Supplementary Material: Morphologically Constrained and Data Informed Cell Segmentation of Budding Yeast

Elco Bakker, Peter S. Swain, and Matthew M. Crane

S1 Experimental Details

S1.1 Growth Conditions

The software was evaluated on images both acquired by us and by the Hao laboratory [7]. The performance in segmentation was evaluated on three datasets and in tracking on four datasets. One dataset for both area and tracking was acquired from [7], and the remaining datasets were acquired by us. For the images acquired in our lab, we used the following culture conditions. The cells in the experiment were plated on 2% XY Dextrose plates, inoculated in synthetic complete (SC) media at a very low starting concentration and loaded directly into the microfluidic device after 24 hours while still in exponential growth. A range of strains, imaging conditions and growth media were used to reflect the range of experiments undertaken in the laboratory (Table 1).

experiment (date)	**strain**	**imaging period**	**overnight growth**	**experiment**
1 (2016-04-25)	*GAL1*-mEOS2	10 minutes	SC raffinose (2%)	SC galactose (2%)
2 (2016-07-22)	HIS3:Pgal1-GFP-Pest	5 minutes	SC raffinose (2%) galactose (0.05%): HIS3	SC raffinose (2%) galactose (0.05%)
3 (2015-12-12)	*GAL1*-EGFP	5 minutes	SC raffinose (2%)	SC raffinose (2%) galactose (0.04%) glucose (0.02%)
4 (2016-04-03)	*WHI5*-EGFP	3 minutes	SC glucose (2%)	SC glucose (2%) (5 hours) → SC glucose (0.005%) (13 hours)

Table S1: Details of different experiments used to test DISCO. The strains for experiments 1, 3, and 4 were made by transformation with lithium acetate using DNA produced by PCR in a background strain of BY4741. The strain for experiment 2 was provided by Dr. Gael Yvert (ENS Lyon).

The preparation of and loading of cells in microfluidic devices was performed following previous protocols [4]. All experiments used a Nikon Ti-2000 microscope with a 60X 1.4 NA objective.

To generate the ground-truth dataset to estimate segmentation errors, cells were imaged using standard GFP filters and then time points selected at which expression was bright and consistent. Experiments 1 and 2 were used for area-tracking metrics; experiments 2, 3 and 4 were used for tracking metrics.

S2 Algorithm Details

S2.1 Identification of microfluidic features (traps)

The microfluidic features are automatically identified by the software and curated by the user at the first time-point before being identified at subsequent time-points by tracking. This series of steps identifies the features' locations throughout the experiment.

To identify the pixels associated with these features, a thresholded out-of-focus image is compared to a rough map for the pillars instantiated when the SVM is trained (see below). The thresholded region most overlapping each pillar is taken to be the pixels for that pillar for a given trap at a given time point. This region is calculated individually at each time-point for each trap to give an accurate map of the pillars' locations, which is used for image registration and in feature construction for the SVM.

S2.2 SVMs for Pixel Classification

Feature Set

To accurately predict cell seeds, images at each time-point were filtered and then processed using two linear SVMs. There is a tradeoff between increased filter numbers, the corresponding increase in time required to perform the image processing and classification, and the increased performance that can be achieved with the larger filter set. We found that the custom set of filters listed below balances these competing interests well. In particular, because the preponderance of cells having either circular or elliptical contours, the filter sets rely heavily on either the circular Hough transform or modifications of this transform. Although the classifier is therefore similar to methods relying just on the Hough transform, accuracy is improved.

The features are constructed from two normalised bright-field images: one above and one below the plane of focus. The second is subtracted from the first to produce a 3rd image, which is again transformed by setting all pixels outside a dilated mask of the trap features to the mean value to provide a 4th image. To each of these four images, the following transformations are applied to produce the final set of features:

1. the image itself
2. the absolute value of the image after mean subtraction
3. a local standard deviation filter of image 1
4. a Gaussian filter of the above filtered image
5. a gradient magnitude of image 1
6. a disk smoothing filter of image 4
7. an accumulation array of the circular Hough transform applied to image 4
8. a circular moving-average smoothing of the accumulation array (filter image 7)
9. a modified circular Hough transform to emphasize *dark to white* transitions applied to image 1
10. a circular moving-average smoothing of the accumulation array from filter 8
11. a modified form of filter 8 to emphasize circles of specific radii
12. a circular moving-average smoothing of the accumulation array from filter 10
13. a modified circular Hough transform to emphasize *white to dark* transitions applied to image 1
14. a circular moving-average smoothing of the accumulation array from filter 12

15. a modified form of filter 12 to emphasize circles of specific radii

16. a circular moving-average smoothing of the accumulation array from filter 14.

The relative importance of the features used for the classification can be ranked according to the **w** of the trained SVM [2] (Fig. S1). The relative importance of features changes depending on what is being classified and on the imaging method. By having a large number of different features, DISCO is flexible and works for applications and imaging approaches for which it was not originally designed.

