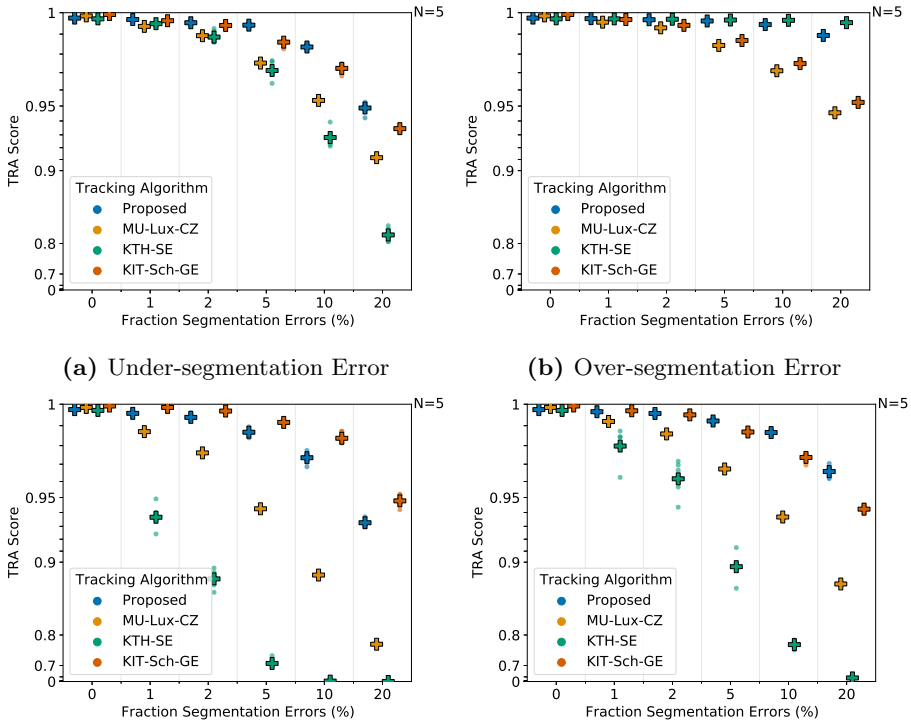
(a) Under-segmentation Error

(b) Over-segmentation Error

(c) FN Error

(d) Mixed Error

Fig. S3. Comparing tracking algorithms on Fluo-N2DH-SIM+ 02
TRA scores of tracking by detection algorithms on 2D data set Fluo-N2DH-SIM+ 02 when provided with the same erroneous segmentation data. Scores of a single run are shown as circles, while + shows the TRA score averaged over $N = 5$ runs.



(a) Under-segmentation Error

(b) Over-segmentation Error

(c) FN Error

(d) Mixed Error

Fig. S4. Comparing tracking algorithms on Fluo-N3DH-SIM+ 02
TRA scores of tracking by detection algorithms on 3D data set Fluo-N3DH-SIM+ 02 when provided with the same erroneous segmentation data. Scores of a single run are shown as circles, while + shows the TRA score averaged over $N = 5$ runs.

Table S1. Run-times of tracking algorithms on image sequences 02. Run times of the tracking algorithms on 2D and 3D data sets, image sequence 02, when provided with perfect ground truth (GT) segmentation as well as when provided with erroneous segmentation data.

Tracking Algorithm	**Data Sets**			
	Fluo-N2DH-SIM+ 02		Fluo-N3DH-SIM+ 02	
	GT	Erroneous	GT	Erroneous
Proposed	28.85 s	29.19 s	1206.05 s	1312.72 s
MU-Lux-CZ	45.15 s	46.12 s	2312.94 s	2323.22 s
KTH-SE	49.71 s	48.85 s	477.17 s	375.26 s
KIT-Sch-GE	20.38 s	31.27 s	1160.88 s	1678.79 s