OrganoidTracker: efficient cell tracking using machine learning and manual error correction

Rutger N.U. Kok, Laetitia Hebert, Guizela Huelsz-Prince,
Yvonne J. Goos, Xuan Zheng, Katarzyna Bozek,
Greg J. Stephens, Sander J. Tans and Jeroen S. van Zon

Abstract

Time-lapse microscopy is routinely used to follow cells within organoids, allowing direct study of division and differentiation patterns. There is an increasing interest in cell tracking in organoids, which makes it possible to study their growth and homeostasis at the single-cell level. As tracking these cells by hand is prohibitively time consuming, automation using a computer program is required. Unfortunately, organoids have a high cell density and fast cell movement, which makes automated cell tracking difficult. In this work, a semi-automated cell tracker has been developed. To detect the nuclei, we use a machine learning approach based on a convolutional neural network. To form cell trajectories, we link detections at different time points together using a min-cost flow solver. The tracker raises warnings for situations with likely errors. Rapid changes in nucleus volume and position are reported for manual review, as well as cases where nuclei divide, appear and disappear. When the warning system is adjusted such that virtually error-free lineage trees can be obtained, still less than 2% of all detected nuclei positions are marked for manual analysis. This provides an enormous speed boost over manual cell tracking, while still providing tracking data of the same quality as manual tracking.

1 Introduction

Increasingly, scientists are looking at development and homeostasis of organs at the single-cell level [1–3]. The invention of organoids, which are mini-organs grown in vitro that mimic the cellular structure and some of the functionality of an organ [4], has made it possible to study tissues at a level of detail that was not possible before. Nowadays, organoids are able to mimic an increasing number of organs with an improving accuracy and complexity [5, 6]. The ability to follow individual cells would allow one to directly reveal the the historical genealogical relations between cells, their changing spatial organization, their cell divisions and deaths, and may

be combined with markers for cell differentiation and other vital cellular processes. Elucidating these cellular spatio-temporal dynamics will be central to understanding organ development and tissue homeostasis, as well as associated pathologies.

However, the automated tracking of cells within organoids presents specific challenges that have so far been difficult to overcome. Since the 1960's, researchers have used algorithms to automatically detect cells from microscopy images [7]. Today, a large number of cell trackers exist [2, 8, 9]. For example, for tracking cells in fruit flies, zebrafishes and mouse embryos there are existing software packages [10]. An important class of cell trackers uses a set of rules to detect cells, and in doing that they have to make certain assumptions about the system. Here, we use a different approach making use of supervised machine learning. Convolutional neural networks have already been used successfully for microscopy cell images in 2D [11–14] as well as in 3D [15–17]. In our approach, a convolutional neural network is trained to find nucleus center positions in 3D microscopy images.

Here, we consider intestinal organoids, which are organoids that model the small intestinal epithelium [18]. This intestinal epithelium is known to rapidly renew itself; in mammalians every five days the stem cells replace almost all of the differentiated cells. This results in a flow of cells from the stem cell compartment (crypt) to the compartment where fully differentiated cells die (villus). Intestinal organoids contain all cell types from the intestinal epithelium [19], making them a valuable model system. By tracking all individual cells in these intestinal organoids we can gain new insights into the dynamics of the intestinal epithelium. Not only can we directly observe divisions, movement and cell death, we can also study differences in fate of cells that are close together in space or related by lineage. This gives us insight in how and when cells make decisions. Moreover, by looking at the positions of the cells, it is also possible to get an estimate of the crowdedness and curvature of the tissue over time.

Automated cell tracking in intestinal organoids is complicated by several factors. The organoids have a high nuclei density, fast cellular movement and intrinsic differences in the shape and intensity of nuclei. Moreover, the requirement to limit phototoxicity results in low-contrast images. As can be seen in Fig. 1, the images have visually overlapping nuclei, a low intensity further away from the objective and a low resolution in the z-direction. All of these factors make it challenging to automatically recognize nuclei. Nevertheless, for our application high accuracy is necessary. One single mistake at a single time point can already cause that cell to become wrongly connected in time, which makes conclusions based on the history of the cell unreliable. Therefore, manual curation of the automatic tracking data is required to eliminate such tracking mistakes [20–22], ideally in a way that allows the user to systematically go through all tracking data that is considered improbable by the program.

Here, we have developed a cell tracker plus manual tracking procedure using a convolutional neural network, a min-cost flow solver [23] and an algorithm that

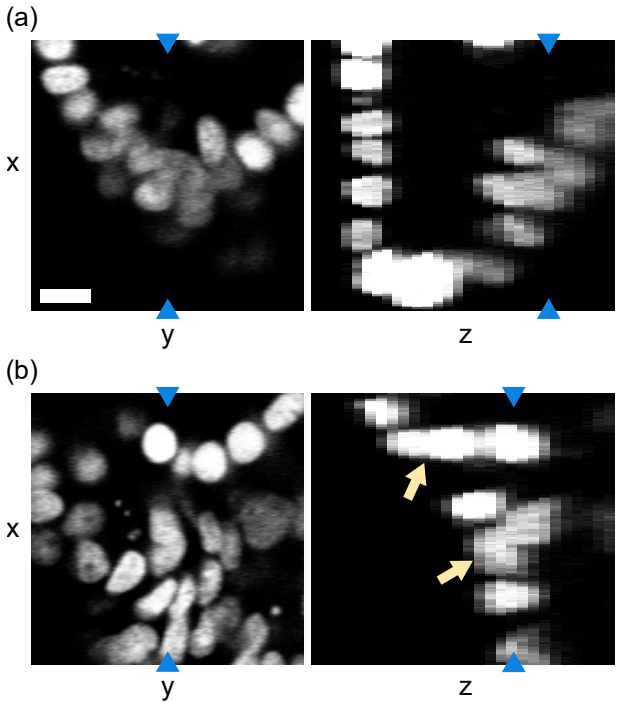

Figure 1: Example of a tracked organoid. Microscopy image slices in the xy and xz planes using H2B-mCherry to visualize the cell nuclei. Blue arrows indicate that both image slices in the same panel represent the same pixels. Note that the resolution in the z-direction is lower. The images show that there is both variation in intensity and visual overlap in the nuclei (indicated by the orange arrows), especially in the z-direction. The scale bar is 10 µm.

detects likely errors in the tracking data. Using this approach, we were able to obtain tracking data of the same quality as manual data, though vastly faster.

