BRIEF COMMUNICATION
https://doi.org/10.1038/s41587-022-01427-7
nature biotechnology

Check for updates

OPEN

Automated reconstruction of whole-embryo cell lineages by learning from sparse annotations

Caroline Malin-Mayor[1], Peter Hirsch[2,3], Leo Guignard[1,4], Katie McDole[1,5], Yinan Wan[1,6], William C. Lemon[1], Dagmar Kainmueller[2,3], Philipp J. Keller[1], Stephan Preibisch[1] and Jan Funke[1]✉

We present a method to automatically identify and track nuclei in time-lapse microscopy recordings of entire developing embryos. The method combines deep learning and global optimization. On a mouse dataset, it reconstructs 75.8% of cell lineages spanning 1 h, as compared to 31.8% for the competing method. Our approach improves understanding of where and when cell fate decisions are made in developing embryos, tissues, and organs.

With recent advances in light-sheet imaging techniques, it is possible to acquire whole embryo developmental datasets over long time scales with high spatial and temporal resolution in complex organisms such as mouse, *Drosophila*, and zebrafish[1]. The resulting datasets contain information required to track the movement and division of nuclei over time, yielding lineage trees and quantitative data on cellular dynamics that are crucial to the study of developmental biology at the cellular level[2]. However, manually tracing lineages with dedicated tools like MaMuT[3] or Mastodon (https://github.com/mastodon-sc/mastodon) is arduous, and for complex, developing organisms it is only feasible to annotate a small percentage of all tracks, making automatic cell tracking necessary for holistic analysis.

Cell-tracking algorithms have been developed for and tested on diverse datasets with different characteristics. While hand-engineered features are sufficient for cell detection and tracking in some model organisms[4,5], learned dataset-specific features, given sufficient training data, improve performance for datasets with heterogeneous cell or nucleus phenotypes and varying imaging statistics over time and space. In particular, deep learning has been shown to improve cell detection[6,7], segmentation[8–11], and tracking[7,11–14] on a variety of datasets. Additionally, it has been shown that tracking methods that take into account global spatiotemporal context perform better, especially for datasets with more movement between time frames[13]. Tracking by graph optimization over a large spatiotemporal context allows inclusion of biological knowledge about track length and cell cycle, improving track continuity[6,15,16] and even allows recovery from noisy detection and segmentation[17].

Only a few of the aforementioned cell-tracking methods are readily applicable to the unique challenges posed by contemporary four-dimensional (3D+t) light-sheet datasets, the focus of this work. Practical methods for this kind of data should take into account temporal and 3D spatial context, easily scale to multi-terabyte datasets, and ideally should not require a manual segmentation of cells for training, owing to the time required to generate per-pixel ground truth. Of methods that fulfill these requirements, tracking with Gaussian mixture models (TGMM)[5] has been shown to work well on model organisms with approximately ellipsoid nuclei. More recently, the ELEPHANT tracking platform employed deep learning for cell detection and per-frame linking in light-sheet datasets with diverse cell appearance and movement[12]. ELEPHANT requires a manual pseudo-segmentation of nuclei by ellipsoid fitting, which takes less time to generate than a per-pixel manual segmentation, but more than point annotations.

Our method combines global optimization and learned features, generating cell lineages through global graph optimization with learned costs. We show that this combination substantially decreases tracking error on three diverse datasets of different model organisms with different temporal resolution, signal-to-noise ratio, and nuclear appearance. Features are learned from sparse point annotations produced by current manual lineage tracking tools like MaMuT and Mastodon, and thus do not require a manual segmentation or dense lineage annotations, which allows rapid generation of training data. Crucially, the steps of our method—including the global optimization—can be computed in a distributed fashion, which is necessary to process multi-terabyte light-sheet datasets and enable the study of whole embryo morphogenesis.

An overview of our method is shown in Fig. 1. Because we are learning features from the data, the method is not tied to a specific input type or format: we use fused and unfused light-sheet recordings with a single fluorescent nuclear channel, and could easily extend to multi-channel input. We use sparse point annotations to train a convolutional neural network to predict at each pixel a 'cell indicator' value that peaks at the center of each nucleus[6,18]. In contrast to ref. [6] we additionally predict a 'movement vector' that points to the center of the same cell nucleus in the previous time frame[7,12]. From these predictions, we generate a candidate graph in two steps: first, we place nodes at the local maxima of the cell indicator values to represent possible cell center locations, with a score to encode the confidence of the network; and second, we locally connect nodes in adjacent frames with edges to represent the possibility that the nodes represent the same cell, and assign a score to each edge on the basis of agreement with the predicted movement vector.

