Supplementary: Large-Scale Multi-Hypotheses Cell Tracking Using Ultrametric Contours Maps

Jordão Bragantini, Merlin Lange, and Loïc Royer

Chan Zuckerberg Biohub, San Francisco, USA
{jordao.bragantini,merlin.lange,loic.royer}@czbiohub.org

This supplementary document presents additional information about our setup for the quantitative experiment on the cell tracking challenge (CTC) [7]. Videos of the cell tracking results and a python library with the proposed method are also included in the supplementary material pack.

1 CTC Quantitative comparison

1.1 Processing details

Here, we describe in detail our CTC submission (Section 4.1).

All datasets were linearly interpolated on the z-axis to make them isotropic. While isotropy is not necessary, it benefits the algorithm's accuracy. They were normalized by their lower and upper quantiles (lq and uq in Table 1), DRO dataset had corrupted voxels which were set to 0.

In the datasets MDA231 and CE, where 3D segmentation labels are available, we trained a U-Net to predict the foreground and the cell boundaries. In the DRO, TRIF and TRIC datasets, where 3D labels are not available, we tried two different approaches:

- IM: where the cells and their contours were estimated using traditional image processing operators;
- PL: where we trained the U-Nets on labels generated by an algorithm and not manually annotated.

Both PL and IM processing began the same way. The cells were detected using a difference of Gaussians with σ_1 and σ_2 thresholding at 0.5, 0.5, and 0.75 times their Otsu for DRO, TRIF and TRIC, respectively. The cell contour maps were simply the inverse of the Gaussian blurred image with σ_1 and normalized to be between 0 and 1. Through visual inspection, we selected $\sigma_1 = 2$ and $\sigma_2 = 8$ for the DRO dataset, $\sigma_1 = (0, 1, 1)$ and $\sigma_2 = (1, 4, 4)$ for the TRIF dataset ((z, y, x) order), and $\sigma_1 = 1$ and $\sigma_2 = 6$ for the TRIC dataset. For the pseudo-labels, we used these contour maps and detections to run the hierarchical watershed and performed a horizontal cut in the hierarchy to obtain the flat segments used for training the network. In IM, they were used as the tracking input.

For the training set, the normalized time-lapses were used to learn the network weights and find the optimal hyper-parameters using grid search and cross-validation between datasets. For that, we trained a network in a single dataset and computed the results on the other. For example, we trained on dataset 01 and predicted the network outputs, tracked, and measured the accuracy on dataset 02.

The parameter power p of the ILP weights ($w^p_{\cdot\cdot}$) was fixed to 4 throughout the experiments.

The U-Nets were trained for 20 epochs using a learning rate of 10^{-4} with exponential weight decay of 0.95 at every epoch. The networks used a kernel size of 5, with simple convolution blocks without residual connections or batch normalization. The number of planes used was 32, 64, 128, and 256. The network features are linearly interpolated in the up-sampling step of the U-Net decoder. The best parameters found through the grid search are reported in Table 1.

The images are processed in overlapping tiles to avoid boundary artifacts of the CNNs' inference. Once a whole frame was computed, the foreground detection channels were binarized with a *Threshold* parameter, and the contours were Gaussian filtered with σ, values at Table 1. The PL and IM approaches shared the same parameters.

The DRO and TRIC datasets, where cells move coherently during embryo development, had their hierarchies registered with a local movement estimation between adjacent frames. The local movement is pre-computed as a low-resolution time-lapse and then applied to translate each candidate segment, influencing their location when accessing their neighbors in adjacent frames and their IoU. The TRIF datasets were registered with a global translation between adjacent frames because some frames are shifted, probably due to acquisition issues.

For the final submission, both training datasets were used for learning the weights, and the segmentation and tracking were done using the hyperparameters found on the training sets as described above.

Table 1: Table of parameters used for our cell tracking challenge submission. †Gaussian filter applied on yx axes only.

	normalization		CNN output		hierarchy construction				tracking			
Dataset	lq	uq	σ	Threshold	Min. Area	Max. Area	Contour Strength	Noise	Link Radius	w_α	w_β	w_δ
MDA231	0.001	0.9999	0.5	0.5	2500	25000	0.2	0	75	-0.01	-0.01	-0.001
CE	0.001	0.9999	1	0.1	5000	1000000	0	0.4	100	-0.005	-0.005	0
DRO	0.001	0.9999	1	0.5	1500	50000	0.1	0.1	50	-0.001	-0.001	-0.001
TRIF	0.5	0.99999	$1^\dagger$	0.5	1000	15000	0	0	50	-0.001	-0.001	0
TRIC	0.05	0.999	2	0.5	250	15000	0	0	25	-0.01	-0.001	-0.001

1.2 Cross-validation results

Table 2 reports our cross-validation results on the training set from Section 4.1, from the main text.