2 Results

2.1 Overview of cell tracking approach

We are interested in tracking cells in time lapse videos of growing intestinal organoids. In the videos, we observe the nuclei of the cells which was made fluorescent using H2B-mCherry. The videos are up to 65 hours in length with the time resolution set to 12 minutes, resulting in videos of 84 to 325 time points. Each pixel is 0.32 x 0.32 µm in size, and the separation between two z-layers is 2 µm. The images have irregular fluorescence intensity, which is not only caused by intrinsic variations of nucleus fluorescence, but also by the location of the cells: some parts of the organoids are oversaturated, while parts deeper inside the organoids are undersatu-

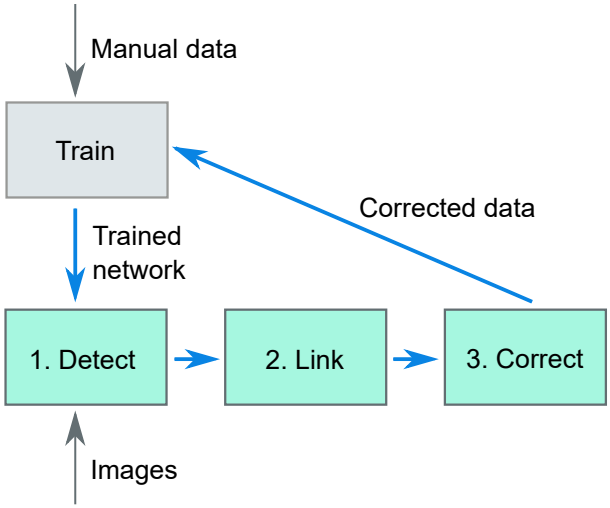

Figure 2: Overview of the tracking software. Using ground-truth data of nucleus locations in microscopy images (obtained from manual tracking) a convolutional neural network is trained. This trained network is then used to detect (step 1) cells in new images. The detections are then linked over time (step 2), after which the user manually corrects the output with the help of an error checker (step 3). It is possible to use the corrected data can then be used as additional ground-truth data to improve the training of the convolutional neural network, which leads to better results for the next organoid that needs to be tracked.

rated. Especially in the oversaturated parts of the images, the nuclei can appear to visually overlap.

The imaged area of the organoid typically contains over 200 cells, which makes manual tracking infeasible. Instead, we had previously obtained (to be submitted) manual tracking data for a part of the imaged area, namely the crypt region. For that region, the center position of about 70 nuclei was recorded manually. The center position of each nucleus was gauged by eye and may therefore be a few pixels off. Additionally, links were created between the same nuclei in different time points, so that the nuclei could be tracked over time. In total, manual tracking was done for 2826 time points, coming from 17 different organoids.

In our cell tracking approach, we use a convolutional neural network for cell detection, a min-cost flow solver [23] for linking cells over time and a small set of rules to detect improbable cell tracking data that the user needs to verify and correct. This complete approach is displayed schematically in Fig. 2. Step 1 is formed by the convolutional neural network: using raw images, it outputs images with peaks at the center location of the nuclei. This network needs to be trained using ground-truth data, which done using the manual tracking data. Then, in Step 2 the cells are linked using a min-cost flow solver that can handle cell divisions [23], which optimizes for the smallest cell movement in between time points, for the smallest changes in nucleus volume in between time points and for the longest cell tracks. Finally, in Step 3 the program reports all cell tracks that it sees as suspicious based on a small set of rules, which the user needs to inspect. For us, this workflow resulted in cell tracks of the same quality as manual tracking though obtained vastly faster.

2.2 Step 1: automated cell detection

The nucleus center positions from our manually annotated tracking dataset were used to train a fully convolutional neural network, where the input and output are 3D images of the same shape. We trained the network with 3D microscopy images as the input images. Here, each image represents a single time point. The network architecture follows a standard U-Net [24] type architecture with all the convolutional layers replaced by the CoordConv layer [25] adapted for 3D images (Fig. 3a). The CoordConv layer concatenates the absolute location of pixels to each convolution input. Compared to the simple convolution layer which is translation equivariant, this layer is useful for cases where the absolute position of the pixels is important to the task.

For the output images of the network, we want images where the brightness of each pixel corresponds to the chance of that pixel being a nucleus center. As multiple pixels close together can have a high likeliness of being a nucleus center, this results in mostly dark output images with bright spots at the locations of the nucleus centers. To make the neural network produce such output images given microscopy images at the input images, it needed to be trained on pairs of microscopy images

and artificially generated output images. The generated output images are black images that contain bright, Gaussian-shaped spots with standard deviations of 2 px at the locations of the nuclei (Fig. 3b). The locations of the nucleus centers were determined from the manual tracking data. The Gaussian shape of the spots acknowledges the fact that there is an uncertainty in the exact location of the nucleus center. Because the resolution in the z axis of the images is lower than in the x and y axes, the Gaussians for the labeled volumes are in 2D and don't cross several xy-planes.

We trained the network on 9 organoids, containing 1388 time points in total. For training, columns of 64x64 px from random locations within the images were used. This made the training algorithm use less memory compared to using the full 512x512 px, and also randomized the x and y coordinates of the nuclei, but unfortunately it does prevent the network from using information in the images that is located further away from the nucleus. To make the training more robust, random perturbations were applied to the training images [24]. The images were randomly flipped in the x, y and/or z direction, rotated a random number of degrees (0-360), scaled to 80%-120% of their original size, made brighter or darker with at most 10 % of the maximum intensity in the image and finally the contrast was adjusted according to $I_{new} = (I - \langle I \rangle) \cdot c + \langle I \rangle$ where I_{new} is the resulting intensity of a pixel, I the original intensity of that pixel, $\langle I \rangle$ the mean intensity of all pixels in the image and c the contrast factor, which varied from 0.5 to 1.5. We use a weighted mean squared error as the loss function between the network output and the labeled volume. Because the labeled volumes were mostly composed of zeroes, we gave more importance to the Gaussian spots by applying weights that correspond to the percentage of non zero values in the labeled volume.

Once the network was trained, it generated output images that show where the nucleus centers are located (Fig. 3c). Each pixel in the 3D image represents the probability of that pixel being the nucleus center, resulting in a probability distribution with small peaks at the location of the nucleus centers. We interpolated linearly the empty space between the slices so that the resulting volume had the same resolution in the z axis as in x and y. This allows us to apply a a 3D peak detection algorithm (`peak_local_max` in scikit-image 1.1.0 [26]) to detect these local maxima in the interpolated 3D volumes. The resulting 3D coordinates are considered to be the locations of the nucleus centers in the full 3D volume. We then map back these coordinates to the closest image slice.

To evaluate the performance of the network, we needed to know how many of the detections are true positives or false positives, and how many false negatives there are. To do this, we compared the automatic tracking data to manual tracking data of 8 organoids (1438 time points) that were not used for training the neural network. Because these images are from separate organoids, we can use this tracking data to evaluate the model generalization. One challenge in the performance evaluation was that it is difficult to measure the number of false positives from the neural network,