Next, we solve a globally constrained optimization problem on the candidate graph to select a subset of nodes and edges that form coherent lineage trees. We know that between time frames, cells can move, divide into two, enter or leave the field of view, or die, but not merge or split into more than two. Thus, we introduce tree

[1]HHMI Janelia, Ashburn, VA, USA. [2]Max Delbrück Center for Molecular Medicine in the Helmholtz Association, Berlin, Germany. [3]Faculty of Mathematics and Natural Sciences, Humboldt-Universität zu Berlin, Berlin, Germany. [4]CNRS, UTLN, LIS 7020, Turing Centre for Living Systems, Aix Marseille University, Marseille, France. [5]MRC Laboratory of Molecular Biology, Cambridge, UK. [6]Biozentrum, University of Basel, Basel, Switzerland.
✉e-mail: funkej@janelia.hhmi.org

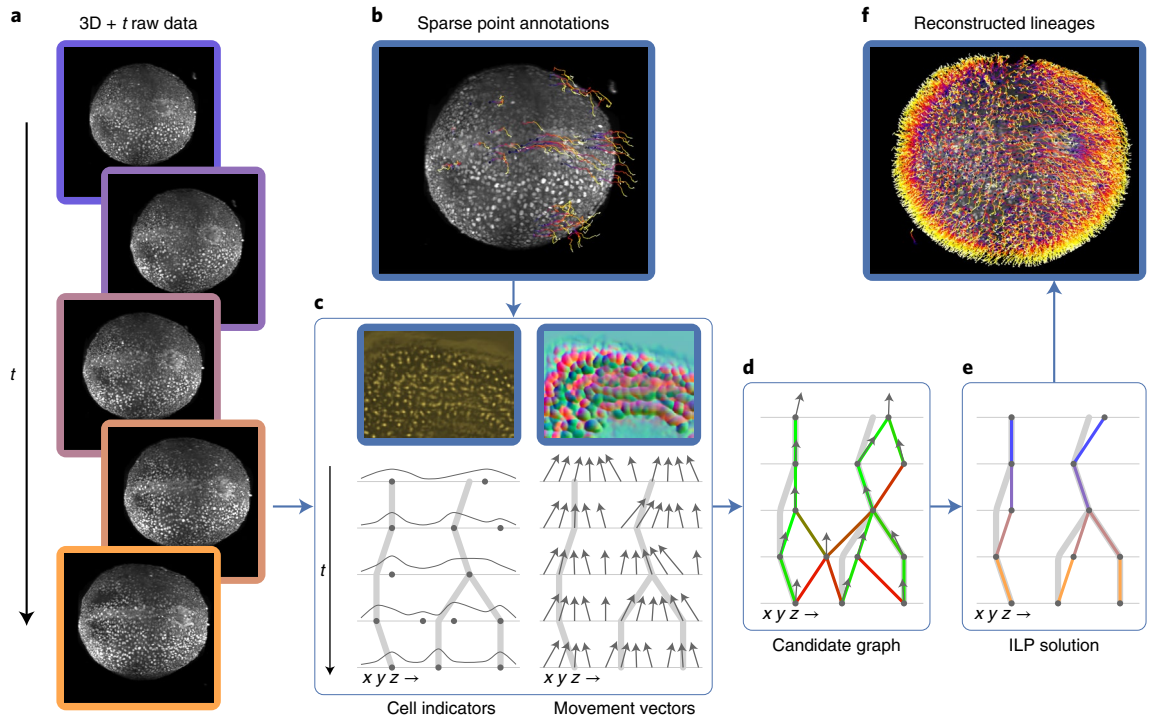


Fig. 1 | Overview of the method, including data and results from the mouse dataset. a, Raw mouse data over 50 time points, visualized as a max intensity projection. **b**, Sparse point annotations superimposed over the first frame of the raw data. Purple dots show the locations of annotated cells in the first time point, and the tails show the movement over time. **c**, Top, examples of the cell indicators (shown as intensity values) and 3D movement vectors (shown as RGB). The cell indicator is trained to have maxima at the center of each nucleus, and the movement vector network is trained to predict the relative location of the same cell nucleus in the previous time point. Bottom, sketch of the predictions over time. Thick gray lines represent ground truth lineages, dots are cell indicator maxima, and arrows are movement vectors. **d**, Candidate graph extracted from the network output. Candidate cells are at cell indicator maxima, and nearby cells are connected with edges that are scored by agreement with the movement vector. **e**, Consistent lineage trees extracted from the candidate graph by global optimization using learned features and biological priors. **f**, Densely reconstructed lineages visualized over the mouse data.

constraints to prevent merging and divisions producing more than two progeny. The objective function incorporates prior knowledge that cell movement is much more common than division or death. We further encourage long and continuous lineages by penalizing the start of new tracks. These tree constraints and continuity costs are similar to those in previous work[6,15,17]; however, we also incorporate the node and edge scores generated by the neural networks into the objective function as learned costs. Thus, we optimize for valid lineages that are both continuous and supported by the learned cell location and movement features. While ref. [6] employ a minimum-cost flow solver on the whole dataset, our integer linear program (ILP) formulation of the optimization problem additionally allows block-wise solving in parallel on large datasets by introducing additional constraints to ensure consistent solutions between adjacent blocks.

