Instance Segmentation and Tracking with Cosine Embeddings and Recurrent Hourglass Networks

Christian Payer[1,*], Darko Štern[2], Thomas Neff[1],
Horst Bischof[1], and Martin Urschler[2,3]

[1]Institute of Computer Graphics and Vision, Graz University of Technology, Austria
[2]Ludwig Boltzmann Institute for Clinical Forensic Imaging, Graz, Austria
[3]BioTechMed-Graz, Graz, Austria

Abstract. Different to semantic segmentation, instance segmentation assigns unique labels to each individual instance of the same class. In this work, we propose a novel recurrent fully convolutional network architecture for tracking such instance segmentations over time. The network architecture incorporates convolutional gated recurrent units (ConvGRU) into a stacked hourglass network to utilize temporal video information. Furthermore, we train the network with a novel embedding loss based on cosine similarities, such that the network predicts unique embeddings for every instance throughout videos. Afterwards, these embeddings are clustered among subsequent video frames to create the final tracked instance segmentations. We evaluate the recurrent hourglass network by segmenting left ventricles in MR videos of the heart, where it outperforms a network that does not incorporate video information. Furthermore, we show applicability of the cosine embedding loss for segmenting leaf instances on still images of plants. Finally, we evaluate the framework for instance segmentation and tracking on six datasets of the ISBI celltracking challenge, where it shows state-of-the-art performance.

Keywords: cell, tracking, segmentation, instances, recurrent, video, embeddings

1 Introduction

Instance segmentation plays an important role in biomedical imaging tasks like cell migration, but also in computer vision based tasks like scene understanding. It is considerably more difficult than semantic segmentation (e.g., [10]), since instance segmentation does not only assign class labels to pixels, but also distinguishes between instances within each class, e.g., each individual person on an image from a surveillance camera is assigned a unique ID.

Mainly due to the high performance of the U-Net [12], semantic segmentation has been successfully used as a first step in medical instance segmentation tasks, e.g., cell tracking. However, for instances to be separated as connected components during postprocessing, borders of instances have to be treated with special

input *t*   embeddings *t*   embeddings *t+1*   instances *t+1*
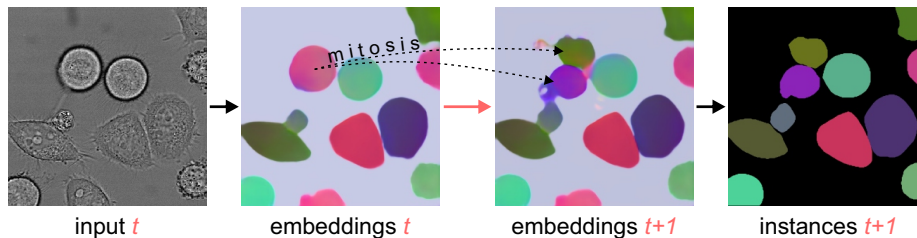
Fig. 1: Overview of our proposed framework showing input image, propagation of cosine embeddings from frame t to frame $t+1$ (three randomly chosen embedding dimensions as RGB channels), and resulting clustered instances.

care. In the computer vision community, many methods for instance segmentation have in common that they solely segment one instance at a time. In [4], all instances are first detected and independently segmented, while in [11], recurrent networks are used to memorize which instances were already segmented. Segmenting solely one instance at a time can be problematic when hundreds of instances are visible in the image, as often is the case with e.g., cell instance segmentation. Recent methods are segmenting each instance simultaneously, by predicting embeddings for all pixels at once [8,5]. These embeddings have similar values within an instance, but differ among instances. In the task of cell segmentation and tracking, temporal information is an important cue to establish coherence between frames, thus preserving instances throughout videos. Despite improvements of instance segmentation using embeddings, to the best of our knowledge, combining them with temporal information for tracking instance segmentations has not been presented.

In this paper, we propose to use recurrent fully convolutional networks for embedding-based instance segmentation and tracking. To memorize temporal information, we integrate convolutional gated recurrent units (ConvGRU [2]) into a stacked hourglass network [9]. Furthermore, we use a novel embedding loss based on cosine similarities, where we exploit the four color map theorem [1], by requiring only neighboring instances to have different embeddings.