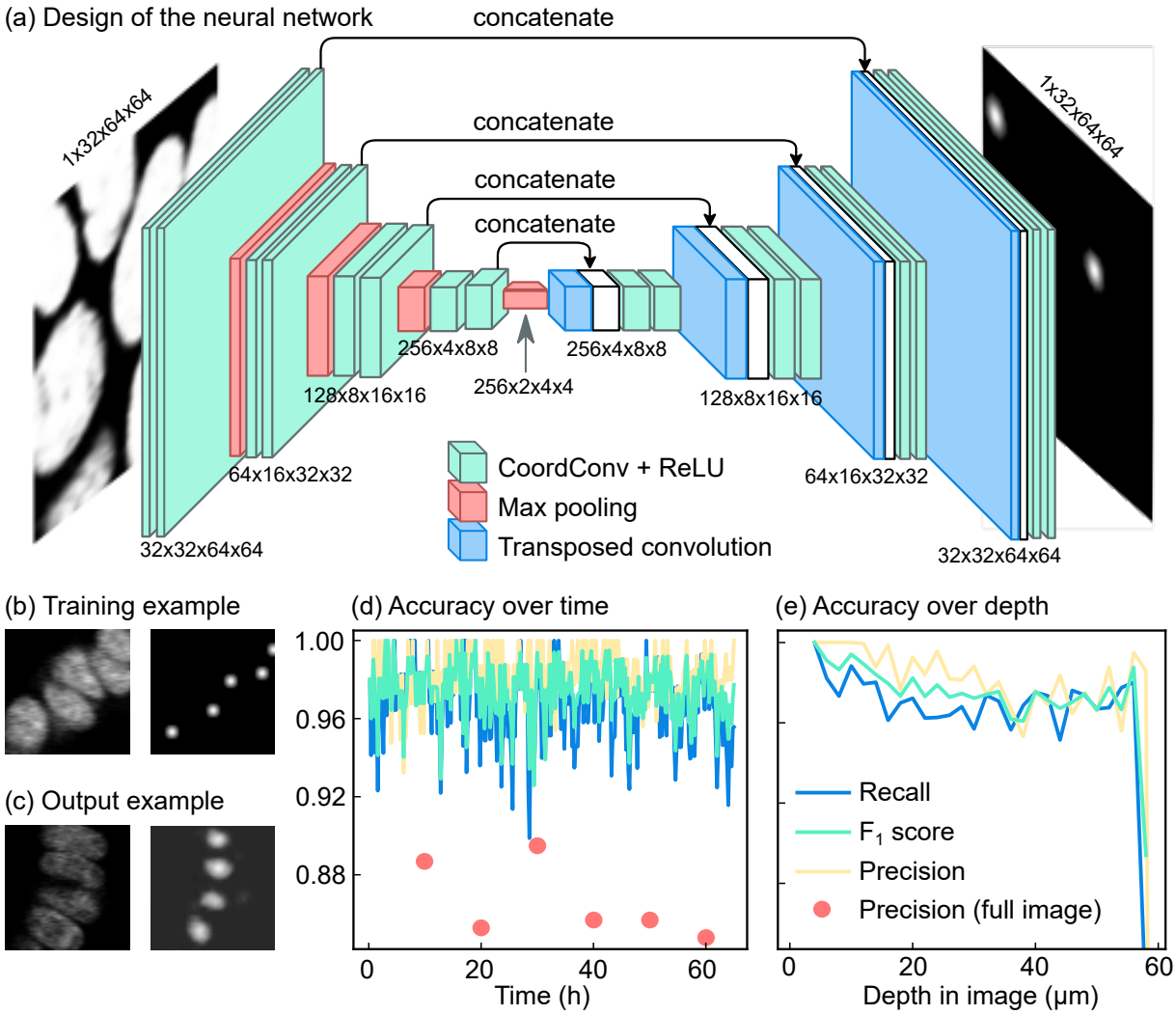

Figure 3: Automated detection of cell nuclei by a convolutional neural network. **(a)** Schematic overview of the network. The network is a standard convolutional network with absolute pixel coordinates added as input values. This makes it possible to adjust its detection network both for cells deep inside the organoid and cells near the objective. **(b)** Example of training data. On the left a slice of a 3D input image, on the right Gaussian predictions created from manual tracking data. The brightness of each pixel represents the likeliness of that pixel being the nucleus center. **(c)** Example of a network prediction, which shows the probability of each pixel being a nucleus center, according to the neural network. **(d)** Accuracy of the neural network over time compared to manual tracking data obtained in a subsection of a single organoid. For six time points (annotated with red circles) tracking data was compared to manually annotated data for an entire organoid. In this case, we observed a lower precision. For this organoid, the recall is 0.97 while the precision is 0.98. **(e)** Same data as in Panel d, now plotted over z. The accuracy of the network drops at the deepest imaged parts, likely because part of each nucleus falls outside the image stack here.

as only 30% to 40% of all cells visible in the images were tracked. Therefore, at any location where the neural network reports the presence of a nucleus while the manual annotations do not, we cannot a priori be sure whether there is a false positive or whether that part of the image was simply not manually annotated. To overcome, we used the following approach.

Any nucleus center detected by the neural network was assigned to the closest nucleus center from the manually tracking data, under the condition that the distance was no longer than 5 µm. Every nucleus center cannot have more than one assignment. Each successful assignment was a true positive. Then, any manually tracked nucleus center that was left with no assignments became a false negative. Finally, any nucleus center from the neural network that was left with no assignments was regarded as a false positive if it was within 5 µm from a manually tracked nucleus center, otherwise it was rejected. This ensured that misdetections within the manually tracked area were still detected.

We measured three values to quantify the performance of the network: the precision, recall and the F_1 score. The precision is the fraction of the nucleus detections done by the neural network that were correct. Mathematically, it is defined as

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

with TP the number of true positives and FP the number of false positives. The recall is the fraction of nucleus detections in the ground truth that was detected by the neural network, and it is defined as

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

with FN the number of false negatives. For a neural network it is important that both the precision and the recall are high. In one extreme example, if a neural network would mark almost every pixel as a nucleus center, it would have a high recall, but a precision of almost zero. In the other extreme, if the neural network would only find back the most obvious nuclei, it would have a high precision but a low recall. To express the quality of a neural network as a single number, the harmonic average of the precision and recall was calculated, which is known as the F_1 score:

$$\text{F}_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{3}$$

The precision turned out to be 0.98 with a standard deviation of 0.007, which means that 98% of the detections done by the neural network were consistent with the manual data. The recall turned out to be on average 0.963 with a standard deviation of 0.017, which means that 96% of all manually tracked cells were detected by the

neural network. Based on these values for the precision and the recall, the F_1 score of the network is 0.97.

Fig. 3d displays the accuracy over time, which shows that the accuracy did not degrade over time, when the organoids have grown larger. As we can see in Fig. 3e, the accuracy is the highest close to the objective and then drops before levelling off from 20 µm. In the deepest part of the organoid, the accuracy drops further. This is because the full nucleus is no longer visible; part of the nucleus now falls outside of the image stack.

The manually tracked region may not be representative for the whole organoid as there are also untracked regions which mostly consist of fluorescent material coming from dead cells. To check the performance of the network for the organoid as a whole, for six time points all nuclei were manually annotated in the entire image stack. This resulted in 1318 nuclei being annotated. The recall for these images was still 0.96, but the precisions, shown in Fig. 3d as red dots, decreased to 0.87. This was mainly because dead cell material was recognized by the neural network as if it were a live cell nucleus. To obtain correct cell tracking data, these spurious annotations will need to be filtered out later in the tracking process, either during the automatic linking step or during the manual error correction step.

With on average 96% of all nuclei correctly recognized, we have an accurate way to detect nucleus centers. The next step is to link the nucleus detections at different time points together, such that nuclei can be followed over time.

