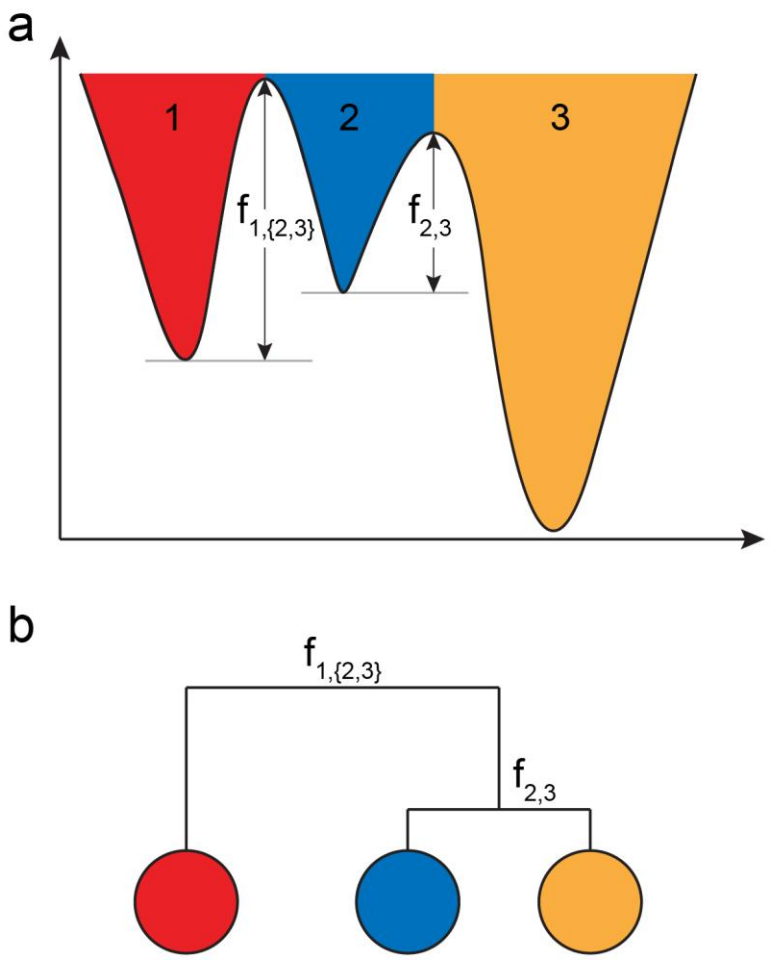
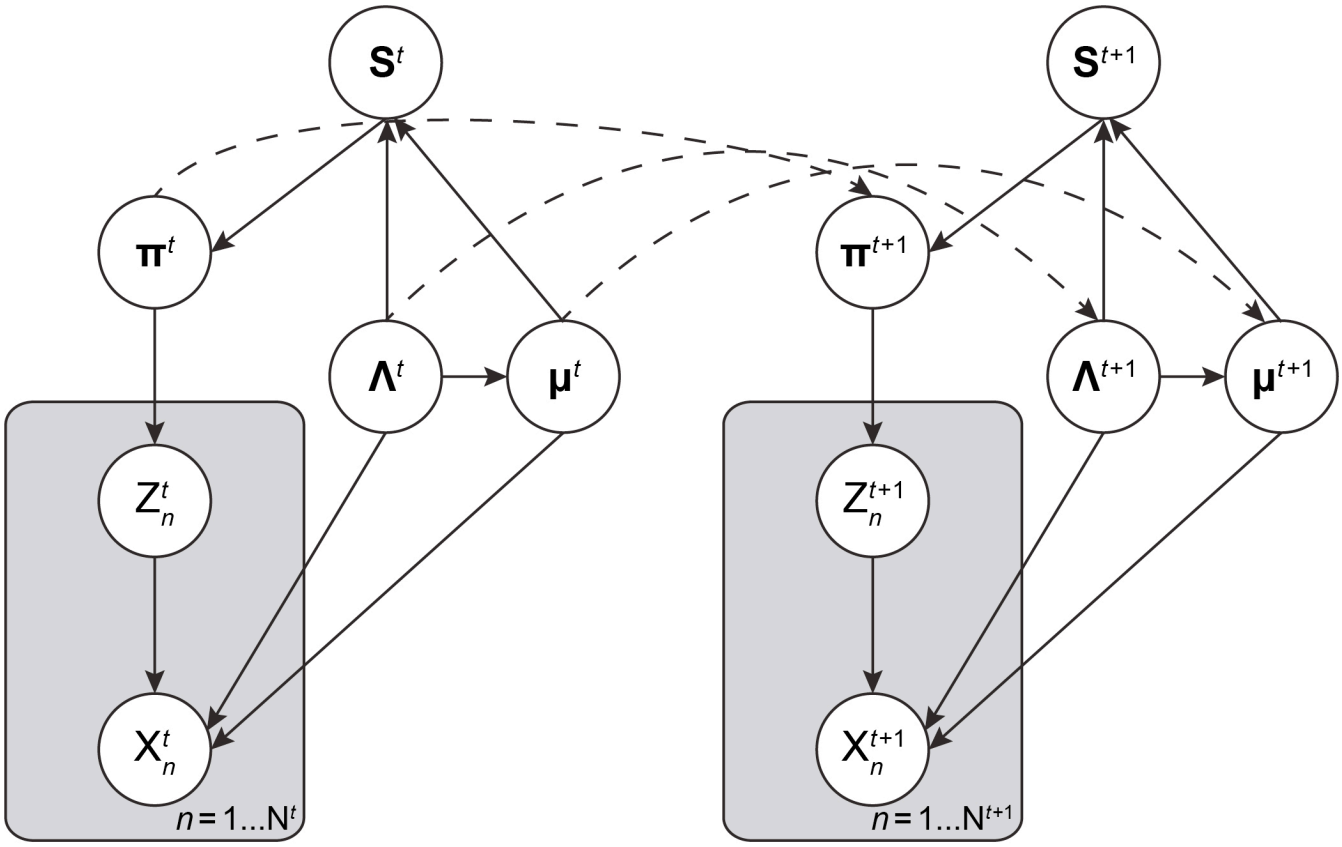


Supplementary Figure 1

Principles of persistence clustering with watershed

(**a**) One-dimensional example of a watershed. The function profile is segmented into three basins (red, blue, orange) by grouping convex segments associated with the same local minima.

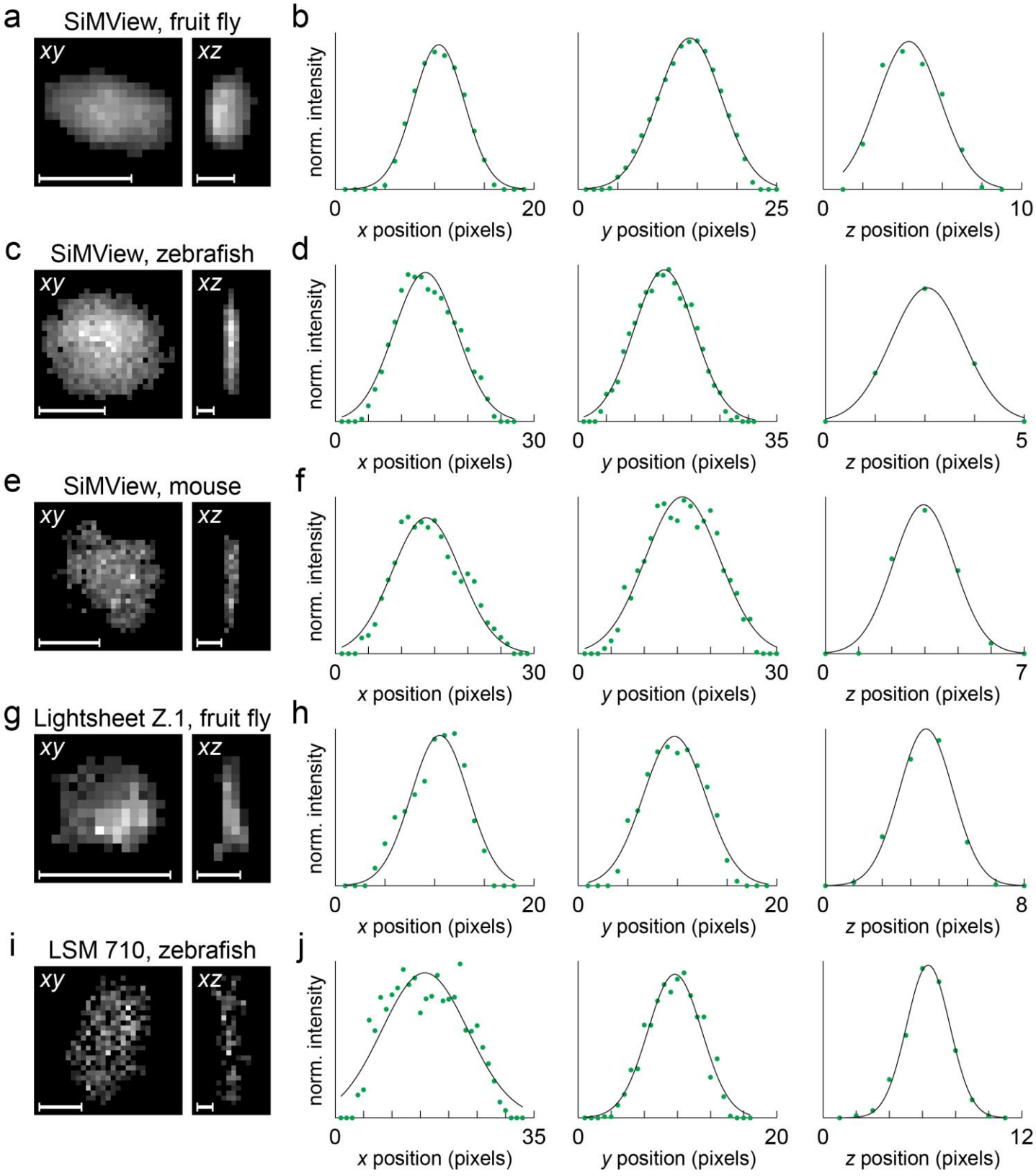
(**b**) Graphical representation using a dendrogram of how persistence-based clustering (PBC) can establish a hierarchical order for merging the different regions generated by the watershed in (a). In order to merge two regions, the parameter τ needs to be set to a value larger than the difference between the function value at the contact point between the two regions and the larger of the two local minima. Every time two regions in the dendrogram are merged, the τ value for the next merge can change. Thus, the dendrogram needs to be calculated sequentially. For example, the merging of region 1 (red) and region {2,3} (blue plus orange) is associated with a τ equal to $f_{1,\{2,3\}}$ instead of $f_{1,2}$, since region 3 has a lower local minimum than region 1.



Supplementary Figure 2

Graphical model for sequential Gaussian mixture models

Graphical model indicating conditional independence between all random variables present in our sequential Gaussian Mixture Model (GMM). $\mathbf{X}^t$ represents the observed intensity values in the image stack at time point t. $\mathbf{Z}^t$ represents the hidden variables that assign each $\mathbf{X}^t$ to a mixture in the GMM. The grey box indicates that these two random variables are considered independent and identically distributed (i.i.d.). $\boldsymbol{\mu}^t$ and $\boldsymbol{\Lambda}^t$ represent the mean matrix and precision matrix, respectively, for each mixture. $\boldsymbol{\pi}^t$ represents the responsibility for each mixture. The sub-graph defined by variables $\mathbf{X}^t$, $\mathbf{Z}^t$, $\boldsymbol{\mu}^t$, $\boldsymbol{\Lambda}^t$ and $\boldsymbol{\pi}^t$ is the standard graphical model defining a GMM. We incorporate binary variable $\mathbf{S}^t$ for each mixture to define the probability that a nucleus is dividing. Finally, we can unroll the model to perform inference sequentially in time as a dynamic Bayesian network. Thus, values at time point $t - 1$ act as priors for time point t.



Nature Methods: doi:10.1038/nmeth.3036

Supplementary Figure 3

Modeling of nuclei intensity profiles as Gaussian distributions

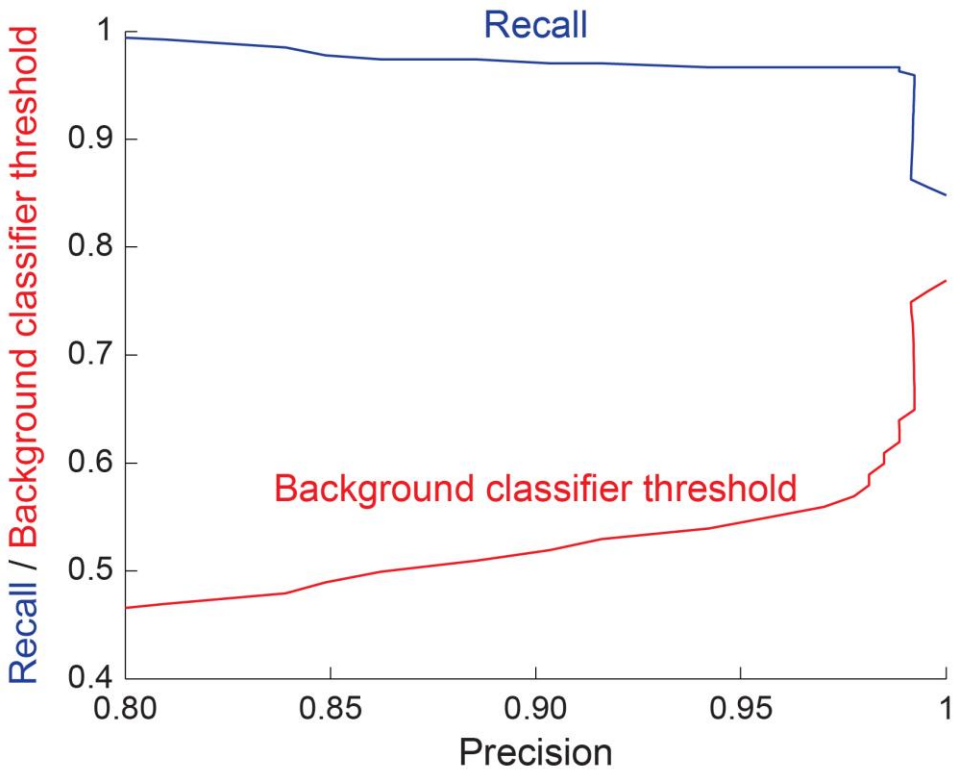
(**a**) Slices of an image stack in orthogonal planes *xy* (sliced orthogonally to detection axis) and *xz* (sliced parallel to detection axis) passing through the center of a segmented nucleus in a *Drosophila* dataset recorded with SiMView microscopy.

(**b**) Gaussian fits (black lines) to the projected intensity levels (green dots) of the stack in (a) along each of three main axes (marginal projections). The high accuracy of the fits in all three directions validates the assumption that nucleus intensity profiles can generally be well approximated by a Gaussian Mixture Model.

(**c**,**e**,**g**,**i**) Same as in (a), but for image slices obtained from a zebrafish data set recorded with SiMView microscopy (c), a mouse data set recorded with SiMView microscopy (e), a *Drosophila* data set recorded with a Carl Zeiss Lightsheet Z.1 light-sheet microscope (g) and a zebrafish data set recorded with a Carl Zeiss LSM 710 confocal microscope (i).

(**d**,**f**,**h**,**j**) Same as in (b), but for the image data shown in (c) (for (d)), (e) (for (f)), (g) (for (h)) and (i) (for (j)).

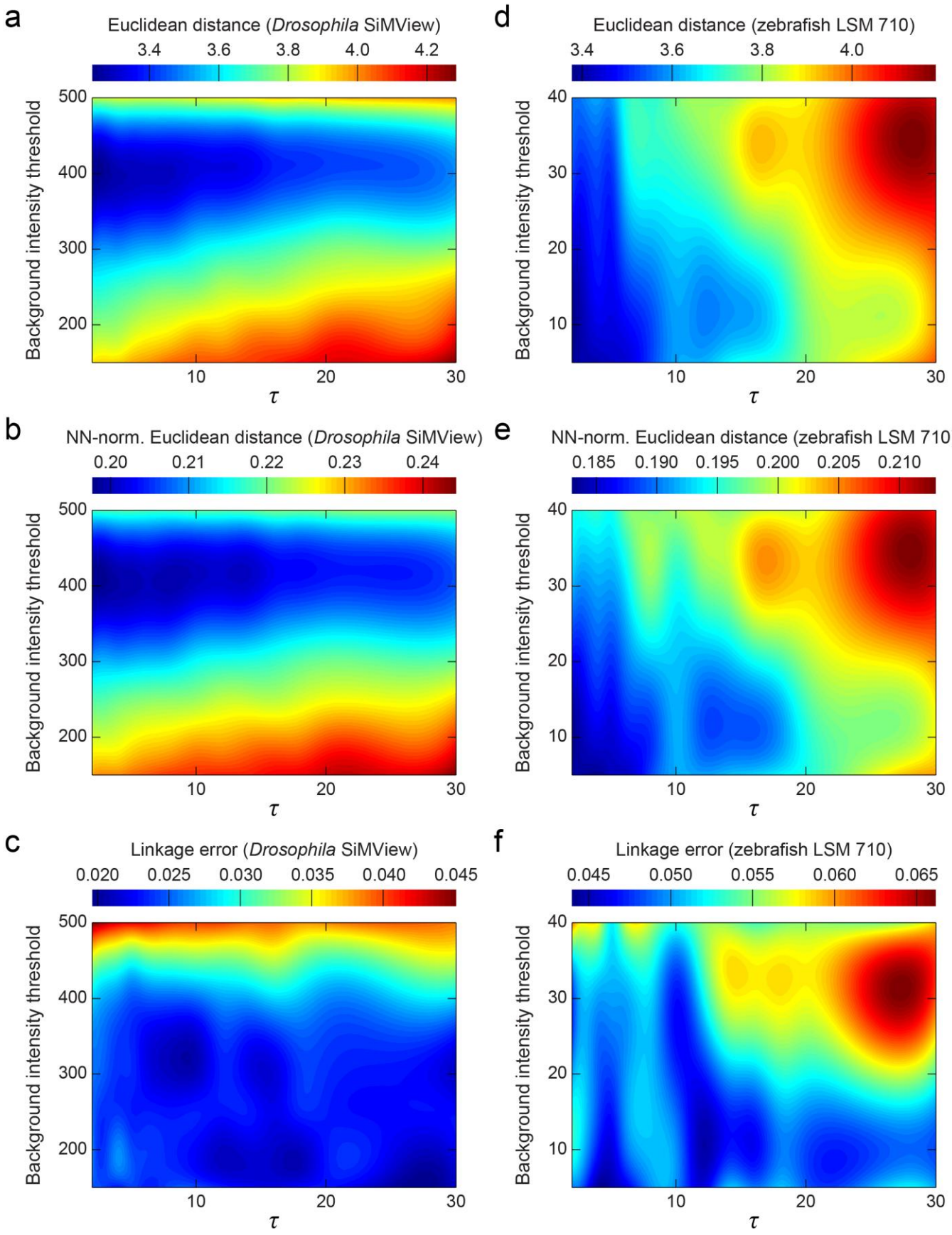
Scale bars, 5 µm (*xy* slices), 10 µm (*xz* slices).



