RESEARCH ARTICLE

EmbedTrack–Simultaneous Cell Segmentation and Tracking Through Learning Offsets and Clustering Bandwidths

KATHARINA LÖFFLER[1,2] AND RALF MIKUT[1]
[1]Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, 76344 Eggenstein-Leopoldshafen, Germany
[2]Institute of Biological and Chemical Systems-Biological Information Processing, Karlsruhe Institute of Technology, 76344 Eggenstein-Leopoldshafen, Germany

Corresponding author: Katharina Löffler (katharina.loeffler@kit.edu)

ABSTRACT To shed light on the processes driving cell migration, a systematic analysis of the cell behavior is required. Since the manual analysis of hundreds or even thousands of cells is infeasible, automated approaches for cell segmentation and tracking are needed. While for the task of cell segmentation deep learning has become the standard, there are few approaches for simultaneous cell segmentation and tracking using deep learning. Here, we present EmbedTrack, a single convolutional neural network for simultaneous cell segmentation and tracking which predicts human comprehensible embeddings. As embeddings, offsets of cell pixels to their cell center and bandwidths are learned which are processed in a subsequent clustering step to generate an instance segmentation and link the segmented instances over time. We benchmark our approach on nine 2D data sets from the Cell Tracking Challenge, where our approach performs on seven out of nine data sets within the top 3 contestants including three top 1 performances. The source code is publicly available at https://git.scc.kit.edu/kit-loe-ge/embedtrack.

INDEX TERMS Cell segmentation, cell tracking, deep learning, image segmentation, instance segmentation, multiple object tracking, object segmentation.

I. INTRODUCTION

The growing availability of biomedical data, such as omics data [1], [2], spectral data [3], images [4] and videos [5], [6], offers the possibility to shed light on the cause of diseases as well as on the biological mechanisms driving the formation of life. Cell migration, the directed movement of cells, is a mechanism of particular interest as it is vital for numerous biological processes such as wound healing, embryonic development or immune response [7]. However, to better understand this mechanism, a quantitative analysis of the cell behavior is needed, which requires tracking hundreds or even thousands of cells over time. Therefore, automated cell segmentation and tracking approaches are needed.

Cell segmentation is the identification of cell instances in single image frames, where the shape of each cell is captured accurately on the pixel level. Cell tracking extends the task of cell segmentation by establishing links between cell instances over time. These links usually either mark two segmented cells in different image frames as the same cell identity or indicate mother-daughter cell relationships after a cell division. To solve the task of cell tracking, two paradigms exist: tracking-by-detection and tracking-by-contour-evolution [5]. In tracking-by-detection, cell segmentation and the linking of the identified instances are done separately, whereas in tracking-by-contour-evolution, segmentation and linking are done simultaneously by propagating an initial segmentation over time.

For long, cell tracking has been dominated by traditional approaches such as simple Nearest Neighbors linking [5], [8], Bayesian filters [9]–[12], and graph-based matching

The associate editor coordinating the review of this manuscript and approving it for publication was Santosh Kumar.

[13]–[19]. The recent success of deep learning based cell segmentation, surpassing traditional segmentation approaches [20]–[23] and the successful application of deep learning for multiple object tracking [24]–[26], give reason to apply deep learning to the task of cell tracking as well.

Currently, the number of approaches incorporating deep learning for cell tracking is still small. In [27] and [28] a Convolutional Neural Network (CNN) shaped like a stacked hourglass with convolutional gated recurrent units is used to predict pixel-wise embeddings, which are subsequently clustered into instances. Hayashida *et al.* train a single model to detect and track cells by predicting offsets and a magnitude which quantifies how likely a cell is at a specific position [29]. He *et al.* combine a particle filter with multi task learning to learn an observation model to select the most probable candidate in the next frame [30]. Hayashida and Bise train two separate models for detection and tracking and predict a motion flow of detected centroids between successive frame pairs [31]. Wen *et al.* train a multi layer perceptron to predict similarity scores of pairs of already segmented cells between successive frames based on position and their Euclidean distances to their neighbors [32]. In [33], after applying a cell segmentation, pairs of image patches are forwarded to a CNN to predict similarity scores which are used to match cells using the Hungarian method. Chen *et al.* extend a Mask R-CNN with a Siamese branch for tracking, using handcrafted and learned features as embeddings to link corresponding cells [34]. Sugawara *et al.* learn, based on sparse annotations, two separate models for cell detection and tracking, where for tracking flows between successive frames are estimated [35]. In [36], two separate models are trained for segmentation and tracking of cells, whereas the tracking model receives a pair of successive frames, a binary mask and the mask of the cell to track and learns to predict the mask of the cell in the next frame. Ben-Haim *et al.* use a cascade of approaches to track already segmented cells: first deep metric learning is used extract cell features, then a graph neural network learns to predict linking costs based on the extracted embeddings, finally, the cells are matched based on the linking costs using a graph-based matching [37]. To handle the lack of fully annotated data, [38] trains a CycleGAN to generate synthetically annotated training data, whereas [39] learns to generate pseudo tracking labels from detection annotations.

Methods based on a cascade of models [30]–[33], [35]–[37], [40] can be complicated and can usually not be trained end to end, whereas approaches using only a single model [27]–[29], [34] can be trained at once. Approaches using recurrent network elements [27], [28], [30], can need careful re-initialization or updates in contrast to models without recurrent elements [29], [31]. Embeddings can be estimated for all cells simultaneously, followed by calculating their pairwise distances afterwards [27]–[31], [35], [39] or a model can predict similarity scores for distinct pairs of cells [30], [32]–[34], [36]. Often, the learned embeddings are not human comprehensible [27], [28], [34], in contrast,

learning embeddings that represent offsets are easy to interpret for humans [29], [31], [35], [39].

In summary, we can categorize recent deep learning based cell segmentation and tracking approaches by: (i) the complexity of their setup, e.g. needing several models for segmentation and tracking vs. a single model, (ii) their network architecture, e.g. if recurrent network elements are used, (iii) their learned output, e.g. predicting similarity scores or embeddings, and (iv) the type of the learned embedding, e.g. whether the embedding is comprehensible for humans.

Here, we propose a single CNN for cell segmentation and tracking, without any recurrent network elements learning human comprehensible embeddings. Our work is based on the instance segmentation approach of Neven *et al.* [41], which trains a branched ERFNet [42] to learn the offset of pixels to its object center as well as a clustering bandwidth to cluster pixels into instance masks. Lalit *et al.* applied this approach first to the task of cell segmentation [43]. We extend the instance segmentation approach to simultaneous segmentation and tracking, by including an estimation of offsets for cell pixels to their cell center between successive frames. Therefore, we adapt the branched ERFNet to process pairs of images and extend the network with an additional decoder branch for tracking. To show the competitiveness of our approach named EmbedTrack, we evaluate our approach on diverse 2D data sets of the well-established Cell Tracking Benchmark (CTB) from the Cell Tracking Challenge (CTC) [5], [6]. We provide our code at https://git.scc.kit.edu/kit-loe-ge/embedtrack.