2.3 Step 2: linking of cell positions

To follow cells over time, the locations of the same cell nucleus at different time points need to be linked together. A link goes from a detected nucleus center at time point t to the same nucleus center imaged at time point $t + 1$. Normally, every nucleus has one link to the next time point and one link to the previous time point. However, in case of a division a nucleus will split into two nuclei and therefore the nucleus will also have two links to the next time point. A straightforward way to create these links is to always assume that the nearest detected nucleus in the previous time point represents the same nucleus; this is called nearest neighbor linking. By going back in time, in theory we get detection of cell divisions for free: if two nuclei at time point $t + 1$ both have the same, single nucleus at time point t as their closest nucleus, a division is generated.

Unfortunately, nearest-neighbor linking does not provide us with accurate lineage trees. We can see in Fig. 4a that nearest neighbor linking creates unrealistically short cell cycles. Moreover, although rare, there is nothing that prevents a mother cell from having three or more daughters. Clearly, the assumption of the nearest nucleus always being the same nucleus does not hold. For this reason it is necessary to let the linking algorithm also consider links to nuclei further away than the nearest

neighbor. In our method we consider all links that are at most two times further away in distance than the nearest neighbor. For most detected cell positions, this results in multiple linking options for creating links (Fig. 4c). Each of those options is given a score (an energy), of which the sum of all created links is minimized. This minimization is carried out using a min-cost flow solver adapted for tracking of dividing cells, developed by Haubold *et al.* [23]. Here, we used the following rules:

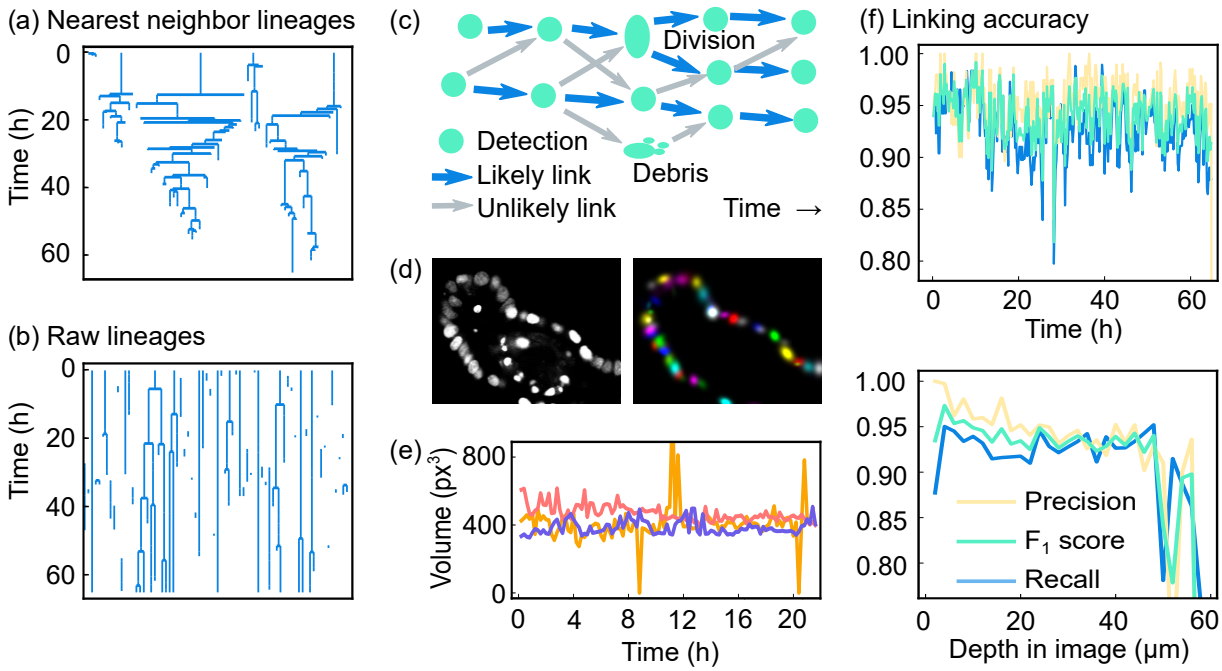

Figure 4: Overview and results of the linking algorithm. (a) Lineage trees obtained by linking using the nearest-neighbor method. **(b)** Raw lineage trees obtained by our linking algorithm. **(c)** Example of a network of links. There are many possibilities to link the cell detections in different time points together. Of all possible links (displayed as arrows), the most likely ones (displayed as blue arrows) are selected using a scoring system. **(d)** Original microscopy image and Gaussian fit. **(e)** Nucleus volume of three nuclei extracted from a Gaussian fit. The volume ($V = \mathrm{Cov}(x,x) \cdot \mathrm{Cov}(y,y) \cdot \mathrm{Cov}(z,z)$) is calculated for three non-dividing cells. The outliers are caused by errors in the Gaussian fit. **(f)** Accuracy of the linking algorithm presented here for a single organoid, over time and over the image depth. Note that the precision is higher than for nearest-neighbor linking.

1. *An ordinary link is scored as score* $= \Delta X + \sqrt[3]{\Delta V}$*, with* ΔX *the change in position* μm*, and* ΔV *the change in volume of the nucleus in* μm^3 This ensures that links with the lowest change in position and volume are favored.

2. *Every track end is given a penalty of 1.* This favors creating longer tracks

over multiple shorter tracks, which allows for the occasional long-distance link.

3. *Every track start is given a penalty of 1.* Again, this favors the creation of longer tracks.

4. *Every cell position that is ignored by the linking algorithm results in a penalty of 1.* This ensures that as much as possible of the data from the neural network is linked. If this rule would not exist, a globally optimal score could be reached by not creating any links at all.

5. *Cell divisions have a scoring system based on the change in volume and intensity*. This scoring system is described in the supplementary information. A score below 1 means that a cell division here is unlikely, in which case no division is allowed here.

6. *If the cell division score is at least 1, the score from rule 1 is halved.* In the time around a cell divisions, the cell nuclei are expected to quickly move and change their volume, so it is appropriate to lower the penalty for position and volume changes.

The scores resulting from rules 1 to 5 are multiplied by a factor specified by the user of the software. Here, these factors are 20, 100, 150, 150 and 30, respectively. By changing these values, the scoring system can be tuned. For example, if the neural network accidentally detects dead fluorescent material as living cells, resulting in many short paths, this can be fixed by increasing the penalty for appearances and disappearances of cells. In this way, the high penalties at the beginning and end of the track together are larger than the penalty of ignoring those detected nucleus positions. As a result, the track is not created at all. To give another example, if the cells in a system tend to move slowly, the movement penalty can be increased, so that the likeliness of creating long-distance links is decreased. We used this in our system, to avoid creating a link to a neighbor nucleus when the correct nucleus was not detected by the neural network. For us this works well, because a cell track that abruptly ends will easily be found if the user rechecks all track ends, while a switch in cell identity resulting from a link to another cell is harder to detect.