Supplementary Figure 4

Precision-recall curve for background detection module

Precision-recall curve (blue line) for the background classifier trained on spatio-temporal features (**Supplementary Note 2**). The size of the training set was 43,500 samples and the size of the test set was 29,000 samples. All samples were annotated using the CATMAID interface discussed in section "*Visualization and manual curation of lineaging results*" in the main text. The machine learning classifier is based on the RUSBoost implementation in Matlab with 600 classification trees as weak classifiers. Each tree is grown and then pruned such that each leaf contains a minimum of 20 training samples. In order to avoid losing real nuclei, the threshold for detecting trajectories with background objects (red line) should be set to a value higher than 0.6.


a
Euclidean distance (*Drosophila* SiMView)
3.4 3.6 3.8 4.0 4.2
Background intensity threshold
500 400 300 200
10 20 30
τ
d
Euclidean distance (zebrafish LSM 710)
3.4 3.6 3.8 4.0
Background intensity threshold
40 30 20 10
10 20 30
τ
b
NN-norm. Euclidean distance (*Drosophila* SiMView)
0.20 0.21 0.22 0.23 0.24
Background intensity threshold
500 400 300 200
10 20 30
τ
e
NN-norm. Euclidean distance (zebrafish LSM 710)
0.185 0.190 0.195 0.200 0.205 0.210
Background intensity threshold
40 30 20 10
10 20 30
τ
c
Linkage error (*Drosophila* SiMView)
0.020 0.025 0.030 0.035 0.040 0.045
Background intensity threshold
500 400 300 200
10 20 30
τ
f
Linkage error (zebrafish LSM 710)
0.045 0.050 0.055 0.060 0.065
Background intensity threshold
40 30 20 10
10 20 30
τ

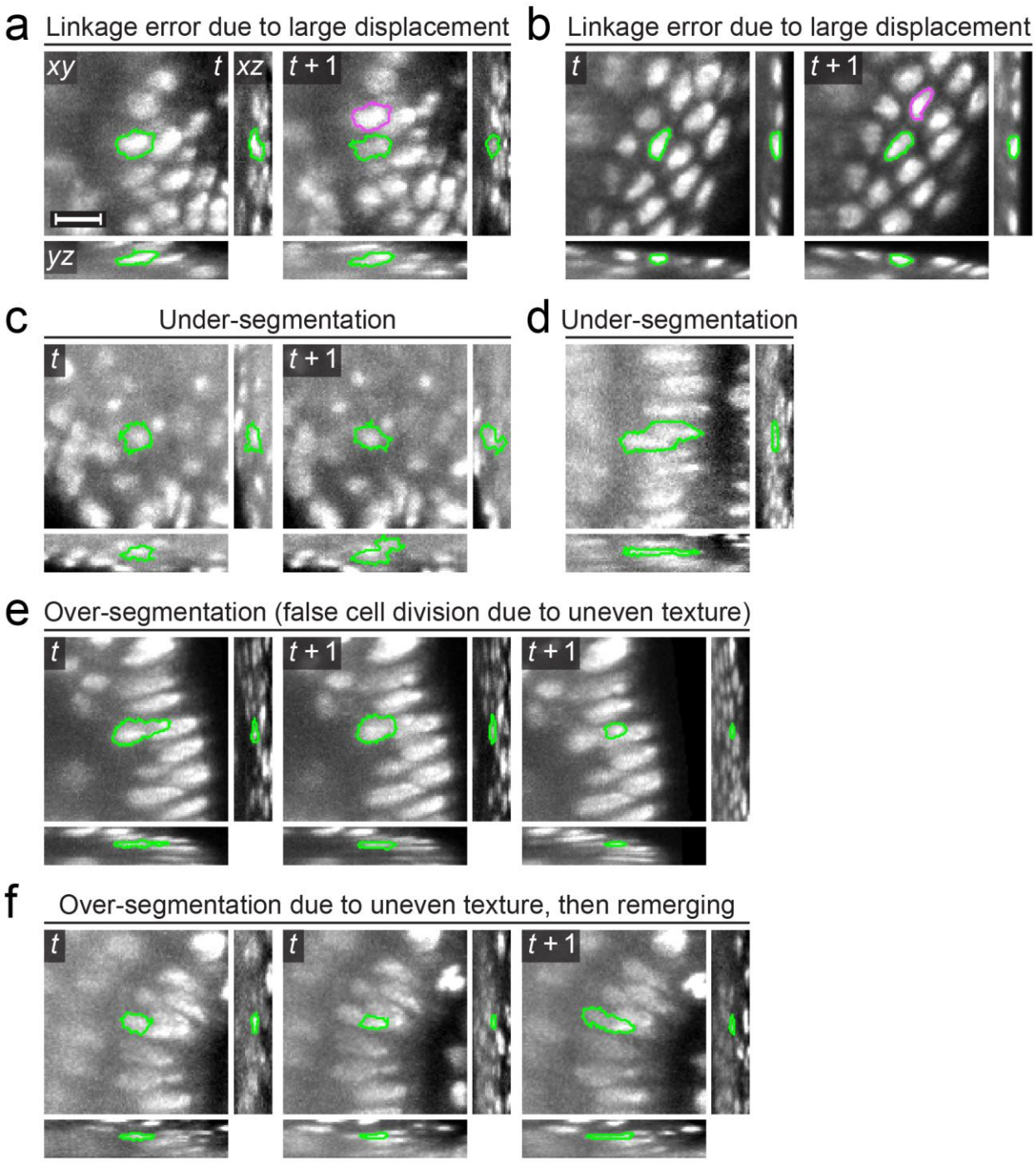
Supplementary Figure 5

Parameter sensitivity analysis

Parameter sensitivity analysis for the two tunable parameters of the processing pipeline, the image background intensity threshold and the PBC threshold τ for watershed agglomeration, with respect to Euclidean distance error metric (**a**,**d**), nearest neighbor (NN) normalized Euclidean distance error metric (**b**,**e**) and linkage error metric (**c**,**f**). Please see **Supplementary Note 3** for details on these metrics.

Panels (**a**-**c**) show results for the *Drosophila* SiMViev dataset. In this scenario, the image background intensity threshold has a larger impact on accuracy than τ, owing to the high signal-to-noise ratio (SNR) of the data set. However, sensitivity with respect to this threshold is low enough to ensure close-to-optimal results for a wide range of parameters values.

Panels (**d**-**f**) show results for the zebrafish confocal dataset. In this scenario, τ has a larger impact on accuracy due to the lower SNR. While the ranges of close-to optimal values for image background intensity are quite different in (**a**-**c**) and (**d**-**f**), the range of close-to-optimal values for τ is almost identical. This observation is true for all datasets investigated in this study. Thus, for new datasets, we generally recommend using τ values between 5 and 15.



Nature Methods: doi:10.1038/nmeth.3036

Supplementary Figure 6

Examples of segmentation and tracking errors

Orthogonal optical slices at different locations in the SiMView recording, each centered on a nucleus representing a different type of error in the automated segmentation and tracking pipeline. The result of the automated segmentation is indicated using green outlines.

(**a**,**b**) Linkage errors due to large displacements of cell nuclei between consecutive time points. Magenta outlines indicate the respective correct solutions.

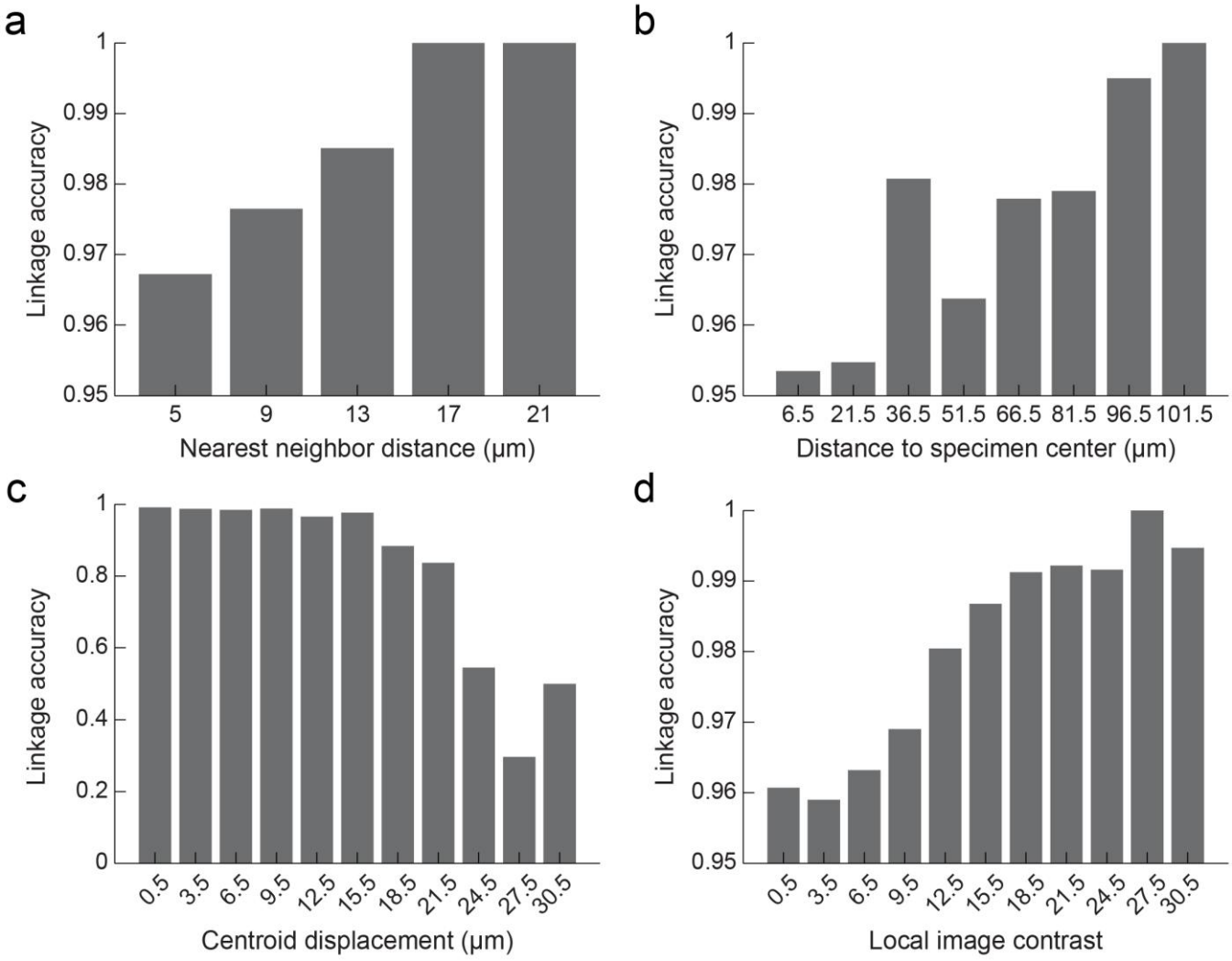
(**c**) Under-segmentation along the optical detection axis, owing to low image contrast.

(**d**) Under-segmentation within the image plane, owing to proximity of cell nuclei.

(**e**) Over-segmentation owing to uneven staining of the nuclear fluorescent label. The algorithm interprets the image content as a cell division.

(**f**) Over-segmentation owing to uneven staining of the nuclear fluorescent label, and subsequent error recovery. The algorithm recovers from an error similar to the one shown in panel (e) by ending one of the two tracks and propagating the other one in time with the correct segmentation solution. Thus, errors do not accumulate over time in the sequential propagation of the Gaussian mixture model.

Scale bar, 10 μm.



Supplementary Figure 7

Error analysis as a function of developmental stage, signal-to-noise ratio, cell density and imaging depth

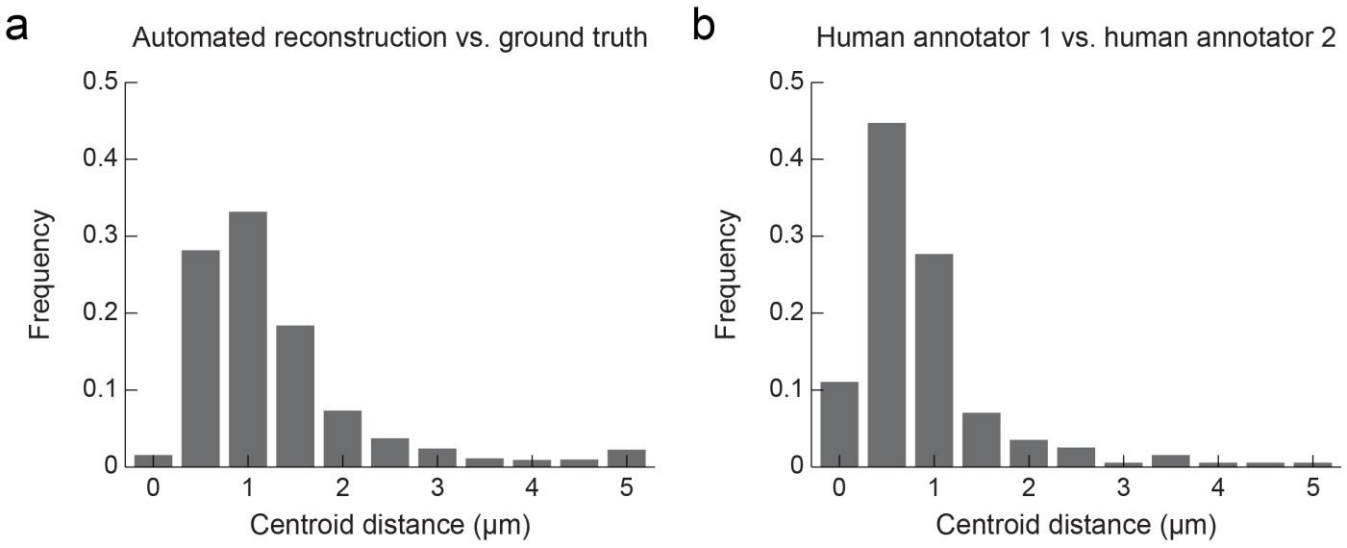
Histograms displaying how different factors affect lineage reconstruction accuracy of the automated segmentation and tracking method presented in this study. All histograms were extracted from *n* = 5,331 linkage annotations in the SiMView time-lapse recording of *Drosophila* embryogenesis. The ranges of the plots span nearly the full physical limits of parameters measured in this data set, which are as follows: the nuclei nearest neighbor distance ranges from 4 µm to 21 µm, the nuclei distance to the center of the embryo (with a diameter of ~200 µm) ranges from 0 to 106 µm, centroid displacements range from 0 µm to 44 µm between consecutive time points, and the 90th percentile of the local image contrast is 33.

(**a**) The larger the distance between adjacent nuclei, the higher the linkage accuracy.

(**b**) The shorter the optical detection path length inside the specimen, the higher the image quality and the linkage accuracy.

(**c**) The quality of the cell lineage reconstruction is independent of the nucleus displacement between two time points, as long as this displacement is not too large. This is the main assumption in our framework and for very large displacements the method breaks down (note that the scale of the vertical axis is very different from the other plots). Optical flow techniques can extend this range substantially and enable successful application of the automated segmentation and tracking framework also in the presence of larger displacements, if necessary. However, such an extension was not required for the data examples presented in this study.

(**d**) The higher the local image contrast (ratio of nucleus brightness versus background level), the higher the linkage accuracy.



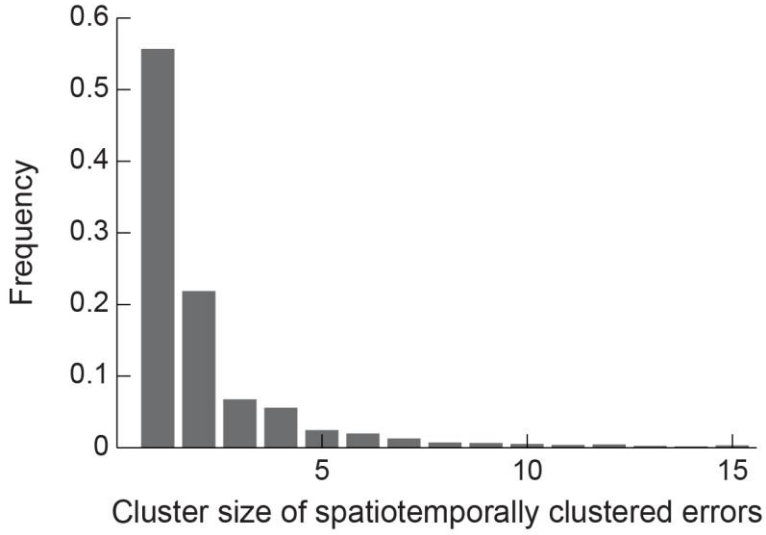
Supplementary Figure 8

Disagreement between annotators

Comparison of Euclidean distance accuracy between different annotators for time point 400 of the SiMView time-lapse recording of zebrafish embryonic development (**Supplementary Video 19**).