Our work has most similarity with the cell detection and tracking approach of [29], in the sense that we train a CNN learning offsets. However, the approaches differ in the used network architecture, the loss function, and the post-processing, our approach is learning a clustering bandwidth and [29] learns a 3D embedding including position and time. Furthermore, [29] only detects cells as points while our network learns cell segmentation.

In the following, we start by introducing the instance segmentation approach of Neven *et al.* [41]. Next, we show how to extend the instance segmentation approach to joint cell segmentation and tracking. In the experiments section, we participate as team KIT-Loe-GE on the CTB. Besides reporting our results on the benchmark, we provide insights into the cell statistics of the data sets concerning cell counts and cell motility as well as provide run time estimates of our algorithm.

II. INSTANCE SEGMENTATION OF NEVEN *et al.*

Instance segmentation allows to distinguish 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
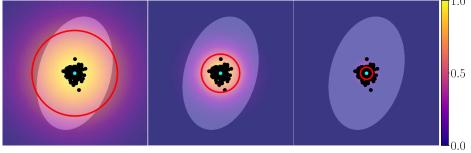
Gaussian Kernel $s = 0.5$ Gaussian Kernel $s = 0.1$ Gaussian Kernel $s = 0.01$

FIGURE 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 range [0, 1] (heat maps from blue to yellow) of each pixel $\mathbf{p}$ 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) will be assigned to the same instance if they lay within the red circle.

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 [41]. As distance measure, [41] use the Gaussian kernel

$$d(\mathbf{c}, \mathbf{p}) = \exp\left(-\frac{(c_x - p_x)^2}{s_x} - \frac{(c_y - p_y)^2}{s_y}\right), \quad (1)$$

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. Fig. 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.

As the true object center is unknown during inference, the idea is that by learning to predict the clustering bandwidths and offsets for each pixel, pixels can be clustered into instances by selecting any pixel, since its shift by the learned offset is an estimate of the object center and the clustering bandwidth defines how far other shifted pixels can be apart to be still assigned to the same instance.

III. METHOD
In the following, we show how to extend the instance segmentation approach of Neven *et al.* to cell segmentation and tracking by adding an additional decoder path to the CNN for tracking and introduce an additional tracking step.

An overview of our approach is shown in Fig. 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 a cell at t to their cell 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 cell (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 tracking step are explained in more detail.

A. EmbedTrack MODEL
We use a branched ERFNet [42] with one shared encoder and three decoder paths – two for segmentation and one for tracking. A visualization of the model and the predicted outputs is shown in Fig. 2. An image pair is fed through 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.

For raw images of shape $[H, W]$, where H is the height and W the width, the first segmentation decoder predicts segmentation offsets $\mathbf{O}^S$ of shape $[2, H, W]$ between pixels belonging to a cell and their corresponding cell center, and clustering bandwidths $\mathbf{S}$ of shape $[2, H, W]$. The second segmentation decoder predicts the seediness $\mathbf{D}$ of shape $[H, W]$, which is a score between 0 and 1 which serves as a foreground-background estimation as well as indicates if a cell pixel estimates its cell center correctly. The tracking
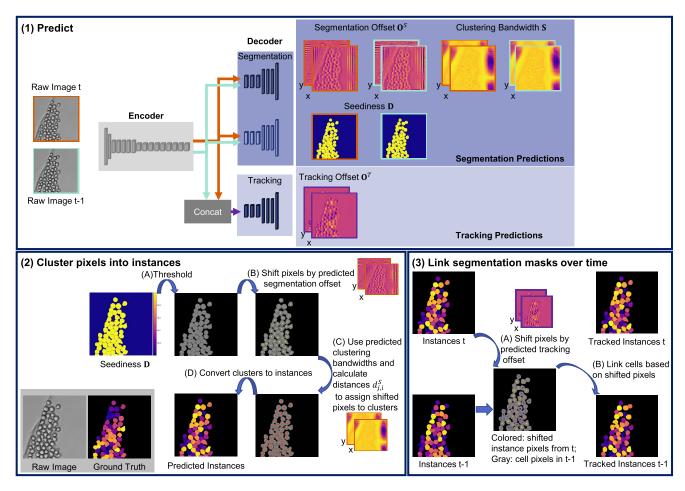
FIGURE 2. Method Overview. The approach consists of three steps: (1) prediction of offsets, clustering bandwidths and seediness maps using a CNN, (2) processing the predicted segmentation offsets and clustering bandwidths in a clustering step to retrieve an instance segmentation, and (3) linking the instance segmentation masks over time by processing the by the CNN predicted tracking offsets. (1) The CNN, a branched ERFNet, receives pairs of raw image crops, of time points t and $t-1$. The CNN predicts two sets of segmentation predictions, one set for each of the two time points, and one tracking offset tensor. The offsets and the clustering bandwidths are tensors, where their values along the x-and y-dimension are shown as maps, where H and W are the height and the width of the raw image crops. (2) The segmentation predictions are processed in a clustering step to retrieve instance segmentation masks for time points t and $t-1$, where gray and black show the prediction of the network for cell and background and the yellow pixels show the cell pixels after adding the predicted offset. The red circles indicate $d_{j,i}^S = 0.5$ with the predicted clustering bandwidths. (3) The instance segmentation masks are linked backwards in time using the predicted tracking offsets, which is a tensor here shown with its values along x- and y-dimension. Raw image crops of the data set BF-C2DL-HSC from the CTC [5], [6].

decoder learns to predict tracking offsets $\mathbf{O}^T$ of shape $[2, H, W]$ between pixels belonging to a cell at t and their corresponding cell center at $t-1$. We apply a tanh activation 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 range $[-1, 1]$, whereas the clustering bandwidth prediction and the seediness prediction are in range $[0, 1]$. In summary, the network predicts one set of segmentation predictions $\mathcal{P}_{\text{seg}} = \{\mathbf{S}, \mathbf{O}^S, \mathbf{D}\}$ for each image t and $t-1$, and one tensor of tracking offsets $\mathbf{O}^T$ to link pixels belonging to a cell at t to its corresponding cell center at $t-1$.

B. 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 cell instances and the ground truth instance masks are used. We like to emphasize that the offsets and clustering bandwidths have no direct supervision during training.