For evaluating rule 1 and 5 it is necessary to know the volume of the nuclei. This volume is estimated using a 3D multivariate Gaussian fit. While nuclei do not always have a Gaussian-like shape, they can still be approximated by a Gaussian function in order to get an estimate of their volume [27]. Because we only want to fit Gaussian functions to living nuclei and not to fluorescent material coming from dead cells, we cannot fit the image as a whole. Instead, we need to separate dead cell material from the living nuclei, and only fit Gaussian functions to the latter. This is done by first performing a segmentation to divide the image into clusters, each containing zero, one or multiple cell positions detected by the neural network. The segmentation starts by performing an adaptive threshold on the original image

I_{image} to separate the foreground and background, resulting in a mask named I_{mask}. This mask is then improved by first removing all pixels from I_{mask} that in the original image I_{image} have a negative iso-intensity Gaussian curvature [27], which means that all pixels of lower intensity in between two places with higher intensity are removed. This is expected to remove the pixels in between two nuclei. Next, all holes in the mask are filled, resulting in the mask named $I_{mask,filled}$. On this mask, three rounds of erosion are performed with a 3x3x1 cross-shaped kernel on the mask. This step makes the area of foreground smaller, which makes the foreground fall apart into separate clusters. For that reason, this eroded mask was called $I_{mask,eroded}$. From the convolutional neural network, we know how many cells there are in each cluster and where their nucleus centers are. A Gaussian fit is then performed on $I_{mask,filled}$, which was the mask before erosion was applied. One Gaussian function is fit for each nucleus position and each Gaussian function is centered around the position given by the neural network, using a default intensity and a default covariance matrix. The following function is used to fit a single nucleus:

$$G\left(\vec{x}\right) = a\, e^{(\vec{x}-\vec{\mu})^T\, \mathbf{\Sigma}^{-1}\, (\vec{x}-\vec{\mu})} \tag{4}$$

Here, a is the maximum intensity of the function, μ is a vector representing the mean and $\mathbf{\Sigma}$ is the covariance matrix. Together, these three variables describe the intensity, the position and the shape of the cell nucleus, respectively. $(\vec{x}-\vec{\mu})^T$ denotes that we are taking the transverse of $(\vec{x}-\vec{\mu})$, which results in a row vector. These functions are fitted to a blurred version of the original image, $I_{image,blurred}(\vec{x})$, so that the nuclei have a more Gaussian-like shape. For all pixels $\vec{x}$ in the cluster, the quadratic difference between that image and the outcome of the Gaussian functions describing the nuclei is minimized:

$$\text{difference} = \sum_{\vec{x}} \left(I_{image,blurred}\left(\vec{x}\right) - \sum_i G_i\left(\vec{x}\right) \right)^2 \tag{5}$$

In Fig. 4d, we can see an example of the Gaussian fit. To show the consistency of the Gaussian fit over time, we have plotted a volume ($\text{Cov}(x,x)\cdot\text{Cov}(y,y)\cdot\text{Cov}(z,z)$) over time for three arbitrary cells in Fig. 4e. As can be seen in that figure, the Gaussian fit predicts a volume over time without significant noise. However, sometimes the fit predicts a volume more than twice the median volume, and sometimes only predicts a very low volume. This can happen if the fit of two nuclei results in one large Gaussian shape that covers both nuclei, and one very small Gaussian shape.

In Fig. 4b the raw lineage trees obtained from the linking algorithm are displayed. By comparing both these lineage trees and lineage trees obtained from nearest-neighbor linking (Fig. 4a) to lineage trees from manual tracking data (Fig. S1) it is clear that the method proposed here is more accurate than nearest-neighbor linking.

To quantitatively measure the accuracy of our method, we need to calculate the precision (Equation 1) and recall (Equation 2). For this it is necessary to define true positives, false positives and false negatives in terms of links. We considered all detected nuclei at most 5 µm from the location of the manually annotated nucleus. If any of these nearby nuclei centers at time point t had a link to any of the nearby nuclei centers at time point $t + 1$, this was considered to be the same link and hence a true positive. If no link existed between the nearby positions, this was considered to be a false negative. Naturally, every link in the manual tracking data can only be assigned to one corresponding link the automatic tracking data. Every remaining link in the automatic tracking data with no corresponding link in the manual tracking data was therefore in principle a false positive. However, as not the whole organoid was tracked, links were rejected if, at both time points, they did not have any nucleus centers in the manual tracking data within 5 µm.

Using this approach, we calculated the precision, recall and F_1 score for the same 8 organoids (1438 time points) as used for evaluating the detection performance. The recall of all organoids is 0.91 with a standard deviation of 0.044, the precision is 0.94 with a standard deviation of 0.020 and the F_1 score is therefore 0.93. Note here that the recall score is limited by the recall of the neural network: if the neural network fails to detect a nucleus, the two links of that nucleus will automatically also not be created. As can be seen in Fig. 4f, the results are relatively constant over time, but not over depth: the accuracy is worse near the edge of the imaged area.

2.4 Step 3: manual error corrections

While the lineages trees obtained from the previous step already show promising results, still the lineage trees are not correct: divisions are missed, cell are switched in identity, cell tracks end prematurely and cell divisions are created where there are none. Based on these results, manual error correction is needed to obtain reliable tracking data. Time-wise, it would not be practical to revisit each nucleus at every time point. For this reason, the program flags all nuclei that violate one the following rules, so that the user can manually make corrections:

1. *Cells must have a link to the previous and to the next time point.* This detects cells that are appearing out of nowhere and cells disappearing into nothing. These cases usually indicate that a cell was not detected at a time point, or that a cell was detected where there was none. In case the cell actually dies, or if the cell goes out of the microscope view, the user can ignore this rule violation.

2. *At least 10 hours must have passed before a cell divides again.* A violation of this rule indicates that a cell identity switch has happened. The exact number of hours can be changed.

3. *Cells must not have moved more than 10 μm in between two time points.* Such a quick movement can point to a switch in cell identity. The exact number of micrometers can be changed.

4. *The nucleus volume of a dividing cell must be larger than the volume of the daughter cells combined.* This rule helps identifying misdetected divisions.

5. *The volume of a nucleus may not shrink more than 3 times.* This rule helps finding switches in cell identity and missed cell divisions.

6. *Two cells may not merge together, and cells may not have more than two daughters.* Even though linking does not allow these situations, this rule can be violated if a person makes a mistake while manually correcting the tracking data.

Any violations of the above rules are automatically flagged by an error checker runs as the last step of the linking process. Additionally, when manually correcting the data, the error checker immediately checks every edit done by the user. The strictness of the rules can be varied, with stricter rules leading to more false warnings. For our organoids 1 to 2 % of all cell detections needed to be manually verified.

2.5 Analysis of tracking data

In Fig. 5 we show the results of this whole-organoid tracking approach. First, this approach makes it possible to connect lineage dynamics to spatial position of cells in an organoid. In Fig. 5a 2D slices of microscopy images of an organoid are displayed, with its 3D reconstruction shown in Fig. 5b. In the reconstruction, the cells have been colored by lineage: selected cells were colored based on which cell they originated from in the first time point. In Fig. 5c lineage trees are shown of the same organoid and using the same coloring. We can see that cells of the same lineage tree tend to stay connected in space, although cells of multiple lineage trees do intermingle. In addition, we can see that the largest lineage trees reside at the lowest position in the crypt region, which is the growing protruded region in Fig. 5b.