(**a**) Histogram of Euclidean distances between centroid locations annotated by human annotator 1 (reference) and those obtained automatically with our automated cell lineage reconstruction framework (**Supplementary Table 2**). The average centroid distance (i.e. accuracy of the automated framework) is 1.29 µm (n = 734).

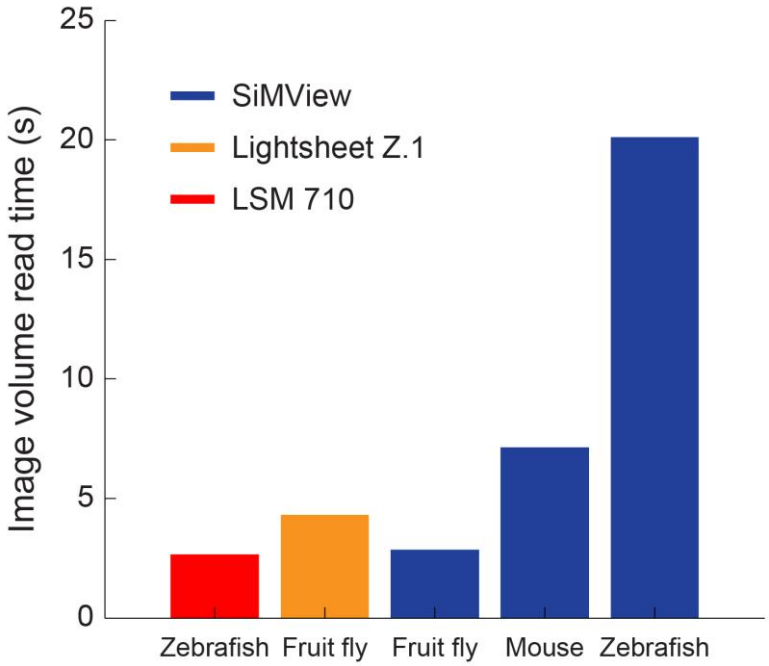
(**b**) Histogram of Euclidean distances between centroid locations annotated by human annotator 1 (reference) versus human annotator 2. Both annotators independently marked the centroid locations for the same set of cells, using the CATMAID interface and without knowledge of the other user's annotation. The average centroid distance (i.e. inter-user accuracy) is 0.89 µm (n = 200).



Supplementary Figure 9

Spatiotemporal clustering of tracking and segmentation errors

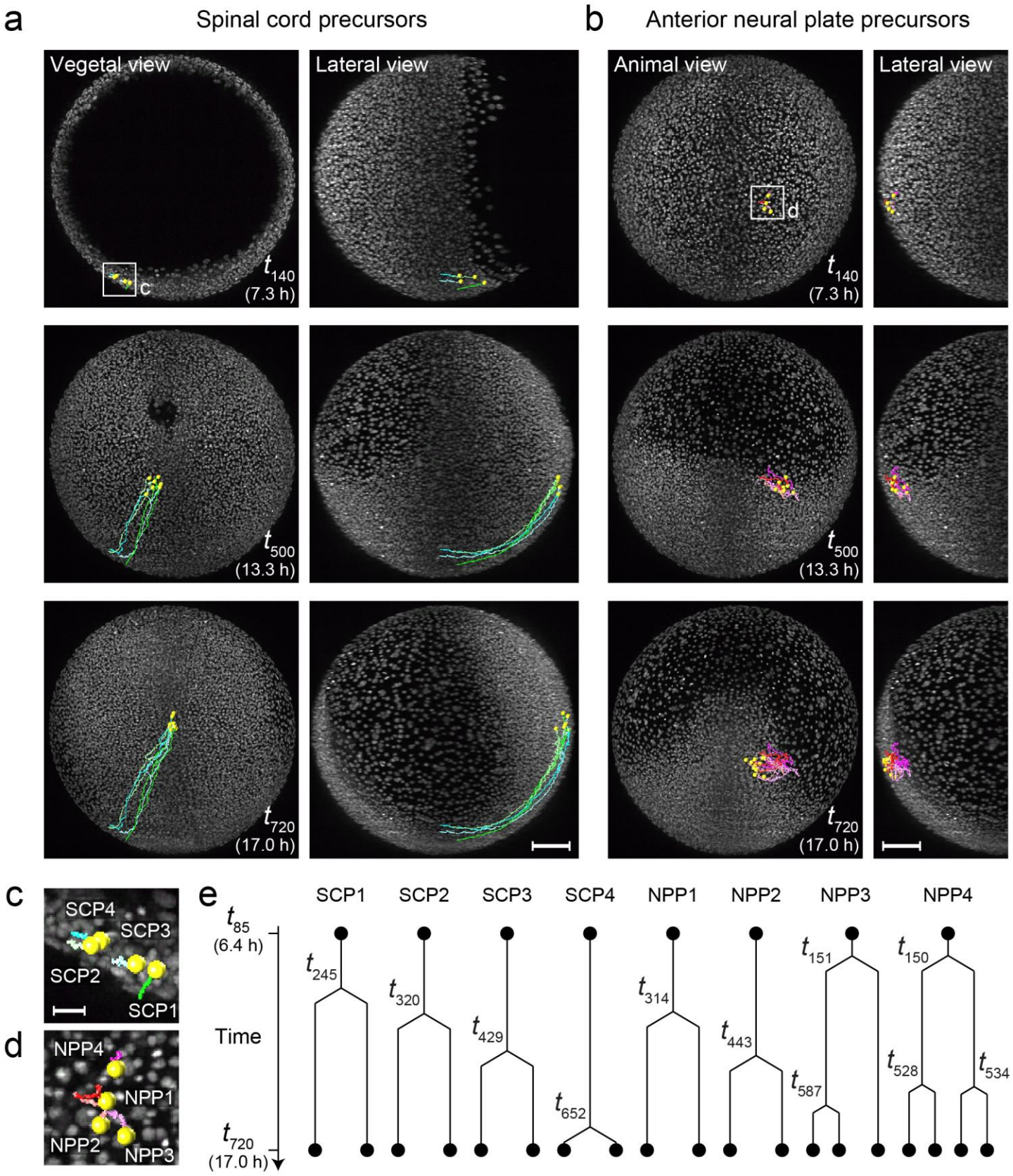
Analysis of co-localization in space and time of all errors (4,982) found in the automated reconstruction of cell lineages described in **Fig. 4**. To obtain this histogram we constructed a graph where each error defines one node. A node is connected by an edge to another node from the same time point if (and only if) this second node is one of the first node's four nearest neighbors. Two nodes belonging to different time points are connected by an edge if (and only if) they belong to the same lineage in the automatic reconstruction and are less than two time points apart. The figure shows the histogram of sizes of the different connected components in the resulting graph.



Supplementary Figure 10

Image volume read time overhead in cell lineage reconstructions

Time required to read a single three-dimensional image stack for each of the data sets discussed in the main text. For each time point of a time-lapse data set, the respective image stack is read twice: once for the initial hierarchical segmentation and once in the sequential GMM tracking module. All image data were stored using lossless compression in three-dimensional JPEG2000 format using libraries from the PICTools Medical software package. Read time increases quadratically with image size and was measured on a processing workstation equipped with two Intel Xeon E5-2687W CPUs, six Seagate Savvio 10K.5 ST9900805SS hard disks combined in a RAID-6 disk array and an Intel RMS25CB080 RAID module (**Online Methods**).



Nature Methods: doi:10.1038/nmeth.3036

Supplementary Figure 11

Cell lineage reconstructions in the early zebrafish embryo

Proof-of-principle automated reconstruction and manual curation of cell lineages in the early zebrafish embryo. The underlying automated reconstructions are based on the data set shown in **Supplementary Video 19** and are visualized in **Supplementary Videos 20** and **21**. The entire data set comprises 10.7 million data points (with one data point corresponding to the positions and dimensions of a cell nucleus at one time point). Manual data curation of the automatically reconstructed cell tracks was performed at a rate of 1,019 data points per hour.

(**a**) Spinal cord cells were manually identified in the automated cell lineage reconstruction, based on their spatial location at time point 720 in the zebrafish recording shown in **Supplementary Videos 19-21**. Using the data curation and annotation interface provided as **Supplementary Software 2**, the tracks of these cells were then curated backwards in time for the time interval 85-720 (10.6 hours of live imaging data, recorded at 21.5°C). When encountering a cell division, the respective other daughter cell was followed forwards in time and its track was fully curated as well. The panels in (a) visualize the resulting cell lineage reconstruction at three different time points, with yellow spheres indicating cell positions at the given time point and colored lines indicating the tracking information up to the respective time point.

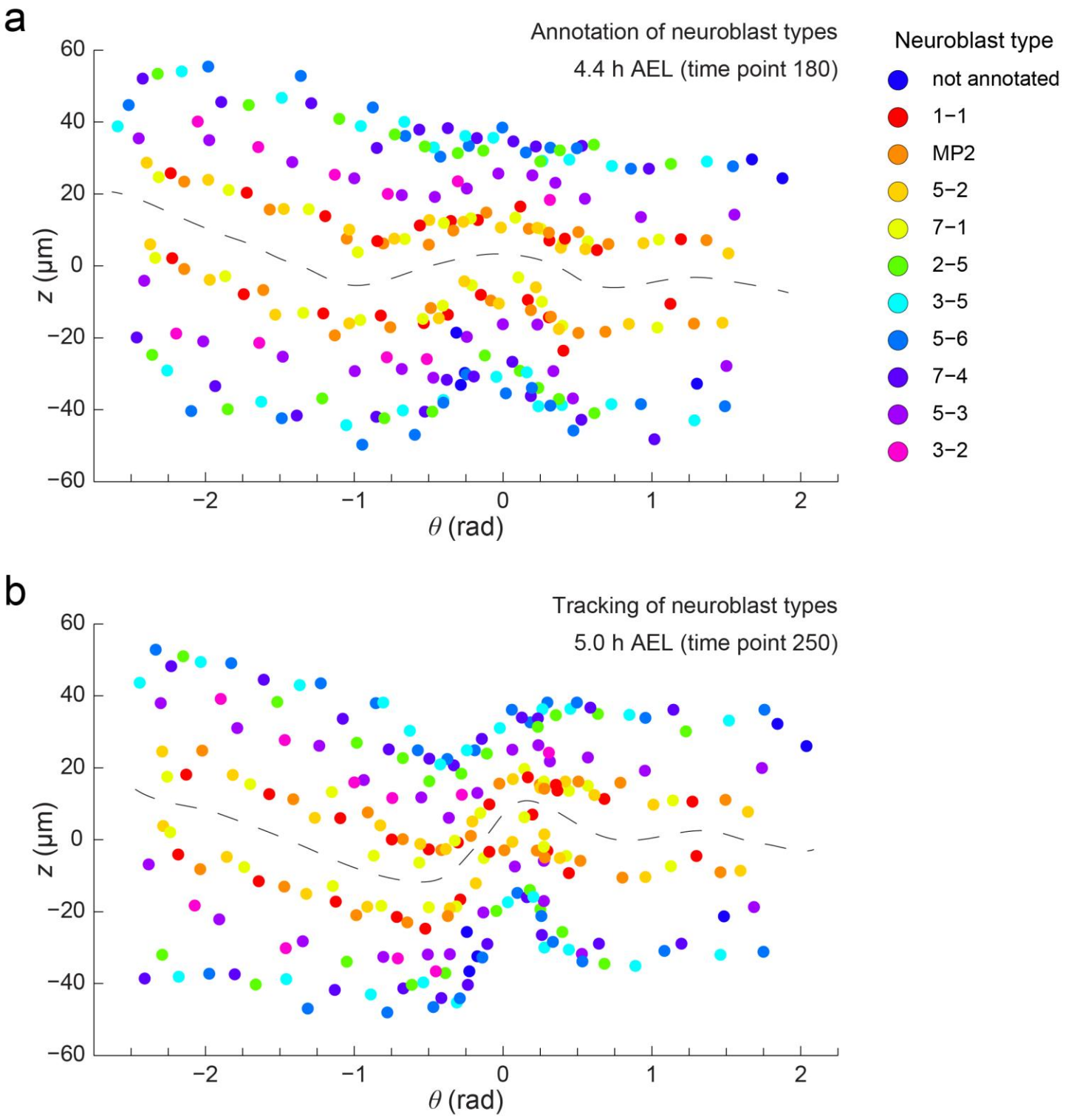
(**b**) As in (a), but for a set of cells located in the anterior neural plate.

(**c**,**d**) Enlarged view of the boxed regions shown in (a) and (b) at time point 140, with annotation labels for the four spinal cord precursors (c) and four anterior neural plate precursors (d).

(**e**) Cell lineage tree representation of the cell lineage reconstructions visualized in (a) an (b). In addition to the start and end points of the reconstructed time interval, these trees include annotations of cell division time points. Each precursor divided at least once within the 10.6-hour imaging interval.

SCP = Spinal cord precursor, NPP = Anterior neural plate precursor.

Scale bars, 100 µm (a,b), 20 µm (c).



Nature Methods: doi:10.1038/nmeth.3036

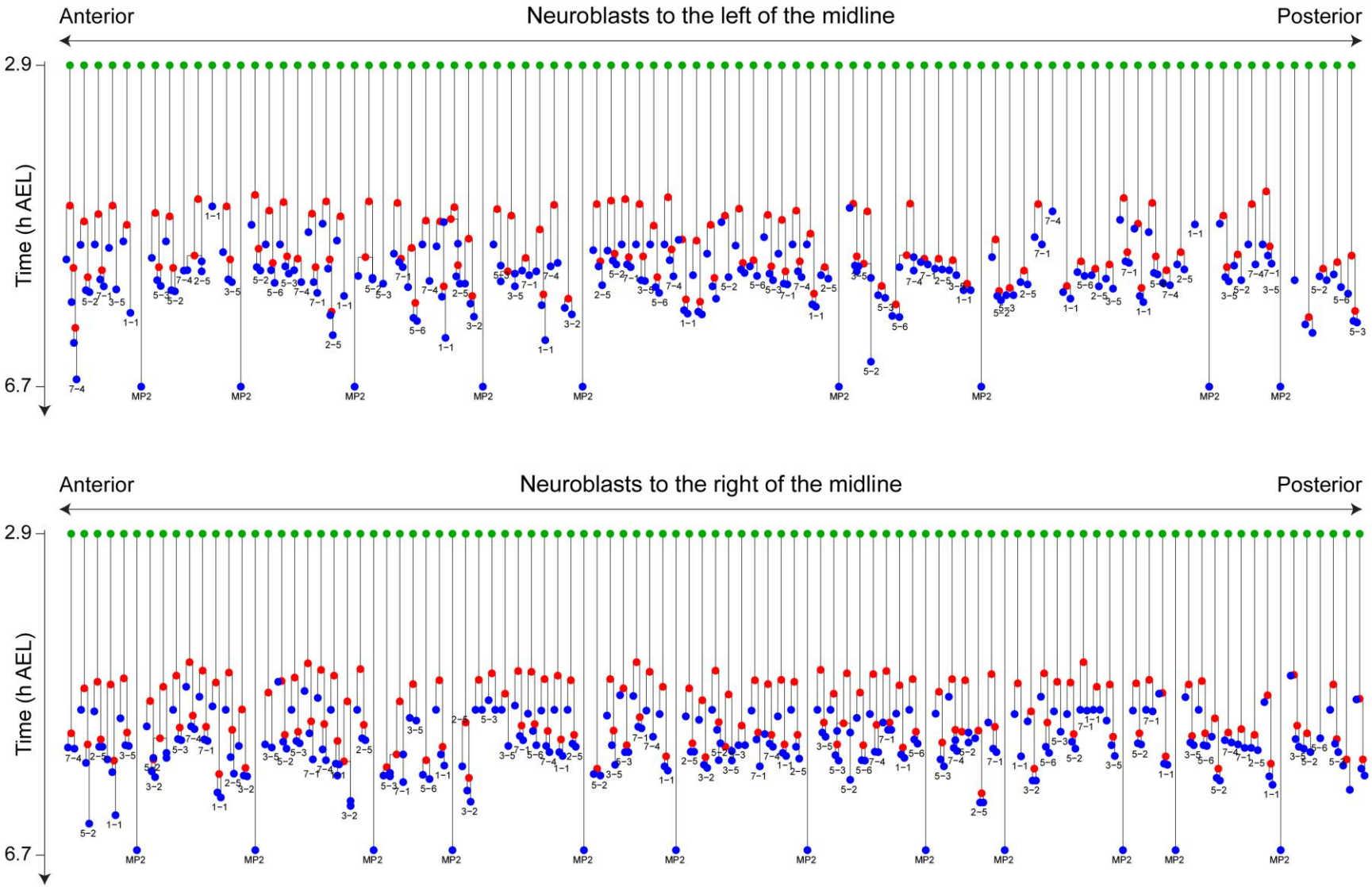
Supplementary Figure 12

Spatial maps of annotated neuroblasts

(**a**) Spatial map of the neuroblast array from our reconstruction of early *Drosophila* embryonic nervous system development at 4.4 h AEL, shortly after neuroblast internalization, using a color code for the manual neuroblast type annotation. Neuroblast coordinates in this map correspond to those shown in **Fig. 5e** and **Supplementary Fig. 15**. The manual annotation of neuroblast types was performed on the basis of relative positional information within the stereotypic neuroblast array, as previously described.

(**b**) Spatial map of the neuroblast array at 5.0 h AEL, approximately half an hour after neuroblast internalization. The color code was propagated from neuroblasts in panel (a) to the corresponding neuroblasts at this later time point, using the tracking information from the curated cell lineage reconstruction shown in **Supplementary Fig. 13**. Neighbor relationships are largely preserved over this time interval.