We evaluate our method on three sparsely annotated datasets from different commonly used model organisms to study embryogenesis: mouse [19], *Drosophila*[5], and zebrafish [20] (see Supplementary Note 1 for details about the datasets and annotations). We compare the performance of our method against TGMM, the previous state-of-the-art method on these datasets[5,19], and greedy tracking using a per-frame nearest neighbor linking algorithm similar to the ELEPHANT tracking method[12]. We compute multiple metrics, including the fraction of perfectly constructed lineages over a range of time periods, and errors per ground truth edge, broken into the following error types: false negative edges (FN), identity switches (IS; when two tracks switch off following the same cell), false positive divisions (FP-D), and false negative divisions (FN-D) (Supplementary Figure 1). False positive edges cannot be computed using sparse ground truth, because we cannot tell if unmatched reconstructions are false positives or tracking unannotated cells, and thus they are not included in our quantitative analysis. We show in Fig. 2 that, with around 20 hours of ground-truth annotation effort, our method correctly reconstructs more cell lineages than both baselines over all time ranges for all datasets. The largest improvement over TGMM is on the mouse dataset: our method correctly tracks 75.8% of mouse cells over a time span of 1 h (12 time frames), compared with 31.8% for TGMM. By 175 min (35 frames), our method still correctly tracks more than half of all cells, while TGMM tracks less than 8%. On all three datasets, our method greatly reduces false negative edges as compared to TGMM, while when compared with the greedy baseline, our method produces far fewer false positive divisions. Supplementary Note 2 contains a detailed description of the evaluation metrics and baselines, and further observations about the performance on various metrics across organisms and evaluation regions.

Both the candidate graph generation and lineage optimization steps of our method are fully parallelizable and scale linearly with the size of the recording, which enables dense lineage reconstruction on very large datasets in reasonable time. On 20 GPUs and 100 CPU cores, reconstruction of dense lineages took about 44 h on the 4.7 TB mouse dataset, generating more than 7 million cell detections and 360,000 tracks over the 44-h recording. Given the estimates that there are 6 million true cell detections in the dataset and that an annotator can click on a cell center every 1.5–3 s, it would take 2,500–5,000 annotator-hours to manually trace all lineages in this dataset. The source code of our method is publicly available,

together with training and inference scripts and extensive documentation (https://github.com/funkelab/linajea).

The ability to densely reconstruct cell lineages in such large, information-rich datasets opens up vast opportunities for exploring cell fate dynamics and tissue morphogenesis. Accurately following cells and their progeny over extended time periods allows identification of individual cell behaviors that are not visible with shorter and less accurate lineages. For example, being able to accurately track more than half of all cells over a time window of 175 min in the mouse dataset, as compared to only 30 min with previous methods, greatly reduces the manual curation needed to test hypotheses such as the existence of neuromesodermal progenitors that can produce neural or mesodermal progeny. While further work is required to improve detection of cell division, the dense cell lineages we publish with this method are a rich source of information about the development and cell fate dynamics of common model organisms.

Method

Network architecture, training, and prediction. To attain per-voxel predictions for cell locations and movements, we use a U-Net architecture with four resolution levels[21]. Our choice of the U-Net architecture over alternatives like Mask-RCNN or YOLO is motivated by two points: first, we found that the performance of the U-Net for cell detection is sufficient for our use case (in particular, regarding high recall for subsequent filtering through the ILP); and second, our proposed movement vectors require dense predictions, which can be naturally generated with a U-Net architecture. We also chose the same architecture for the cell indicators to unify, and thus simplify, our pipeline. To incorporate temporal as well as spatial context, we concurrently feed seven 3D frames centered on the target time point and use four-dimensional convolutions until, owing to valid convolutions, the time dimension is reduced to 1. We downsample by factors (2, 2, 2) in xyz, except for anisotropic datasets (mouse, *Drosophila*) where we do not downsample in z in the first downsampling layer. We use 12 initial feature maps and increase by a factor of 3 at each level. When upsampling, we restrict our upsampling convolutional kernels to constant values, as we have observed this reduces artifacts in the output.