2   Instance Segmentation and Tracking

Figure 1 shows our proposed framework on a cell instance segmentation and tracking example. To distinguish cell instances, they are represented as embeddings at different time points. By representing temporal sequences of embeddings in a recurrent hourglass network, a predictor can be learnt from the data, which allows tracking of embeddings also in the case of mitosis events. To finally generate instance segmentations, clustering of the predicted embeddings is performed.
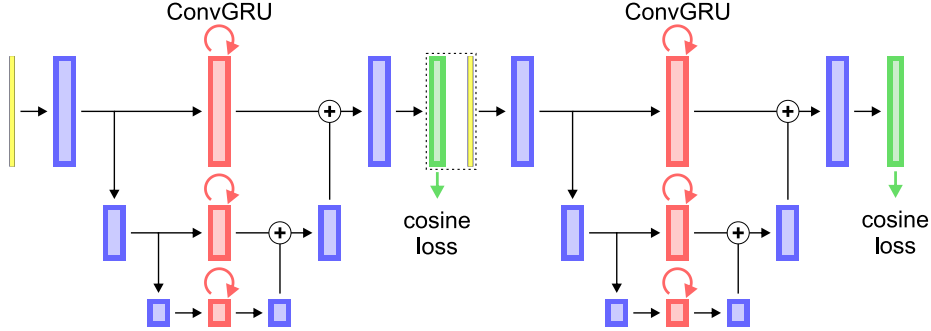
Fig. 2: Overview of the recurrent stacked hourglass network with two hour-glasses and three levels. Yellow bars: input; blue boxes: convolutions; red boxes: ConvGRU; dashed black box: concatenation; green boxes: embeddings.

2.1 Recurrent Stacked Hourglass Network

We modify the stacked hourglass architecture [9] by integrating ConvGRU [2] to propagate temporal information, as shown in Fig. 2. Differently from the original stacked hourglass network, we use single convolution layers with 3×3 filters and 64 outputs for all blocks in the contracting and expanding paths, while we use ConvGRU with 3×3 filters and 64 outputs in between paths. As proposed by [9], we also stack two hourglasses in a row to improve network predictions. Therefore, we concatenate the output of the first hourglass with the input image to use it as input for the second hourglass. We apply the loss function on the outputs of both hourglasses, while we only use the outputs of the second hourglass for the clustering of embeddings.

2.2 Cosine Embedding Loss

We let the network predict a d-dimensional embedding vector $\boldsymbol{e}_p \in \mathbb{R}^d$ for each pixel p of the image. To separate instances $i \in \mathbb{I}$, firstly, embeddings of pixels $p \in \mathbb{S}^{(i)}$ belonging to the same instance i need to be similar, and secondly, embeddings of $\mathbb{S}^{(i)}$ need to be dissimilar to embeddings of pixels $p \in \mathbb{S}^{(j)}$ of other instances $j \neq i$. Here, we treat background as an independent instance. Following from the four color map theorem [1], only neighboring instances need to have different embeddings. Thus, we relax the need of dissimilarity between different instances only to the neighboring ones, i.e., $\mathbb{N}^{(i)} = \bigcup_j \mathbb{S}^{(j)}$ for all instances $j \neq i$ within pixel-wise distance $r_\mathbb{N}$ to instance i. This relaxation simplifies the problem by assigning only a limited number of different embeddings to a possibly large number of different instances.

We compare two embeddings with the cosine similarity

$$\cos(\boldsymbol{e}_1, \boldsymbol{e}_2) = \frac{\boldsymbol{e}_1 \cdot \boldsymbol{e}_2}{\|\boldsymbol{e}_1\| \, \|\boldsymbol{e}_2\|}, \tag{1}$$

which ranges from -1 to 1, while -1 indicates the vectors have the opposite, 0 orthogonal, and 1 the same direction. We define the cosine embedding loss as

$$L = \frac{1}{|\mathbb{I}|} \sum_{i \in \mathbb{I}} \left(1 - \frac{1}{|\mathbb{S}^{(i)}|} \sum_{p \in \mathbb{S}^{(i)}} \cos(\bar{\boldsymbol{e}}^{(i)}, \boldsymbol{e}_p) \right) + \left(\frac{1}{|\mathbb{N}^{(i)}|} \sum_{p \in \mathbb{N}^{(i)}} \cos(\bar{\boldsymbol{e}}^{(i)}, \boldsymbol{e}_p)^2 \right), \quad (2)$$