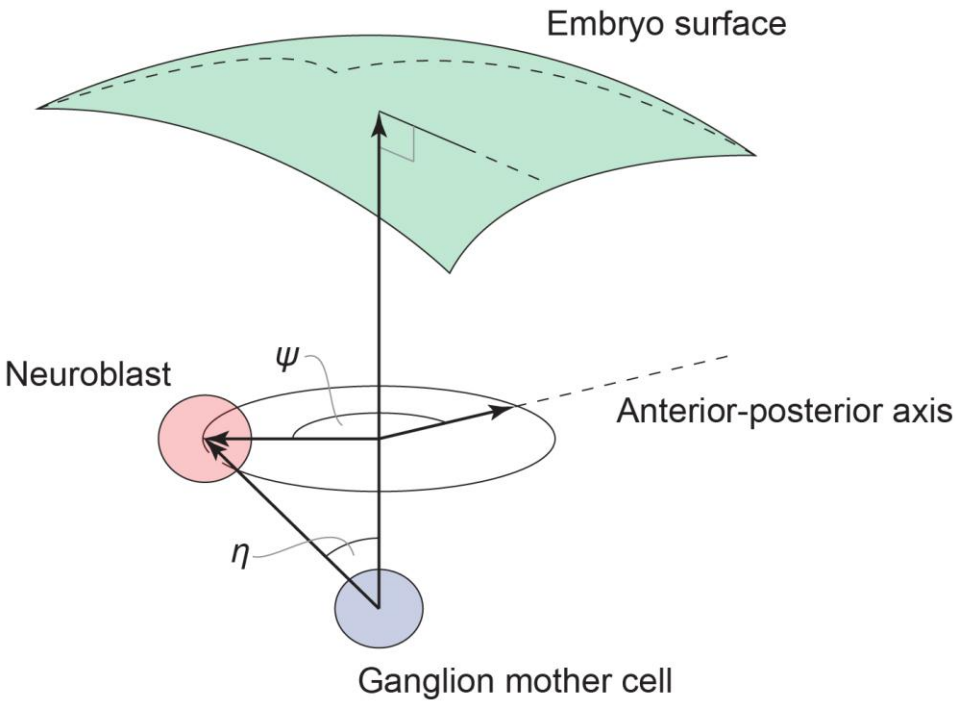
Nature Methods: doi:10.1038/nmeth.3036



Supplementary Figure 13

Cell lineage reconstruction of the early *Drosophila* embryonic nervous system

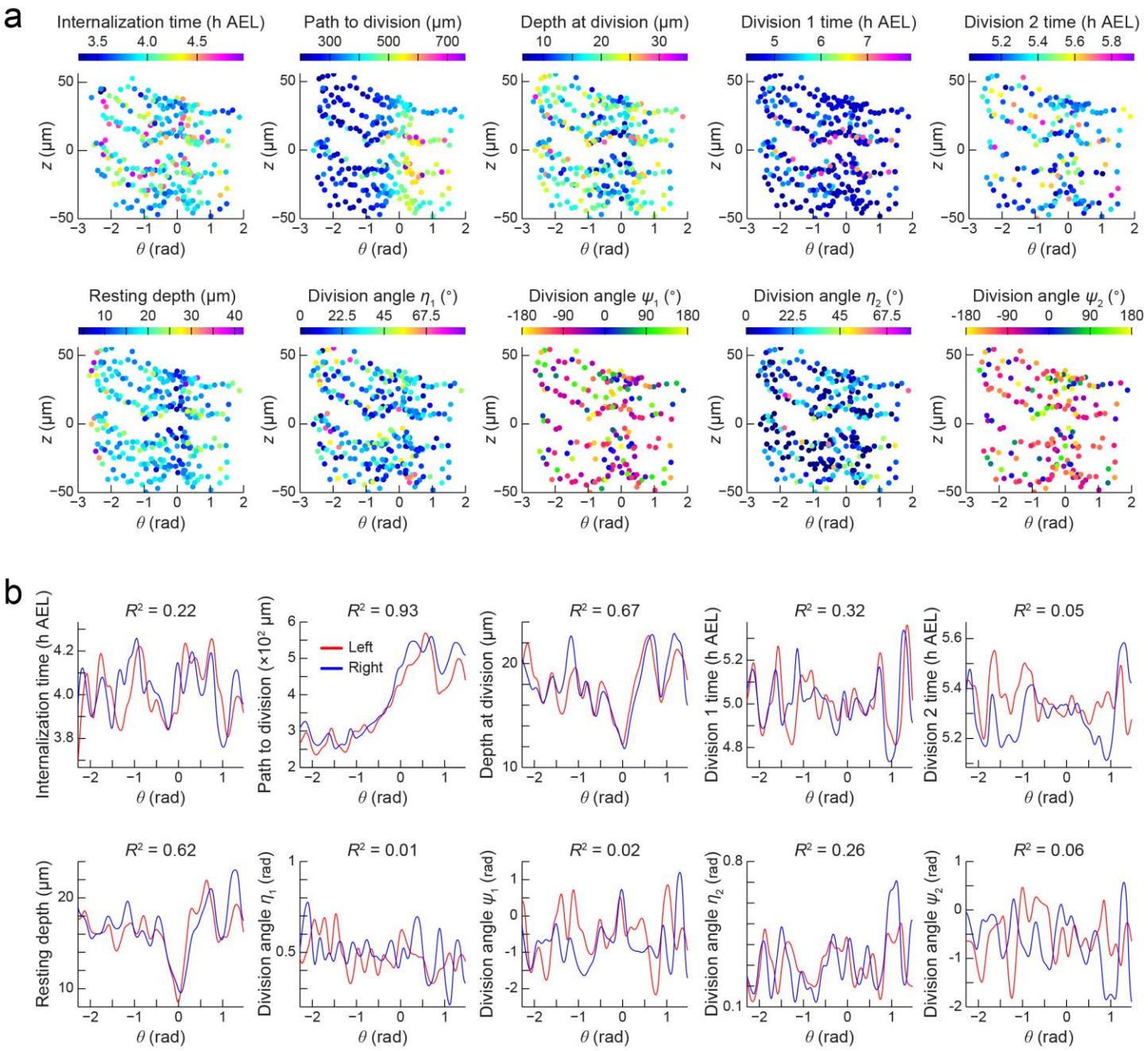
Cell lineage tree for all neuroblasts tracked in our reconstruction of early *Drosophila* embryonic nervous system development (**Fig. 5**, **Supplementary Videos 24-28**). Green circles indicate lineage origins in the blastoderm, red circles indicate cell division events and blue circles indicate the end time point of cell tracks for the respective neuroblasts or ganglion mother cells. All neuroblasts identified by manual inspection are annotated at the end of the respective lineage branch.



Supplementary Figure 14

Neuroblast division angles

Local coordinate system used to analyze neuroblast division angles (ψ, η). The angles ψ, η define a spherical coordinate system. The origin corresponds to the center of the ganglion mother cell. The north pole is defined by the normal vector to the surface of the embryo at the point closest to the ganglion mother cell. The division angle η defines the deviation of the surface normal from the vector connecting ganglion mother cell and neuroblast. The division angle ψ defines the orientation of the division axis relative to the anterior-posterior and medio-lateral axes of the embryo. $\psi = 0°$ corresponds to the direction along the germ band facing the anterior end of the embryo, and $\psi = 270°$ corresponds to a medial direction.


a
Internalization time (h AEL)
3.5 4.0 4.5
Path to division (μm)
300 500 700
Depth at division (μm)
10 20 30
Division 1 time (h AEL)
5 6 7
Division 2 time (h AEL)
5.2 5.4 5.6 5.8
Resting depth (μm)
10 20 30 40
Division angle η_1 (°)
0 22.5 45 67.5
Division angle ψ_1 (°)
-180 -90 0 90 180
Division angle η_2 (°)
0 22.5 45 67.5
Division angle ψ_2 (°)
-180 -90 0 90 180
z (μm)
θ (rad)
b
R^2 = 0.22
R^2 = 0.93
R^2 = 0.67
R^2 = 0.32
R^2 = 0.05
R^2 = 0.62
R^2 = 0.01
R^2 = 0.02
R^2 = 0.26
R^2 = 0.06
Left
Right
Internalization time (h AEL)
Path to division (×10² μm)
Depth at division (μm)
Division 1 time (h AEL)
Division 2 time (h AEL)
Resting depth (μm)
Division angle η_1 (rad)
Division angle ψ_1 (rad)
Division angle η_2 (rad)
Division angle ψ_2 (rad)
θ (rad)

Supplementary Figure 15

Morphodynamic measurements using neuroblast trajectories

(**a**) Overview of features of dynamic cell behavior measured for all neural precursors analyzed in the reconstruction of early *Drosophila* embryonic nervous system development (**Fig. 5**, **Supplementary Videos 24-28**). This overview figure includes the four features shown in the main text (**Fig. 5e**). Each feature is represented in the neuroblast array at 4.4 h AEL using an individual color code. These features were used for the prediction of neuroblast cell types, shown in **Fig. 6e,f**.

(**b**) Overview of bilateral symmetry analysis for the features of dynamic cell behavior measured for all neural precursors analyzed in the reconstruction of early *Drosophila* embryonic nervous system development. This overview figure includes the plot shown in the main text (**Fig. 6b**). Neuroblasts to the left of the midline are represented by red lines, neuroblasts to the right of the midline by blue lines. Continuity of the plots was achieved by using a Gaussian-weighted average along the midline to convert the discrete neuroblast data points to a continuous graph (σ = 0.1 rad). The level of feature correlation between the left and right halves of the neuroblast array is indicated in the form of an R^2 score above each plot.

Supplementary Table 1 | Cell division detection accuracy of the cell lineaging framework

Microscope	Model organism	Time point	Segmented objects[1]	**Estimated number of cell divisions**	**True positives[2,3]**	**False positives (correcting under-segmentation)[2]**	**False positives (background objects)[2]**	**False positives (other)[2,4]**
SiMView	Fruit fly embryo	30 (3.2 h AEL)	5,664	18	0	11	4	3
		80 (3.6 h AEL)	4,597	37	15	15	5	2
		130 (4.0 h AEL)	5,593	25	5	15	1	4
		180 (4.4 h AEL)	5,882	39	9	23	6	1
		230 (4.8 h AEL)	6,799	53	16	24	8	5
		280 (5.2 h AEL)	7,302	62	9	33	11	9
	Zebrafish embryo	100 (6.7 hpf)	9,519	20	12	6	0	2
		200 (8.3 hpf)	11,823	16	8	7	0	1
		300 (10.0 hpf)	13,765	26	15	10	0	1
		400 (11.7 hpf)	16,086	45	20	21	0	4
		500 (13.3 hpf)	18,255	46	15	31	0	0
		600 (15.0 hpf)	19,499	53	15	34	0	4
		700 (16.7 hpf)	19,397	53	9	29	0	15
LSM 710		30 (6.0 hpf)	2,982	22	6	4	0	12
		60 (7.0 hpf)	3,374	32	10	12	0	10
		90 (8.0 hpf)	3,347	22	5	10	0	7

[1] This parameter also serves as an estimate of the number of cells imaged by the microscope at the respective time point.

[2] These parameters and metrics are defined in **Supplementary Note 3**.

[3] We also manually annotated all cell divisions in the time interval [100 101] of the SiMView zebrafish recording to estimate false negatives. Out of 22 user-annotated divisions, 17 were correctly identified by the cell lineaging framework (77%). 4 out of the 5 false negative cases corresponded to divisions that were correctly identified by the GMM but then discarded by the spatio-temporal context rules, as a result of parameter settings required for striking a good balance between false negatives and false positives.

[4] This category generally refers to over-segmentation as a result of heterogeneous texture of the nucleus.

Supplementary Table 2 | Performance overview of the cell lineaging framework

Microscope	Model organism	Time point	Segmented objects[1]	Ground truth samples	Processing time (s)	**Linkage accuracy (%)[2]**	**Euclidian distance (µm)[2]**	**Normalized Euclidian distance (%)[2]**
SiMView	Fruit fly embryo	30 (3.2 h AEL)	5,664	1,211	25.2	$97.7^{+0.9}_{-1.5}$	$1.51^{+0.05}_{-0.05}$	$23.9^{+0.8}_{-0.8}$
		80 (3.6 h AEL)	4,597	1,718	21.2	$95.4^{+1.2}_{-1.2}$	$1.44^{+0.08}_{-0.08}$	$18.6^{+1.0}_{-0.9}$
		130 (4.0 h AEL)	5,593	2,408	20.2	$98.0^{+0.7}_{-1.0}$	$1.50^{+0.06}_{-0.05}$	$20.8^{+0.7}_{-0.7}$
		180 (4.4 h AEL)	5,882	2,008	21.8	$97.2^{+0.9}_{-1.2}$	$1.23^{+0.05}_{-0.05}$	$18.8^{+0.8}_{-0.7}$
		230 (4.8 h AEL)	6,799	2,041	23.0	$98.2^{+0.7}_{-1.0}$	$1.42^{+0.05}_{-0.05}$	$22.5^{+0.8}_{-0.8}$
		280 (5.2 h AEL)	7,302	1,436	22.9	$97.1^{+1.1}_{-1.6}$	$1.45^{+0.07}_{-0.07}$	$21.9^{+1.0}_{-0.9}$
	Zebrafish embryo	100 (6.7 hpf)	9,519	1,375	29.7	$99.3^{+0.4}_{-1.0}$	$1.53^{+0.05}_{-0.05}$	$11.3^{+0.4}_{-0.3}$
		200 (8.3 hpf)	11,823	2,655	32.7	$99.1^{+0.4}_{-0.7}$	$1.92^{+0.19}_{-0.07}$	$14.0^{+0.5}_{-0.4}$
		300 (10.0 hpf)	13,765	2,416	36.5	$99.2^{+0.4}_{-0.7}$	$1.38^{+0.05}_{-0.04}$	$12.2^{+0.5}_{-0.4}$
		400 (11.7 hpf)	16,086	1,469	49.1	$98.9^{+0.5}_{-1.0}$	$1.16^{+0.05}_{-0.05}$	$10.6^{+0.5}_{-0.5}$
		500 (13.3 hpf)	18,255	1,210	45.9	$98.3^{+0.8}_{-1.4}$	$1.27^{+0.08}_{-0.07}$	$12.0^{+0.8}_{-0.7}$
		600 (15.0 hpf)	19,499	1,369	42.8	$99.4^{+0.4}_{-0.9}$	$1.21^{+0.07}_{-0.06}$	$11.2^{+0.7}_{-0.6}$
		700 (16.7 hpf)	19,397	1,246	46.9	$98.4^{+0.7}_{-1.3}$	$1.57^{+0.08}_{-0.07}$	$14.4^{+0.8}_{-0.7}$
	Mouse embryo	0 (t_0 = E6.25)	886	715	9.8	$91.3^{+2.5}_{-3.4}$	$3.38^{+0.18}_{-0.17}$	$33.1^{+1.9}_{-1.7}$
		12 (t_0 + 1.0 h)	959	740	9.0	$88.9^{+2.8}_{-3.6}$	$3.41^{+0.20}_{-0.18}$	$32.7^{+1.8}_{-1.7}$
		24 (t_0 + 2.0 h)	1093	841	9.3	$89.8^{+2.5}_{-3.3}$	$3.21^{+0.17}_{-0.16}$	$31.3^{+1.7}_{-1.6}$
Lightsheet Z.1	Fruit fly embryo	90 (3.0 h AEL)	3,784	1,330	11.2	$96.8^{+1.1}_{-1.6}$	$1.68^{+0.06}_{-0.05}$	$28.4^{+1.0}_{-0.9}$
		140 (3.4 h AEL)	2,911	1,365	8.6	$96.6^{+1.1}_{-1.7}$	$1.50^{+0.07}_{-0.06}$	$22.1^{+0.9}_{-0.9}$
		190 (3.8 h AEL)	3,499	1,744	9.5	$94.7^{+1.3}_{-1.7}$	$1.51^{+0.07}_{-0.06}$	$24.1^{+1.0}_{-0.9}$
		240 (4.3 h AEL)	4,308	1,998	12.9	$96.2^{+1.0}_{-1.4}$	$1.50^{+0.06}_{-0.05}$	$25.8^{+0.9}_{-0.9}$
		290 (4.7 h AEL)	4,800	1,327	12.2	$96.5^{+1.1}_{-1.7}$	$1.53^{+0.11}_{-0.09}$	$23.8^{+1.2}_{-1.1}$
		340 (5.1 h AEL)	5,200	1,594	14.1	$98.0^{+0.8}_{-1.2}$	$1.45^{+0.12}_{-0.07}$	$23.2^{+1.0}_{-1.0}$
LSM 710	Zebrafish embryo	30 (6.0 hpf)	2,982	1,206	5.6	$96.8^{+1.1}_{-1.7}$	$1.98^{+0.12}_{-0.11}$	$16.3^{+1.0}_{-0.9}$
		60 (7.0 hpf)	3,374	1,219	5.2	$95.2^{+1.4}_{-2.0}$	$2.00^{+0.12}_{-0.10}$	$18.4^{+1.0}_{-0.9}$
		90 (8.0 hpf)	3,347	1,202	3.9	$94.2^{+1.6}_{-2.2}$	$2.33^{+0.15}_{-0.13}$	$21.8^{+1.2}_{-1.1}$

[1] This parameter also serves as an estimate of the number of cells imaged by the microscope at the respective time point.

[2] These metrics are defined in **Supplementary Note 3**. 95th and 5th percentile confidence intervals are provided as sub-/superscripts.