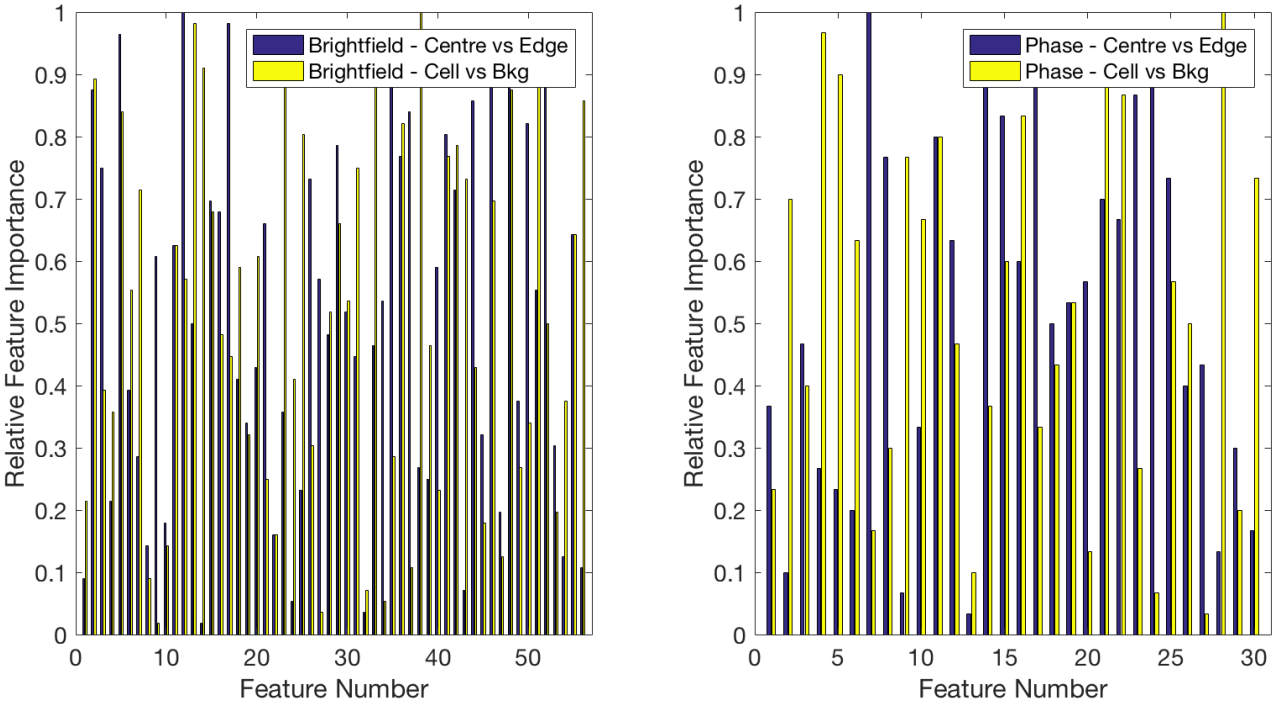

Figure S1: The relative weights of each feature following training of the SVM. The features are sorted according to the absolute weight and then normalized so the most important feature is ranked 1. Left) The relative rank of the features for the bright-field classification on our images. Right) The relative rank of features for the phase-contrast images acquired from [7]. Only a single z-slice was used for the phase-contrast images and so fewer features are generated.

Construction of decision and edge images

The set of features is passed to two SVMs: a foreground-background classifier and a centre-edge classifier. For each SVM, the signed distance from the decision plane (g_{BG} and g_{edge} respectively) are interpreted as Bayes factors:

$$g_{\mathrm{BG}} = \log\left(\frac{p_{\mathrm{background}}}{p_{\mathrm{interior}} + p_{\mathrm{edge}}}\right) \quad (1)$$

$$g_{\mathrm{edge}} = \log\left(\frac{p_{\mathrm{edge|foreground}}}{p_{\mathrm{interior|foreground}}}\right) \quad (2)$$

With this interpretation, we can calculate the probabilities for each class as:

$$p_{\text{interior}} = \frac{1}{1+e^{g_{\text{BG}}}}\frac{1}{1+e^{g_{\text{edge}}}} \tag{3}$$

$$p_{\text{edge}} = \frac{1}{1+e^{g_{\text{BG}}}}\frac{e^{g_{\text{edge}}}}{1+e^{g_{\text{edge}}}} \tag{4}$$

$$p_{\text{background}} = \frac{e^{g_{\text{BG}}}}{1+e^{g_{\text{BG}}}}. \tag{5}$$

The trap pixels previously identified are then forcibly set to have $(p_{\text{interior}}, p_{\text{edge}}, p_{\text{background}}) = (0.1, 0.2, 0.7)$. We can calculate the edge and decision images pixel-wise as:

$$\text{decision image} = \log\left(\frac{p_{\text{background}}+p_{\text{edge}}}{p_{\text{interior}}}\right) \tag{6}$$

$$\text{edge image} = \log\left(\frac{p_{\text{interior}}+p_{\text{background}}}{p_{\text{edge}}}\right). \tag{7}$$