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 will be assigned to the same cluster and therefore the same instance. To enforce similar clustering bandwidths, [41] use a loss component based on the variance between the clustering bandwidth vectors belonging to the same instance. Therefore, for each instance m the mean clustering bandwidth vector $\bar{\mathbf{s}}_m$, in x- and y-dimension, is calculated over the set of pixel indices $\mathcal{P}_m$ belonging to the instance mask m, where $\mathbf{s}_k$ is the clustering bandwidth vector at pixel index k of the clustering

bandwidth prediction $\mathbf{S}$

$$\bar{\mathbf{s}}_m = \frac{1}{|\mathcal{P}_m|} \sum_{k \in \mathcal{P}_m} \mathbf{s}_k. \qquad (2)$$

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

$$\mathcal{L}_{\text{var}} = \frac{1}{M_{\text{inst}}} \sum_{m=1}^{M_{\text{inst}}} \frac{1}{|\mathcal{P}_m|} \sum_{k \in \mathcal{P}_m} (\bar{\mathbf{s}}_m - \mathbf{s}_k)^2, \qquad (3)$$

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

To get the predictions of the cell centers for segmentation and tracking, the predicted offsets $\mathbf{o}_i^{\text{S}}$ for segmentation and $\mathbf{o}_i^{\text{T}}$ for tracking, where i is a multi-index, referring to the pixel position in x- and y-dimension, are added to the position of each pixel

$$\begin{aligned} \mathbf{e}_i^{\text{S}} &= \mathbf{p}_i + \mathbf{o}_i^{\text{S}}, \\ \mathbf{e}_i^{\text{T}} &= \mathbf{p}_i + \mathbf{o}_i^{\text{T}}, \end{aligned} \qquad (4)$$

where $\mathbf{e}_i^{\text{S}}$ is the predicted cell center of pixel i at the same time point, $\mathbf{e}_i^{\text{T}}$ is the predicted cell center of pixel i from time point t for its cell 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 (1), we get for each pixel index i a prediction how close its shifted position $\mathbf{e}_i^{\text{S}}$ is to the cell center

$$\mathbf{s}_m = \exp(w_{\text{s}} \cdot \bar{\mathbf{s}}_m),$$
$$d_{m,i}^{\text{S}}\left(\mathbf{c}_m, \mathbf{e}_i^{\text{S}}\right) = \exp\left(-\frac{\left(c_{m,x} - e_{i,x}^{\text{S}}\right)^2}{s_{m,x}} - \frac{\left(c_{m,y} - e_{i,y}^{\text{S}}\right)^2}{s_{m,y}}\right), \qquad (5)$$

where $d_{m,i}^{\text{S}}(\mathbf{c}_m, \mathbf{e}_i^{\text{S}})$ is the distance between the cell center $\mathbf{c}_m$ of instance mask m and the predicted cell center $\mathbf{e}_i^{\text{S}}$, $s_{m,x}$ and $s_{m,y}$ are the x- and y-dimensions of a scaled mean clustering bandwidth vector $\mathbf{s}_m$ and w_s a scaling weight which we set to -10. As cell center $\mathbf{c}_m$ for each instance mask m we choose as in [43] the medoid, so the cell center will always lay inside of the cell.

In [41] the segmentation offsets and clustering bandwidths are learned jointly using an instance loss based on the Lovász hinge loss [44], [45]. 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 [45]. The Jaccard loss is given by

$$\mathcal{L}_{\text{Jaccard}} = 1 - J(\mathcal{P}_{\text{pred}}, \mathcal{P}_{\text{GT}}), \qquad (6)$$

where $J(\mathcal{P}_{\text{pred}}, \mathcal{P}_{\text{GT}})$ is the Jaccard index between the set of pixels $\mathcal{P}_{\text{pred}}$ belonging to the predicted instance mask and the set of pixels $\mathcal{P}_{\text{GT}}$ 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 will be maximized

$$\begin{aligned} \mathbf{B}_m &= (b_{m,i}) = \begin{cases} 1 & \text{if } i \in \mathcal{P}_m \\ 0 & \text{otherwise}, \end{cases} \\ \mathbf{D}_m^{\text{S}} &= (d_{m,i}^{\text{S}}(\mathbf{c}_m, \mathbf{e}_i^{\text{S}})), \\ \mathcal{L}_{\text{instance}} &= \sum_{m=1}^{M_{\text{inst}}} \mathcal{L}_{\text{Lovász}}\left(2 \cdot \mathbf{D}_m^{\text{S}} - \mathbf{1}, 2 \cdot \mathbf{B}_m - \mathbf{1}\right), \quad (7) \end{aligned}$$

where, $\mathbf{B}_m$ is a binary mask which is 1 at indices belonging to the instance mask m and 0 otherwise, $\mathbf{D}_m^{\text{S}}$ is a distance matrix of shape $[H, W]$ containing the distance of each shifted pixel to the cell center $\mathbf{c}_m$ of instance mask m, $\mathbf{1}$ is a matrix of ones the same size as $\mathbf{D}_m^{\text{S}}$, and $\mathcal{L}_{\text{Lovász}}$ 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 background or a cell, and if the pixel belongs to a cell how close the predictions of the cell center $\mathbf{e}_i^{\text{S}}$ is to the actual cell center. It therefore serves as a foreground-background estimation and as a prediction of how well pixels estimate their cell center. To learn the seediness map, an additional seed loss part is used in [41]. For the seediness map, the CNN learns to predict for each pixel i the distance measure $d_{m,i}^{\text{S}}(\mathbf{c}_m, \mathbf{e}_i^{\text{S}})$ for pixels belonging to a cell and regressing to 0 for background pixels

$$\begin{aligned} \mathcal{L}_{\text{seed}} = w_{\text{fg}} \sum_{m=1}^{M_{\text{inst}}} \frac{1}{|\mathcal{P}_m|} \sum_{k \in \mathcal{P}_m} (d_{m,k}^{\text{S}}(\mathbf{c}_m, \mathbf{e}_k^{\text{S}}) - d_k)^2 \\ + \frac{1}{|\mathcal{P}_{\text{bg}}|} \sum_{j \in \mathcal{P}_{\text{bg}}} (d_j - 0)^2, \quad (8) \end{aligned}$$

where d_k is the predicted seediness score of pixel k belonging to the set of pixel indices $\mathcal{P}_m$ of instance mask m, d_j is the predicted seediness score at background pixel j in the seediness map, $\mathcal{P}_{\text{bg}}$ the set of background pixel indices, and w_{fg} is a weight for foreground pixels.