Next, we used our tracking data to study dynamics of cell divisions in space and time. We observed that almost all cell divisions are happening in the crypt (supplementary video), which is consistent with the literature [28]. We could see that at the bottom of the crypt, the number of divisions is the largest. However, there were a few cells at the bottom of the crypt that, in sharp contrast to their neighbors, did not divide, or divided only once. These are likely to be Paneth cells, which are differentiated cells that reside in the stem cell compartment [29]. In Fig. 5d we saw that divisions were initially synchronized, while later in time this synchronization was less apparent.

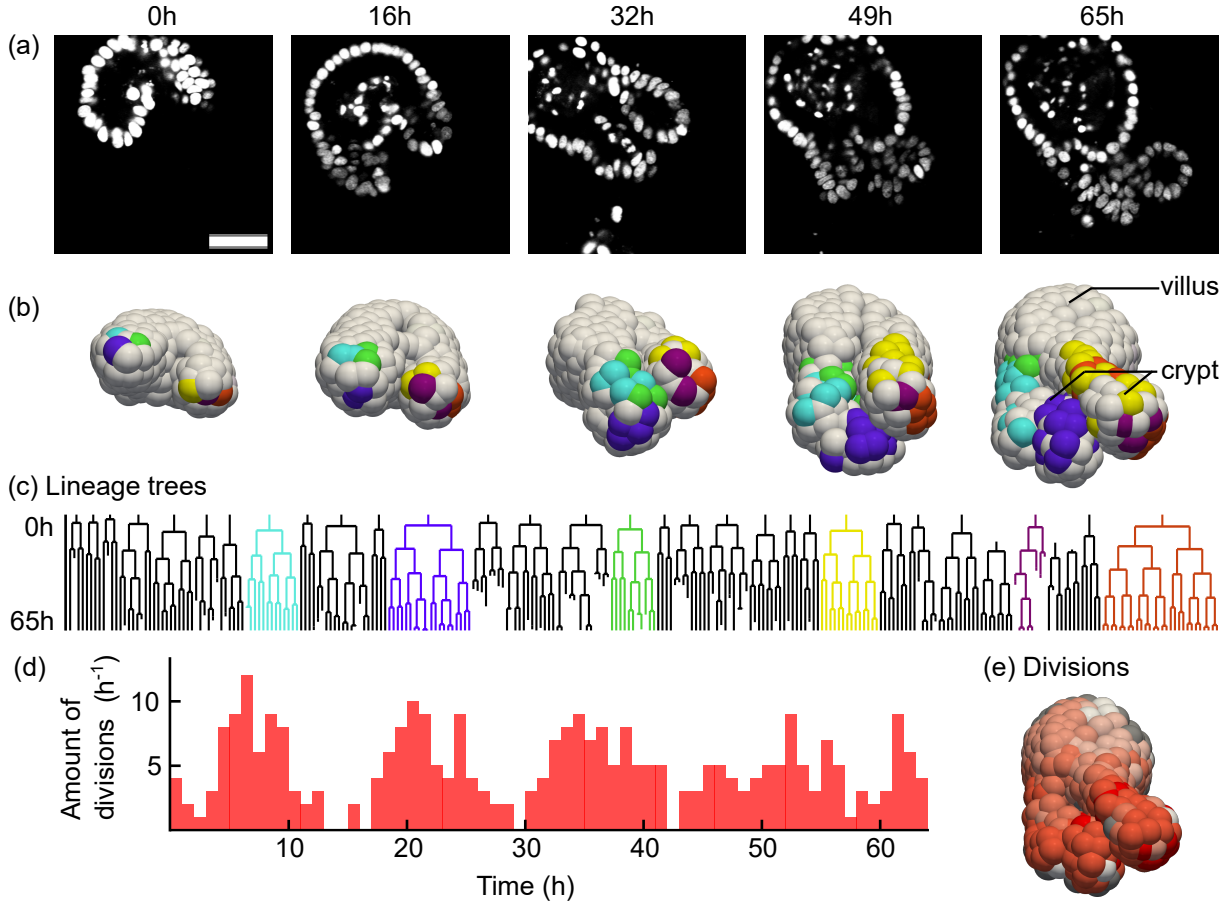

Figure 5: Analysis of a fully tracked organoid. (a) Single microscopy image slices of an intestinal organoid from a time lapse movie of 65 hours, in steps of 16.25 hours. H2B-mCherry was used to visualize the nuclei. Scale bar is 40 µm.**(b)** Digital reconstruction of the same organoid as in Panel a. Cells of the same color originate from the same cell at the beginning of the experiment. **(c)** Selected lineage trees. The colors match the colors in the reconstruction from Panel b. **(d)** Number of cellular divisions over time in a single organoid. **(e)** Number of times each cell divided during the time-lapse integrated over time, from 0 (white) to 5 (red). For gray cells, we could not count the amount of divisions because the cell was outside the image at an earlier point in time.

2.6 Software

Source code and software packages are available at `https://github.com/jvzonlab/OrganoidTracker`. Documentation including tutorials is available at the same address. The trained model used for this article has been made available as Supplementary Data 1; this model is suitable for 3D confocal images of cell nuclei, with a resolution of approximately 0.32 µm/px, z-separation of 2 µm

and low background noise. Another model is available for trained for the lower resolution of 0.40 µm/px, still with the same z-separation and noise tolerance.

The software was written in Python 3.6, making use of the established software packages Mahotas 1.4.4 [30], matplotlib 2.2.2 [31], numpy 1.15.1 [32], OpenCV 3.4.1 [33], PyQT 5.9.2 [34] and Tensorflow 1.9.0 [35]. Image loading support is provided by tifffile 2020.2.16 [36] and nd2reader 3.1.0 [37]. The software saves all tracking data as JSON files. Lineage tree drawing is supported by the software, and 3D visualization is possible by exporting the nucleus positions to Paraview [38], which can render those points as spheres. The software is also able to load and save files in the data format of the Cell Tracking Challenge [8], which makes it possible to exchange data with most other cell trackers. The software supports loading scripts written in Python, which can be used for data analysis. These scripts can be reloaded while the program is running, making it unnecessary to restart the program if you made a change to your analysis script.

3 Discussion

We have developed a program for cell tracking based on nuclear markers inside organoids that uses a convolutional neural network for detection and a min-cost flow solver for linking cell positions. For our dataset 91% of all links in the ground truth were recognized. This is mainly because the neural network was only able to recognize 96% of the cells, and a single missing detection already means that the linking algorithm cannot reconstruct the cell track. However, the software facilitates manual error-correction so from this data cell lineages at the same quality as for manual tracking can still be obtained.

The convolutional neural network and the min-cost flow solver used for linking are both very generic: they make no assumptions on the microscopy pictures, other than that they are grayscale 3D images. In contrast, the Gaussian fit, which is used to provide an estimate of the nucleus volume and to recognize cell divisions, is quite specific to our images: it assumes bright nuclei on a dark background with not too much noise. Currently, eliminating the Gaussian fit is not possible, as the recognition of cell divisions depends on this.

As this approach uses a supervised form of machine learning, it is necessary to have a diverse set of training data available. Although the network will already automatically generate variations of your training data by adjusting the brightness, scale and rotation, still the network can have suboptimal performance for microscope images taken with different settings, like for example images with a different contrast or resolution. To solve this, one needs to retrain the network on training data created with those settings.