Training

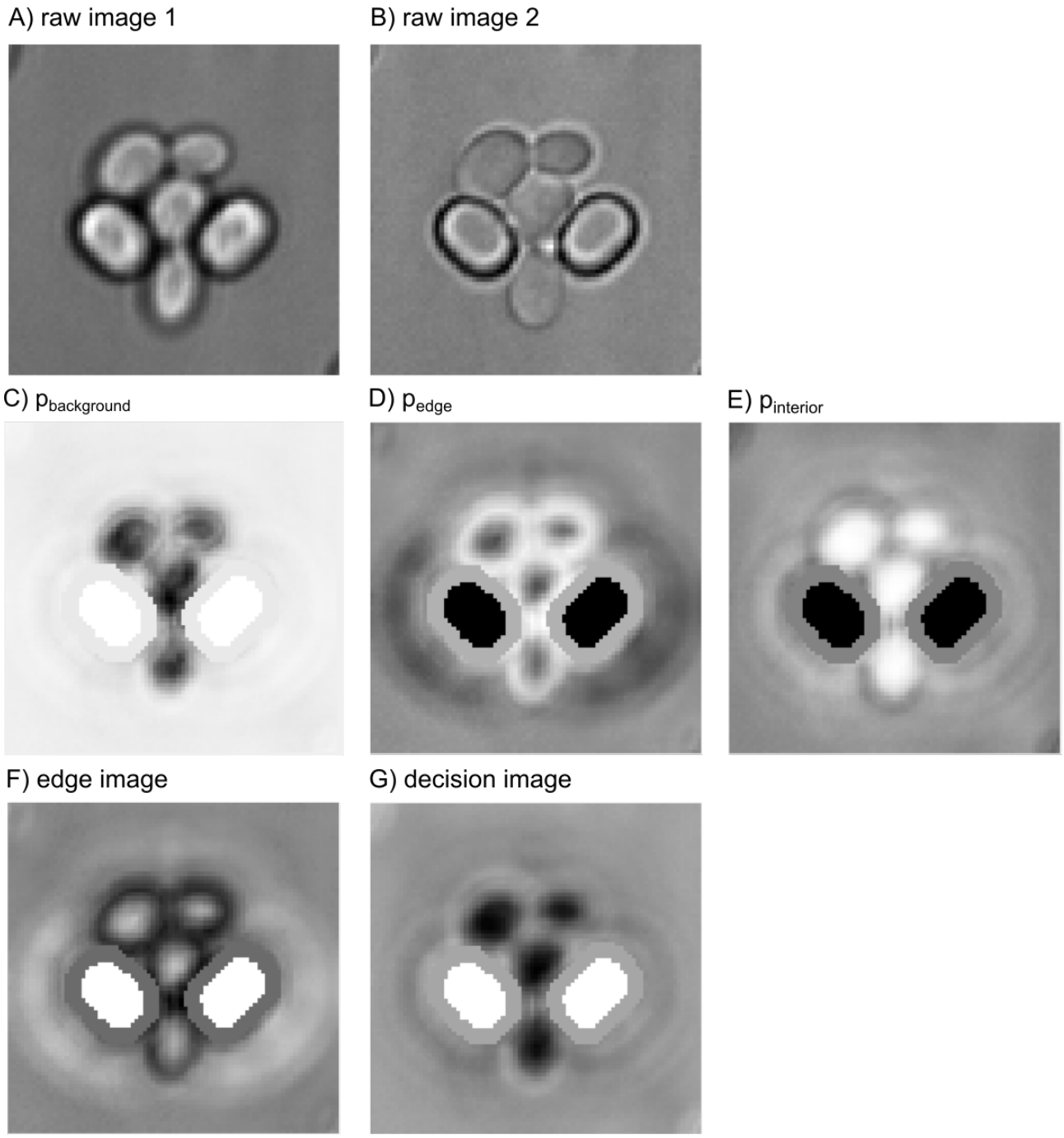

Figure S2: Examples of output from the classifier. A) and B) are the raw images used in the classifier. The other images are described in the main text.

A set of experiments for training are obtained and a random subset of the time-points are curated using a custom GUI written in Matlab. For each of the curated time-points, the traps are selected as in a normal experiment and the cells outlined by hand (a relatively straightforward procedure because the radial shape space used is easily edited by human intervention). Once this editing is complete, the features for each curated trap are calculated, and random selection of pixels from the cell interior, cell edge, and background are used as a training set for the SVMs. Using only a subset of pixels from each image allows us to build a diverse training set from many different images while still training the classifier in a reasonable time. The foreground-background SVM is trained to discriminate between background pixels and the union of interior and edge pixels; the edge-interior SVM is trained to

separate interior from edge pixels with no reference to the background pixels.

Ground-truth background pixels are taken preferentially from the region between the two pillars because these pixels are the most difficult for the SVM to distinguish from cell pixels. The pixels from the borders between the edge, interior and background regions are not used because classifier is more robust in their absence. An example of the images generated by the classifier are shown in Fig. S2.

The classifier used for experiments 1–4, and for all experiments performed in the Swain laboratory, was trained on example images from 3 different experiments, corresponding to approximately 100 individual traps. Once instructed, a user can accomplish this training in under a day. A classifier once trained can be used for all experiments using the same imaging modality and trap design. We have routinely used the same classifier for multiple years.

S2.3 Motivating the cost function

From a Bayesian perspective, it would ideally be possible to take the edge image and find the collection of cell outlines that maximises the log-likelihood over all configurations, i.e. the set of cell outlines that maximise:

$$\begin{aligned} C &= \sum_{i\in\text{all edge pixels}} \log\Big[p_{\text{edge}}(i)\Big] + \sum_{i\in\text{all interior pixels}} \log\Big[p_{\text{interior}}(i)\Big] \\ &\quad + \sum_{i\in\text{all remaining pixels}} \log\Big[p_{\text{background}}(i)\Big] \\ &= \text{constant} + \sum_{i\in\text{all edge pixels}} \log\Big[p_{\text{edge}}(i)\Big] - \log\Big[p_{\text{background}}(i)\Big] \\ &\quad + \sum_{i\in\text{all interior pixels}} \log\Big[p_{\text{interior}}(i)\Big] - \log\Big[p_{\text{background}}(i)\Big] \end{aligned} \tag{8}$$

To this cost function we should add a fixed penalty for each cell because each new cell adds a shape term that must have a probability of less than 1. Including this penalty gives an ideal cost function of:

$$\begin{aligned} C &= \text{constant} + \sum_{i\in\text{all edge pixels}} \Big\{\log\Big[p_{\text{edge}}(i)\Big] - \log\Big[p_{\text{background}}(i)\Big]\Big\} \\ &\quad + \sum_{i\in\text{all interior pixels}} \Big\{\log\Big[p_{\text{interior}}(i)\Big] - \log\Big[p_{\text{background}}(i)\Big]\Big\} \\ &\quad -\gamma N_{\text{cells}} \end{aligned} \tag{9}$$

where γ is a positive constant. Optimising such a cost function would be computationally intensive. Nevertheless, we can use this cost function to motivate our single-cell cost function.