Supplementary Table 3 | Segmentation accuracy of the cell lineaging framework

Microscope	Model organism	Time point	Segmented objects[1]	Ground truth samples	Max. nucleus radius (μm)[2]	**True positives[2]**	**Over-segmented nuclei[2]**	**False detections[2]**	**False negatives[2]**
SiMView	Zebrafish embryo	100 (6.7 hpf)	9,519	685	4.325	683	0	0	2
		101 (6.7 hpf)	9,560	690	4.331	690	0	0	2
		200 (8.3 hpf)	11,823	1,332	4.016	1,332	1	0	26
		201 (8.3 hpf)	11,857	1,323	4.016	1,323	1	0	24
		300 (10.0 hpf)	13,765	1,207	3.890	1,207	2	0	21
		301 (10.0 hpf)	13,811	1,209	3.883	1,209	3	0	16
		400 (11.7 hpf)	16,086	734	3.801	734	1	0	5
		401 (11.7 hpf)	16,125	735	3.808	735	1	0	6
		500 (13.3 hpf)	18,255	605	3.780	605	0	0	13
		501 (13.3 hpf)	18,307	605	3.780	605	1	0	13
		600 (15.0 hpf)	19,499	683	3.815	683	0	0	10
		601 (15.0 hpf)	19,475	686	3.822	686	0	0	11
		700 (16.7 hpf)	19,397	623	3.883	623	1	0	21
		701 (16.7 hpf)	19,361	623	3.890	623	1	0	17
LSM 710		30 (6.0 hpf)	2,982	603	4.340	603	0	0	32
		31 (6.0 hpf)	2,980	603	4.327	603	1	0	24
		60 (7.0 hpf)	3,374	612	4.184	612	1	0	37
		61 (7.0 hpf)	3,378	607	4.184	607	0	0	38
		90 (8.0 hpf)	3,347	601	4.104	601	1	0	42
		91 (8.0 hpf)	3,327	601	4.094	601	4	0	50

[1] This parameter also serves as an estimate of the number of cells imaged by the microscope at the respective time point.

[2] These metrics are defined in **Supplementary Note 3**.

Supplementary Table 4 | Comparative performance of cell lineaging methods for SiMView images of the *Drosophila* blastoderm

Method	Automatic seed at t_0?[1]	Method detects cell divisions?	Time point	Segmented objects	Ground truth samples	Processing time (s)	**Linkage accuracy (%)[2]**	**Euclidian distance (µm) [2]**	**Normalized Euclidian distance (%)[2]**
This study	Yes	Yes	90 (2.0 h AEL)	2,877	1,241	19.5	$94.4^{+1.6}_{-2.1}$	$1.32^{+0.11}_{-0.09}$	$16.5^{+1.3}_{-1.1}$
			120 (2.2 h AEL)	3,916	1,435	14.4	$84.2^{+2.6}_{-3.0}$	$4.12^{+0.14}_{-0.13}$	$50.7^{+1.3}_{-1.2}$
			150 (2.4 h AEL)	5,899	1,224	15.8	$96.5^{+1.2}_{-1.8}$	$0.86^{+0.05}_{-0.04}$	$16.0^{+0.8}_{-0.7}$
			180 (2.6 h AEL)	6,627	1,202	13.9	$95.2^{+1.4}_{-2.0}$	$1.29^{+0.06}_{-0.05}$	$23.2^{+0.9}_{-0.9}$
Kausler *et al*. 2012 (CGT)[4]	Yes	Yes	90 (2.0 h AEL)	2,824	1,241	282.8	$84.7^{+2.6}_{-3.0}$	$1.56^{+0.08}_{-0.07}$	$21.7^{+1.2}_{-1.0}$
			120 (2.2 h AEL)	3,516	1,435	292.1	$73.5^{+3.2}_{-3.5}$	$4.48^{+0.14}_{-0.14}$	$57.2^{+1.4}_{-1.4}$
			150 (2.4 h AEL)	3,516	1,224	292.1	$77.3^{+3.1}_{-3.5}$	$3.47^{+0.11}_{-0.10}$	$48.3^{+1.4}_{-1.4}$
			180 (2.6 h AEL)	7,202	1,202	328.5	$73.7^{+3.4}_{-3.7}$	$2.44^{+0.13}_{-0.11}$	$45.9^{+1.6}_{-1.6}$
Giurumescu *et al*. 2012 (NT4D)[5]	No	No	90 (2.0 h AEL)	2,373	1,241	2.9	$85.3^{+2.6}_{-3.0}$	$4.15^{+0.15}_{-0.14}$	$43.9^{+1.5}_{-1.5}$
			120 (2.2 h AEL)	3,932	1,435	8.1	$83.5^{+2.7}_{-3.0}$	$4.29^{+0.14}_{-0.13}$	$54.2^{+1.3}_{-1.2}$
			150 (2.4 h AEL)	3,858	1,224	7.8	$89.4^{+2.2}_{-2.7}$	$3.13^{+0.11}_{-0.11}$	$49.1^{+1.7}_{-1.6}$
			180 (2.6 h AEL)	6,871	1,202	21.8	$93.8^{+1.7}_{-2.2}$	$1.75^{+0.05}_{-0.05}$	$34.6^{+1.1}_{-1.0}$
Tomer *et al*. 2012 (NM12)[6]	Yes	Yes	90 (2.0 h AEL)	2,755	1,241	18.4	$95.6^{+0.7}_{-1.3}$	$1.00^{+0.06}_{-0.05}$	$13.0^{+0.7}_{-0.6}$
			120 (2.2 h AEL)	3,596	1,435	19.3	$85.7^{+2.5}_{-2.9}$	$4.24^{+0.14}_{-0.13}$	$52.3^{+1.3}_{-1.3}$
			150 (2.4 h AEL)	5,635	1,224	20.6	$97.1^{+1.1}_{-1.6}$	$1.03^{+0.08}_{-0.06}$	$18.6^{+0.9}_{-0.8}$
			180 (2.6 h AEL)	7,078	1,202	29.2	$94.8^{+1.5}_{-2.1}$	$1.12^{+0.04}_{-0.04}$	$20.7^{+0.7}_{-0.7}$

[1] If "No", manual segmentation is required for the first time point (t_0) to initialize the algorithm.

[2] These metrics are defined in **Supplementary Note 3**. 95th and 5th percentile confidence intervals are provided as sub-/superscripts.

Supplementary Table 5 | Comparative performance of cell lineaging methods for SiMView images of *Drosophila* germ band extension

Method	Automatic seed at t_0?[1]	Method detects cell divisions?	Time point	Segmented objects	Ground truth samples	Processing time (s)	**Linkage accuracy (%)[2]**	**Euclidian distance (µm)[2]**	**Normalized Euclidian distance (%)[2]**
This study	Yes	Yes	30 (3.2 h AEL)	5,664	1,211	25.2	$97.7^{+0.9}_{-1.5}$	$1.51^{+0.05}_{-0.05}$	$23.9^{+0.8}_{-0.8}$
			80 (3.6 h AEL)	4,597	1,718	21.2	$95.4^{+1.2}_{-1.6}$	$1.44^{+0.08}_{-0.08}$	$18.6^{+1.0}_{-0.9}$
			130 (4.0 h AEL)	5,593	2,408	20.2	$98.0^{+0.7}_{-1.0}$	$1.50^{+0.06}_{-0.05}$	$20.8^{+0.7}_{-0.7}$
			180 (4.4 h AEL)	5,882	2,008	21.8	$97.2^{+0.9}_{-1.2}$	$1.23^{+0.05}_{-0.05}$	$18.8^{+0.8}_{-0.7}$
			230 (4.8 h AEL)	6,799	2,041	23.0	$98.2^{+0.7}_{-1.0}$	$1.42^{+0.05}_{-0.05}$	$22.5^{+0.8}_{-0.8}$
			280 (5.2 h AEL)	7,302	1,436	23.0	$97.1^{+1.1}_{-1.6}$	$1.45^{+0.07}_{-0.07}$	$21.9^{+1.0}_{-0.9}$
Kausler *et al*. 2012 (CGT)	Yes	Yes	30 (3.2 h AEL)	7,334	1,211	489.8	$67.5^{+3.6}_{-3.8}$	$2.29^{+0.09}_{-0.08}$	$41.0^{+1.4}_{-1.3}$
			80 (3.6 h AEL)	6,868	1,718	485.5	$51.0^{+3.3}_{-3.3}$	$2.16^{+0.12}_{-0.10}$	$41.3^{+1.3}_{-1.3}$
			130 (4.0 h AEL)	7,092	2,408	486.9	$54.3^{+2.8}_{-2.8}$	$2.44^{+0.11}_{-0.10}$	$44.4^{+1.1}_{-1.1}$
			180 (4.4 h AEL)	7,436	2,008	490.5	$55.2^{+3.0}_{-3.1}$	$2.07^{+0.08}_{-0.07}$	$43.7^{+1.2}_{-1.2}$
			230 (4.8 h AEL)	7,436	2,041	499.0	$44.1^{+3.1}_{-3.0}$	$4.29^{+0.11}_{-0.10}$	$76.3^{+1.1}_{-1.1}$
			280 (5.2 h AEL)	7,277	1,436	484.8	$54.0^{+3.9}_{-3.9}$	$2.67^{+0.16}_{-0.15}$	$46.4^{+1.6}_{-1.5}$
Giurumescu *et al*. 2012 (NT4D)	No	No	30 (3.2 h AEL)	5,814	1,211	19.6	$86.0^{+2.5}_{-3.0}$	$3.77^{+0.11}_{-0.11}$	$61.4^{+1.6}_{-1.5}$
			80 (3.6 h AEL)	4,561	1,718	20.8	$79.6^{+2.6}_{-2.8}$	$4.48^{+0.14}_{-0.13}$	$64.3^{+1.4}_{-1.4}$
			130 (4.0 h AEL)	5,652	2,408	18.4	$84.1^{+2.0}_{-2.2}$	$4.06^{+0.09}_{-0.09}$	$61.3^{+1.1}_{-1.2}$
			180 (4.4 h AEL)	5,716	2,008	18.9	$86.2^{+2.0}_{-2.3}$	$3.45^{+0.09}_{-0.09}$	$55.4^{+1.3}_{-1.3}$
			230 (4.8 h AEL)	6,686	2,041	23.5	$89.6^{+1.7}_{-2.0}$	$3.34^{+0.09}_{-0.08}$	$54.8^{+1.2}_{-1.2}$
			280 (5.2 h AEL)	7,318	1,436	26.4	$92.0^{+1.9}_{-2.4}$	$3.20^{+0.11}_{-0.10}$	$51.6^{+1.6}_{-1.5}$
Tomer *et al*. 2012 (NM12)	Yes	Yes	30 (3.2 h AEL)	8,721	1,211	90.8	$88.2^{+2.3}_{-2.8}$	$1.61^{+0.08}_{-0.06}$	$32.3^{+1.0}_{-0.9}$
			80 (3.6 h AEL)	9,139	1,718	79.9	$74.2^{+2.8}_{-3.0}$	$1.65^{+0.14}_{-0.11}$	$34.0^{+1.2}_{-1.1}$
			130 (4.0 h AEL)	6,285	2,408	39.7	$89.1^{+1.6}_{-1.9}$	$1.90^{+0.13}_{-0.11}$	$30.0^{+1.0}_{-1.0}$
			180 (4.4 h AEL)	6,094	2,008	51.9	$86.8^{+2.0}_{-2.2}$	$1.72^{+0.13}_{-0.11}$	$29.2^{+1.0}_{-0.9}$
			230 (4.8 h AEL)	7,126	2,041	52.1	$88.4^{+1.8}_{-2.1}$	$1.44^{+0.10}_{-0.08}$	$26.9^{+0.9}_{-0.9}$
			280 (5.2 h AEL)	7,842	1,436	54.5	$88.5^{+2.3}_{-2.7}$	$2.14^{+0.23}_{-0.19}$	$30.5^{+1.3}_{-1.2}$

[1] If "No", manual segmentation is required for the first time point (t_0) to initialize the algorithm.

[2] These metrics are defined in **Supplementary Note 3**. 95th and 5th percentile confidence intervals are provided as sub-/superscripts.

Supplementary Table 6 | Comparative performance of cell lineaging methods for confocal images of early zebrafish embryogenesis

Method	Automatic seed at t_0?[1]	Method detects cell divisions?	Time point	Segmented objects	Ground truth samples	Processing time (s)	**Linkage accuracy (%)[2]**	**Euclidian distance (µm)[2]**	**Normalized Euclidian distance (%)[2]**
This study	Yes	Yes	30 (6.0 hpf)	2,982	1,206	5.4	$96.8^{+1.1}_{-1.7}$	$1.98^{+0.12}_{-0.11}$	$16.3^{+1.0}_{-0.9}$
			60 (7.0 hpf)	3,374	1,219	4.8	$95.2^{+1.4}_{-2.0}$	$2.00^{+0.12}_{-0.10}$	$18.4^{+1.0}_{-0.9}$
			90 (8.0 hpf)	3,347	1,202	4.3	$94.2^{+1.6}_{-2.2}$	$2.33^{+0.15}_{-0.13}$	$21.8^{+1.2}_{-1.1}$
Kausler *et al*. 2012 (CGT)	Yes	Yes	30 (6.0 hpf)	3,393	1,206	293.6	$82.8^{+2.8}_{-3.2}$	$4.54^{+0.21}_{-0.20}$	$36.8^{+1.6}_{-1.5}$
			60 (7.0 hpf)	3,511	1,219	297.7	$79.1^{+3.0}_{-3.4}$	$4.93^{+0.20}_{-0.19}$	$42.9^{+1.6}_{-1.6}$
			90 (8.0 hpf)	3,433	1,202	300.7	$75.9^{+3.2}_{-3.6}$	$5.02^{+0.23}_{-0.21}$	$44.9^{+1.7}_{-1.7}$
Giurumescu *et al*. 2012 (NT4D)	No	No	30 (6.0 hpf)	2,845	1,206	11.0	$89.7^{+2.2}_{-2.7}$	$5.55^{+0.20}_{-0.19}$	$47.2^{+1.6}_{-1.6}$
			60 (7.0 hpf)	3,293	1,219	13.2	$89.6^{+2.2}_{-2.7}$	$7.08^{+0.23}_{-0.22}$	$63.0^{+1.7}_{-1.7}$
			90 (8.0 hpf)	3,311	1,202	14.1	$90.0^{+2.1}_{-2.7}$	$5.24^{+0.19}_{-0.18}$	$51.6^{+1.7}_{-1.6}$
Tomer *et al*. 2012 (NM12)	Yes	Yes	30 (6.0 hpf)	3,854	1,206	65.2	$93.7^{+1.7}_{-2.2}$	$2.56^{+0.11}_{-0.11}$	$19.1^{+0.8}_{-0.8}$
			60 (7.0 hpf)	4,985	1,219	76.4	$96.4^{+1.2}_{-1.8}$	$2.16^{+0.09}_{-0.08}$	$19.0^{+0.8}_{-0.7}$
			90 (8.0 hpf)	5,680	1,202	78.4	$96.0^{+1.3}_{-1.9}$	$2.03^{+0.08}_{-0.08}$	$19.7^{+0.8}_{-0.8}$

[1] If "No", manual segmentation is required for the first time point (t_0) to initialize the algorithm.

[2] These metrics are defined in **Supplementary Note 3**. 95th and 5th percentile confidence intervals are provided as sub-/superscripts.

Supplementary Note 1 | Sequential Bayesian estimation of Gaussian Mixture Models

Probabilistic model for segmentation and tracking

As explained in the main text, the main idea behind our tracking and segmentation approach is to take advantage of the temporal coherence between consecutive time points and the relatively simple shape that fluorescent-labeled nuclei present in light microscopy images. We model the image intensity at each time point *t* as a Gaussian Mixture Model (GMM) (**Supplementary Fig. 3**):

$$I^t[n] \propto \sum_{k=1}^{K^t} \pi_k^t \mathcal{N}(x_n; \mu_k^t \Sigma_k^t) \quad (1)$$

where K^t is the number of nuclei at time t, x_n are the three-dimensional (3D) coordinates for the nth voxel, and π_k^t, μ_k^t and Σ_k^t define the kth Gaussian mixture at time t. The temporal coherence allows us to estimate the parameters sequentially using the solution at time t as initialization for time $t+1$. The main challenge in the parametric model from Eq. (1) is a strategy for updating the value of K^t during cell division and apoptosis (or loss of cells owing to constraints in the imaging process). *De novo* cell "birth" events are not considered explicitly, since we assume consistent physical sample coverage in the imaging experiment. Below we describe in detail how we address the challenge by solving Eq. (1) for different values of K^t and by adding *a priori* probabilities for the loss of cells. Although non-parametric models for contour evolution, such as multi-level-sets[7], handle topology changes automatically, the challenges associated with live image data of complex multicellular specimens (**Fig. 2**, **Supplementary Videos 5**, **6**, **10**, **11**, **15**, **18**, **22** and **23**) make level-sets boundaries unreliable. Moreover, the number of possible topology changes in cell lineage reconstructions is highly constrained (irrespective of the occurrence of cell divisions). The parametric model allows setting a constrained combinatorial problem to avoid biologically unrealistic solutions. For example, a cell is not allowed to divide into three or more objects at a given time point.

The intuition presented above can be formalized as a full Bayesian framework by inferring the following posterior distribution:

$$P(\boldsymbol{\Theta}, \mathbf{S}, \mathbf{Z}|\mathbf{I}) \tag{2}$$

where $\mathbf{I} = \{I^1, \dots, I^T\}$ is the set of image stacks for time points 1 to T, and $\boldsymbol{\Theta} = \{\Theta^1, \dots, \Theta^T\}$ with $\Theta^t = \{\theta_1^t \dots \theta_{K^t}^t\}$ represents the GMM parameters for each time point ($\theta_k^t = \{\pi_k^t, \mu_k^t, \sigma_k^t\}$). $\mathbf{S}$ and $\mathbf{Z}$ are two sets of hidden random variables that allow decomposing the problem into easier sub-problems. $\mathbf{Z} = \{Z_v^t\}$ for $t = 1 \dots T$, $v = 1 \dots v^t$ represents the standard assignment variable in a mixture model setting. Z_v^t is the probability that the vth sample (in our case the vth super-voxel at time point t) belongs to the kth mixture, and can take any value between 1 and K^t. Aside from simplifying the solution to the problem, knowing the probability of Z_v^t directly returns a probabilistic segmentation for each time point, since it indicates which super-voxels belong to which nuclei. Finally, $\mathbf{S} = \{S_k^t\}$ for $t = 1 \dots T$, $k = 1 \dots K^t$ is a vector of binary random variables representing the probability that the kth mixture (or cell nucleus, in our case) is dividing at time point t.