The cell indicator network is trained on sparse point annotations and predicts the centers of cell nuclei. The training signal for this network, called the cell indicator value, is a Gaussian with maximum value 1 at the cell center annotation and decreasing according to a hyperparameter σ. With only sparse annotations, it is unknown if pixels far from cell center annotations are background or cells that were not annotated. To avoid training on unknown regions, we construct a 'training mask' around each annotation with a user-defined radius. This radius should be small enough that the mask will not overlap with neighboring cell nuclei. We only train on the mean squared error loss within the training mask. We are not training our cell indicator network on any background regions, so the behavior is unconstrained in the background. After prediction, we use local non-maximum suppression (NMS) to extract cell center candidates, with the goal of detecting all cell centers along with potential false positives owing to the unconstrained background behavior. The NMS window size is dataset dependent and should be a bit smaller than the minimal distance between two cell centers. To reduce the number of false positives, especially in background regions, we only consider detections above a threshold cell indicator value, determined empirically for each dataset and model. Additionally, if a foreground mask is available (as in the zebrafish dataset) we filter detections to those that lie in the foreground.

In addition to the cell indicator network, we train a movement vector network to predict the movement of cells between frames. For a pixel near to a cell in frame t, the movement vector is a 3D offset vector that points to the relative location of the center of the same cell in frame $t-1$. Predicting the offset to the same cell in the previous time frame, rather than the next time frame, allows divisions to be represented naturally, since each daughter cell points to the center of the parent cell. We calculate the loss on two different masked regions. Loss L_A is the mean squared error between the ground truth and predicted movement vectors, calculated over the same training mask as the cell indicator network. Loss L_B limits the error to voxels with maximal cell indicator values after NMS that also are within the training mask. The total loss is the weighted sum $L = \alpha L_A + (1-\alpha) L_B$, with $\alpha = \frac{1}{1+\exp 0.01\,(-i+20{,}000))}$, and i being the number of training iterations. This weighting scheme weights L_A higher at the beginning of training, when the cell indicator network is still converging, with a smooth transition to L_B at 20,000 iterations.

These networks are trained simultaneously for 400,000 iterations, with batch size 1. Batches are randomly sampled from annotated locations, and random augmentations including elastic deformation, mirroring, transposing axes, and intensity augmentation are applied using the GUNPOWDER library (https://github.com/funkey/gunpowder). Prediction is then performed block-wise using parallelization over multiple GPUs to process large datasets efficiently. To eliminate edge artifacts, we ensure that our prediction stride is a multiple of the network downsample factors[22].