First, if cells are minimally overlapping we can assume that any interior pixel of the current cell will not be an edge pixel of another cell and vice versa. Second, if we assume that there are some fixed set of 'good' pixels, with edge and interior pixels in proportion to the edge and interior ratio of the current cell, and that will constitute cells similar to this one, then the cost function is approximation by:

$$\begin{aligned} C &\simeq N_{\text{cells}} \sum_{i\in\text{cell edge pixels}} \Big\{\log\Big[p_{\text{edge}}(i)\Big] - \log\Big[p_{\text{background}}(i)\Big] - \log\Big[p_{\text{interior}}(i)\Big]\Big\} \\ &\quad + N_{\text{cells}} \sum_{i\in\text{cell interior pixels}} \Big\{\log\Big[p_{\text{interior}}(i)\Big] - \log\Big[p_{\text{background}}(i)\Big] - \log\Big[p_{\text{edge}}(i)\Big]\Big\} \\ &\quad -\gamma N_{\text{cells}} \end{aligned} \tag{10}$$

The summations are now over the particular cell we are segmenting, and the inclusion of the extra terms is from the assumption of non-overlapping cells. We have dropped the constant for clarity.

Further, the total number of cells is given by $\frac{N_{\text{good pixels}}}{N_{\text{cell pixels}}}$ because the number of 'good pixels' is fixed. Therefore:

$$C \simeq \frac{N_{\text{good pixels}}}{N_{\text{cell pixels}}} \times \Bigg[\sum_{i \in \text{cell edge pixels}} \Big\{ \log\Big[p_{\text{edge}}(i)\Big] - \log\Big[p_{\text{background}}(i)\Big] - \log\Big[p_{\text{interior}}(i)\Big] \Big\} + \sum_{i \in \text{cell interior pixels}} \Big\{ \log\Big[p_{\text{interior}}(i)\Big] - \log\Big[p_{\text{background}}(i)\Big] - \log\Big[p_{\text{edge}}(i)\Big] \Big\} - \gamma \Bigg] \tag{11}$$

In practice this cost function was seen to undervalue edge pixels, so the $N_{\text{cell pixels}}$ in the denominator of the edge pixel term was replaced $N_{\text{edge pixels}}$: turning the term into an average over edge pixels.

This reasoning gives some justification for the form of the cost function we use in segmenting the cells and in particular for the inflation term $-\frac{\gamma}{N_{\text{cell pixels}}}$.

S2.4 Powell-like line optimiser

Similar to [5], we implemented a Powell-like line algorithm to optimise the cost function ([9, 11]). DISCO is implemented in Matlab, and so the cost function was quicker to calculate in batches, and as such the line search sub-algorithm was altered to test multiple candidate moves at once and pick the move that offered the greatest improvement.

For new cells, the outline was initialised as a small circle within the cell, which was taken as a starting point for the optimisation. For tracked cells, the algorithm was run twice: once initialised as the outline at the previous time-point and once as a small circle. Although this approach effectively doubles the run-time, the improvement in performance was considerable.

S2.5 Shape space and motion training

In this section, the term 'radii' will generally refer to the six parameters, or spline knots, that define the cell outline (Fig. 3).

The shape space has two components: one for newly identified cells and one for tracked cells. To train the shape model for newly identified cells, cell outlines were curated for 50 time points (>5000 cells) with a 6 parameter spline fitted to each cell. The radii of these outlines were extracted and regularised by permuting and reversing so that the longest element was first and its longest neighbour was second while maintaining the ordering. permuting the radii is equivalent to rotating the outline of the cell so that the longest axis points to the right; reversing the radii is equivalent to reflecting the cell in the x-axis. We then have a tightly confined distribution of cell shapes, which is agnostic to orientation but maintains the essential shape of the cell.

A mean and covariance matrix was calculated for the regularised radii and then used in a multi-dimensional normal distribution to provide a probability for a given shape.

In the shape component of the active contour's cost function for newly identified cells, the radii defining the proposed contour are regularised as described above and the log probability density calculated assuming the fitted mean and covariance matrix and a multivariate normal distribution. This log probability is large and negative for unlikely cell shapes.

For tracked cells (i.e. a cell that has been identified based on the cells identified at the previous time-point), the shape space is different. The radii at the previous time point and the proposed radii for the current time point are independently regularised as described above. An element-wise division of the regularised proposed radii by the regularised radii from the previous time point is then calculated and assumed to follow a multivariate log-normal distribution. The covariance of this distribution was calculated from a set of curated pairs of cells (> 1000 pairs of cells at consecutive time points) and calculated separately for small cells (average radii of less than 1.6 μm, based on a visual inspection of the average value of the radii ratio) and large cells. We chose a log-normal rather than a normal distribution because, having a positive kurtosis, a log-normal better captures that cells are more likely to grow than shrink if they do not retain their shape from one time point to the next. To test the validity of this distribution, the Jarque-Bera test was performed on the log of the element-wise

division of the curated cell pairs for both large and small cells, and the null hypothesis of a normal distribution could not be rejected at the 5% significance level. We included this shape constraint in the cost function by subtracting the log probability of the cells' shapes following this distribution.

To generate the motion prior used for tracking cells and calculating the probable location images, we also used this set of curated pairs. When calculating the probable motion for a given cell, its size and location are used to extract two probability densities for its relative motion, and the average of these densities is the motion prior for the given cell. Cell centres were extracted for the pairs of cells and used to calculate the relative motion from one time point to the next. First, cell pairs were divided into four groups by the size of the cells at time point $t-1$, and an estimate of the probability density of the relative motion found by convolving the relative motion of these cells with a circular Gaussian. Second, cell pairs were separately divided by the location of the cells in the trap at time point $t-1$, and a motion probability density estimated for these cells. To ensure good statistics, cell 'locations' were grouped into three areas of the trap thought to behave similarly based on the flow profile (i.e. the centre, front and back of the trap).