where the mean embedding of instance i is defined as $\bar{\boldsymbol{e}}^{(i)} = \frac{1}{|\mathbb{S}^{(i)}|} \sum_{p \in \mathbb{S}^{(i)}} \boldsymbol{e}_p$. By minimizing L, the first term urges embeddings $\boldsymbol{e}_p$ of pixels $p \in \mathbb{S}^{(i)}$ to have the same direction as the mean $\bar{\boldsymbol{e}}^{(i)}$, which is the case when $\cos(\bar{\boldsymbol{e}}^{(i)}, \boldsymbol{e}_p) \approx 1$, while the second term pushes embeddings $\boldsymbol{e}_p$ of pixels $p \in \mathbb{N}^{(i)}$ to be orthogonal to the mean $\bar{\boldsymbol{e}}^{(i)}$, i.e., $\cos(\bar{\boldsymbol{e}}^{(i)}, \boldsymbol{e}_p) \approx 0$.

2.3   Clustering of Embeddings

To get the final segmentations from the predicted embeddings, individual groups of embeddings that describe different instances need to be identified. As the number of instances is not known, we perform this grouping with the clustering algorithm HDBSCAN [3] that estimates the number of clusters automatically. For each dataset, two HDBSCAN parameters have to be adjusted: minimal points m_{pts} and minimal cluster size m_{clSize}. To simplify clustering and still be able to detect splitting of instances, we cluster only overlapping pairs of consecutive frames at a time. Since our embedding loss allows same embeddings for different instances that are far apart, we use both image coordinates and value of the embeddings as data points for the clustering algorithm. After identifying the embedding clusters with HDBSCAN and filtering clusters that are smaller than t_{size}, the final segmented instances for each frame pair are obtained.

For merging the segmented instances in overlapping frame pairs, we identify same instances by the highest intersection over union (IoU) between each segmented instance in the overlapping frame. The resulting segmentations are then upsampled back to the original image size, generating the final segmented and tracked instances.

3   Experimental Setup and Results

We train the networks with TensorFlow[1] and perform on-the-fly data augmentation with SimpleITK[2]. We use hourglass networks with seven levels and an input size of 256×256, while we scale the input images to fit. All recurrent networks are trained on sequences of ten frames. We refer to the supplementary material for individual training and augmentation parameters, as well as individual values of parameter described in Section 2.
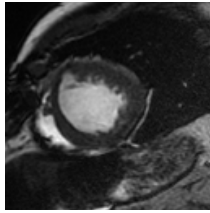
---

[1] https://www.tensorflow.org/
[2] http://www.simpleitk.org/

Left Ventricle Segmentation: To show that our proposed recurrent stacked hourglass network is able to incorporate temporal information, we perform semantic segmentation on videos of short-axis MR slices of the heart from the left ventricle segmentation challenge [14]. We compare the recurrent network with a non-recurrent version, where we replace each ConvGRU with a convolution layer to keep the network complexity the same. Since outer slices do not contain parts of the left ventricle, the networks are evaluated on the three central slices that contain both left ventricle myocardium and blood cavity (see Fig. 3a). We train the networks with a softmax cross entropy loss to segment three labels, i.e., background, myocardium, and blood cavity. We use a three-fold cross-validation setup, where we randomly split datasets of 96 patients into three equally sized folds. Table 1a shows the IoU for our internal cross-validation of both recurrent and non-recurrent stacked hourglass networks.
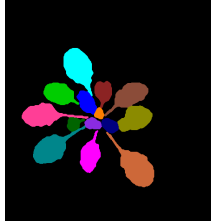
Leaf Instance Segmentation: We show that the cosine embedding loss and the subsequent clustering are suitable for instance segmentation without temporal information, by evaluating on the A1 dataset of the CVPPP challenge for segmenting individual plant leaves [7] (see Fig. 3b). We use the non-recurrent version of the proposed network from the previous experiment to predict embeddings with 32 dimensions. Consequently, the clustering is also performed on single images. As we were not able to provide results on the challenge test set in time before finalizing this paper, we report results of an internal three-fold cross-validation of the 128 training images. In consensus with [13], we report the symmetric best Dice (SBD) and the absolute difference in count (|DiC|) and compare to other methods in Table 1b.