Once we have defined all random variables in our problem we can use the graphical model presented in **Supplementary Fig. 2** to factorize Eq. (2) as follows:

$$P(\boldsymbol{\Theta}, \mathbf{S}, \mathbf{Z}|\mathbf{I}) \propto P(\boldsymbol{\Theta}, \mathbf{S}, \mathbf{Z}, \mathbf{I}) \tag{3}$$

$$= \prod_{t=1}^{T} P(I^t|Z^t, \sigma^t, \mu^t) P(Z^t|\pi^t) P(\pi^t|S^t, \pi^{t-1}) P(\mu^t|\sigma^t, \mu^{t-1}) P(\sigma^t|\sigma^{t-1}) P(S^t|\sigma^t, \mu^t) \tag{4}$$

$$= \prod_{t=1}^{T} \prod_{v=1}^{V^t} P(I_v^t|Z_v^t, \sigma^t, \mu^t) P(Z_v^t|\pi^t) \prod_{k=1}^{K^t} P(\pi_k^t|S_k^t, \pi_k^{t-1}) P(\mu_k^t|\sigma_k^t, \mu_k^{t-1}) P(\sigma_k^t|\sigma_k^{t-1}) P(S_k^t|\sigma_k^t, \mu_k^t) \tag{5}$$

Eq. (3) is obtained using Bayes' formula. Eq. (4) applies the conditional independence information contained in the graphical model in **Supplementary Fig. 2** (in particular, the solution at time point *t* depends on the solution at time point *t* – 1, following a Markovian process). Finally, Eq. (5) introduces independence assumptions between mixtures and image super-voxels.

If we ignore the random variable **S** and consider only a single time point in the graphical model (**Supplementary Fig. 2**), we obtain a standard graphical model to describe a GMM. Each sample (intensity level of a voxel, in our case) is considered independently of the others[1]. Eq. (5) adds three extensions to the standard GMM. First, priors to the mixture parameters are added based on

values from the previous time point. These priors account for the fact that object shape, position and intensity are correlated in consecutive time points. Second, all voxels belonging to the same super-voxel share the same hidden variable Z_v^t and are assigned to the same mixture, although the likelihood of the model is computed over all voxels (Eq. (1)). Third, we include a second set of hidden variables (**S**) in order to account for cell divisions. Instead of including the number of Gaussians in the mixture (K^t) as a random variable in the model and using hyper-parameters to infer the value of (K^t)[8, 9], we add the binary variable **S** to restrict topological changes to biologically feasible processes. Finally, to simplify the notation we omitted the fact that we do not have priors for $t = 0$. We assume that an initial segmentation is provided for this first time point $t = 0$. As explained in the main text, we use the super-voxels as an initial segmentation for $t = 0$ in our pipeline. Our strategy for performing approximate inference in Eq. (5) is to fit different GMMs at each time point, depending on the value of S^t. The following sections describe each individual step of this procedure in detail.

Variational inference for approximate inference in GMM

If we fix (or assume that we know) the value of S^t for each object we can solve the GMM for each time point efficiently. We follow the exposition by Bishop (chapter 10)[1] to use a variational inference procedure to perform approximate inference on the mixture model. Here, we summarize the key points and insights that are relevant for our implementation.

We choose the following (conjugate) priors, which allow defining an analytical solution to the inference problem:

$$P(\pi^t|\pi^{t-1}) = \mathrm{C}(\alpha_o)\prod_{k=1}^{K^t}(\pi_k^t)^{\alpha_{o,k}^t-1} \tag{6}$$

$$P(\Lambda^t|\Lambda^{t-1}) = \prod_{k=1}^{K^t} W_3(\Lambda_k^{t-1}, \nu_{o,k}^t) \tag{7}$$

$$P(\mu^t|\Lambda^t, \mu^{t-1}) = \prod_{k=1}^{K^t} N(\mu_k^{t-1}, (\beta_{o,k}^t\Lambda_k^t)^{-1}) \tag{8}$$

Eq. (6) represents a Dirichlet distribution with hyper-parameter $\alpha_{o,k}^{t}(\pi_k^{t-1})$, Eq. (7) is a Wishart distribution with three degrees of freedom and hyper-parameter $\nu_{o,k}^{t}(\Lambda_k^{t-1})$, and Eq. (8) is a Normal distribution with hyper-parameter $\beta_{o,k}^{t}(\mu_k^{t-1})$. Λ^t is the precision matrix, which is the inverse of the covariance matrix Σ^t. We use Λ^t since it makes it easier to define a conjugate prior distribution.

Following Eqs. (1) and (5) our likelihood for a single time point is:

$$P(I^t|Z^t,\sigma^t,\mu^t)P(Z^t|\pi^t)=\prod_{n=1}^{N^t}\left[\sum_{k=1}^{K^t}\pi_k^t\mathcal{N}(x_n;\mu_k^t\Sigma_k^t)\right]^{I_n^t} \tag{9}$$

Note that in Eq. (5) all voxels (n) belonging to the same super-voxel (v) share the same hidden variable Z_v^t, but the likelihood is still evaluated over all voxels.

In variational inference, we pose inference for Eq. (5) when the value of **S** is known as an optimization problem over probability distributions as follows:

$$argmin_q KL(P(I^t|Z^t,\sigma^t,\mu^t)P(Z^t|\pi^t)P(\pi^t|\pi^{t-1})P(\Lambda^t|\Lambda^{t-1})P(\mu^t|\Lambda^t,\mu^{t-1})||q) \tag{10}$$

$$subject\ to\ q(Z^t,\Lambda^t,\mu^t,\pi^t)=q(Z^t)q(\Lambda^t,\mu^t,\pi^t) \tag{11}$$

where KL indicates the Kullback-Leibler distance between two probability distributions. The main assumption in Eq. (11) is the fact that the hidden variables Z^t are independent from the parameters. This assumption returns an iterative optimization strategy very similar to the expectation-maximization (EM) algorithm, which alternates between updating the hidden variables and the mixture model parameters. However, we chose variational inference instead of the traditional expectation-maximization (EM) approach because it handles prior probabilities more naturally and in practice avoids singularities in the covariance matrices that can affect numerical stability in the implementation. Moreover, the computational cost is very similar and the three hyper-parameters in Eq. (6-8) (α_o^t,ν_o^t,β_o^t) allow easy control of the behavior of the algorithm with only few degrees of freedom. Formally, the main difference between EM and variational inference is that EM returns a single optimal value as a solution to the fitting of the mixture model, while in variational inference the parameters are treated as random variables to

obtain a probability distribution as a solution. In our case, we take the mean of the distribution as the final solution and use the standard deviation as a confidence estimate.

The complete solution to Eq. (11) can be found in Bishop (chapter 10, section 2)[1]. For our purposes, the key insight is that the three hyper-parameters α_o^t,ν_o^t,β_o^t allow us to control the influence of the priors from the previous time point seamlessly. In particular, ν_o^t controls the extent to which object shape (precision matrix) can change from time point to time point; β_o^t controls the extent to which object centroids can move between two consecutive time points, and α_o^t controls the likelihood for objects to disappear. Setting $\alpha_{o,k}^t$ to high values makes sure that the kth component does not disappear in the next time point, while setting $\alpha_{k_o}^t$ to low values makes the disappearance or apoptosis of cells more likely. Even if a Dirichlet distribution is only formally defined for $\alpha_{o,k}^t \geq 0$, we can artificially set negative values for the priors to indicate that a particular cell nucleus is likely to disappear unless the likelihood indicates otherwise. All hyper-parameters depend on N_k^t, which represents the sum intensity of the voxels corresponding to the kth mixture at time point t. Thus, by defining all three hyper-parameters relative to N_k^t, constant values can be assigned to these hyper-parameters for all objects at all time points in the configuration file needed to run the software framework.

Practical implementation

As explained above, we perform sequential inference in the dynamic Bayesian network (**Supplementary Fig. 2**), where we test different GMM models at each time point using variational inference to account for cell divisions. The main reason for performing such a greedy inference across time points is that each cell division changes the number of possible states and the dimensionality of the probability distribution in Eq. (2), which makes global inference intractable even when considering jumps between probability spaces. Moreover, the fact that our image data sets are correlated in consecutive time points (more specifically, nucleus centroids tend to not move further than the average nearest neighbor distance between time points) allows us to solve the problem using a parametric contour evolution method. As explained in the main text (**Fig. 1**), we first evolve the ellipsoids without changing the number of elements. Thus, only loss of cells can occur in the first round of variational inference. Then, we look for nuclei that

Nature Methods: doi:10.1038/nmeth.3036

group non-connected super-voxels and split them, effectively increasing K^t. Thus, in the second round of variational inference loss of cells and cell divisions are allowed.

In practice, we have to make sure that our images are as close to the model represented by Eq. (1) as possible. However, raw light microscopy data sets usually contain uneven background levels, a circumstance that is in conflict with the assumption that the Gaussian probability distributions tend to approach zero in their tails. We use the super-voxel partition of the images to perform local background estimation and subtract this estimate from the original image. Since our super-voxel segmentation is a full partition of the image, each super-voxel contains voxels belonging to a nucleus and to background, respectively (**Fig. 1**). Thus, for each super-voxel, we calculate the background level using Otsu's method[10]. Before subtracting the background, we smooth the background estimate with a Gaussian kernel to avoid sharp transitions between super-voxel boundaries.

Another important consideration in the final pipeline is the strategy for setting the hyper-parameters α_o^t, ν_o^t, β_o^t. We define all of these parameters relative to N_k^{t-1} and keep them fixed across all experiments presented in this work. However, it is possible to change the settings in the advanced parameters section of the software configuration file. In our case, $\nu_{o,k}^t$ is set equal to N_k^{t-1}. Thus, the shape prior carries as much weight as the image data, since we do not expect sudden shape changes and N_k^{t-1} should be similar to N_k^t. We set $\beta_o^t = 0.1N_k^{t-1}$, since we need to account not only for average cell movements, but also for exceptionally large, local displacements during fast developmental processes. If, in a particular experiment, one can incorporate a good motion predictor for each cell, it would be possible to increase the value of this prior to reflect higher certainty in predicting nucleus centroid location. Finally, we set $\alpha_{o,k}^t = 0.8 \times 10^{-5} \sum_{k=1}^{K^{t-1}} \alpha_k^{t-1}$, such that only mixtures where N_k^t contributes less than 0.0002% to the total value of α^t are allowed to disappear. In practice, aside from the priors, we also set absolute boundaries on the size and eccentricity of each Gaussian mixture to avoid biologically unrealistic results. These constraints are easily implemented by calculating the eigenvalues of the precision matrix after each iteration. Again, these parameters were kept constant across all biological model systems and imaging methods (we set a maximum radius of 10 pixels in each principal direction and a maximum eccentricity of 3 between each pair of principal directions) but they can be modified in the advanced parameters section of the software configuration file.

Finally, it is important to keep in mind that high computational efficiency (i.e. fast processing speed) is crucial when processing time-lapse imaging experiments with thousands of time points, each comprising up to hundreds of millions of voxels and tens of thousands of cell nuclei. We therefore take advantage of GPU computing to implement variational inference. Kumar *et al*. previously demonstrated a 100-fold speed-up for the EM algorithm using GPU computing[11]. We implemented our variational inference algorithm using Nvidia CUDA obtaining similar speed-ups and improved memory scalability by sparsifying the matrix of responsibilities. Thereby, our implementation solves the variational inference problem for over 10,000 Gaussians in less than one second on a single computer workstation.

Supplementary Note 2 | Spatio-temporal features for the background detector

In order to detect background objects we trained a machine learning binary classifier using several features calculated in a sliding temporal window for each object track. The classifier predicts the probability that the object at the center of the temporal sliding window does not represent a cell nucleus. The features, which include object shape, object intensity, and dynamic and temporal characteristics of each object track, are described below in detail. We used our CATMAID interface to generate a training set of object tracks for real cell nuclei and background objects obtained from different automatic reconstructions. Then, we generated a classifier with the Matlab implementation of Random UnderSampling Boost (RUSBoost)[12] with classification trees. RUSBoost is designed to handle class imbalance problems and was chosen because our training set contains fewer background annotations than foreground annotations. **Supplementary Fig. 4** shows the precision-recall curve for the test set. The same classifier was used for all model organisms and microscope types and is provided together with the source code.

Below we provide a detailed list of the features used for background classification. All features, except for the temporal features, are calculated over branches in the tracks without cell divisions. These features are based on the intuition that cell nuclei exhibit coherent behavior within short time intervals (sliding window), whereas object tracks corresponding to background structures exhibit incoherent behavior.

- *Straight line fit to x, y and z positions (dynamic feature)*. We use the least-squares method to fit a straight line trajectory to the x, y and z positions of all centroids in the spatiotemporal window. We then determine the mean and standard deviation (SD) of the distance from each point to the fitted line. Using these features, we expect cell nuclei to exhibit more robust movement directions within the sliding window, compared to background objects. We also recalculate the mean and SD after removing the point furthest from the fitted line.

- *Constant velocity model (dynamic feature)*. We measure the displacement in x, y and z between all consecutive time points. We define the SD along all three axes as a feature.

In contrast to background objects, we expect cell nuclei to follow a constant velocity model within the sliding window.

- *Change in relative size (shape feature)*. We calculate the relative change in number of voxels (object size) between all consecutive time points. We use the SD of this metric as a feature.

- *Offset Jaccard distance (shape feature)*. We calculate the Jaccard distance between segmented objects in consecutive time points. In order to eliminate a possible contribution to the distance measure arising from object movement, we translate the daughter object in x, y and z to maximize object overlap. We use the mean and SD of this metric as a feature.

- *Number of holes inside an object (texture feature)*. For each object, we count the number of holes (background voxels) within the segmented volume. We use the mean of this metric as a feature. If the segmented object is a fluorescently labeled cell nucleus, it usually does not enclose background voxels. In contrast, background objects frequently contain background voxels within the segmented volume.

- *Normalized k-nearest neighbor distance (spatial feature)*. For each nucleus, we calculate the distances to its first six nearest neighbors. We then normalize these distances to the distance to the nearest neighbor and use them as features (disregarding the first normalized distance since it is always 1). If the segmented object is a fluorescently labeled cell nucleus, it tends to be surrounded by nearest neighbors at similar distances. This distribution is typically less even for autofluorescent background objects.

- *Cell activity recognition (temporal feature)*. Inspired by the "bag-of-words" approach of Bettadapura *et al*.[13], we calculate multiple histograms to determine the frequencies of different types of events within siding windows for each object track. Specifically, we count the number of track deaths, track divisions and track displacements, and we also determine histograms of the lengths of intervals between events.

The background detector is provided as an optional post-processing step that is usually helpful in samples with high levels of auto-fluorescence. The time-lapse recordings of *Drosophila*

embryogenesis presented in this study are a good example of such a scenario. During early developmental stages, auto-fluorescent regions of the fruit fly embryo can represent up to 15% of the total recorded signal. Since we model each 3D image as a Gaussian mixture model (GMM), the entire intensity content of the image needs to be accounted for. Thus, without the background detector's ability to eliminate tracked objects that do not represent real cells, the automated reconstruction would contain a significant number of false positive detections.

The background detector introduces only three advanced parameters in the tracking framework: the size of the temporal window used to calculate the spatio-temporal features described above and the upper and lower thresholds that are applied to the result of the machine learning classifier in order to produce a temporal filter with hysteresis. The features themselves are not parameters. Rather, they are internally calculated in the code and needed for the machine learning classifier to determine the probability of a detected object to represent a real cell. In this regard, the precision-recall curve provided in **Supplementary Fig. 4** shows that by setting the right threshold, we can detect most of the background objects (recall) without falsely classifying true cells as background objects (precision). Thus, in cases like the fruit fly embryo, this post-processing step is critical to eliminate large numbers of trajectories that do not represent real cells. In other cases, such as zebrafish embryos, this post-processing step is less critical, owing to the low level of auto-fluorescence.

Supplementary Note 3 | Estimation of segmentation and tracking accuracy

Obtaining a comprehensive ground truth annotation for our data sets is practically impossible, since we estimate the time required for manually drawing all nuclei boundaries and connecting them across all time points to be on the order of at least several years. The accuracy of the algorithm can thus not be practically evaluated on this basis. Moreover, since cells in the developing embryo frequently move over long distances and image quality and reconstruction challenges vary dramatically across the embryo (**Fig. 2**), it is also challenging to obtain an exhaustive, unbiased subset of ground truth annotations for a significant fraction of embryonic development. For example, if the sampled region only contains superficial cells, the accuracy analysis will overestimate global accuracy. Thus, we devised four types of metrics that reflect different properties of the accuracy of the cell lineage reconstruction and can be calculated by random sampling of the data. The random sampling furthermore allows us to establish confidence intervals for each metric. We have incorporated a random annotation function in our CATMAID interface (accessible by pressing Q from within the Tracing Tool) that enables users to efficiently generate random samples in the image space for unbiased annotations.

The first metric is *linkage accuracy*. The annotator marks the centroid of a random nucleus at time point t and links it to the correct centroid location at time $t + 1$. Using this pairwise annotation, we can determine if the algorithm connected the correct objects between consecutive time points. For each annotated centroid, we determine which nucleus it was assigned to in the automatic reconstruction. Then, we check if both nuclei are connected in time. Thus, the *linkage accuracy* measures tracking accuracy. However, it does not measure under-segmentation accuracy (imagine constructing a single ellipsoid encompassing the entire image, which would always lead to a perfect success rate of 1 in the *linkage accuracy* metric).