The classifier makes little distinction between cell interior and centre and so we applied substantial smoothing to the motion prior. A strict motion prior will frequently propose seeds inside the cell but not at the actual centre: reducing edge detection efficacy.

S2.6 Algorithm for greedy identification of cells

The following is a description of the greedy optimisation in pseudo code.

```
for timepoint = 1 do
    calculate decision and edge images using the pixel classifier
    find new cells (decision image, edge image)
end

for timepoint = 2 to end do
    calculate decision and edge images using the pixel classifier
    for n = 1 to number of cells identified at previous timepoint do
        calculate probable location image for cell n
        probable location image stack (:,:,n) = probable location image
    end
    m = max (probable location image stack)
    while m > probable location image threshold do
        (x, y, z) = location of m in probable location image stack
        find cell outline using active contour algorithm with a center (x, y), cell z outline from
         previous timepoint for tracked cell shape probability and edge image
        if cell score < (score threshold) and
        cell shape probability > (shape probability threshold) then
            add cell with center (x, y) and proposed outline to the data structure
            give the cell the tracking number of cell z at the previous timepoint
            remove slice z from the probable location image stack
            set cell area to (probable location image threshold)-1 in all slices of probable
             location image stack
            set cell area to (decision image threshold)+1 in decision image
        else
            set cell area to (probable location image threshold)-1 in slice z of probable location
             image stack
        end
        m = max (probable location image stack)
    end
    find new cells (decision image, edge image)
end


def find new cells(decision image, edge image):
    m = max (decision image)
    while m > decision image threshold do
        (x, y) = location of m in decision image
        find cell outline using active contour algorithm with a center (x, y), new cell shape
         probability and edge image
        if cell score < (score threshold) and
        cell shape probability > (shape probability threshold) then
            add cell with center (x, y) and proposed outline to the data structure
            give the cell a new tracking number
            set cell area to (decision image threshold)+1 in decision image
        else
            set cell area to (decision image threshold)+1 in decision image
        end
        m = max (decision image)
    end
```

For clarification, if a tracked cells fails on any of these thresholds, the cell is not stored as a true cell, and the putative area of the cell is blotted out of the probable location image but not blotted out of the decision image, allowing new cells to be identified in this region. If a new cell fails its score threshold, the cell is blotted out of the decision image because a cell cannot be successfully identified in that region.

Usually the thresholds for the decision image and probable location image are chosen by eye based on the decision image of a few time points, but for the purposes of comparison the thresholds were fixed across the experiments. The threshold for the shape probability is selected from the probability of the curated shapes to which the distribution was fitted and is fixed for all experiments. The threshold for the cell score was set similarly.

S3 Details of the comparison with alternative software

S3.1 Curation for segmentation

To score segmentation, two experiments (Sec. S1) in which a bright, cytoplasmic fluorescent reporter had been imaged were selected and used to create fully curated ground-truth datasets. The workload of manually correcting each image is substantial, and so we instead segmented using the images acquired in the GFP channel. This fluorescence segmentation used a maximum projection of the fluorescence images and a Chan-Vese-based active contour applied to this projection. Following the segmentation, a subset of regularly spaced time-points were then manually curated against the bright-field images using a GUI built into DISCO and any false positive or false negative cell objects corrected. Only these manually corrected time points are used for comparing segmentation. The data provided by the Hao laboratory was created manually from the phase contrast images.

Imaging conditions can change slightly between experiments and so the ground-truth datasets were acquired over several months to ensure that robust performance of one algorithm over the other is not a result of better alignment to the specific imaging conditions. Furthermore, the ground truth datasets come from images that were acquired over 24h because cell morphology can change over the course of their lifespan. During this time, the median cell has divided nearly 19 times, which is >75% of the median lifespan. As such, the dataset provides a wide range of cell morphologies.

Algorithms for segmentation often rely on specific aspects of the cell boundary, and one algorithm may choose the inner boundary and another choose the outer boundary. To avoid bias from consistent over or under segmentation, the cell outlines identified by the alternative software were subjected to dilation and erosion. A grid search of various dilation and erosion sizes was performed and the best score over this grid was taken as the score for the given algorithm (Fig. S4 for CellX; Fig. S3 for CellSerpent; Fig. S5 for CellStar). We note that this process of dilation and erosion was not performed for DISCO, but the raw segmentation output used instead.

S3.2 Curation for tracking

Three different datasets were used to generate the curated dataset for tracking (Sec. S1). To generate datasets that represent the normal experimental variation, the images were acquired in experiments run at different times over several months. All cells within the selected traps were curated for tracking except those cells far from the centre of the traps: cells more than three cell diameters from the centre of the trap (30 microns) were excluded because these cells are unlikely to be retained for long periods of time.