The segmentation loss is then given as

$$\mathcal{L}_{\text{seg}} = w_{\text{instance}}\mathcal{L}_{\text{instance}} + w_{\text{var}}\mathcal{L}_{\text{var}} + w_{\text{seed}}\mathcal{L}_{\text{seed}}, \qquad (9)$$

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 a cell at t to their cell center at $t-1$. Similar to the instance segmentation loss, we propose 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 $t-1$, we add time indices in the following to highlight to which time point the predicted segmentation components belong. $d_{m,i}^{\text{T}}(\mathbf{c}_m^{t-1}, \mathbf{e}_i^{\text{T}})$ is the distance between $\mathbf{c}_m^{t-1}$, the cell center of instance m at time point $t-1$, and $\mathbf{e}_i^{\text{T}}$,

which is the predicted cell center of pixel i belonging to time point t shifted to its cell center at $t-1$

$$d_{m,i}^{\mathrm{T}}\left(\mathbf{c}_m^{t-1}, \mathbf{e}_i^{\mathrm{T}}\right) = \exp\left(-\frac{\left(c_{m,x}^{t-1}-e_{i,x}^{\mathrm{T}}\right)^2}{s_{m,x}^t}-\frac{\left(c_{m,y}^{t-1}-e_{i,y}^{\mathrm{T}}\right)^2}{s_{m,y}^t}\right),$$

$$\mathbf{D}_m^{\mathrm{T}} = \left(d_{m,i}^{\mathrm{T}}\left(\mathbf{c}_m^{t-1}, \mathbf{e}_i^{\mathrm{T}}\right)\right),$$

$$\mathbf{B}_m^t = \left(b_{m,i}^t\right) = \begin{cases} 1 & \text{if } i \in \mathcal{P}_m^t \\ 0 & \text{otherwise} \end{cases},$$

$$\mathcal{L}_{\text{track}} = \sum_{m=1}^{M_{\text{inst}}^t} \mathcal{L}_{\text{Lovász}}\left(2\cdot\mathbf{D}_m^{\mathrm{T}}-\mathbf{1}, 2\cdot\mathbf{B}_m^t-\mathbf{1}\right), \quad (10)$$

where $\mathcal{P}_m^t$ are the pixel indices of the 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 cell 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_{inst}^t is the number of instances at time point t.

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

$$\mathcal{L} = w_{\text{seg}}\mathcal{L}_{\text{seg}} + w_{\text{track}}\mathcal{L}_{\text{track}}, \quad (11)$$

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

C. PIXEL CLUSTERING

To convert the pixel-wise predictions from the segmentation decoders to an instance segmentation, a clustering step is applied. We modify the clustering proposed in [41] as follows: First, the clustering bandwidth tensor is smoothed with a 3×3 kernel along x- and y- dimension and will be referred to as $\mathbf{S}^{\text{smooth}}$. Since the seediness map $\mathbf{D}$ estimates how well pixels predict their cell center, we select pixels with scores larger than 0.5 in the seediness map and compute the shifted pixel positions $\mathbf{e}_i^{\text{S}}$ to find potential cell centers. We convert each $\mathbf{e}_i^{\text{S}}$, which is a normalized pixel position in range $([0, 1], [0, 1])$ to a pixel index in range $([0, H], [0, W])$, accumulate them, and find potential cell centers by selecting pixel indices with more than five clustered pixels in their 3×3 neighborhood. Next, the pixel indices j that refer to potential cell centers are sorted by their seediness map score $\mathbf{d}_j$. Then, starting with the pixel with the highest score, the list of likely cell centers is processed and pixels i are assigned to the cell centers $\mathbf{e}_j^{\text{S}}$ using a similar distance measure as in the loss

$$\mathbf{s}_j = \exp(w_{\text{s}} \cdot \mathbf{S}^{\text{smooth}}[j]),$$

$$d_{j,i}^{\text{S}}(\mathbf{e}_j^{\text{S}}, \mathbf{e}_i^{\text{S}}) = \exp\left(-\frac{\left(e_{jx}^{\text{S}}-e_{i,x}^{\text{S}}\right)^2}{s_{jx}}-\frac{\left(e_{jy}^{\text{S}}-e_{i,y}^{\text{S}}\right)^2}{s_{jy}}\right),$$

$$\mathbf{B}_j = \begin{cases} 1 & \text{if } \mathbf{d}_{j,i}^{\text{S}} > 0.5 \\ 0 & \text{otherwise}, \end{cases} \quad (12)$$

where $\mathbf{s}_j$ is as in the loss a scaled clustering bandwidth vector at pixel j with the same weight w_{s} as used in the loss, s_{jx} and s_{jy} the x and y components of the clustering bandwidth vector, and $\mathbf{B}_j$ the resulting instance mask. Pixel indices referring to potential cell centers that are assigned to an instance mask will be removed from the list of likely cell centers. To filter out false positives, we set the minimum size of an instance mask to half the size of the 1% percentile of all mask sizes in the training data set. On cell boundaries, pixels can have high distance scores for several cell centers. To assign the pixel to the best cell center, we keep for each pixel its highest distance score $d_{j,i}^{\text{S}}$ to allow to reassign a pixel to a subsequent cell center. Therefore, three conditions need to be fulfilled: the pixel receives a higher score $d_{j,i}^{\text{S}}$ if it is assigned to the new cell 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 Fig. 3.

D. 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 cell 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.
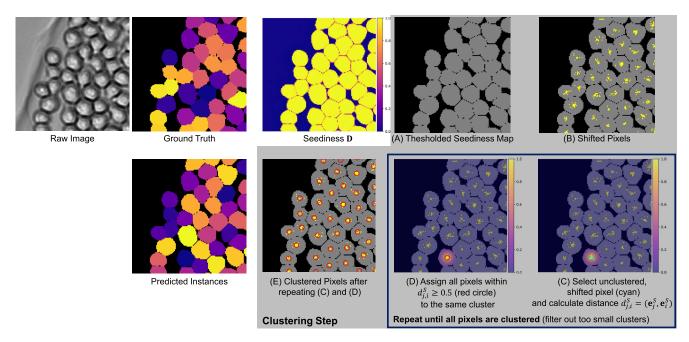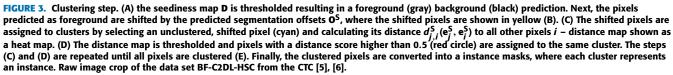
IV. EXPERIMENTS

All models where trained 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.
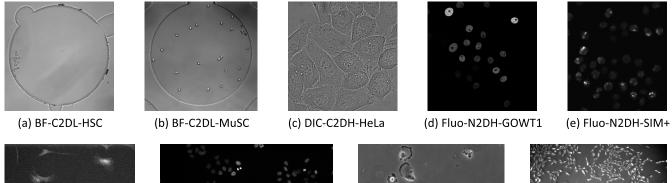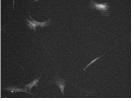
A. DATA SETS