It is possible to think of modifications to the algorithm. One could start training the neural network with a smaller training dataset, which results in a cell tracker

of which its output requires a large amount of manual curation to get error-free results. After this correction step is over, it is then possible to use the manually corrected data to retrain the neural network. In this way, a feedback loop is created that improves the neural network every time more tracking data becomes available. In this paper no such feedback loop has been used: because we already had a large amount of training data beforehand, for us it was not helpful to retrain the network.

In general, we believe the way forward is to make the neural network as accurate as possible, and then give the linking algorithm the smallest possible amount of options to create links. As long as the biologically correct option is still part of this smaller set of options the linking algorithm can choose from, this approach would reduce the chance of the linking algorithm making mistakes. One possible step in this direction would be to let the neural network also classify cells: if the network could point out where the mother and daughter cells are, the linking algorithm could be restricted to only creating divisions at those locations.

In conclusion, we are now able to observe a growing organoid at the cellular level using cell tracking in microscopy videos. For this, we have developed a cell tracker using a convolutional neural network for cell detection, a min-cost flow solver [23] for linking of cell positions and a small set of rules to assist with data curation. It is our hope that cell tracking, with its coupling of lineage tree dynamics to time and space, will lead to in a better understanding of organoid growth and maintenance.

4 Author contributions

The software was written by R.K., L.H. and K.B. The article was written by R.K., L.H. and J.Z. G.H, Y.G. and X.Z. cultured the organoids and manually tracked the cells. G.S., S.T. and J.Z. supervised the project and secured the funding.

References

[1] Elizabeth Pennisi. “Development cell by cell”. In: *Science* 362.6421 (2018), pp. 1344–1345.

[2] Stavroula Skylaki, Oliver Hilsenbeck, and Timm Schroeder. “Challenges in long-term imaging and quantification of single-cell dynamics”. In: *Nature Biotechnology* 34.11 (2016), pp. 1137–1144.

[3] Carl Magnus Svensson et al. “Untangling cell tracks: Quantifying cell migration by time lapse image data analysis”. In: *Cytometry Part A* 93.3 (2018), pp. 357–370.

[4] Sandra Montes-Olivas, Lucia Marucci, and Martin Homer. “Mathematical Models of Organoid Cultures”. In: *Frontiers in Genetics* 10 (2019), p. 873.

[5] Xiaolei Yin et al. “Engineering Stem Cell Organoids”. In: *Cell Stem Cell* 18.1 (2016), pp. 25–38.

[6] Benedetta Artegiani and Hans Clevers. “Use and application of 3D-organoid technology”. In: *Human Molecular Genetics* 27.May (2018), pp. 99–107.

[7] Judith M. S. Prewitt and Mortimer L. Mendelsohn. “The Analysis of Cell Images”. In: *Annals of the New York Academy of Sciences* 128.3 (1966), pp. 1035–1053.

[8] Vladimír Ulman et al. “An objective comparison of cell-tracking algorithms”. In: *Nature Methods* 14.12 (2017), pp. 1141–1152.

[9] Yao Yao, Ihor Smal, and Erik Meijering. “Deep neural networks for data association in particle tracking”. In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. Vol. 2018-April. ISBI. IEEE, 2018, pp. 458–461.

[10] Fernando Amat et al. “Fast, accurate reconstruction of cell lineages from large-scale fluorescence microscopy data”. In: *Nature Methods* 11.9 (2014), pp. 951–958.

[11] Runkai Zhu et al. “An Extended Type Cell Detection and Counting Method based on FCN”. In: *2017 IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE)*. Washington, DC: IEEE, 2017, pp. 51–56.

[12] Weidi Xie, J. Alison Noble, and Andrew Zisserman. “Microscopy cell counting and detection with fully convolutional regression networks”. In: *Computer Methods in Biomechanics and Biomedical Engineering: Imaging and Visualization* 6.3 (2018), pp. 283–292.

[13] Yuanpu Xie et al. “Efficient and robust cell detection: A structured regression approach”. In: *Medical Image Analysis* 44.3 (2018), pp. 245–254.

[14] Xipeng Pan et al. “Cell detection in pathology and microscopy images with multi-scale fully convolutional neural networks”. In: *World Wide Web* 21.6 (2018), pp. 1721–1743.

[15] Chichen Fu et al. “Three Dimensional Fluorescence Microscopy Image Synthesis and Segmentation”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Vol. 2018-June. IEEE, 2018, pp. 2302–23028. eprint: `1801.07198`.

[16] David Joon Ho et al. “Center-Extraction-Based Three Dimensional Nuclei Instance Segmentation of Fluorescence Microscopy Images”. In: *2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*. IEEE, 2019, pp. 1–4. arXiv: `1909.05992`.

[17] Edouard A. Hay and Raghuveer Parthasarathy. “Performance of convolutional neural networks for identification of bacteria in 3D microscopy datasets”. In: *PLoS Computational Biology* 14.12 (2018), pp. 1–17.

[18] Hans Clevers. “Modeling Development and Disease with Organoids”. In: *Cell* 165.7 (2016), pp. 1586–1597.

[19] Thomas E. Wallach and James R. Bayrer. “Intestinal Organoids: New Frontiers in the Study of Intestinal Disease and Physiology”. In: *Journal of Pediatric Gastroenterology and Nutrition* 64.2 (2017), pp. 180–185.

[20] Nadya Dimitrova et al. “Vertebrate Neural Stem Cell Segmentation, Tracking and Lineaging with Validation and Editing”. In: *PLoS ONE* 32.7 (2017), pp. 736–740.

[21] Jean Yves Tinevez et al. “TrackMate: An open and extensible platform for single-particle tracking”. In: *Methods* 115 (2017), pp. 80–90.

[22] Hongqing Han et al. “eDetect: A Fast Error Detection and Correction Tool for Live Cell Imaging Data Analysis”. In: *iScience* 13 (2019), pp. 1–8.

[23] Carsten Haubold et al. “A generalized successive shortest paths solver for tracking dividing targets”. In: *European Conference on Computer Vision*. Amsterdam, The Netherlands: Springer, 2016, pp. 566–582.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *Lecture Notes in Computer Science* 9351 (2015), pp. 234–241.

[25] Rosanne Liu et al. “An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. 31. Curran Associates, Inc., 2018, pp. 9605–9616.

[26] Stéfan van der Walt et al. “scikit-image: image processing in Python”. In: *PeerJ* 2 (2014), e453.

[27] Yu Toyoshima et al. “Accurate Automatic Detection of Densely Distributed Cell Nuclei in 3D Space”. In: *PLoS Computational Biology* 12.6 (2016), pp. 1–20.

[28] Helmuth Gehart and Hans Clevers. “Tales from the crypt: new insights into intestinal stem cells”. In: *Nature Reviews Gastroenterology & Hepatology* 16.1 (2019), pp. 19–34.

[29] Hans C. Clevers and Charles L. Bevins. “Paneth Cells: Maestros of the Small Intestinal Crypts”. In: *Annual Review of Physiology* 75.1 (2013), pp. 289–311.

[30] Luis Pedro Coelho. “Mahotas: Open source software for scriptable computer vision”. In: *Journal of Open Research Software* 1.1 (2013), e3.