S3.3 Cell Serpent

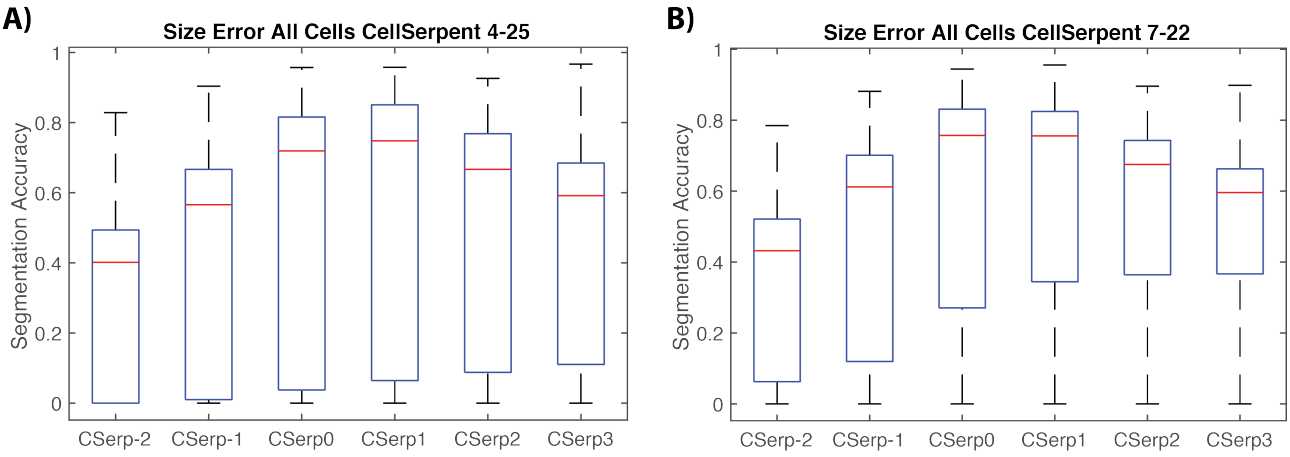

Figure S3: A grid search for the optimal sizes of erosion and dilation was performed against the results for segmentation for CellSerpent. For x-labels less than 0, the image was eroded; for x-labels greater than 0, the image was dilated. (Top) Performance for the 7-22 dataset. (Bottom) Performance for the 4-25 dataset.

The source code for CellSerpent[1] was modified to allow it to be used directly from our own data structure and to use seeds proposed by our SVM because CellSerpent was not designed for bright-field images. Parameters were selected using the GUI supplied.

Further, we implemented a commonly used tracking method [3] that relies on the area overlap between adjacent time-points because CellSerpent does not include a cell tracking algorithm. This algorithm assumes that segmented objects that overlap in subsequent frames are likely to be the same cell. For the overlap metric, we used the score from the segmentation (union divided by intersection). We performed a grid search to determine the optimum overlap score for tracking, which was used when measuring the tracking performance.

S3.4 CellX

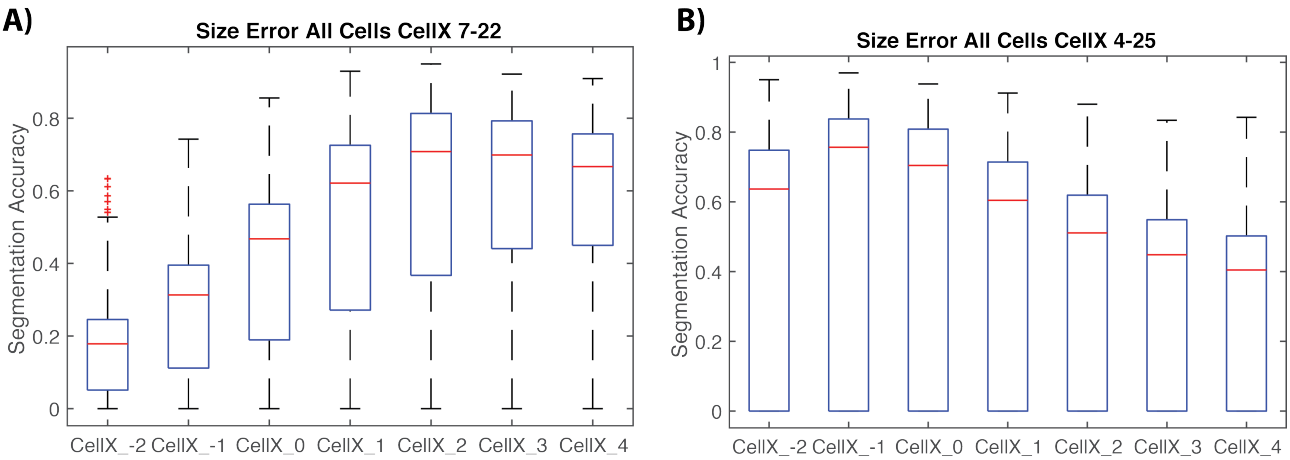

Figure S4: A grid search for the optimal sizes of erosion and dilation was performed against the results for segmentation for CellX. For x-labels less than 0, the image was eroded; for x-labels greater than 0, the image was dilated. (Top) Performance for the 7-22 dataset. (Bottom) Performance for the 4-25 dataset.

Segmentation and tracking by CellX [6] was run from the provided Java GUI and the result imported into our data structure. The software was configured separately for each of the five experiments

according to the published guide [8] and with input from the authors. A single out-of-focus bright-field image where the edge was dark, and the center bright was used to segment each timepoint. Tracking was performed using the CellX tracking algorithm.