The CTC provides two benchmarks based on the same cell data sets: the Cell Segmentation Benchmark (CSB) for segmentation and the Cell Tracking Benchmark (CTB) for tracking. We train EmbedTrack on nine 2D data sets from the CTC, which are shown in Fig. 4. The CTC data consists of two labeled images sequences for training, which we will refer to as training sequences 01 and 02, and two unlabeled image sequences for testing, which we will refer 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 nine selected data sets. All of the selected data sets, apart from Fluo-N2DH-SIM+ which is simulated and hence has a
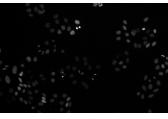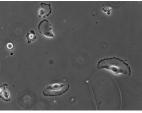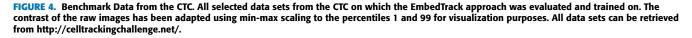
FIGURE 3. Clustering step. (A) the seediness map **D** is thresholded resulting in a foreground (gray) background (black) prediction. Next, the pixels predicted as foreground are shifted by the predicted segmentation offsets $\mathbf{O}^S$, where the shifted pixels are shown in yellow (B). (C) The shifted pixels are assigned to clusters by selecting an unclustered, shifted pixel (cyan) and calculating its distance $d_{j,i}^S(\mathbf{e}_j^S, \mathbf{e}_i^S)$ to all other pixels i – 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. The steps (C) and (D) are repeated until all pixels are clustered (E). Finally, the clustered pixels are converted into a instance masks, where each cluster represents an instance. Raw image crop of the data set BF-C2DL-HSC from the CTC [5], [6].

FIGURE 4. Benchmark Data from the CTC. All selected data sets from the CTC on which the EmbedTrack approach was evaluated and trained on. The contrast of the raw images has been adapted using min-max scaling to the percentiles 1 and 99 for visualization purposes. All data sets can be retrieved from http://celltrackingchallenge.net/.

fully annotated GT, also contain a Silver Truth (ST). The ST are segmentation masks obtained from averaging predictions of previous submissions to the CTC and are provided by the CTC.

B. EVALUATION METRICS

To compare the submitted cell segmentation and tracking algorithms, the CTC uses the metrics SEG, DET and TRA, which all lay in range [0, 1], where a higher score corresponds

with a better performance. The SEG metric is based on the Jaccard similarity index, which calculates the ratio between the intersection of ground truth mask and predicted mask and the union of ground truth mask and predicted mask. Ground truth masks are matched to predicted masks, if the intersection between ground truth mask and predicted mask covers at least half of the ground truth mask. Ground truth masks without a matched predicted mask are penalized with a score of 0. The DET and TRA score are based on the

TABLE 1. Inference time and training data statistics on our system with Ubuntu 18.04, an Intel i9 99000k, 32 GB RAM and two Titan RTX with 24 GB VRAM each. Inference time per sequence is the average time needed to process the challenge data sets from loading the raw images from disk to saving the predicted tracking masks, averaged over both sequences. By dividing the average run time through the number of frames per sequence, we get a rough estimate for processing a single frame. The cell statistics, cell counts and the overlap of the 10% most motile cells calculated from the ST annotations of the training data sets.

Data Set	Inference Time		Training Data Statistics			
			N Cells Last Frame		Overlap fastest 10% Cells	
	per sequence	per frame	01	02	01	02
BF-C2DL-HSC	∼55:30min	1.9s	12	159	0.388	0.553
BF-C2DL-MuSC	∼43:00min	1.9s	23	22	0.098	0.275
DIC-C2DH-HeLa	∼2:30min	1.8s	18	17	0.774	0.848
Fluo-C2DL-MSC	∼2:20min	2.9s	8	3	0.709	0.031
Fluo-N2DH-GOWT1	∼3:00min	2s	20	28	0.927	0.909
Fluo-N2DL-HeLa	∼2:50min	1.9s	137	363	0.721	0.759
PhC-C2DH-U373	∼2:00min	1.1s	7	5	0.875	0.881
PhC-C2DL-PSC	∼7:35min	1.5s	661	498	0.862	0.875
Fluo-N2DH-SIM+	∼2:40min	1.7s	45	54	0.863	0.841

acyclic oriented graph matching (AOGM) measure [46]. The AOGM penalizes the number of transformations needed to transform the predicted tracking graph into the ground truth tracking graph. The measure penalizes errors concerning the detection: false positives, false negatives, and merged cells and tracking: missing links, wrong links, and links with wrong semantics. The DET score only includes errors concerning detection, whereas TRA includes all penalties for detection and tracking. On the CTB, the tracking algorithms are compared on the metrics SEG, TRA, and OP$_{CTB}$ which is the average between SEG and TRA score.

C. TRAINING

We merge the segmentation masks from the ST with the point-wise annotations of the tracking GT to create fully labeled training data with reasonable annotation accuracy. To train our approach, we split the training sequences, 01 and 02, keeping the first 90% of each image sequence for training and the last 10% of each image sequence for evaluation. For submission to the CTB, we select the model with the best Intersection over Union (IoU) score on the evaluation data set. During training, we generate overlapping crops of size 256x256 (512x512 for Fluo-C2DL-MSC). As augmentations during training we use CLAHE, blur, rotation, flipping and add shifts to simulate larger cell movement. Each image crop is normalized to range [0, 1] using the 1% and 99% percentiles per image. We train each model for 15 epochs using the Adam optimizer [47] with learning rate $5 \cdot 10^{-4}$ and a one cycle learning rate scheduler.

D. INFERENCE

For inference, we generate overlapping crops of of size 256x256 (512x512 for Fluo-C2DL-MSC). We apply as test time augmentation rotation and flipping and min-max normalize each crop to range [0, 1] using the 1% and 99% percentiles per image, then we forward the augmented crops to the trained model. We calculate the average over the augmented outputs and stitch the averaged predictions to predictions covering the entire image. Next, the clustering step generates the instance segmentation on the segmentation predictions of the entire image. Finally, we link instances to tracks by processing the tracking offsets and the instance segmentation masks as explained in the tracking step.

E. EVALUATION ON THE CTB

To evaluate the performance of our approach on diverse 2D data sets, we train and submit a model for each of the nine selected 2D data sets from the CTC. We report the time needed to process the two challenge sequences on our system, averaged over both challenge sequences, which includes all processing steps from loading the model, generating image crops, inferring them with the trained model, clustering, tracking and saving the predicted masks to disk in Table 1. In addition, to provide additional insight on the data sets, we calculate the cell counts and cell motility based on the training sequences 01 and 02, which are shown in Table 1. As measure for cell motility, we use the ST annotations and 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 report the overlap for the 10% of the most motile cells. 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, we identify challenging data sets based on their cell motility and their cell count. BF-C2DL-HSC and BF-C2DL-MuSC are data sets that have high cell motility: 10% of the cells have an overlap of less than 0.6, furthermore, both data sets have more than 10 cells in the last frame. Data sets with high cell count, more than 100 cells in the last frame, are BF-C2DL-HSC, Fluo-N2DL-HeLa and PhC-C2DL-PSC.