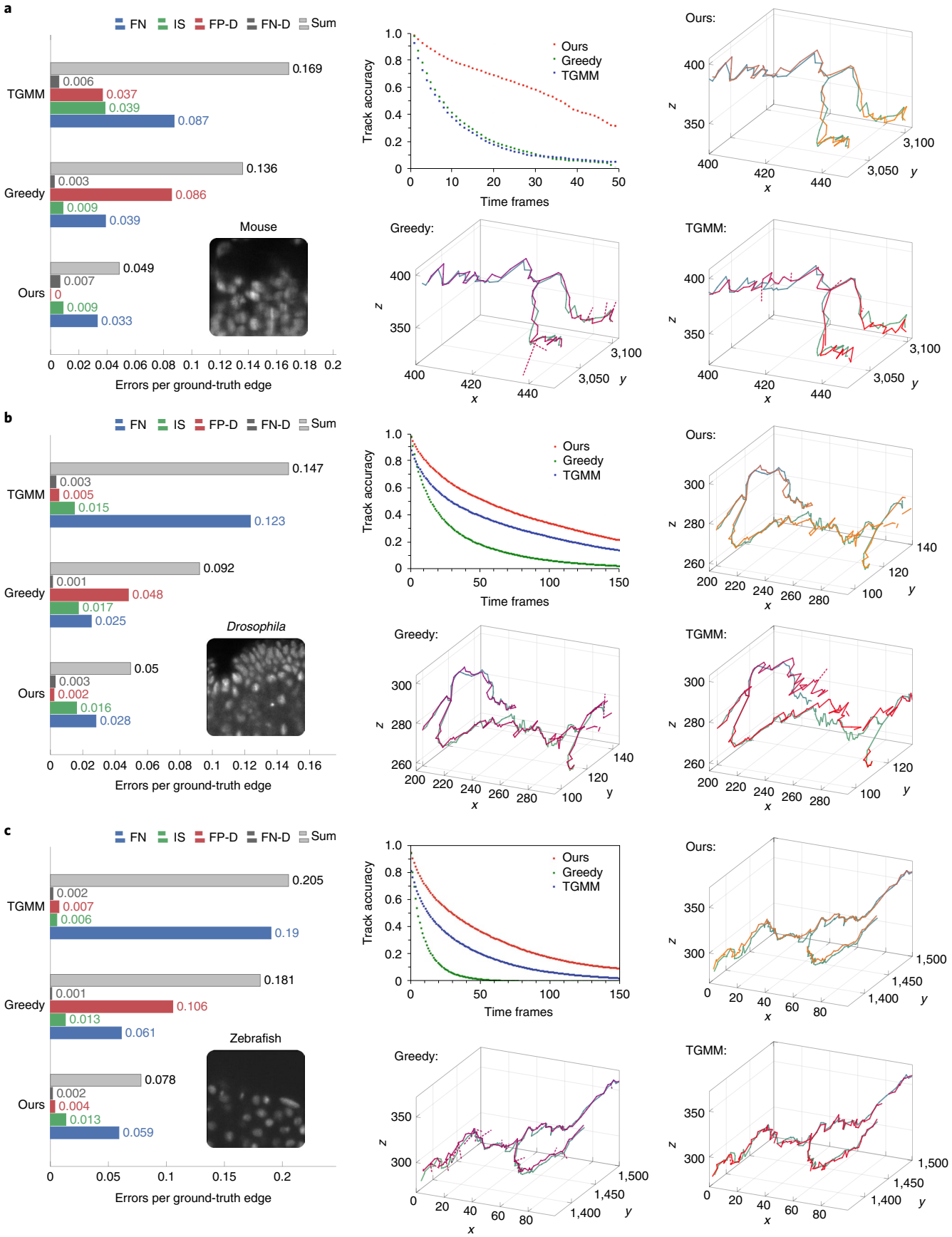
Candidate graph extraction. After prediction, we create a directed candidate graph $G = (V, E)$ with nodes that represent possible cell center locations and edges that represent possible movements of the same cell between frames. G is expected to contain extra nodes and edges, which will be filtered out in the final step.

V is the set of NMS detections. Each $v \in V$ has a three-dimensional location l_v, a time t_v, a predicted cell indicator score s_v, and a predicted movement vector $\mathbf{m}_v$. We avoid storing the predicted cell indicators and movement vectors at every pixel by performing NMS on the cell indicator values during prediction and only save the predicted values at the detection.

We construct the set of directed edges E by locally connecting nodes in adjacent frames with edges that point one frame backwards in time. For each candidate v at time t_v, we compute the predicted location $\hat{l}_v$ of the same cell in the previous frame: $\hat{l}_v = l_v + m_v$. Then, we add an edge from v to each node candidate u at time $t_v - 1$ where the predicted distance $\hat{d}_e = \| \hat{l}_v - l_u \|^2$ is less than a hyperparameter θ. $\hat{d}_e$ is stored as a score on each edge.

Discrete optimization to find lineage trees. We construct a lineage tree by selecting a subset of nodes and edges from G. We define a vector $\mathbf{y} = \left[\mathbf{y}^V, \mathbf{y}^E\right]^\intercal \in \{0, 1\}^{|V|+|E|}$ such that each element of the vector corresponds to a node or edge in G. Then $G(\mathbf{y})$ is the subgraph induced by $\mathbf{y}$ that only contains nodes and edges with the corresponding element of $\mathbf{y}$ equal to 1.

Fig. 2 | Comparison of tracking errors on three datasets. a–c, Comparison of tracking errors on mouse (**a**), *Drosophila* (**b**), and zebrafish (**c**) datasets. **a–c**, Left, average errors per ground truth edge for each error type; middle top, fraction of error-free tracks for a given track length; middle bottom, top right, and bottom right, example ground truth track (green) with superimposed tracking result (orange or red) for our method (top right), the greedy baseline (middle bottom), and TGMM (bottom right), respectively. Other than the dashed false positive divisions, we only show detections that matched the selected ground truth track.


a
FN IS FP-D FN-D Sum
TGMM
0.169
0.006
0.037
0.039
0.087
Greedy
0.136
0.003
0.086
0.009
0.039
Ours
0.049
0.007
0
0.009
0.033
Mouse
0 0.02 0.04 0.06 0.08 0.1 0.12 0.14 0.16 0.18 0.2
Errors per ground-truth edge
Track accuracy
Time frames
Ours
Greedy
TGMM
Ours:
Greedy:
TGMM:
b
FN IS FP-D FN-D Sum
TGMM
0.147
0.003
0.005
0.015
0.123
Greedy
0.092
0.001
0.048
0.017
0.025
Ours
0.05
0.003
0.002
0.016
0.028
Drosophila
0 0.02 0.04 0.06 0.08 0.1 0.12 0.14 0.16
Errors per ground-truth edge
c
FN IS FP-D FN-D Sum
TGMM
0.205
0.002
0.007
0.006
0.19
Greedy
0.181
0.001
0.106
0.013
0.061
Ours
0.078
0.002
0.004
0.013
0.059
Zebrafish
0 0.05 0.1 0.15 0.2
Errors per ground-truth edge