The image processing solution showed superior results to the pseudo-label alternatives in the DRO, TRIF, and TRIC datasets. Hence, it was the regime used in the final submission. Notably, the gap in evaluation metrics is the largest in the extremely challenging DRO dataset, where the leaderboard scores are smaller than other datasets. This is one of the datasets in which we achieved the top score in the hidden test set (Table 1 from the main text).

Table 2: Results on the training set of the cell tracking challenge; Fully supervised (Sup) and pseudo-labels (PL) regimes used cross-validation; Image processing (IM) does not require training and achieved better results, especially on the DRO dataset.

Dataset	Regime	File	TRA	SEG	CTB
MDA231	Sup	1	0.888	0.613	0.751
		2	0.923	0.626	0.774
		AVG	0.905	0.620	0.763
CE	Sup	1	0.970	0.705	0.838
		2	0.955	0.649	0.802
		AVG	0.963	0.677	0.820
DRO	PL	1	0.693	0.425	0.559
		2	0.923	0.528	0.726
		AVG	0.808	0.476	0.642
	IM	1	0.812	0.512	0.662
		2	0.934	0.556	0.745
		AVG	0.873	0.534	0.704
TRIF	PL	1	0.813	0.607	0.710
		2	0.811	0.623	0.717
		AVG	0.812	0.615	0.714
	IM	1	0.870	0.687	0.779
		2	0.918	0.679	0.799
		AVG	0.894	0.683	0.789
TRIC	PL	1	0.946	0.552	0.749
		2	0.880	0.690	0.785
		AVG	0.913	0.621	0.767
	IM	1	0.980	0.527	0.754
		2	0.853	0.733	0.793
		AVG	0.917	0.630	0.773

2 Softwares

For our implementation, we used Gurobi [1] to solve the ILP. The deep learning routines used PyTorch [4]. The hierarchical segmentation was done using Higra [5]. Throughout the code, we heavily used NumPy [2], SciPy [8], CuPy [3] and scikit-image [9].

3 Videos

The supplementary material pack contains eight videos: One time-lapse per dataset of the final submissions of CTC, five in total; one video each for the peripodial (PRE) and proper discs (PRO) cells from the Epithelial Cell Benchmark, and a video of the whole embryo tracking.

These time lapses were visualized, and the videos were generated using napari [6].

References

1. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2023), `https://www.gurobi.com`
2. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., et al.: Array programming with numpy. Nature **585**(7825), 357–362 (2020)
3. Nishino, R., Loomis, S.H.C.: Cupy: A numpy-compatible library for nvidia GPU calculations. Advances in Neural Information Processing Systems Workshop **151**(7) (2017)
4. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems. pp. 8026–8037 (2019)
5. Perret, B., Chierchia, G., Cousty, J., Guimaraes, S.J.F., Kenmochi, Y., Najman, L.: Higra: Hierarchical graph analysis. SoftwareX **10**, 100335 (2019)
6. Sofroniew, N., Lambert, T., Evans, K., Nunez-Iglesias, J., Bokota, G., Winston, P., Peña-Castellanos, G., Yamauchi, K., Bussonnier, M., Doncila Pop, D., Can Solak, A., Liu, Z., Wadhwa, P., Burt, A., Buckley, G., Sweet, A., Migas, L., Hilsenstein, V., Gaifas, L., Bragantini, J., Rodríguez-Guerra, J., Muñoz, H., Freeman, J., Boone, P., R Lowe, A., Gohlke, C., Royer, L., Pierré, A., Har-Gil, H., McGovern, A.: napari: a multi-dimensional image viewer for Python. `https://doi.org/10.5281/zenodo.3555620`, `https://github.com/napari/napari`
7. Ulman, V., Maška, M., Magnusson, K.E., Ronneberger, O., Haubold, C., Harder, N., Matula, P., Matula, P., Svoboda, D., Radojevic, M., et al.: An objective comparison of cell-tracking algorithms. Nature Methods **14**(12), 1141–1152 (2017)
8. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al.: Scipy 1.0: fundamental algorithms for scientific computing in python. Nature Methods **17**(3), 261–272 (2020)
9. Van der Walt, S., Schönberger, J.L., Nunez-Iglesias, J., Boulogne, F., Warner, J.D., Yager, N., Gouillart, E., Yu, T.: scikit-image: image processing in python. PeerJ **2**, e453 (2014)