S3.5 CellStar

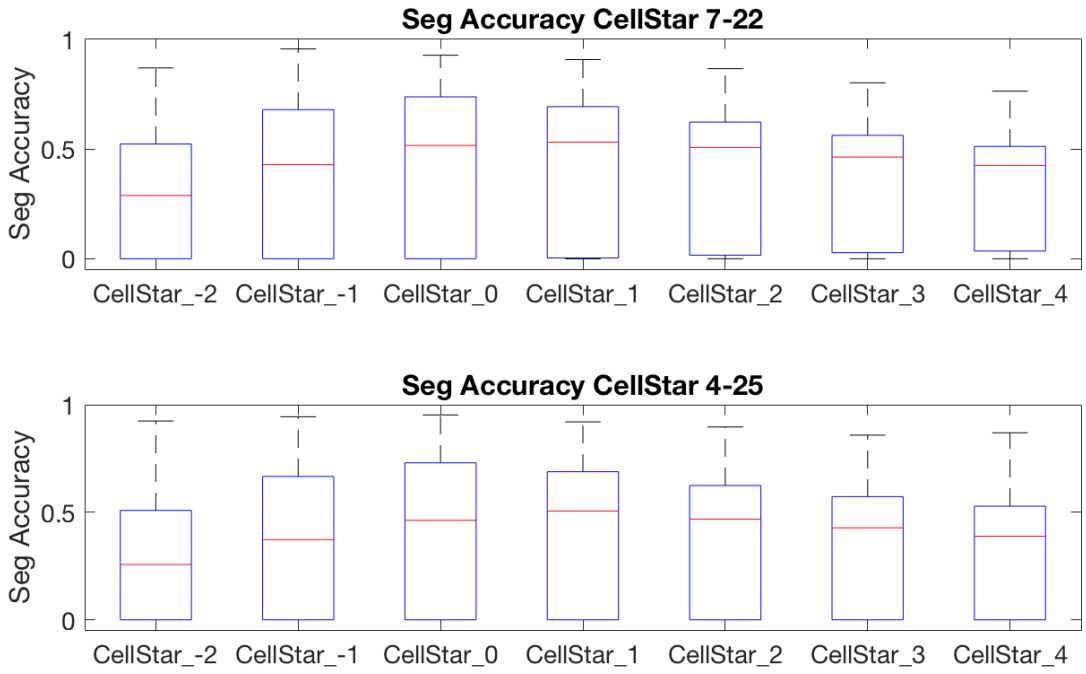

Figure S5: A grid search for the optimal sizes of erosion and dilation was performed against the results for segmentation for CellStar. For x-labels less than 0, the image was eroded; for x-labels greater than 0, the image was dilated. (Top) Performance for the 7-22 dataset. (Bottom) Performance for the 4-25 dataset.

Segmentation and tracking by CellStar [10] was run from the command line in batch mode in Matlab, and the results imported into our data structure. The software was configured according to the published user guide. For the bright-field images, CellStar was run as follows. A single out-of-focus image where the cell was dark and the outline was light was used for segmentation. A single dataset was used to create a ground-truth that was then used for customized training of the CellStar classifier by using the Matlab GUI provided with CellStar. This ground-truth was generated using 31 different cells selected from five time-points. The segmentation parameters generated by this training and optimization were used for all classification by CellStar. Segmentation was performed using an accuracy parameter of 7, and the background image was automatically generated using the first frame of the time-lapse movie.

For the phase-contrast images from [7], the entire segmentation process was run using the GUI. The average diameter of cells was set manually to 17 pixels, and 22 cells were used to generate the ground-truth dataset for training. Unlike the training for DISCO on these images, where we trained the classifier on images that were not part of the test set, we created the training set directly from the test images for CellStar.

S3.6 DISCO without history

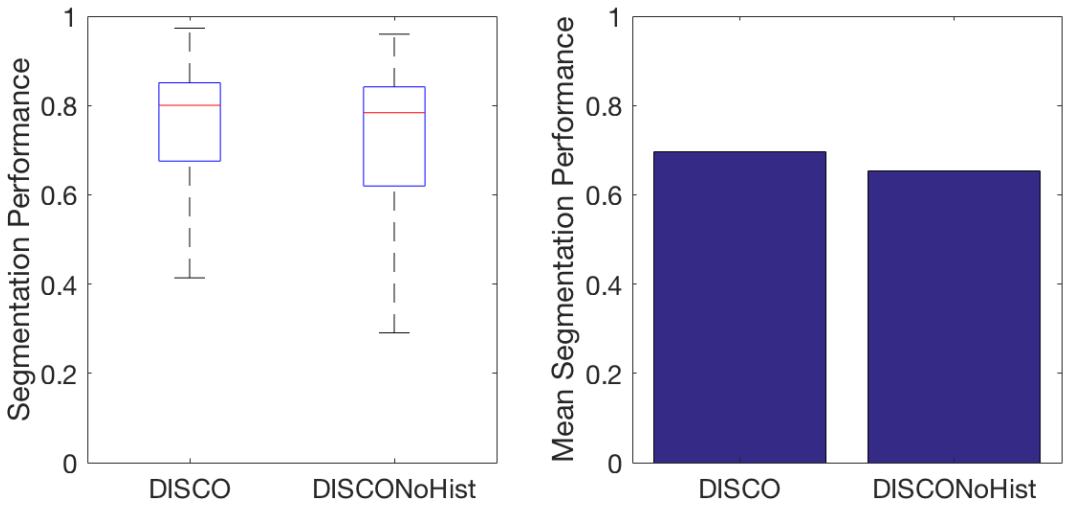

Figure S6: The performance for segmentation of DISCO without history: the lack of information from prior time-points reduces performance. To demonstrate the importance of including prior temporal information, DISCO was run with and without access to cell history. (Left) Box plots showing the distribution of measurements of segmentation performance for each cell. (Right) The mean performance on all cells for DISCO run with and without history information.

DISCO incorporates information about the cell shape and location at the previous time-point. By using such temporal information, continuity is introduced into the cell shape and location during segmentation. To demonstrate the advantages of this additional information, we separately ran DISCO on the curated datasets for segmentation excluding the information from previous time-points. There is a reduction in performance (Fig. S6).