For better comparison, we provide in Table 2 a short summary of all cell segmentation and tracking algorithms that perform on at least one of the nine selected data sets within the top 3. Apart from TUG-AT [27], [54] which submitted a joint approach for segmentation and tracking, all other teams handled segmentation and tracking separately. The performance on the CTB is shown in Table 3, where the metric scores SEG, TRA and OP$_{CTB}$ are the averages over the two challenge sequences. Compared to the other participating teams, our approach performs well on a wide range of data sets, showing competitive performance on eight out of nine data sets, including surpassing all other submitted approaches on three data sets.

Our approach handles both data sets with high cell motility – BF-C2DL-HSC and BF-C2DL-MuSC – well, outperforming all other submitted approaches. Also, on data sets with high cell count, BF-C2DL-HSC, Fluo-N2DL-HeLa

TABLE 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 nine selected data sets.

Algorithm	Segmentation	Tracking
KIT-Sch-GE [19], [22]	dual U-Net predicting cell and neighbor distances, watershed-based post-processing	graph-based tracking, based on coupled-minimum cost flow problem
MU-Lux-CZ [48]	modified U-Net predicting markers and cell boundaries, watershed-based post-processing	overlap-based tracking
MU-Ba-US [49], [50]	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
KTH-SE [14], [51]	distinct traditional image segmentation methods for each data set	tracking based on the Viterbi algorithm
UVA-NL [52]	same segmentation as MU-Lux-CZ	Siamese tracking using SiamFC tracker, re-segmentation if collision of multiple cells detected
MU-US [50], [53]	modified U-Net predicting markers and binary cell masks	linear assignment based on IoU score, additional gating and Kalman filter modules to improve robustness
TUG-AT [27], [54]	stacked hourglass neural network with recurrent units, predict unique pixel-wise embeddings for each instance for joint segmentation and tracking	
BGU-IL [55], [56]	U-Net with convolutional long short term memory (LSTM) blocks	overlap-based tracking
DESU-US [57], [58]	segmenting cells in spatio-temporal domain using a sequence of traditional image processing methods	model tracking as graph-partitioning problem
AC	anonymous contribution; no description available	
HIT-CN [59]	two U-Nets one predicting cell masks, one predicting cell centroids	overlap-based tracking
FR-Fa-GE [60]	U-Net	overlap-based tracking
PURD-US [61], [62]	U-Net; apply connected component analysis to generate instance segmentation	volumetric particle tracking velocimetry
FR-Ro-GE [63]	U-Net	overlap-based tracking
ND-US [64]	FCN model	tracking based on earth mover's distance

TABLE 3. Results of the Cell Tracking Benchmark (CTB) – status 14.03.2021. We participated as team KIT-Loe-GE on the CTB http://celltrackingchallenge.net/latest-ctb-results/. Per data set 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.

Data Set	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)**
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

Algorithm
KIT-Loe-GE (ours)
KIT-Sch-GE
MU-Lux-CZ
MU-Ba-US
KTH-SE
UVA-NL
MU-US
TUG-AT
BGU-IL
DESU-US
AC
HIT-CN
FR-Fa-GE
PURD-US
FR-Ro-GE
ND-US

and PhC-C2DL-PSC, our approach performs on at least one of the CTB metrics within the top 2 approaches. Besides that, we perform on three more data sets, Fluo-N2DH-GOWT1,PhC-C2DL-U373, and Fluo-N2DH-SIM+ within the top 3 concerning the overall OP_{CTB} metric. Moreover, all nine data sets can be processed within a reasonable time.

V. CONCLUSION

We proposed learning cell segmentation and tracking jointly in a single CNN by learning offsets of cell pixels to their cell centers and a clustering bandwidth. The proposed network architecture does not require any recurrent network parts and the predicted embeddings, offsets and clustering bandwidth, are human comprehensible. To show the performance of EmbedTrack, we evaluated our approach on nine 2D data sets from the Cell Tracking Challenge, which was presented in Table 3. While having a reasonable run time, our approach performs on at least one metric on seven out of nine data sets within the top 3 contestants including three top 1 performances. Our approach can handle data sets with high cell motility, surpassing previous submissions on the CTB, as well as data sets with many cells.

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 [38], or adapting the training to learn on sparse labeled data. In addition, extending the approach to 3D data sets should be investigated.

Cell segmentation and tracking provide the basis for analyzing the cell behavior, for instance by quantifying changes in cell characteristics, such as size or shape. To shed light on the processes driving cell migration, future cell segmentation and tracking approaches should ideally keep track of each cell and its lineage over time. We envision EmbedTrack as a contribution in the steady improvement of cell segmentation and tracking algorithms towards the goal of reaching virtually error free segmentation and tracking performance.

REFERENCES

[1] B. Berger, J. Peng, and M. Singh, "Computational solutions for omics data," *Nature Rev. Genet.*, vol. 14, no. 5, pp. 333–346, May 2013.

[2] A. Conesa and S. Beck, "Making multi-omics data accessible to researchers," *Sci. Data*, vol. 6, no. 1, p. 251, Oct. 2019.

[3] V. S. Chaudhary, D. Kumar, and S. Kumar, "Gold-immobilized photonic crystal fiber-based SPR biosensor for detection of malaria disease in human body," *IEEE Sensors J.*, vol. 21, no. 16, pp. 17800–17807, Aug. 2021.

[4] X. Yu, J. Wang, Q.-Q. Hong, R. Teku, S.-H. Wang, and Y.-D. Zhang, "Transfer learning for medical images analyses: A survey," *Neurocomputing*, vol. 489, pp. 230–254, Jun. 2022.

[5] V. Ulman, M. Maška, and C. Ortiz-de-Solorzano, "An objective comparison of cell-tracking algorithms," *Nature Methods*, vol. 14, pp. 1141–1152, Oct. 2017.

[6] M. Maška, V. Ulman, D. Svoboda, P. Matula, P. Matula, C. Ederra, A. Urbiola, T. España, S. Venkatesan, D. M. Balak, and P. Karas, "A benchmark for comparison of cell tracking algorithms," *Bioinformatics*, vol. 30, no. 11, pp. 1609–1617, 2014.

[7] D. A. Lauffenburger and A. F. Horwitz, "Cell migration: A physically integrated molecular process," *Cell*, vol. 84, no. 3, pp. 359–369, Feb. 1996.

[8] J. Stegmaier and R. Mikut, "Fuzzy-based propagation of prior knowledge to improve large-scale image analysis pipelines," *PLOS ONE*, vol. 12, no. 11, Nov. 2017, Art. no. e0187535.

[9] Y.-H. Chang, H. Yokota, K. Abe, C.-T. Tang, and M.-D. Tasi, "Automated detection and tracking of cell clusters in time-lapse fluorescence microscopy images," *J. Med. Biol. Eng.*, vol. 37, no. 1, pp. 18–25, Jan. 2017.