We then construct a constrained optimization problem that minimizes the objective

$$\min_{\mathbf{y}} C(\mathbf{y}) \quad \text{s.t.} \quad G(\mathbf{y}) \in \mathcal{T}^2,$$

where $\mathcal{T}^2$ is the set of binary forests and $C : \mathbf{y} \rightarrow \mathbb{R}$ assigns a cost for each set of selected nodes and edges. Thus, the goal is to select the cost-minimal subset of nodes and edges from G that form a binary forest.

To simplify the presentation of the cost function, we introduce two auxiliary indicator vectors of length $|V|$ that can be entirely derived from $\mathbf{y}$. The indicator for tracks appearing, $\mathbf{y}^T$, is 1 for nodes at the beginning of a track and 0 otherwise. $\mathbf{y}^D$ represents a division and is 1 for the parent node of a division. For a formalization of the definition of these auxiliary vectors from $\mathbf{y}$, see below.

With these auxiliary indicator variables, we define a linear cost function as follows:

$$C(\mathbf{y}) = \langle \mathbf{c}, \mathbf{y} \rangle + \langle \mathbf{c}^T, \mathbf{y}^T \rangle + \langle \mathbf{c}^D, \mathbf{y}^D \rangle, \tag{1}$$

where $\mathbf{c} = \left[\mathbf{c}^V, \mathbf{c}^E\right]^\intercal$ is a vector containing the cost for selecting each node and edge, $\mathbf{c}^T$ is a vector containing the cost c^T of having to start a new track and $\mathbf{c}^D$ is a vector containing the cost c^D of having a division occurring. The costs c^T and c^D are constant parameters of the method, but the predicted cell indicator values and movement vectors are used to individualize the cost vector $\mathbf{c}$ for selecting each node and edge.

With s_i as the cell indicator score for node i, we define the node selection cost for node i as $c_i^V = \tau^V + w^V s_i$, where τ^V and w^V are parameters of the method. To encourage selection of higher cell indicator scores during minimization, w^V should be negative.

Similarly, with $\hat{d}_i$ as the distance between the predicted and actual offsets at edge i, we define the edge selection cost for edge i as $c_i^E = w^E \hat{d}_i$. Unlike with node scores, w^E should be positive to encourage selection of edges with lower scores, since those edges align better with the predicted cell movement.

To determine the optimal values of the ILP parameters c^T, c^D, τ^V, w^V, and w^E, we performed a grid search where we fixed $c^D = 1$ to eliminate redundant solutions. We selected the parameter set that minimized the sum of errors over the validation set (Supplementary Note 2).

Integer linear program formulation. We use an ILP to solve the constrained optimization problem with the Gurobi solver (https://www.gurobi.com/). The objective is the cost function $C(\mathbf{y})$ (Eq. 1). To ensure a binary forest with correctly set auxiliary variables, we implement three kinds of constraints: consistency, continuation, and split constraints.

The first consistency constraint requires that if an edge is selected, the incident nodes are selected as well. This constraint for edge $e=(v,u)$ is represented by the equation $2y_e - y_v - y_u \leq 0$. The second consistency constraint ensures that the number of selected incoming edges is ≤2, that is, for each node v: $\sum_{n \in N_v} y_n - 2y_v \leq 0$ with N_v being the set of edges from nodes in $t_v + 1$ to v. The continuation constraint ensures that if a node is selected either it is marked as the beginning of a track or the track continues. Let P_v be the set of edges from node v in time t_v to nodes in $t_v - 1$. We define the track continuation constraint as $\sum_{p \in P_v} y_p + y_v^T - y_v = 0$, ensuring that if node v is selected, either there is exactly one selected edge to time $t_v - 1$ or the track (appear) indicator y_v^T is set to 1 (and thus the associated cost has to be paid). Finally, the two split constraints ensure that the division indicator y_v^D is set for every cell that divides, that is, for each node v: $\sum_{n \in N_v} y_n - y_v^D \leq 1$ and $\sum_{n \in N_v} y_n - 2y_v^D \geq 0$.