[31] John D. Hunter. “Matplotlib: A 2D Graphics Environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95.

[32] Stéfan Van Der Walt, S. Chris Colbert, and Gaël Varoquaux. “The NumPy array: A structure for efficient numerical computation”. In: *Computing in Science and Engineering* 13.2 (2011), pp. 22–30.

[33] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. Ed. by Mike Loukides. O'Reilly Media, Inc., 2008, pp. 421–474.

[34] Riverbank Computing Limited. *What is PyQt?* URL: https://www.riverbankcomputing.com/software/pyqt/intro (visited on 03/13/2020).

[35] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. URL: http://tensorflow.org/.

[36] Christoph Gohlke. *tifffile*. URL: https://pypi.org/project/tifffile/ (visited on 03/13/2020).

[37] Jim Rybarski and Ruben Verweij. *nd2reader*. URL: https://rbnvrw.github.io/nd2reader/ (visited on 03/13/2020).

[38] Utkarsh Ayachit. *The ParaView Guide: A Parallel Visualization Application*. Kitware, 2015.

5 Supplementary information

5.1 Cell division scoring system

The fluorescent H2B-mCherry marker labels the histones, which normally results in the entire nucleus being observed. However, during a cell division the nuclear envelope breaks down and instead we observe the compacted chromosomes. It would therefore not be correct to say that we observe the nucleus during a cell division, as there is no nucleus during certain parts of the division. However, to make the text easier to read we will still call these compacted chromosome structures "nuclei" in the following text, instead of constantly writing "nuclei or compacted chromosome structures".

The scoring system takes into account both the mother cell and two daughter cells. The scoring system is designed such that lower values are more optimal; the score can therefore be considered as a penalty score or an energy. If the score is above -1, the possibility of a cell division is not even considered by the min-cost flow solver.

As stated in the main text, the linking algorithm starts by defining all possible links, which are the nearest-neighbor links along with links that span a distance at most twice as long as the distance to the nearest neighbor. Division scores are calculated for every case where a nucleus at one time point has at least two possible links to the next time point. If there are three or more possible links, then there are also three or more possible daughters. As biologically only two daughter cells are possible, scores are calculated for all possible combinations of two daughter cells and the lowest score is chosen.

Every potential mother cell gets a score using the following equation:

$$\begin{aligned}\texttt{score} &= \texttt{mother_intensity_delta} + \texttt{daughter_intensity_difference} \\ &\quad + \texttt{daughter_intensity_delta} + \texttt{mother_volume}\end{aligned} \tag{6}$$

Of these scores, mother_volume carries the heaviest weight. By default, it is 10 which is so high that it makes it impossible for the min-cost flow solver to create a division here. Only if the volume of the mother nucleus is almost as large as the volume of the new daughter cells combined, the score becomes 0 which allows for a division.

$$\texttt{mother_volume} = \begin{cases} 0 & \text{if } V_{mother}/(V_{daughter1} + V_{daughter2}) > 0.95 \\ 10 & \text{otherwise} \end{cases}$$

The other rules are then used to pick the most likely division candidate, if multiple possibilities of nuclei show such a volume change. The next rule makes use of the observation that the daughter nuclei are located at a different position than the mother nucleus was. Therefore, the fluorescence intensity at the location of the mother cell before the division should have decreased after the division. For non-dividing nucleus, it is rare that nuclei move this rapidly, so non-dividing nuclei will get a lower score.

$$\texttt{mother_intensity_delta} = \begin{cases} -1 & \text{if } I_{current}/(I_{next} + 0.0001) > 2 \\ 0 & \text{if } 2 \geq I_{current}/(I_{next} + 0.0001) > 1.4 \\ 1 & \text{otherwise} \end{cases}$$

Here, $I_{current}$ is the average pixel intensity at the location of the mother nucleus at the last time point where a single nucleus was visible, with the intensities of the image normalized from 0 to 1. The location of the mother nucleus is derived from the Gaussian fit: we look at all pixels at the XY plane of the nucleus center where the Gaussian fit still has 20% of its maximum intensity. I_{next} is the intensity at exactly the same pixel locations in the next time point. The $+0.0001$ is used to prevent a division by zero for the rare cases where the image becomes completely black at the location where the mother location was.

For the daughter nuclei the opposite should happen: where there was no nucleus before, there now is a nucleus. The intensity should therefore go up. For a single daughter, the score is calculated as follows:

$$\texttt{daughter1_intensity_delta} = \begin{cases} -1 & \text{if } I_{current}/(I_{prev} + 0.0001) > 2 \\ 0 & \text{if } 2 \geq I_{current}/(I_{prev} + 0.0001) > 1 \\ 1 & \text{otherwise} \end{cases}$$

Here, $I_{current}$ is the intensity of the daughter cell nucleus, in the first time point where the two daughter nuclei were visible. Again, the Gaussian fit is used to find out which pixels are inside the daughter cell nucleus, in exactly the same way as this was done for the mother nucleus. I_{prev} is the intensity at the same pixel locations, but for the previous time point.

For daughter nucleus 2 the same equation is used. Both values are then summed:

$$\begin{aligned}\texttt{daughter_intensity_delta} = &-1 + \texttt{daughter1_intensity_delta} \\ &+ \texttt{daughter2_intensity_delta}\end{aligned}$$

1 is subtracted here to relax the scoring system and make divisions more likely.

Finally, we expect both daughter nuclei to have the same intensity, as they were created out of a single nucleus minutes ago. The only deviations of intensity between the two nuclei should occur if one daughter nucleus ends up further away from the microscope objective than the other. Therefore, we can lower the score for any difference in the average intensity of the two:

$$\texttt{daughters_intensity_difference} = \frac{|I_{daughter1} - I_{daughter2}|}{2}$$

Again, the intensity is the average intensity of the pixels within the nucleus, as defined by the Gaussian fit.

5.2 Lineage tree obtained using manual data

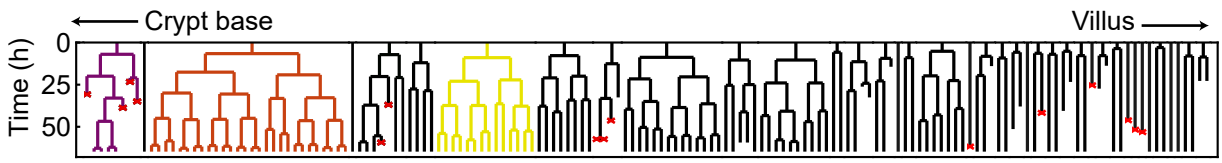

Figure S1: Full lineages of the manual tracking data of the same organoid that is displayed in Fig. 5. Red crosses denote the positions where a cell death was observed in the microscopy images. This figure will also become part of an upcoming publication.

5.3 Video's

Fig. 5a and b both showed five stills of a growing organoid: panel **(a)** showed microscopy slices and panel **(b)** a 3D reconstruction highlighting certain lineages. In the supplementary information, the full videos are included as Supplementary

Video 1 and 2, respectively. In addition, a 3D reconstruction video highlighting the cell divisions is included (Supplementary Video 3).