[10] O. Hirose, S. Kawaguchi, T. Tokunaga, Y. Toyoshima, T. Teramoto, S. Kuge, T. Ishihara, Y. Lino, and R. Yoshida, "SPF-CellTracker: Tracking multiple cells with strongly-correlated moves using a spatial particle filter," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 15, no. 6, pp. 1822–1831, Dec. 2018.

[11] J. Xu, Y. Cao, Z. Zhang, and H. Hu, "Spatial-temporal relation networks for multi-object tracking," in *Proc. IEEE ICCV*, Oct. 2019, pp. 3987–3997.

[12] M. I. Hossain, A. K. Gostar, A. Bab-Hadiashar, and R. Hoseinnezhad, "Visual mitosis detection and cell tracking using labeled multi-Bernoulli filter," in *Proc. 21st Int. Conf. Inf. Fusion (FUSION)*, Jul. 2018, pp. 1–5.

[13] D. Padfield, J. Rittscher, and B. Roysam, "Coupled minimum-cost flow cell tracking for high-throughput quantitative analysis," *Med. Image Anal.*, vol. 15, no. 4, pp. 650–668, 2011.

[14] K. E. G. Magnusson, J. Jaldén, P. M. Gilbert, and H. M. Blau, "Global linking of cell tracks using the Viterbi algorithm," *IEEE Trans. Med. Imag.*, vol. 34, no. 4, pp. 911–929, Apr. 2015.

[15] A. Arbelle, J. Reyes, J.-Y. Chen, G. Lahav, and T. R. Raviv, "A probabilistic approach to joint cell tracking and segmentation in high-throughput microscopy videos," *Med. Image Anal.*, vol. 47, pp. 140–152, Jul. 2018.

[16] E. Türetken, X. Wang, C. J. Becker, C. Haubold, and P. Fua, "Network flow integer programming to track elliptical cells in time-lapse sequences," *IEEE Trans. Med. Imag.*, vol. 36, no. 4, pp. 942–951, Apr. 2017.

[17] M. Schiegg, P. Hanslovsky, C. Haubold, U. Koethe, L. Hufnagel, and F. A. Hamprecht, "Graphical model for joint segmentation and tracking of multiple dividing cells," *Bioinformatics*, vol. 31, no. 6, pp. 948–956, Mar. 2015.

[18] S. U. Akram, J. Kannala, L. Eklund, and J. Heikkilä, "Cell tracking via proposal generation and selection," 2017, *arXiv:1705.03386*.

[19] K. Löffler, T. Scherr, and R. Mikut, "A graph-based cell tracking algorithm with few manually tunable parameters and automated segmentation error correction," *PLoS ONE*, vol. 16, no. 9, Sep. 2021, Art. no. e0249257.

[20] C. Stringer, T. Wang, M. Michaelos, and M. Pachitariu, "Cellpose: A generalist algorithm for cellular segmentation," *Nature Methods*, vol. 18, no. 1, pp. 100–106, Jan. 2021.

[21] K. J. Cutler, C. Stringer, P. A. Wiggins, and J. D. Mougous, "Omnipose: A high-precision morphology-independent solution for bacterial cell segmentation," *bioRxiv*, Dec. 2021, doi: 10.1101/2021.11.03.467199.

[22] T. Scherr, K. Löffler, M. Böhland, and R. Mikut, "Cell segmentation and tracking using CNN-based distance predictions and a graph-based matching strategy," *PLoS ONE*, vol. 15, no. 12, Dec. 2020, Art. no. e0243219.

[23] M. Weigert, U. Schmidt, R. Haase, K. Sugawara, and G. Myers, "Star-convex polyhedra for 3D object detection and segmentation in microscopy," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 3655–3662.

[24] S. K. Pal, A. Pramanik, J. Maiti, and P. Mitra, "Deep learning in multi-object detection and tracking: State of the art," *Int. J. Speech Technol.*, vol. 51, no. 9, pp. 6400–6429, Apr. 2021.

[25] L. Kalake, W. Wan, and L. Hou, "Analysis based on recent deep learning approaches applied in real-time multi-object tracking: A review," *IEEE Access*, vol. 9, pp. 32650–32671, 2021.

[26] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera, "Deep learning in video multi-object tracking: A survey," *Neurocomputing*, vol. 381, pp. 61–88, Mar. 2020.

[27] C. Payer, D. Stern, M. Feiner, H. Bischof, and M. Urschler, "Segmenting and tracking cell instances with cosine embeddings and recurrent hourglass networks," *Med. Image Anal.*, vol. 57, pp. 106–119, Oct. 2019.

[28] M. Zhao, Q. Liu, A. Jha, R. Deng, T. Yao, A. Mahadevan-Jansen, M. J. Tyska, B. A. Millis, and Y. Huo, "VoxelEmbed: 3D instance segmentation and tracking with voxel embedding based deep learning," in *Machine Learning in Medical Imaging*. Cham, Switzerland: Springer, Sep. 2021, pp. 437–446.

[29] J. Hayashida, K. Nishimura, and R. Bise, "MPM: Joint representation of motion and position map for cell tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 3822–3831.

[30] T. He, H. Mao, J. Guo, and Z. Yi, "Cell tracking using deep neural networks with multi-task learning," *Image Vis. Comput.*, vol. 60, pp. 142–153, Apr. 2017.

[31] 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* (Lecture Notes in Computer Science), vol. 11764. Cham, Switzerland: Springer, Oct. 2019, pp. 397–405.

[32] C. Wen, T. Miura, V. Voleti, K. Yamaguchi, M. Tsutsumi, K. Yamamoto, K. Otomo, Y. Fujie, T. Teramoto, T. Ishihara, K. Aoki, T. Nemoto, E. M. Hillman, and K. D. Kimura, "3DeeCellTracker, a deep learning-based pipeline for segmenting and tracking cells in 3D time lapse images," *eLife*, vol. 10, Mar. 2021, Art. no. e59187.

[33] Y. Xie, M. Liu, S. Zhou, and Y. Wang, "A deep local patch matching network for cell tracking in microscopy image sequences without registration," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, early access, Sep. 24, 2021, doi: 10.1109/TCBB.2021.3113129.

[34] Y. Chen, Y. Song, C. Zhang, F. Zhang, L. O'Donnell, W. Chrzanowski, and W. Cai, "CellTrack R-CNN: A novel end-To-end deep neural network for cell segmentation and tracking in microscopy images," 2021, *arXiv:2102.10377*.

[35] K. Sugawara, C. Cevrim, and M. Averof, "Tracking cell lineages in 3D by incremental deep learning," *bioRxiv*, Feb. 2021, doi: 10.1101/2021.02.26.432552.

[36] J.-B. Lugagne, H. Lin, and M. J. Dunlop, "DeLTA: Automated cell segmentation, tracking, and lineage reconstruction using deep learning," *PLOS Comput. Biol.*, vol. 16, no. 4, Apr. 2020, Art. no. e1007673.