The *Euclidean distance metric* aims to provide a precise measure of under-segmentation and uses the same annotations as the *linkage accuracy*. For each annotated centroid, we again determine which nucleus it was assigned to in the automatic reconstruction. Then, we measure the Euclidean distance between the annotated centroid and the centroid of the nucleus from the automatic reconstruction. The *Euclidean distance metric* is the average of all Euclidean distances for all annotated time points. Thus, a good algorithm will have a *Euclidean distance metric*

smaller than the expected radius of a nucleus. However, this metric could be made exactly zero by assigning one centroid to each voxel in the image. In other words, it does not account for over-segmentation.

The *nearest neighbor (NN) normalized distance metric* measures over-segmentation and uses the same annotations as the *linkage accuracy*. For each annotated data point, for which we have calculated the *Euclidean distance metric* explained above, we normalize the distance measure by the distance to the nearest neighbor in the automatic reconstruction. Thus, if the region is over-segmented by the algorithm, the *NN normalized distance metric* measure will increase significantly. In fact, we limit the maximum value to 1, in order assign comparable weight to correct reconstructions versus incorrect reconstructions. The closer the *NN normalized distance metric* measure is to zero, the higher the quality of the automatic reconstruction.

To complement the segmentation metrics described above, we estimate true positive and false positive rates using a surrogate based on our centroid annotations performed in CATMAID (**Supplementary Table 3**). For each time point, we estimate the maximum nucleus radius by calculating the radius of a sphere of equal volume for each segmented nucleus and determining the 75^{th} percentile of this radius across all nuclei for this time point. We then count the number of centroids in the automated segmentation that are located within this radius for each ground truth annotation. If no centroids are found within the radius, we consider the annotated object a false negative. In our results, most false negatives correspond to under-segmented nuclei, i.e. the centroid has been placed in between two neighboring nuclei. If only one centroid is found within the radius, we consider the annotated object a true positive. If two centroids or more are found within the radius, the annotated object is considered an over-segmented nucleus. We only present results for zebrafish embryos because this approximation is less accurate in other specimens, such as *Drosophila* embryos, where nuclei can have elongated shapes that differ significantly from a sphere. Finally, we measure the number of false positives by visual inspection of the automated reconstruction results in CATMAID: two different annotators visited all centroid locations returned by the automated pipeline and evaluated if these locations corresponded to a background object instead of a true cell.

We also evaluate cell division accuracy (**Supplementary Table 1**). For each analyzed time point, we visually inspected all cell divisions returned by the automated pipeline and classified

these as true or false positives. For false positives, we distinguished between (1) cases, in which the algorithm recovered from under-segmentation, (2) false divisions detected in background objects and (3) other cases. We made this distinction because the first category of false positives listed above tends to dominate and is in fact required to avoid propagating errors in our sequential approach. Finally, we inspected 3D image stacks in the SiMView and confocal microscopy recordings of zebrafish embryogenesis and manually annotated all cell divisions at specific time points. These annotations allowed us to measure the number of false negatives for these time points.

Nature Methods: doi:10.1038/nmeth.3036

Supplementary Note 4 | Morphodynamic features of cell trajectories

For each neuroblast cell lineage, we calculated the following morphodynamic features (**Supplementary Fig. 15**) as a basis for the analysis presented in the section "*Reconstruction of early Drosophila nervous system development at the single-cell level*" in the main text.

- *Internalization time*: at each time point, we estimate the distance of the nucleus to the surface of the embryo (depth). In the case of neuroblasts, depth as a function of time follows approximately a sigmoidal shape due to the internalization process (data not shown). We therefore fit a sigmoid to each time-vs.-depth profile and use the half rising time as an estimate of the cell's internalization time point.

- *Final depth*: using the same sigmoidal fit as above, we estimate the resting depth of the neuroblast in the embryo.

- *Division time*: we note the time points, at which each cell undergoes a cell division.

- *Frequency of division*: for neuroblast lineages with more than one division of the neuroblast in the time window of the curated reconstruction, we determine the time difference between divisions.

- *Angles of division*: for each cell division event, we determine the two division angles (ψ and η) using a local spherical coordinate system (**Supplementary Fig. 14**).

- *Depth at division*: for each cell division event we note the current depth of the nucleus prior to division.

- *Path length to division*: we integrate all displacements between consecutive time points to calculate the total distance a neural precursor cell travels from its origin in the blastoderm up to the first cell division event.

Nature Methods: doi:10.1038/nmeth.3036

Supplementary Note 5 | Comparative performance of cell lineaging methods

We compared our cell lineage reconstruction pipeline to three other recently published algorithms: Chain Graph Tracking (CGT)[4], Nuclei Tracker 4D (NT4D)[5] and our own previous tracking pipeline (NM12)[6] for cell dynamics in the *Drosophila* syncytial blastoderm. All methods were specifically developed for cell lineage reconstructions from time-lapse light microscopy images of fluorescently labeled nuclei. In the case of CGT and NT4D, we used the code made publicly available by the authors. For CGT, the available code is a slightly simplified version of the method published in Kausler *et al*.[4], since it uses a constant prior probability to determine whether a detection is a true cell nucleus or not, instead of a random forest classifier for object detection to establish this prior probability. For NT4D, we modified basic data structures from the original Matlab code to improve data throughput rates when working with data sets containing thousands of cells per time point.

CGT is a data-association method using linear programming to perform inference in a graphical model connecting detections between time points. These detections are created using a segmentation module that assigns to each pixel the probability of belonging to a nucleus based on a trained random forest classifier. The probability map is then thresholded, and each resulting connected component represents a detected object that can be linked in time. This approach performs well in early developmental stages, such as the *Drosophila* blastoderm and the early zebrafish embryo, where nuclei are well separated (**Supplementary Tables 4** and **6**). In later developmental stages, cells are very close to each other, and after thresholding a single connected component can encompass tens of cells. Thus, the detection step contains too many merged cells and precludes the return of a meaningful solution in the data-association step (**Supplementary Table 5**). However, when attempting to set a higher threshold to better separate nuclei, the detection misses too many objects located in parts of the image with lower contrast. Moreover, since the segmentation step has to calculate features and apply a random forest classifier for each voxel, it is computationally slower than our watershed approach. Inference in the graphical model for the linkage step seems to scale quadratically with the number of objects, which makes it harder to scale this approach to tens of thousands of objects (**Supplementary Table 5**).

NT4D was devised as a semi-automatic cell lineaging program, which propagates the solution sequentially from the previous time point to the next, using blob detection techniques for segmentation and nearest neighbor assignments for linkage. Prior to each propagation of the solution, the user verifies the proposed solution at the current time point using a well-designed Matlab user interface. The user is required to manually annotate all cell divisions as well as the segmentation of the initial time point. In the original paper, the authors tracked and validated cells in *C. elegans* and zebrafish at a rate of 4,000 data points per day. In order to evaluate the accuracy of the automatic sequential propagation, we initialized NT4D using the solution provided by our own algorithm ten time points before the time point where we sample ground truth. For those ten time points, we accepted every proposed move by the NT4D algorithm and then evaluated the linkage accuracy and Euclidean distance at the time point with ground truth (**Supplementary Tables 4-6**). Due to the simplicity of assumptions in the sequential propagation of the solution, accuracy metrics degrade much faster in NT4D than in our approach, even after only ten time points, although both methods require comparable computation time. Similarly to CGT, this method performs better in early developmental stages (**Supplementary Tables 4** and **6**), where the data sets match the conditions the authors developed the algorithm for.

Our previous tracking pipeline (NM12) was designed to reconstruct cell behavior in early developmental stages in *Drosophila*, specifically the synchronized mitotic waves in syncytial blastoderm stages. Since nuclei are well separated in these early stages, the Gaussian Mixture Model can be applied on a per voxel basis (instead of per super-voxel) and still produce results comparable to those of our new method (**Supplementary Table 4**). However, in later stages, the boundaries between cell nuclei are less clear and the Gaussians tend to under-segment, producing single mixtures explaining multiple nuclei. In addition, more iterations of the variational inference approach are required to converge to a solution. Finally, more pixels are part of the foreground in later stages, and the GPU-based approach runs out of memory trying to save multiple responsibility values per voxel (**Supplementary Table 5**). Thus, to be able to run NM12 at all in later stages of *Drosophila* embryogenesis, we had to raise the image background threshold at the expense of losing lower contrast cell nuclei. All of these issues (under-segmentation, scalability and convergence speed) are overcome by our hierarchical segmentation approach based on super-voxels as the basic image unit. Finally, the addition of data-association rules in a local spatio-temporal window further improves tracking and segmentation results in

our new method, and enhances the detection of mixtures that do not represent cell nuclei (**Supplementary Table 5**).

Supplementary Software 1 | Automated modules of the cell lineaging framework

This section contains basic instructions for installing and running the cell lineaging software package "Tracking with Gaussian Mixture Models" (TGMM). The code provided here has been tested with the 64-bit version of Windows 7 and with the 64-bit version of Ubuntu Linux 12.04 LTS, using a variety of CUDA-compatible NVIDIA GPUs.

1. Contents of the software archive

We assume that the user has uncompressed the file "Supplementary_Software_1.zip" in a folder of their choice, referred to here as $ROOT_TGMM. The subfolders in $ROOT_TGMM contain the following components:

- *src:* All source code files. This folder also includes a CMakeList.txt file that can be used to generate a Visual Studio solution (using CMake) and compile the source code.
- *doc:* Documentation of the TGMM software.
- *build:* A Visual Studio C++ 2010 project generated from src using CMake. This subfolder also contains precompiled binaries, suitable for running the code without the need for re-compiling the source code.
- *data:* Contains a three-dimensional time-lapse data set with 31 time points (corresponding to a cropped sub-region of the *Drosophila* SiMView recording presented in the main text), for testing the TGMM code and ensuring that the software is running as expected.

Note: The Visual Studio project will not compile unless the folder "build" is copied to the same absolute path as that used to generate the project. We provide the full project folder primarily as a reference for the final structure of a successful Visual Studio solution.

2. Installation and software requirements

In order to run the precompiled binaries, the following auxiliary software package must be installed as well:

- CUDA Toolkit 5.5: required to run algorithms on an NVIDIA GPU
  Download: https://developer.nvidia.com/cuda-toolkit-archive

We provide precompiled binaries for the 64-bit version of Windows 7. The folder with the precompiled binaries also contains all required DLLs, and thus no external software packages other than the CUDA drivers mentioned above need to be installed. The software can effectively be run out-of-the-box, as detailed below in section 3.

For Linux, compilation of the source code is required (see detailed instructions in section 2.1). For compiling the source code, any software version equal to or above the CUDA Toolkit software version listed above should suffice.

For possible common runtime errors and solutions see section 5.

2.1 Source code compilation in Linux

- Make sure CMake is installed (http://www.cmake.org/). For Ubuntu distributions, you can simply use the following command:
  sudo apt-get install cmake cmake-gui
- Go to the folder $ROOT_TGMM and create a new folder called "build", where the binaries will subsequently be generated:
  cd $ROOT_TGMM
  mkdir build
  cd build
- In the build folder, execute the following commands:
  cmake -D CMAKE_BUILD_TYPE=RELEASE ../src/
  make

The first command locates all libraries (for example, from the CUDA Toolkit) and generates all necessary makefiles to compile the code. The second command calls these makefiles. After executing the second command, you should see messages in the terminal commenting on the compilation progress. If the progress report reaches 100%, the program has compiled successfully. After successful compilation, the following executables should be present:

$ROOT_TGMM/build/nucleiChSvWshedPBC/ProcessStack
$ROOT_TGMM/build/TGMM

You can use cmake-gui or cmake options to change different parameters in the makefiles (for example, final destination folder or CUDA architecture level).

3. Running the TGMM software

We provide a test data set that allows the user to test the code and familiarize themselves with software configuration before applying the code to their own data sets. Currently, 2D + time and 3D + time datasets with 8-bit or 16-bit unsigned integer TIFF stacks are supported as the input data format. The two-dimensional or three-dimensional image data recorded for each time point should be provided as a single TIFF file.

3.1 Configuration file

The file "$ROOT_TGMM\data\TGMM_configFile.txt" serves as a template for the configuration file and contains all parameters required to run the TGMM code. In principle (and for all results presented in this study), only parameters listed under "main parameters" need to be modified for each new experiment. Access to parameters listed under "advanced parameters" is provided as well and is intended for experienced users who wish to experiment further with the code.

Each parameter is accompanied by a description of its functionality (see section "Overview of advanced framework parameters" below for more details). In order to process a new data set, simply copy the configuration text file and adjust parameters as needed.

Important note: Before applying the TGMM software to the test data set, the variables *debugPathPrefix* and *imgFilePattern* in the configuration file need to be adjusted, so the software can locate the image stacks (*imgFilePattern*) and save the results (*debugPathPrefix*).

Nature Methods: doi:10.1038/nmeth.3036

3.2 Watershed segmentation with persistence-based agglomeration

Windows

In order to generate the hierarchical segmentation for each time point, follow these three steps:

1. Open a Windows command line terminal (run "cmd.exe").
2. Go to the folder "$ROOT_TGMM\build\nucleiChSvWshedPBC\Release".
3. Execute the command:
   ProcessStackBatchMulticore.exe $ROOT_TGMM\data\TGMM_configFile.txt 0 30

The program automatically detects how many processing cores are present in the workstation and parallelizes the image segmentation task accordingly. The last two arguments are the first time point and the last time point of the time-lapse image data set.

Once processing is complete, new files "$ROOT_TGMM\data\TM?????_timeFused_blending\SPC0_CM0_CM1_CHN00_CHN01.fusedStack_?????_hierarchicalSegmentation_conn3D74_medFilRad2.bin" should have been generated (one for each time point). These binary files store all information required to restore the hierarchical segmentation for each time point. If the binary files were not created, an error occurred during execution of "ProcessStackBatchMulticore.exe" and a corresponding error message is displayed in the terminal.

Linux

In order to generate the hierarchical segmentation for each time point, follow these three steps:

1. Open a terminal.
2. Go to the folder "$ROOT_TGMM/build/nucleiChSvWshedPBC".
3. Execute the command:
   parallel -j8 ./ProcessStack $ROOT_TGMM\data\TGMM_configFile_linux.txt -- {0..30}

The option -j8 indicates how many cores should be used in parallel (in this case 8). The last option, {0..30}, indicates that the program ProcessStack should be executed for time points 0 to 30.

Important note: The command *parallel* is part of the GNU software (http://www.gnu.org/software/parallel/). The program presents an easy interface to call programs in parallel. If this software is not already installed, it can be downloaded from the GNU website or installed from official repositories. For example, in Ubuntu you can simply use the following command: "*sudo apt-get install moreutils*".

Important note: Make sure to use the configuration file TGMM_configFile_linux.txt instead of TGMM_configFile.txt, since the latter contains Windows end-of-line symbols that will lead to a failure during code parsing in Linux. You can also use the tool dos2unix to ensure that any given text file can be used as a config file.

3.3 Bayesian sequential tracking with Gaussian Mixture Models

In order to track cell nuclei and reconstruct cell lineages, follow these three steps (the same instructions are valid for Windows and Linux):

1. Open a Windows command line terminal (run "cmd.exe" in Windows).
2. Go to the folder "$ROOT_TGMM\build\Release"
3. Execute the command:
   TGMM.exe $ROOT_TGMM\data\TGMM_configFile.txt 0 30

The command line will display notifications about the progress of the tracking and segmentation algorithm. Since the hierarchical segmentation results from step 3.2 are saved separately in the ".bin" files, different tracking parameter settings can be tested without the need for recalculating or changing the segmentation data. The output data format of the tracking module is explained in section 4.

3.4 Verifying successful program execution

In order to simplify the verification of successful TGMM software execution, we provide the output for the test data set in "$ROOT_TGMM\data\TGMMruns_testRunToCheckOutput". The output generated by your execution of the program should be very similar to the contents of this folder.

4. Tracking and segmentation output data format

The folder "*debugPathPrefix*\GMEMtracking3D_%date" contains the output of the TGMM run.

The final result can be found in the subfolder "$debugPathPrefix\GMEMtracking3D_%date\XML_finalResult_lht" or "$debugPathPrefix\GMEMtracking3D_%date\XML_finalResult_lht_bckgRm". The latter directory is used if the user applied the background classifier. The output subfolder contains one XML file and one ".svb" file per time point.

The XML file contains the main tracking and segmentation information. Each object is stored under the tag <GaussianMixtureModel> with the following attributes:

- id [integer]: unique id of the object in this particular time point.
- lineage [integer]: unique id of the cell lineage the object belongs to.
- parent [integer]: id of the linked object at the previous time point. Following the chain of "parent" objects reconstructs the track. A value of -1 indicates the birth of a track.
- splitScore [float]: confidence level for the correct tracking of this particular object. A value of 0 indicates very low confidence and a value of 5 indicates very high confidence. Sorting elements by confidence level can guide the user in the data curation process and facilitate more effective editing of the TGMM results (see main text and **Fig. 4**).
- scale [float[3]]: voxel scaling factors along the *x*-, *y*- and *z*-axis.
- nu, beta, alpha [float]: value of the hyper-parameters for the Bayesian GMM.
- m [float[3]]: mean of the Gaussian Mixture (object centroid, in pixels).
- W [float[3][3]]: precision matrix of the Gaussian Mixture (object shape).
- *Prior: same as before, but for prior values obtained from the previous time point. These values are used during the inference procedure.
- svIdx [integer[]]: list of indices of the super-voxels clustered by this Gaussian. Together with the ".svb" file, this information can be used to obtain precise segmentation regions for each object.

The ".svb" file is a binary file in a custom format that can be read with the constructor "supervoxel::supervoxel(istream& is)". Briefly, it contains information about all super-voxels

generated at a particular time point. Thus, using the "svIdx" attribute, the precise segmentation mask for each object can be recovered.

5. Troubleshooting common runtime errors

1. Program execution starts and one of the following error messages is displayed in the terminal: "no CUDA- capable device is detected" or "CUDA driver version is insufficient for CUDA runtime version".