S3.7 Performance in segmentation depends on cell size

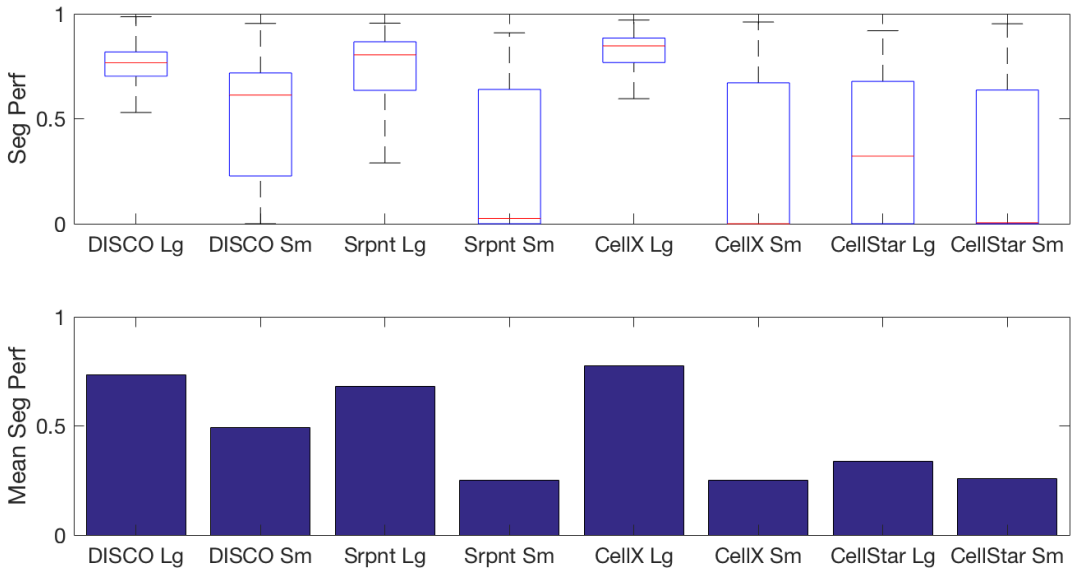

Figure S7: Segmentation performance is a function of cell size. The ground-truth dataset was separated into quartiles based on the cell size and performance evaluated for the upper (Lg) and bottom (Sm) quartiles. (Top) Box plot showing the distribution of performance for single-cell segmentation . (Bottom) Mean segmentation performance over all cells.

Cell size affects the accuracy of segmentation, and we therefore further investigated how each algorithm performs based on cell size. We separated the test datasets based on cell size and segmentation for large (upper quartile of cells) and small (lowest quartile of cells) compared (Fig. S7). Each software perform similarly on large cells, but performance is variable with the smallest cells. With small cells, there is less optical information to interrogate and the prior information on cell morphology and the model of the shape space become more important.

S3.8 Performance for phase-contrast images

Of the three data sets that were provided by the Hao laboratory [7], two were used for training the classifier and the third was used for testing the segmentation. The performance was then compared against the other software. To avoid biasing our ground-truth towards any particular approach, no 'base result' was used in the curation and the images were curated from scratch. Six traps in 20 time-points spread across the experiment were used for the curation.

S3.9 Comments on run-time

The run-time was tested for segmentation of a single position for all 133 time-points. In contrast to the curated datasets, the whole image was analysed and not just a subset of traps. Though DISCO is usually parallelised, it was de-parallelised for the comparison. For CellX, the execution time was taken from the creation of data files by the Java GUI and includes tracking time. All tests where run on an iMac with a 2.5 GHz Intel Core i5 and 12 GB of ram.

software	execution time (s)
DISCO	7855
CellX	6240
CellSerpent	8272
CellStar	8519

As can be seen, DISCO is marginally faster than CellX, though slower than CellSerpent. When parallelised, DISCO's execution time is reduced substantially.

References

[1] K. Bredies and H. Wolinski. An active-contour based algorithm for the automated segmentation of dense yeast populations on transmission microscopy images. *Computing and Visualization in Science*, 14(7):341–352, 2011.

[2] Y.-W. Chang and C.-J. Lin. Feature ranking using linear svm. In *WCCI causation and prediction challenge*, pages 53–64, 2008.

[3] S.-C. Chen, T. Zhao, G. J. Gordon, and R. F. Murphy. A novel graphical model approach to segmenting cell images. In *Computational Intelligence and Bioinformatics and Computational Biology, 2006. CIBCB'06. 2006 IEEE Symposium on*, pages 1–8. IEEE, 2006.

[4] M. M. Crane, I. B. N. Clark, E. Bakker, S. Smith, and P. S. Swain. A Microfluidic System for Studying Ageing and Dynamic Single-Cell Responses in Budding Yeast. *PLoS ONE*, 9(6):e100042, jun 2014.

[5] R. Delgado-Gonzalo, D. Schmitter, V. Uhlmann, and M. Unser. Efficient Shape Priors for Spline-Based Snakes. *IEEE Transactions on Image Processing*, 24(11):3915–3926, 2015.

[6] S. Dimopoulos, C. E. Mayer, F. Rudolf, and J. Stelling. Accurate cell segmentation in microscopy images using membrane patterns. *Bioinformatics*, 30(18):2644–2651, 2014.

[7] Y. Li, M. Jin, R. O'Laughlin, L. S. Tsimring, L. Pillus, J. Hasty, and N. Hao. Multi-generational silencing dynamics control cell aging. *bioRxiv*, 2017. doi: 10.1101/102210. URL `http://biorxiv.org/content/early/2017/01/22/102210`.

[8] C. Mayer, S. Dimopoulos, F. Rudolf, and J. Stelling. Using cellX to quantify intracellular events. *Current Protocols in Molecular Biology*, (SUPPL.101):1–20, 2013.

[9] M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, feb 1964. doi: 10.1093/comjnl/7.2.155.

[10] C. Versari, S. Stoma, K. Batmanov, F. Mroz, A. Kaczmarek, M. Deyell, P. Hersen, and G. Batt. Long-term tracking of budding yeast cells in brightfield microscopy : CellStar and the Evaluation Platform. 2017.

[11] E. Ziegel, W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. Numerical Recipes: The Art of Scientific Computing. *Technometrics*, 29(4):501, nov 1987.