[37] T. Ben-Haim and T. R. Raviv, "Graph neural network for cell tracking in microscopy videos," 2022, *arXiv:2202.04731*.

[38] Q. Liu, I. M. Gaeta, M. Zhao, R. Deng, A. Jha, B. A. Millis, A. Mahadevan-Jansen, M. J. Tyska, and Y. Huo, "Towards annotation-free instance segmentation and tracking with adversarial simulations," Jan. 2021, *arXiv:2101.00567*.

[39] K. Nishimura, J. Hayashida, C. Wang, D. F. E. Ker, and R. Bise, "Weakly-supervised cell tracking via backward-and-forward propagation," in *Computer Vision—ECCV 2020*. Cham, Switzerland: Springer, Oct. 2020, pp. 104–121.

[40] J. Wang, X. Su, L. Zhao, and J. Zhang, "Deep reinforcement learning for data association in cell tracking," *Frontiers Bioeng. Biotechnol.*, vol. 8, p. 298, Apr. 2020.

[41] D. Neven, B. D. Brabandere, M. Proesmans, and L. Van Gool, "Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8829–8837.

[42] E. Romera, J. M. Álvarez, L. M. Bergasa, and R. Arroyo, "ERFNet: Efficient residual factorized ConvNet for real-time semantic segmentation," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 1, pp. 263–272, Jan. 2018.

[43] M. Lalit, P. Tomancak, and F. Jug, "Embedding-based instance segmentation in microscopy," in *Proc. Mach. Learn. Res.*, vol. 143. Lübeck, Germany, Jul. 2021, pp. 399–415.

[44] M. Berman, A. R. Triki, and M. B. Blaschko, "The lovasz-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4413–4421.

[45] J. Yu and M. Blaschko, "Learning submodular losses with the Lovász Hinge," in *Proc. 32nd Int. Conf. Mach. Learn.*, vol. 37. Lille, France, Jul. 2015, pp. 1623–1631.

[46] P. Matula, M. Maska, D. V. Sorokin, P. Matula, C. Ortiz-de Solórzano, and M. Kozubek, "Cell tracking accuracy measurement based on comparison of acyclic oriented graphs," *PLOS ONE*, vol. 10, no. 12, Dec. 2015, Art. no. e0144959.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[48] F. Lux and P. Matula. (Mar. 2020). *MU-Lux-CZ*. [Online]. Available: http://celltrackingchallenge.net/participants/MU-Lux-CZ/

[49] R. Bao, N. M. Al-Shakarji, I. Toubal, K. Palaniappan, and F. Bunyak. (Mar. 2021). *MU-Ba-US*. [Online]. Available: http://celltrackingchallenge.net/participants/MU-Ba-US/

[50] N. M. Al-Shakarji, F. Bunyak, G. Seetharaman, and K. Palaniappan, "Multi-object tracking cascade with multi-step data association and occlusion handling," in *Proc. 15th IEEE Int. Conf. Adv. Video Signal Based Surveill.*, Nov. 2018, pp. 1–6.

[51] K. Magnusson, J. Jaldén, and H. M. Blau. (Mar. 2021). *KTH-SE*. [Online]. Available: http://celltrackingchallenge.net/participants/KTH-SE/

[52] A. Panteli, D. K. Gupta, N. de Bruijn, and E. Gavves. (Feb. 2020). *UVA-NL*. [Online]. Available: http://celltrackingchallenge.net/participants/UVA-NL/

[53] N. M. Al-Shakarji, G. Rahmon, R. Bao, I. Toubal, F. Bunyak, and K. Palaniappan. (Mar. 2021). *MU-U.S.* [Online]. Available: http://celltrackingchallenge.net/participants/MU-US/

[54] C. Payer, D. Stern, M. Feiner, H. Bischof, and M. Urschler. (Feb. 2019). *TUG-AT*. [Online]. Available: http://celltrackingchallenge.net/participants/TUG-AT/

[55] A. Arbelle, S. Cohen, and T. R. Raviv. (Jun. 2018). *BGU-IL*. [Online]. Available: http://celltrackingchallenge.net/participants/BGU-IL/

[56] A. Arbelle and T. R. Rariv, "Microscopy cell segmentation via convolutional LSTM networks," in *Proc. IEEE 16th Int. Symp. Biomed. Imag. (ISBI)*, Apr. 2019, pp. 1008–1012.

[57] Y. Wang, S. Annasamudram, O. Williams, K. Zheng, and S. Makrogiannis. (Nov. 2021). *DESU-US*. [Online]. Available: http://celltrackingchallenge.net/participants/DESU-US

[58] F. Boukari and S. Makrogiannis, "Automated cell tracking using motion prediction-based matching and event handling," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 17, no. 3, pp. 959–971, Jun. 2020.

[59] Z. Zhou and F. Wang. (Apr. 2019). *HIT-CN*. [Online]. Available: http://celltrackingchallenge.net/participants/HIT-CN/

[60] T. Falk, D. Mai, A. Dovzhenko, K. Palme, T. Brox, and O. Ronneberger. (Aug. 2018). *FR-Fa-GE*. [Online]. Available: http://celltrackingchallenge.net/participants/FR-Fa-GE/

[61] T. Guo and Y. Wang. (Dec. 2020). *PURD-US*. [Online]. Available: http://celltrackingchallenge.net/participants/PURD-US/

[62] T. Guo, A. M. Ardekani, and P. P. Vlachos, "Microscale, scanning defocusing volumetric particle-tracking velocimetry," *Experim. Fluids*, vol. 60, no. 6, p. 89, May 2019.

[63] O. Ronneberger, R. Bensch, P. Fischer, and T. Brox. (Mar. 2015). *FR-Ro-GE*. [Online]. Available: http://celltrackingchallenge.net/participants/FR-Ro-GE/

[64] P. Liang. (Mar. 2019). *ND-US*. [Online]. Available: http://celltrackingchallenge.net/participants/ND-US/

KATHARINA LÖFFLER received the B.Sc. and M.Sc. degrees in electrical engineering and information technology from the Karlsruhe Institute of Technology, Karlsruhe, Germany, in 2016 and 2019, respectively. She is currently pursuing the Ph.D. degree with the Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Eggenstein-Leopoldshafen, Germany. Her research interests include tracking, biomedical image processing, and deep learning applications.

RALF MIKUT received the Diploma degree in automatic control from the University of Technology, Dresden, Germany, in 1994, and the Ph.D. and Habilitation degrees in mechanical engineering from the University of Karlsruhe, Karlsruhe, Germany, in 1999 and 2007, respectively. Since 2011, he has been an Adjunct Professor at the Faculty of Mechanical Engineering and the Head of the Research Group Automated Image and Data Analysis, Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology (KIT), Germany. His current research interests include machine learning, image processing, life science applications, and smart grids.

• • •