   First, confirm that the workstation is equipped with an NVIDIA CUDA-capable graphics card. This is a hardware requirement for running the software. If such a card is installed, you most likely need to update the driver in order to be compatible with CUDA Toolkit 5.5. Go to https://developer.nvidia.com/cuda-downloads and download the current toolkit. The toolkit will also install an updated NVIDIA driver.
2. When you try to run the program from the terminal, a Windows dialog pops up with the following message "The program can't start because msvcp100.dll is missing from your computer".

   For some reason, the provided DLL from the Microsoft Visual C++ 2010 SP1 Redistributable Package (x64) is not compatible with your windows version. Delete the DLL from the TGMM software folder and go to http://www.microsoft.com/en-us/download/confirmation.aspx?id=13523 to download and install the appropriate version of the Microsoft Visual C++ 2010 SP1 Redistributable Package.
3. Note that the program needs to be called from a "cmd.exe" terminal in Windows. Cygwin or MinGw terminals cause the program to fail.
4. "ProcessStackBatchMulticore.exe" requires paths to be provided using absolute path names. The use of relative path names also causes the program to fail.
5. Note that the parameter "imgFilePattern" in the configuration file "TGMM_configFile.txt" requires the use of forward slashes in path names (since the image library used to read TIFF files follows the Unix convention), whereas the parameter "debugPathPrefix" in the same file requires the use of double backslashes in path names (since backslashes are special characters that are interpreted by the operating system). On Linux systems, always use forward slashes in both parameters.

6. Program execution starts and one of the following error messages is displayed on the terminal: “invalid device symbol in C:/ROOT_TGMM/src/nucle/iChSvWshedPBC/ CUDAmedianFilter2D/medianFilter2D.cu at line 230”

   The provided binaries were compiled for CUDA compute capability 2.0 or higher. If your NVIDIA GPU card has a lower CUDA compute capability (this information is available from https://developer.nvidia.com/cuda-gpus), the provided binaries will not work. However, you can recompile the source code, which should allow you to run the software. Before compiling, you need to edit the CMakeLists.txt file and modify the line at the top: set (SETBYUSER_CUDA_ARCH sm_20 CACHE STRING “CUDA architecture”). Adjust the flag sm_20 to the appropriate CUDA compute capability of your NVIDIA GPU (for example, sm_13 for CUDA compute capability 1.3).

Key configuration parameters: intensity threshold and persistence agglomeration threshold

Across all computational reconstructions and data sets presented in this study, two parameters were modified: the threshold for persistence-based agglomeration of watershed regions (*persistenceSegmentationTau*) and the intensity threshold for defining the background level in each recording (*backgroundThreshold)*. Both of them refer to image properties and are straight-forward to determine by visual inspection of the image volume at a late time point of the time-lapse recording. In general, inspecting late time points is more useful, since (depending on the experiment) intensity levels are often slightly dimmer and cell densities higher. Measurements in this scenario provide a lower bound constraint for both values.

To determine the background threshold, simply inspect a region in the image volume outside the specimen (for example, by using the open-source software ImageJ) and determine the mean intensity level in this background region. It is preferable to be conservative, i.e. to set a lower value so as not to miss cell nuclei, since false negatives can alter the coherence between time points. Moreover, we compute a local threshold for each super-voxel using Otsu’s method (see section “Detection of Cell Divisions” in the **Online Methods**) and, thus, even if background regions are included in the foreground estimate, this will not affect the final shape of the super-voxels (**Supplementary Fig. 5**). The only drawback of a lower background threshold is a small increase in computation time.

To determine the threshold for persistence-based agglomeration of watershed regions (τ, see also **Fig. 1**), plot the intensity profile across the line connecting two of the dimmest nuclei centroids in the image stack (for example, by using the open-source software ImageJ). The profile should have two peaks (nuclei centroids) and a valley (nuclei borders) (**Supplementary Fig. 1**). The threshold τ should be set to a value smaller than the difference between the intensity values of the peaks and the valley, such that the corresponding nuclei are not merged into a single super-voxel (under-segmentation). In our experience, a value of τ between 5 and 20 tends to be sufficient to compensate for the watershed over-segmentation of noisy regions, without risking merging of dim cell nuclei.

We note that, although care should be taken to set these parameters appropriately, one can obtain close-to-optimal results for a fairly wide range of parameter values (**Supplementary Fig. 5**). For images with lower signal-to-noise ratio (SNR), such as confocal microscopy images, the value of τ is more critical than the background intensity because watershed regions fragment the image into smaller regions. In contrast, in images with high SNR, such as most light-sheet microscopy images, the intensity background is more relevant because the watershed algorithm already produces super-voxels that follow nucleus morphologies fairly well.

Overview of advanced framework parameters

In this section, we provide an overview of all advanced framework parameters. Note that these parameters were not changed across the computational reconstructions and data sets presented in this study. To complement the descriptions below, we also provide the default parameter values in the configuration file "$ROOT_TGMM\data\TGMM_configFile.txt", which is included in "Supplementary_Software_1.zip".

betaPercentageOfN_k

Non-negative floating point scalar, described in Eq. (8) in **Supplementary Note 1**. *betaPercentageOfN_k* defines the prior probability for the centroid position of a nucleus based on its position in the previous time point. No motion model is used, unless the optical flow module is activated. Thus, if cells are moving fast this parameter should be set close to zero. If

cells are moving very little, and the position at time t is a good prediction for the position at time $t + 1$, this parameter should be set to 1 or greater.

nuPercentageOfN_k

Non-negative floating point scalar, described in Eq. (7) in **Supplementary Note 1**. Follows the same concept as *betaPercentageOfN_k*, but for shape variation between consecutive time points. If objects change shape rapidly between two consecutive time points this parameter should be set close to zero. If object shapes change very little between time points this parameter should be set to 1 or greater.

alphaPercentage

Floating point scalar, described in Eq. (6) in **Supplementary Note 1**. *alphaPercentage* controls the prior probability of death for a track. The more likely nuclei are to disappear from the image or undergo apoptosis, the lower the value of this parameter should be.

maxIterEM

Integer positive number. *maxIterEM* defines the maximum number of iterations of variational inference allowed each time a Gaussian mixture model is fitted. In general, very few rounds are needed (less than 10), since the model is initialized with the solution from the previous time point. Thus, this parameter is implemented as a precaution. The terminal output of TGMM.exe can be used to obtain an estimate of the typical number of iterations needed and *maxIterEM* can then be set accordingly.

tolLikelihood

Floating point positive number. *tolLikelihood* is used as a stopping criterion for variational inference of the Gaussian mixture model. Optimization is stopped if the relative increase in likelihood between two consecutive iterations is less than the value of this parameter. Thus, the lower the value, the more iterations of variational inference are run.

Nature Methods: doi:10.1038/nmeth.3036

regularizePrecisionMatrixConstants_lambdaMax

Floating point positive number. This parameter provides the maximum allowed value for any of the eigenvalues (in pixels) of the covariance matrix defining each object in the Gaussian mixture model. Thus, the larger the value of *regularizePrecisionMatrixConstants_lambdaMax*, the larger the ellipsoids fitting each nucleus can grow. This parameter is used during regularization of the variational inference results.

regularizePrecisionMatrixConstants_lambdaMin

Floating point positive number. This parameter provides the minimum allowed value for any of the eigenvalues (in pixels) of the covariance matrix defining each object in the Gaussian mixture model. Thus, the lower the value of *regularizePrecisionMatrixConstants_lambdaMin*, the smaller the ellipsoids fitting each nucleus can shrink. This parameter is used during regularization of the variational inference results.

regularizePrecisionMatrixConstants_maxExcentricity

Floating point positive number. This parameter provides the maximum eccentricity between any two principal axes of the ellipsoid defining a nucleus. This parameter is used during regularization of the variational inference results.

temporalWindowForLogicalRules

Positive integer number. This parameter provides the radius of the temporal window (total window length is [2 × *temporalWindowForLogicalRules* + 1] time points) used to apply spatio-temporal heuristic rules for fixing tracking errors. The larger the value, the more memory is required, since the program will keep the image data of all concerned time points in memory to be able to calculate the features needed to apply the various heuristics.

thrBackgroundDetectorHigh and *thrBackgroundDetectorLow*

Floating point non-negative numbers. These parameters provide the thresholds applied to the results of the background track detector for removing trajectories representing non-nuclei

objects. They control the behavior of a hysteresis filter applied over time to the background probability scores of multiple data points belonging to the same lineage. When the program detects a data point with a background probability above *thrBackgroundDetectorHigh* it proceeds with deleting its descendants until the probability falls below *thrBackgroundDetectorLow*. Thus, the higher the value of *thrBackgroundDetectorHigh*, the fewer objects are removed. If *thrBackgroundDetectorHigh* is above 1, then no background track removal is applied.

SLD_lengthTMthr

Non-negative integer number. Any daughter branch that ends within less than *SLD_lengthTMthr* time points after division is considered a spurious over-segmentation event and is deleted.

radiusMedianFilter

Positive integer number. This parameter provides the radius (in pixels) of the median filter applied before the watershed hierarchical segmentation is performed. The noisier the images, the larger this value should be.

minTau

Non-negative floating point value. This parameter provides the minimum value of τ used for the hierarchical segmentation using persistence-based clustering of watershed regions. The higher *minTau*, the larger the minimum super-voxel size that can be generated at the lower level of the hierarchical segmentation. This value should be kept low so as not to compromise the framework's capability to recover from under-segmentation.

conn3D

Values allowed are 6, 28 and 74. This parameter defines the 3D local neighborhood used to run watershed for generating super-voxels. Values of 6 and 28 define traditional 3D neighborhoods, whereas a value of 74 generates cubes of 5 x 5 x 3 around each point to address the anisotropy of the point-spread-function typically encountered in 3D microscopy images.

estimateOpticalFlow

Values allowed are 0, 1 and 2. This parameter activates/deactivates the use of optical flow calculations between time points for the purpose of compensating for large object displacements. A value of 0 deactivates optical flow calculations. A value of 1 indicates that pre-calculated optical flow files are available and can be used to apply local motion displacements between time points. A value of 2 indicates that the program will calculate optical flow on-the-fly, using a routine provided by the user.

maxDistPartitionNeigh

Floating point positive number. It is only used if *estimateOpticalFlow* is equal to 2 and the routine called is the one described in F. Amat *et al*., "Fast and robust optical flow for time-lapse microscopy using super-voxels" (*Bioinformatics*, 2013). The parameter provides the maximum allowed distance (in pixels) between super-voxels for them to be considered neighbors in the calculation of the optical flow (coherence constraint).

deathThrOpticalFlow

Integer number. If positive, the optical flow module will be activated automatically when the number of deaths at a specific time point is larger than the value of this parameter. Usually, when large motions occur (larger than one nucleus diameter from one time point to the next), many Gaussians in the model disappear, since the solution from the previous time point is not well-suited for initialization of the current time point. Thus, monitoring deaths can be used as a trigger to activate optical flow only when needed.

minNucleiSize

Positive integer number. If the number of voxels belonging to a super-voxel is less than *minNucleiSize* the super-voxel is deleted. This parameter is useful to delete spurious super-voxels representing background intensity.

maxNucleiSize

Positive integer number. This parameter defines the maximum allowed size (in voxels) of a super-voxel after applying Otsu's threshold. If Otsu's threshold generates an object larger than *maxNucleiSize* the threshold is increased until the objet size falls below *maxNucleiSize*.

maxPercentileTrimSV

Floating point number between 0 and 1. This parameter defines the maximum allowed percentage of voxels in a super-voxel belonging to foreground. If Otsu's threshold generates an object larger than *maxPercentileTrimSV* the threshold is increased until the percentage of foreground voxels falls below below *maxPercentileTrimSV*.

conn3DsvTrim

Values allowed are 6, 28 and 74. The final super-voxel generated after trimming the initial super-voxel partition to detect foreground and background is guaranteed to have this connectivity.

maxNumKNNsupervoxel

Positive integer number. This parameter defines the maximum number of nearest neighbors to consider for each super-voxel when building the spatio-temporal graph for tracking. The shorter the nuclear displacement between time points, the lower the parameter value can be.

maxDistKNNsupervoxel

Floating point positive number. This parameter defines the maximum distance (in pixels) to consider for each super-voxel when building the spatio-temporal graph for tracking. The shorter the nuclear displacement between time points, the lower the parameter value can be.

thrSplitScore

Floating point number. If 3D Haar features are used for cell division classification, this parameter sets the threshold for the machine learning classifier to decide whether a cell is dividing or not. The higher the threshold, the fewer divisions are going to be called by the

classifier. In order to activate 3D Haar features, the code needs to be recompiled with preprocessor directive CELL_DIVISION_WITH_GPU_3DHAAR.

thrCellDivisionPlaneDistance

Floating point positive number, defining the threshold of a feature for disregarding cell division false positives. The feature calculates the distance (in pixels) between mother cell and the midplane defined by the two daughter cells. If the value is above *thrCellDivisionPlaneDistance*, the cell division is considered a false positive and the linkage between mother and furthest daughter is removed. The default value of 3.2 was determined empirically from a small training set to maximize precision while maintaining a high recall of true cell divisions. The lower the value, the lower the recall of cell divisions and the higher the precision.

Nature Methods: doi:10.1038/nmeth.3036

Supplementary Software 2 | Modified CATMAID module for visualizing image and cell lineage data, manually curating cell lineage data and annotating cell lineage reconstructions

The file "Supplementary_Software_2.zip" provides all files necessary to set up a CATMAID server for image data and cell lineage data visualization as well as for editing cell lineage reconstructions. This code is a branch from the master repository found at http://catmaid.org/ and users can simply follow the detailed documentation located in the CATMAID master repository to set up the server. However, we recommend downloading the source code by using the following GIT command to obtain the latest version of the CATMAID server for cell lineage reconstructions:

"*git clone https://fernandoamat@bitbucket.org/fernandoamat/catmaid_5d_visualization_annotation.git*"

Importantly, this branch of the CATMAID server contains a *TGMM importer* option on the administrator site that allows importing the output data of the automated TGMM segmentation and tracking framework (**Supplementary Software 1**) into CATMAID.

Nature Methods: doi:10.1038/nmeth.3036

Supplementary Data 1 | Cell lineage reconstruction of early *Drosophila* embryonic nervous system development

The Matlab file "NervousSystem.mat" contains the complete curated and annotated cell lineage data set from our reconstruction of early *Drosophila* embryonic nervous system development (**Fig. 5, Supplementary Videos 24-28**). We included the following variables:

- *trackingNeuroblastCurated:* main data array containing all cell lineage information. Each row represents a nucleus at one time point. Contents of the columns are detailed below.
- *stackResolution:* three-dimensional vector containing information about the pixel size (in micrometers) along each direction. This information is required to convert pixel coordinates to positions in micrometers.
- *stackSize:* three-dimensional vector containing information about the size of each three-dimensional image stack at each time point.
- *tagcell:* cell array of length ten, required to convert the integer identities of the annotated neuroblast types to readable labels.
- *mapLineageToNBType:* array for associating each cell lineage with a neuroblast type. The first column contains a unique set of integer IDs that can be mapped to the first column of *trackingNeuroblastCurated*. The second column contains an integer number from 1 to 10 that can be mapped to the neuroblast identity using *tagcell*.

The variable *trackingNeuroblastCurated* contains six columns with the following information for each nucleus data point:

1. Unique ID (large integer number) in the CATMAID database for identifying each nucleus data point.
2. *x* coordinate (in pixels) of the nucleus.
3. *y* coordinate (in pixels) of the nucleus.
4. *z* coordinate (in pixels) of the nucleus.
5. Parent ID in the cell lineage tree. This ID is equal to -1 if the data point is a root node. Otherwise, it contains the unique ID of the parent (see description of column 1).
6. Time point. Note that the reconstruction starts at time point 0, which corresponds to 2.9 h AEL (shortly before onset of gastrulation). The time interval is 30 seconds.

Nature Methods: doi:10.1038/nmeth.3036

Supplementary References

1. Bishop, C.M. Pattern recognition and machine learning. (Springer, 2007).
2. Amat, F., Myers, E.W. & Keller, P.J. Fast and robust optical flow for time-lapse microscopy using super-voxels. *Bioinformatics* **29**, 373-380 (2013).
3. Broadus, J. et al. New neuroblast markers and the origin of the aCC/pCC neurons in the Drosophila central nervous system. *Mech Dev* **53**, 393-402 (1995).
4. Kausler, B.X. et al. A discrete chain graph model for 3D+t cell tracking with high misdetection robustness. *ECCV* **7574**, 144-157 (2012).
5. Giurumescu, C.A. et al. Quantitative semi-automated analysis of morphogenesis with single-cell resolution in complex embryos. *Development* **139**, 4271-4279 (2012).
6. Tomer, R., Khairy, K., Amat, F. & Keller, P.J. Quantitative high-speed imaging of entire developing embryos with simultaneous multiview light-sheet microscopy. *Nat Methods* **9**, 755-763 (2012).
7. Li, K. et al. Cell population tracking and lineage construction with spatiotemporal context. *Med Image Anal* **12**, 546-566 (2008).
8. Fearnhead, P. Particle filters for mixture models with an unknown number of components. *Stat Comp* **14**, 11-21 (2004).
9. Figueiredo, M.A.T. & Jain, A.K. Unsupervised learning of finite mixture models. *PAMI* **24**, 381-396 (2002).
10. Otsu, N. A threshold selection method from gray-level histograms. *IEEE Trans Syst Man Cy* **9**, 62-66 (1979).
11. Kumar, N.S.L.P., Satoor, S. & Buck, I. Fast parallel expectation maximization for Gaussian mixture models on GPUs using CUDA. *IEEE HPCC*, 103-109 (2009).
12. Seiffert, C., Khoshgoftaar, T.M., Van Hulse, J. & Napolitano, A. RUSBoost: improving classification performance when training data is skewed. *Int C Patt Recog*, 3650-3653 (2008).
13. Bettadapura, V., Schindler, G., Plötz, T. & Essa, I. Augmenting Bag-of-Words: data-driven discovery of temporal and structural information for activity recognition. *CVPR* (2013).