Processing large volumes block-wise. Ideally, we would solve the ILP for the whole candidate graph at once to obtain a globally optimal solution. However, for large volumes, this is too time and memory intensive. Therefore, to obtain lineage trees for arbitrarily large volumes, we divide the candidate graph into a set of blocks B that tile the whole volume and use multiple processes to solve the ILP for many blocks in parallel.

Solving each block $b \in B$ completely independently can result in discontinuities in tracks between blocks, and the constraints would no longer be fulfilled at the boundaries. To ensure a consistent, valid solution across the whole volume, we allow each process to view a context region around the target region b that must be at least as large as the amount a cell can move in space between two frames and contain at least one time frame in each direction. Let $\hat{b}$ be the union of b and the surrounding context area. A process reads all nodes and edges in $\hat{b}$, solves the ILP, and writes the result for only the target region b into a central database. If the database already contains results in the context region, these selections will be introduced as further constraints into the ILP, ensuring the solution will be consistent across boundaries. At the block boundaries, we set the new track cost c^T to zero, because we do not want to penalize solutions that cross block boundaries.

The introduction of a context region introduces dependencies between neighboring blocks, and thus they cannot be run trivially in parallel. By ensuring that overlapping blocks are never run simultaneously using the DAISY library (https://github.com/funkelab/daisy), we obtain a globally consistent solution while retaining a high degree of parallel processing. While there is no guarantee of global optimality, with a large enough context region, we assume that nodes and edges further away do not affect the local solution in a target region.

The number of frames in a block influences the choice of optimal ILP hyperparameters. Therefore the size used in the grid search has to be the same, or at least of the same magnitude, as during inference.

Online content

Any methods, additional references, Nature Research reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41587-022-01427-7.

Received: 20 December 2021; Accepted: 12 July 2022;
Published online: 5 September 2022

References

1. Wan, Y., McDole, K. & Keller, P. J. Light-sheet microscopy and its potential for understanding developmental processes. *Ann. Rev. Cell Dev. Biol.* **35**, 655–681 (2019). Publisher: Annual Reviews.
2. Spanjaard, B. & Junker, J. P. Methods for lineage tracing on the organism-wide level. *Curr. Opin. Cell Biol.* **49**, 16–21 (2017).
3. Wolff, C. et al. Multi-view light-sheet imaging and tracking with the MaMuT software reveals the cell lineage of a direct developing arthropod limb. *eLife* **7**, e34410 (2018).
4. Bao, Z. et al. Automated cell lineage tracing in *Caenorhabditis elegans*. *Proc. Natl Acad. Sci. USA* **103**, 2707–2712 (2006).
5. Amat, F. et al. Fast, accurate reconstruction of cell lineages from large-scale fluorescence microscopy data. *Nat. Methods* **11**, 951–958 (2014).
6. Kok, R. N. U. et al. OrganoidTracker: efficient cell tracking using machine learning and manual error correction. *PLoS ONE* **15**, e0240802 (2020).
7. Hayashida, J., Nishimura, K. & Bise, R. MPM: joint representation of motion and position map for cell tracking. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 3822–3831 (IEEE, 2020).
8. Weigert, M., Schmidt, U., Haase, R., Sugawara, K. & Myers, G. Star-convex polyhedra for 3D object detection and segmentation in microscopy. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)* 3655–3662 (IEEE, 2020).
9. Cao, J. et al. Establishment of a morphological atlas of the *Caenorhabditis elegans* embryo using deep-learning-based 4D segmentation. *Nat. Commun.* **11**, 6254 (2020).