Cell Instance Tracking: As our main experiment, we show applicability of our full framework for instance segmentation and tracking by evaluating six different datasets of cell microscopy videos from the ISBI celltracking challenge [15]. Each celltracking dataset consists of two annotated training videos and two testing videos with image sizes ranging from 512×512 to 1200×1024 and with 48 to 138 frames. We refer to [6] for additional imaging and video parameters. As the instance IDs in groundtruth images are consistent throughout the whole video only for tracking, but not for segmentation, we merge both tracking and segmentation groundtruth for each frame to have consistent instance IDs. Furthermore to learn the background embeddings, we only use the frames on which every cell is segmented. With hyperparameters determined on the two annotated training videos from each dataset, we train the networks for predicting embeddings of size 16 on both videos for our challenge submission.

To compete in the tracking metric of the challenge, the framework is required to identify the parent ID of each cell. As the framework is able to identify splitting cells and to assign new instance IDs (i.e., mitosis as seen on Fig. 1), the parent ID of each newly created instance is determined as the instance with the highest IoU in previous frames. We further postprocess the cells' family tree to be consistent with the evaluation criteria, e.g., an instance ID may not be used after splitting into children. The results in comparison to the top performing methods are presented in Table 2.

(a) Heart MRI input and segmentation.

(b) Plant leaves input and instances.

(a) Quantitative results of the heart MRI left ventricle segmentation.

	IoU_{myo}	IoU_{cav}
non-recurrent	78.3 ± 9.2	89.1 ± 7.7
recurrent	$\mathbf{79.4 \pm 8.5}$	$\mathbf{89.4 \pm 7.2}$

(b) Quantitative results of the CVPPP leaf instance segmentation. Values taken from [13].

	SBD	|DiC|
RIS+CRF	66.6 ± 8.7	1.1 ± 0.9
MSU	66.7 ± 7.6	2.3 ± 1.6
Nottingham	68.3 ± 6.3	3.8 ± 2.0
Wageningen	71.1 ± 6.2	2.2 ± 1.6
IPK	74.4 ± 4.3	2.6 ± 1.8
IS+RA [11]	$\mathbf{84.9 \pm 4.8}$	$\mathbf{0.8 \pm 1.0}$
Ours	84.5 ± 5.5	1.5 ± 1.2

Fig. 3 & Table 1: Results of the left ventricle segmentation and the CVPPP leaf instance segmentation. Values show mean ± standard deviation. Note that we report our results for both datasets based on a three-fold cross-validation setup. Thus, they are not directly comparable to other published results. SBD: symmetric best Dice; |DiC|: absolute difference in count; IoU: intersection over union; myo: myocardium; cav: blood cavity.

4   Discussion and Conclusion

Up to our knowledge, we are the first to present a method that incorporates temporal information into a network to allow tracking of embeddings for instance segmentation. We perform three experiments to show different aspects of our novel method, i.e., temporal segmentation, instance segmentation, and combined instance segmentation and tracking. Thus, we demonstrate the wide applicability of our approach.

We use the left ventricle segmentation experiment to show that our novel recurrent stacked hourglass network can be used for incorporating temporal information. It can be seen from the results of the experiment that incorporating ConvGRU between contracting and expanding path deeply inside the architecture improves over the baseline stacked hourglass network. Nevertheless, since we simplified the evaluation protocol of the challenge, the results of the experiment should not be directly compared to other reported results. Moreover, benefits of such deep incorporation compared to having recurrent layers on other positions in the network [11] remain to be shown.

This paper also contributes with a novel embedding loss based on cosine similarities. Most of the methods that use embeddings for differentiating between instance segmentations are based on maximizing distances of embeddings

Table 2: Quantitative results of the celltracking datasets for overall performance (OP), segmentation (SEG), and tracking (TRA), as described in [15].

		DIC-HeLa	Fluo-MSC	Fluo-GOWT1	Fluo-HeLa	PhC-U373	Fluo-SIM+	
OP	1st	0.864	0.759	0.951	0.942	0.951	0.882	Ours
	2nd	0.828	0.676	0.914	0.940	0.896	0.878	BGU-IL (1–2)
	3rd	0.629	0.658	0.902	0.928	0.895	0.874	CUNI-CZ
			5th 0.631		11th 0.829	4th 0.888	9th 0.810	CVUT-CZ
SEG	1st	0.814	0.645	0.927	0.903	0.920	0.802	FR-Be-GE
	2nd	0.776	0.590	0.893	0.893	0.832	0.791	FR-Ro-GE
	3rd	0.464	0.582	0.887	0.869	0.826	0.781	HD-Har-GE
			5th 0.496	4th 0.880	10th 0.749	5th 0.793	8th 0.718	KIT-GE
TRA	1st	0.915	0.873	0.976	0.991	0.983	0.975	KTH-SE (1–4)
	2nd	0.881	0.765	0.947	0.987	0.981	0.961	LEID-NL
	3rd	0.797	0.763	0.925	0.986	0.977	0.957	
					12th 0.909		10th 0.902	

in the Euclidean space, e.g., [8]. When using such embedding losses, we observed problems when combining them with recurrent networks, presumably due to unrestricted embedding values. To overcome these problems, we use cosine similarities that normalize embeddings. The only other work that suggests cosine similarities for instance segmentation with embeddings is the unpublished work of [5]. However, compared to their embedding loss that takes all instances into account, our novel loss focuses only on neighboring ones, which can be beneficial for optimization in the case of a large number of instances. We evaluate our novel loss on the CVPPP challenge dedicated to instance segmentation from still images. While waiting for the results of the competition, our method evaluated with three-fold cross-validation shows to be in line with the currently leading method, and has a significant margin to the second best. Moreover, compared to the leading method [11], the architecture of our method is considerably simpler.

In our main experiment for segmentation and tracking of instances, we evaluate our method on the ISBI celltracking challenge, showing large variability in visual appearance, size and number of cells. Our method achieves two first and two second places among the six submitted datasets in the tracking metric. For the dataset DIC-HeLa, having a dense layout of cells as seen in Fig. 1, we outperform all other methods in both tracking and segmentation metrics. On the dataset Fluo-GOWT1 we rank overall second. On the datasets Fluo-HeLa and Flou-SIM+, which consist of images with small cells, our method does not perform well due to the need to downsample images for the network to process them. When the downsampling results in drastic reduction of cell sizes, our method fails to create instance segmentations, thus explaining the not satisfying performance also in tracking. To increase the resolution and consequently improve segmentation and tracking, we could split the input image into multiple smaller parts, similarly as done in [12].

In conclusion, our work has shown that embeddings for instance segmentation can be successfully combined with recurrent networks incorporating temporal information to perform instance tracking. In future work, we will investigate the possibility of incorporating the required clustering step inside of a single end-to-end trained network, which could simplify the framework and further improve the segmentation and tracking results.

References

1. Appel, K., Haken, W.: Every planar map is four colorable. Bull. Am. Math. Soc. 82(5), 711–712 (1976)
2. Ballas, N., Yao, L., Pal, C., Courville, A.: Delving Deeper into Convolutional Networks for Learning Video Representations. Int. Conf. Learn. Represent. CoRR, abs:1511.06432 (2016)
3. Campello, R.J.G.B., Moulavi, D., Zimek, A., Sander, J.: Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection. ACM Trans. Knowl. Discov. Data 10(1), 5:1–5:51 (2015)
4. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: Proc. Int. Conf. Comput. Vis. pp. 2980–2988 (2017)
5. Kong, S., Fowlkes, C.: Recurrent Pixel Embedding for Instance Grouping CoRR, abs:1712.08273 (2017)
6. Maška, M., Ulman, V., Svoboda, D., Matula, P., Matula, P., Ederra, C., et al.: A benchmark for comparison of cell tracking algorithms. Bioinformatics 30(11), 1609–1617 (2014)
7. Minervini, M., Fischbach, A., Scharr, H., Tsaftaris, S.A.: Finely-grained annotated datasets for image-based plant phenotyping. Pattern Recognit. Lett. 81, 80–89 (2016)
8. Newell, A., Huang, Z., Deng, J.: Associative Embedding: End-to-End Learning for Joint Detection and Grouping. In: Adv. Neural Inf. Process. Syst., pp. 2277–2287. Curran Associates, Inc. (2017)
9. Newell, A., Yang, K., Deng, J.: Stacked Hourglass Networks for Human Pose Estimation. In: Proc. Eur. Conf. Comput. Vis. pp. 483–499 (2016)
10. Payer, C., Štern, D., Bischof, H., Urschler, M.: Multi-label Whole Heart Segmentation Using CNNs and Anatomical Label Configurations. In: MMWHS Chall. 2017, Stat. Atlases Comput. Model. Hear. pp. 190–198. Springer (2018)
11. Ren, M., Zemel, R.S.: End-To-End Instance Segmentation With Recurrent Attention. In: Proc. Comput. Vis. Pattern Recognit. pp. 6656–6664 (2017)
12. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Proc. Med. Image Comput. Comput. Interv., pp. 234–241. Springer (2015)
13. Scharr, H., Minervini, M., French, A.P., Klukas, C., Kramer, D.M., Liu, X., Luengo, I., Pape, J.M., Polder, G., Vukadinovic, D., Yin, X., Tsaftaris, S.A.: Leaf segmentation in plant phenotyping: a collation study. Mach. Vis. Appl. 27(4), 585–606 (2016)
14. Suinesiaputra, A., Cowan, B.R., Al-Agamy, A.O., Elattar, M.A., Ayache, N., et al.: A collaborative resource to build consensus for automated left ventricular segmentation of cardiac MR images. Med. Image Anal. 18(1), 50–62 (2014)
15. Ulman, V., Maška, M., Magnusson, K.E., Ronneberger, O., Haubold, C., Harder, N., et al.: An objective comparison of cell-tracking algorithms. Nat. Methods 14(12), 1141–1152 (2017)

Appendix

1 Network and Training Parameters

We set the network parameters as follows: The weights of each convolution layer of the stacked hourglass network are initialized with the method as described in [16], the biases with 0. The networks do not employ any normalization layers or dropout, but use an L2 weight regularization factor of 0.00001. Due to the demanding training of recurrent neural networks, in terms of both memory and computational requirements, we set the mini-batch size to 1. We train the recurrent networks for sequences of 10 consecutive frames. For the non-recurrent neural networks, we use a mini-batch size of 10. We train all networks with ADAM [17] for total 40000 iterations and a learning rate of 0.0001, while the learning rate is reduced to 0.00001 after 20000 iterations. Training of a recurrent networks took ≈ 12 hours, training of the non-recurrent networks took ≈ 8 hours on a single NVIDIA Titan Xp with 12 GB.

2 Data Preprocessing and Augmentation Parameters

We perform input data augmentation, by changing intensity values and spatial deformations. First, we change the image intensity values such that the minimum and maximum values are -1 and 1. As MR and microscopy images may contain outliers in terms of minimum and maximum values, we calculate the minimum value as the median of $i_{\min}\%$ of all intensity values of an image, and the maximum as the median of $i_{\max}\%$. Then, for augmentation, we shift each intensity value randomly by i_{shift} and scale each intensity by i_{scale}. For the random spatial deformations in both x and y axes, we translate by t pixels, flip axis with probability f_p, rotate by r degrees and scale by s. Furthermore, we employ elastic deformations, by randomly moving points by b pixels on a grid of size g and interpolating with third order splines. All random augmentations sample from a uniform distribution within the specified intervals.

Left Ventricle Segmentation: The augmentation parameters are as follows: Intensity transformations: $i_{\min} = 10\%$, $i_{\max} = 10\%$, $i_{\text{shift}} \in [-0.25, 0.25]$, $i_{\text{scale}} \in [0.75, 1.25]$. Spatial transformations: $t \in [-20, 20]$, $f_p = 0$, $r \in [-15, 15]$, $s \in [0.75, 1.25]$, $b = 8$, $g \in [-10, 10]$. We set default pixel values outside the defined image region to 0.

Leaf Instance Segmentation: The augmentation parameters are as follows: Intensity transformations: $i_{\min} = 1\%$, $i_{\max} = 1\%$, $i_{\text{shift}} \in [-0.25, 0.25]$, $i_{\text{scale}} \in [0.75, 1.25]$. Spatial transformations: $t \in [-12, 12]$, $f_p = 0.5$, $r \in [-180, 180]$, $s \in [0.75, 1.25]$, $b = 8$, $g \in [-10, 10]$. For each instance $i \in \mathbb{I}$, we define all pixels inside the segmentation mask as $\mathbb{S}^{(i)}$, and all pixels of all other instances as $\mathbb{N}^{(i)}$. We perform mirror padding for pixels outside the defined image region, but we do not calculate the loss for these pixels.

Cell Instance Tracking: Unless otherwise stated, the augmentation parameters for all datasets are as follows: Intensity transformations: $i_{\min} = 20\%$, $i_{\max} = 10\%$ (for Fluo-MSC and Fluo-SIM+ we set $i_{\max} = 1\%$), $i_{\mathrm{shift}} \in [-0.25, 0.25]$, $i_{\mathrm{scale}} \in [0.75, 1.25]$. Due to noise in the intensity values, we smooth the images with a Gaussian function with $\sigma = 2$ pixel. Spatial transformations: $t \in [-25, 25]$, $f_p = 0.5$, $r \in [-180, 180]$, $s \in [0.75, 1.25]$, $b = 8$, $g \in [-10, 10]$. For each instance i, we define all pixels inside the segmentation mask as $\mathbb{S}^{(i)}$, while we set $\mathbb{N}^{(i)}$ to only neighboring instances within a specified radius $r_{\mathbb{N}}$ in pixels. For dataset Fluo-MSC we set $r_{\mathbb{N}} = 150$, for the dataset Fluo-HeLa we set $r_{\mathbb{N}} = 25$. For all other datasets we set $r_{\mathbb{N}} = 50$. For each mini-batch, we use at most 32 different instances for training, to reduce memory consumption. We perform mirror padding for pixels outside the defined image region, but we do not calculate the loss for these pixels.

3   Clustering Parameters

We append the image coordinates scaled with factor c to value of the embeddings as data points for the clustering algorithm. We modify the parameters c and m_{pts} for each dataset, while we set $m_{\mathrm{clSize}} = m_{\mathrm{pts}}$ and $t_{\mathrm{size}} = \frac{m_{\mathrm{pts}}}{2}$. DIC-HeLa: $m_{\mathrm{pts}} = 1000$, $c = 0.02$; Fluo-MSC: $m_{\mathrm{pts}} = 500$, $c = 0.1$; Fluo-GOWT1: $m_{\mathrm{pts}} = 50$, $c = 0.001$; Fluo-SIM+: $m_{\mathrm{pts}} = 100$, $c = 0.001$; Fluo-HeLa: $m_{\mathrm{pts}} = 25$, $c = 0.01$; PhC-U373: $m_{\mathrm{pts}} = 500$, $c = 0.005$; For the CVPPP dataset we set $m_{\mathrm{pts}} = 50$, $c = 0.001$.

Appendix References

16. He, K., Zhang, X., Ren, S., Sun, J.: Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In: Proc. Int. Conf. Comput. Vis. pp. 1026–1034. IEEE (2015)
17. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. Int. Conf. Learn. Represent. CoRR, abs:1412.6980 (2015)
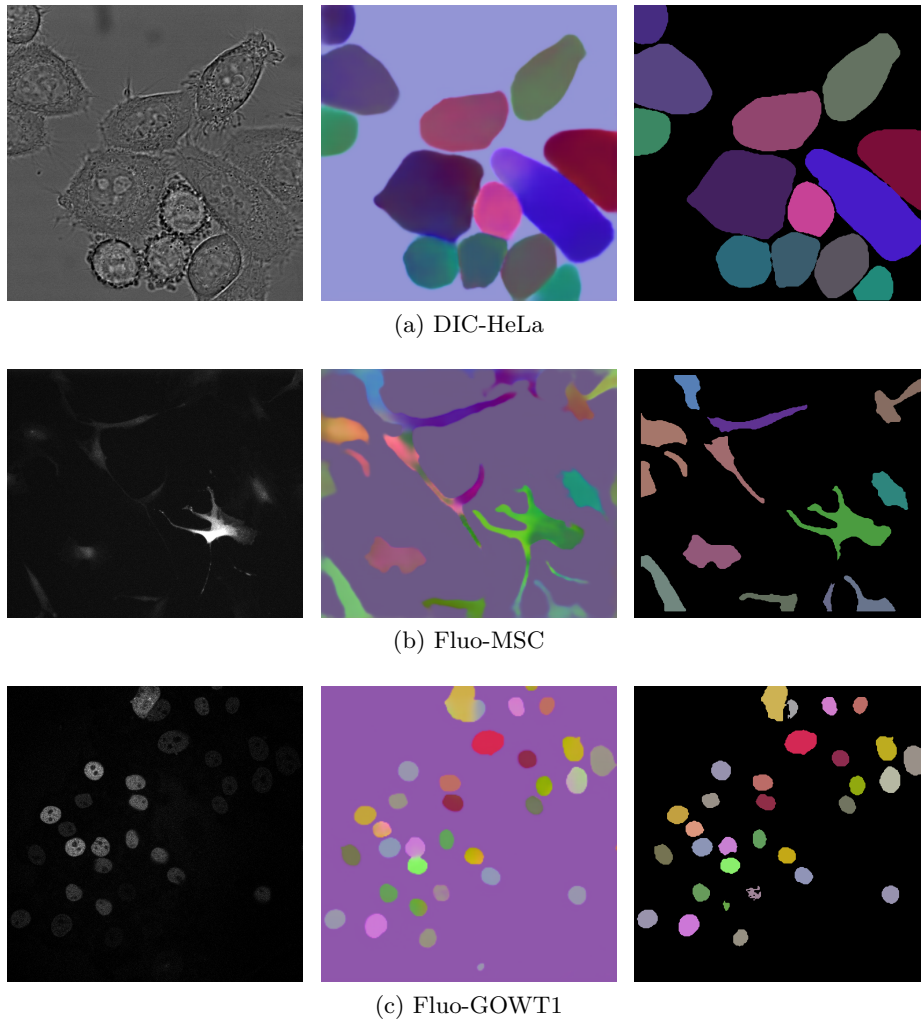
(a) DIC-HeLa

(b) Fluo-MSC

(c) Fluo-GOWT1

Fig. 4: Example results of the evaluated celltracking datasets. Left: normalized input; middle: three randomly chosen dimensions of the embedding as RGB channels; right: final instance segmentation.
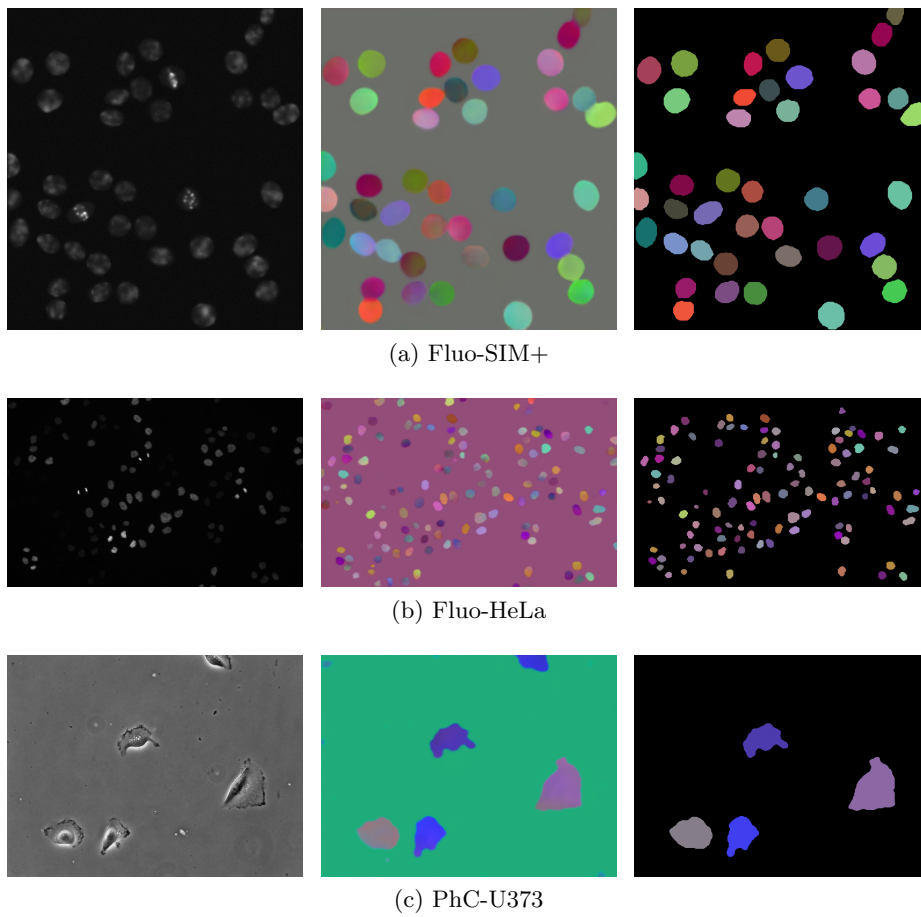
(a) Fluo-SIM+

(b) Fluo-HeLa

(c) PhC-U373

Fig. 5: Example results of the evaluated celltracking datasets, continued.