Appendix A. Self-Supervised Training

Architecture. Our self-supervised training requires a network (here referred to as mini U-Net) with a field of view (FoV) smaller than the expected cell diameter. Since our analyzed datasets contain cells with diameters as small as 20 pixels wide, we use a U-Net architecture [20] with an FoV of only 16×16. To increase the model's capabilities without expanding the FoV, we include additional 1×1 convolutions. Each U-Net block is composed of a series of [3×3, 1×1, 1×1, 3×3] valid convolution layers with ReLU activations. We use a downsampling factor of 2×2, a depth of 1 and constant upsampling layers. In the first layer, we use 64 feature maps and increase it by a factor of 3 after each block.

Training. We train the mini U-Net on batches of 8 randomly chosen images with size 252×252 pixels. We use the Adam optimizer [11] with an initial learning rate of $4e^{-5}$ and train for 50 epochs, reducing the learning rate by a factor of 10 after epochs 20 and 30. In our pairwise loss, defined in Equation 5,

$$\mathcal{L} = \sum_{i,j \in P} \sigma\left(d(i,j) - \hat{d}(i,j)\right) + \lambda_{\text{reg}} \|r(i)\|_2,$$

we use $\sigma(\delta) = \left(1 + \exp\left(-\frac{\|\delta\|_2^2}{\tau}\right)\right)^{-1}$, $\tau = 10$, $\lambda_{\text{reg}} = 1e^{-5}$ and reduce the amount of coordinate pairs to P to reduce the GPU memory footprint. We obtain P by first sampling $\mathcal{P}_1$ as 20% of all pixels. For every sample in $p_1 \in \mathcal{P}_1$ we then sample $p_2 \in \mathcal{P}_2$, a random coordinate within radius $\kappa = 10$ of p_1. In our loss we sample $i, j \in P = \mathcal{P}_1 \times \mathcal{P}_2$.

Appendix B. Supervised Training

Architecture. We use the same architecture as the mini U-Net (see Appendix A) but increase the depth of the network to 3 which expands the network's field of view (FoV).

In conclusion, each U-Net block is composed of a series of $[3 \times 3, 1 \times 1, 1 \times 1, 3 \times 3]$ valid convolution layers with ReLU activations. We use a downsampling factor of 2×2, a depth of 3 and constant upsampling layers. In the first layer, we use 64 feature maps and increase it by a factor of 3 after each block.

Training. All models are trained with identical training setups. We optimize the loss (see $\mathcal{L}_{\text{StarDist}}$) with the Adam optimizer with learning rate $1e^{-5}$ for 200 epochs, reducing the learning rate by a factor of 10 at epochs 30, 80 and 160. We use batches of 8 images with size 252×252 pixels, sampling pairs of patches within radius $\kappa = 10$ (see Equation 5), and set the loss temperature $\tau = 10$.

Appendix C. HSC Dataset

The HSC dataset is especially challenging - a culture plate is visible in the background, which causes all evaluated models to predict additional object instances near the border of the plate.

We compute scores on a range of IoU thresholds to investigate robustness of evaluated methods. At the matching IoU threshold of 0.5, we obtain F1 scores of 0.00 for the pre-trained CELLPOSE model, 0.09 for the pre-trained STARDIST model and 0.06 for CELLULUS *(ours)* (see Table 3). Additionally for the same matching IoU threshold, we obtain RECALL scores of 0.01 for CELLPOSE, 0.26 for STARDIST and 0.55 for CELLULUS. On the HSC dataset, CELLULUS is the most sensitive at detecting cells, but performs less favorably with respect to F1 and ACCURACY metrics. The high RECALL scores obtained with CELLULUS also explains the high SEG scores where false-positive predictions are not heavily penalized. Qualitative results on the HSC dataset can be seen in Figure 7.

Table 3. **Quantitative results on the HSC dataset in the Cell Tracking Challenge [26], for selected matching IoU thresholds (*fully unsupervised setting*).** Pre-trained CELLPOSE and STARDIST baseline models are compared with CELLULUS *(ours)*. Best performing method on each threshold and metric, is shown in bold.

	ACCURACY			F1			RECALL		
Threshold	CELLPOSE	STARDIST	CELLULUS	CELLPOSE	STARDIST	CELLULUS	CELLPOSE	STARDIST	CELLULUS
0.1	**0.68**	0.22	0.06	**0.81**	0.36	0.11	0.92	0.98	**0.99**
0.2	**0.59**	0.21	0.06	**0.74**	0.35	0.11	0.84	**0.97**	0.96
0.3	**0.27**	0.18	0.05	**0.43**	0.30	0.10	0.51	0.83	**0.86**
0.4	0.03	**0.11**	0.04	0.06	**0.20**	0.08	0.07	0.55	**0.72**
0.5	0.00	**0.05**	0.03	0.00	**0.09**	0.06	0.01	0.26	**0.55**
0.6	0.00	0.01	**0.02**	0.00	0.03	**0.05**	0.00	0.07	**0.39**
0.7	0.00	0.00	**0.01**	0.00	0.00	**0.02**	0.00	0.01	**0.13**
0.8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.01**
0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Appendix D. SIMULATED Dataset

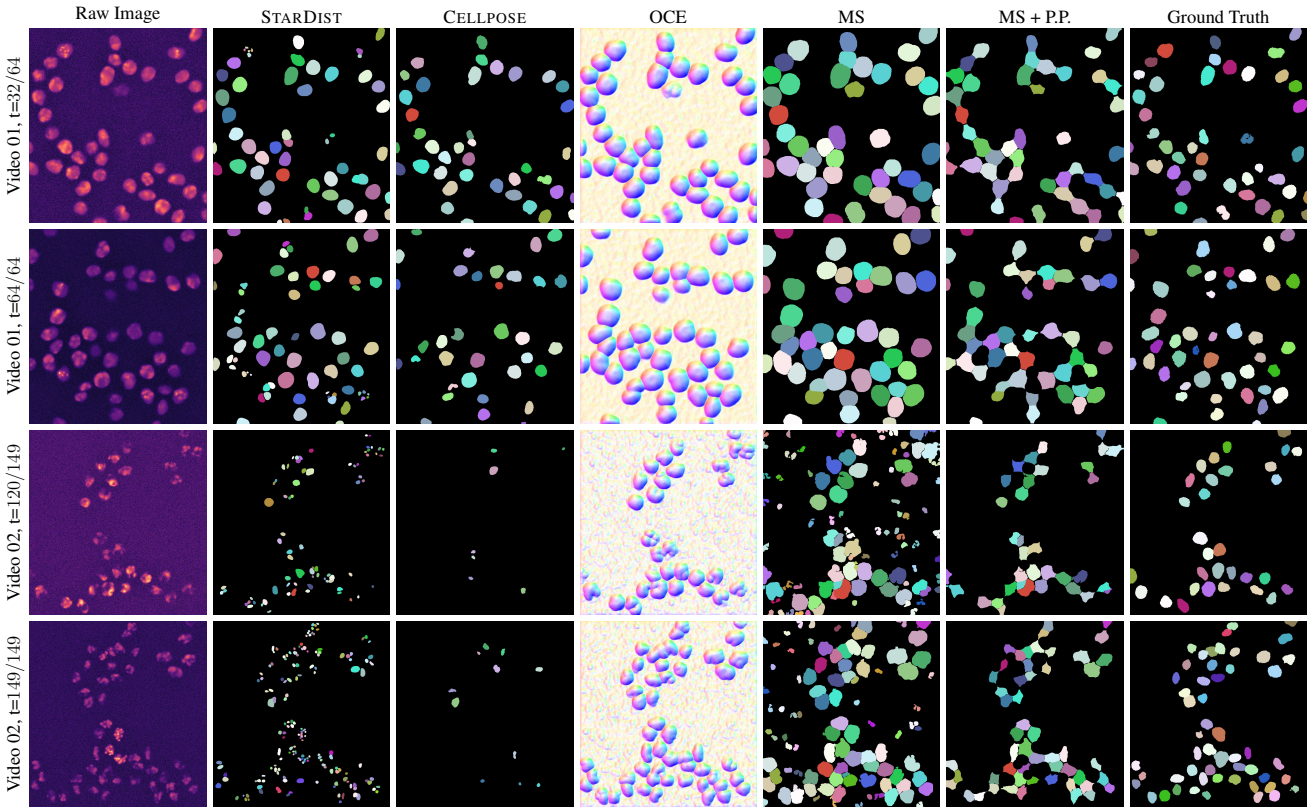

Figure 8. **Qualitative results on the SIMULATED dataset in the Cell Tracking Challenge [26].** The SIMULATED dataset comprises of two time-lapse videos (Videos 01 and 02) which contain 64 and 149 image frames respectively. Shown here are individual raw images from the two videos (first column), predicted instance segmentations obtained using the pre-trained baseline models StarDist (second column) and Cellpose (third column), dense prediction of Object-Centric Embeddings (OCEs) obtained using Cellulus (fourth column), intermediate instance segmentations obtained by applying mean-shift (MS) clustering on the dense OCEs (fifth column), these intermediate instance segmentations are further post-processed (sixth column, see more details in the Section 4.1 - *Scale Informs All Parameter Choices*) and the Ground Truth Instance Segmentation available for evaluation purposes (seventh column).
Video 02 in SIMULATED (see last two rows) contains cells with visible granules which cause over-segmentation for both the evaluated, pre-trained baseline models.