10. Stringer, C., Wang, T., Michaelos, M. & Pachitariu, M. Cellpose: a generalist algorithm for cellular segmentation. *Nat. Methods* **18**, 100–106 (2021).
11. Medeiros, G. d. et al. Multiscale light-sheet organoid imaging framework. Preprint at *bioRxiv* https://doi.org/10.1101/2021.05.12.443427 (2021).
12. Sugawara, K., Cevrim, C. & Averof, M. Tracking cell lineages in 3D by incremental deep learning. *eLife* **11**, e69380 (2022).
13. Ulman, V. et al. An objective comparison of cell-tracking algorithms. *Nat. Methods* **14**, 1141–1152 (2017).
14. Moen, E. et al. Accurate cell tracking and lineage construction in live-cell imaging experiments with deep learning. Preprint at *bioRxiv* https://doi.org/10.1101/803205 (2019).
15. Jug, F., Levinkov, E., Blasse, C., Myers, E. W. & Andres, B. Moral Lineage Tracing. In *2016 IEEE Conference on Computer Vision and Pattern Recognition* 5926–5935 (IEEE, 2016).
16. Haubold, C., Aleš, J., Wolf, S. & Hamprecht, F. A. A generalized successive shortest paths solver for tracking dividing targets. In *Computer Vision—ECCV 2016* (Eds Leibe, B., Matas, J., Sebe, N. & Welling, M.) 566–582 (Springer International Publishing, 2016).
17. Schiegg, M., Hanslovsky, P., Kausler, B. X., Hufnagel, L. & Hamprecht, F. A. Conservation Tracking. In *2013 IEEE International Conference on Computer Vision* 2928–2935 (IEEE, 2013).
18. Höfener, H. et al. Deep learning nuclei detection: a simple approach can deliver state-of-the-art results. *Comput. Med. Imaging Graph.* **70**, 43–52 (2018).
19. McDole, K. et al. In toto imaging and reconstruction of post-implantation mouse development at the single-cell level. *Cell* **175**, 859–876.e33 (2018).
20. Wan, Y. et al. Single-cell reconstruction of emerging population activity in an entire developing circuit. *Cell* **179**, 355–372.e23 (2019).
21. Ronneberger, O., Fischer, P. & Brox, T. U-Net: convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention* 234–241 (Springer, 2015).
22. Rumberger, J. L. et al. How shift equivariance impacts metric learning for instance segmentation. In *Proc. of the IEEE/CVF International Conference on Computer Vision* (ICCV, 2021).

Reporting summary. Further information on research design is available in the Nature Research Reporting Summary linked to this article.

Acknowledgements

We thank W. Patton and T. Nguyen for supporting the GUNPOWDER and DAISY libraries, S. Wolf for his guidance and feedback, and N. Eckstein, J. Buhmann, and A. Sheridan for helpful discussions. This work was supported by the Howard Hughes Medical Institute. K.M. was supported by the Medical Research Council, as part of United Kingdom Research and Innovation (MCUP1201/23). P.H. was supported by HFSP grant RGP0021/2018-102, the MDC Berlin-New York University exchange program. P.H. and D.K. were supported by the HHMI Janelia Visiting Scientist Program.

Author contributions

Conceptualization: J.F., P.J.K. Funding acquisition: J.F., P.J.K., S.P., K.M., P.H., D.K. Software: C.M.-M., P.H., J.F., L.G. Validation and evaluation: C.M.-M., P.H. Data and annotation generation: K.M., Y.W., W.C.L. Supervision: J.F., S.P., P.J.K., D.K. Writing, original draft: C.M.-M., J.F. Writing, review, and editing: C.M.-M., J.F., P.J.K., K.M., S.P., P.H., L.G., D.K.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41587-022-01427-7.

Correspondence and requests for materials should be addressed to Jan Funke.

Peer review information *Nature Biotechnology* thanks Péter Horváth, Talmo Pereira and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

Reprints and permissions information is available at www.nature.com/reprints.

nature research

Corresponding author(s): Jan Funke

Last updated by author(s): 2022/06/28

Reporting Summary

Nature Research wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Research policies, see our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed
☐	☒ The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☐	☒ A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☒	☐ The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐ A description of all covariates tested
☒	☐ A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☒	☐ A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☒	☐ For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐ For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐ For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☒	☐ Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	Standard python packages (networkx, pymongo, scipy, pandas, tensorflow, zarr, sklearn, gunpowder, daisy), see https://github.com/funkelab/linajea for a full list of requirements and https://github.com/funkelab/linajea_experiments for training and inference code.
Data analysis	Standard python packages (pymongo, scipy, pandas, zarr, sklearn), see https://github.com/funkelab/linajea for a full list of requirements and https://github.com/funkelab/linajea_experiments for evaluation code.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Research guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A list of figures that have associated raw data
- A description of any restrictions on data availability

Raw datasets are available through their respective publications (Mouse: McDole et al. (2018), Droso: Amat et al. (2014), ZFish: Wan et al. (2019b)). Ground-truth annotations used for training and evaluation of our method, together with the lineage tracks produced by our method, are publicly available at https://github.com/funkelab/linajea_experiments.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	The number of annotated cells (with links to previous frames) used for training and evaluation is given by the available ground-truth, i.e., 75,745 for the Drosophila dataset, 37,009 for the Mouse dataset, 34,530 for the Zebrafish dataset.
Data exclusions	No data was excluded.
Replication	Results can be replicated. The effect of random initialization of networks for training is reported in Supplementary Figure 6a.
Randomization	N/A
Blinding	N/A

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology and archaeology
☒	☐ Animals and other organisms
☒	☐ Human research participants
☒	☐ Clinical data
☒	☐ Dual use research of concern

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging