Supporting Information

Gelbart et al. 10.1073/pnas.1205258109

SI Text

Description of Embryo Development Geometry Explorer. This section describes in detail the five main steps of the Embryo Development Geometry Explorer (EDGE) pipeline. Datasets processed by EDGE consist of image (.tif) stacks recorded over depth, time, or both. One channel contains images obtained with a membrane marker. Additional channels may be added containing information on other relevant markers. The software then performs three basic steps: (*i*) membrane detection, (*ii*) construction of the polygon data structures, and (*iii*) tracking. The outcome is a set of cells, and each set is represented by a stack of polygons that is tracked over space and time. Because every segmentation and tracking algorithm admits some errors (Fig. S5), the user may optionally use EDGE's built-in interface for manually correcting errors. The user can then also use (*iv*) automatic error correction to improve the results. The last step is (*v*) computing measurements of desired shape properties. If the data to be processed contains additional channels, these channels can be processed as well and related to the cell geometry.

(i) Membrane detection. When a dataset is imported into EDGE, each image is processed independently to form a binary image representing the cell membranes. Images are first normalized to zero average, clipped at ± 2 SDs, and then, denoised with a band-pass filter in Fourier space. The band-pass filter uses the radially symmetric kernel (Eq. **S1**)

$$f(k) = \frac{1}{1+\exp\left(\frac{1}{\beta}(k-2\pi/L)\right)} - \frac{1}{1+\exp\left(\frac{1}{\beta}(k-2\pi/H)\right)}, \quad \textbf{[S1]}$$

where k is the radial polar coordinate in Fourier-space, L and H are the low-pass and high-pass cutoff wavelengths, respectively, and β is the filter sharpness. The filtered image is then thresholded at a threshold defined by the user in units of the filtered image intensity's SD, resulting in a binary image with ones at pixels that were above threshold and zeros elsewhere. EDGE then applies morphological transformations to shrink structures down (implemented by Matlab's *bwmorph* function). Structures with holes (such as putative cells) are reduced to 1-pixel-wide closed curves. Structures without holes shrink down to isolated pixels and are eliminated from the image in subsequent steps. Optionally, cells near the boundary of the image may be removed.

(ii) Reduction to polygons. Rationale. The binary image data structure contains detailed information about membrane locations, but it is cumbersome to work with for several reasons. First, the amount of data is large; even in a sparse representation, binary membrane images still contain a large amount of information. Incorporating too much data can slow steps farther down the pipeline. Second and more importantly, this data structure is difficult for user interaction. For example, EDGE includes an interface that allows the user to fix errors, such as missing edges (see below). If EDGE relied on the binary image data structure alone, the user would need to manually paint in the missing edge pixel by pixel until the mistake was repaired. The program would then need to recognize that a cell has been split into two and reorganize its internal data structures to create two smaller cells. This process would require it to decide which membrane pixels belonged to which new cell, each time dealing with one pixel at a time. Any system that automatically corrected errors would be much more complicated. To avoid these difficulties, EDGE approximates the cells as polygons and stores only ordered lists of vertices. This process compresses the data by a factor of 10–100 depending on the size and details of the image, and furthermore, this representation is insensitive to changes in image resolution.

Method. To obtain the polygon-based representation, we first find the centroids and the vertices of each cell outline in the thresholded binary image (using Matlab's *regionprops* and *bwmorph* commands with the branch points option, respectively). We then label each connected region with an integer index (using Matlab's *bwlabel* command), such that each label corresponds to a putative cell region. Each integer-labeled image, along with its associated cell centroids and vertices, is then used to build the polygon data structure. (This point in the pipeline marks the transition from Matlab to Java code). First, for each vertex, we examine the integer labels in the eight neighboring pixels to build a list of regions (and thus, centroids) associated with that vertex. Vertices with fewer than two neighboring regions are removed. Second, the vertices are recursively merged with other vertices within a user-specified minimum distance. The result is a set of merged vertices and their corresponding neighboring regions (which were combined by taking the union of the two neighbor sets during the merging process). The data structure is then inverted, resulting in a set of regions and their associated vertices instead of vertices and their associated regions. In this paper, we refer to such regions as polygons, because they represent a cell outline as a polygon in 2D space. For each polygon, we sort the vertices by angle using the cell centroid as the origin to obtain vertex connectivity information. We then remove polygons with fewer than two neighbors as well as those polygons containing an interior angle less than some user-specified minimum. We also remove polygons with area less than the user-specified minimum. This process is repeated for all images in the dataset.

Capturing curvature. In cases where the polygon representation of cell outlines is too crude, because edges between vertices are curved, EDGE offers a feature to add additional vertices along edges both manually and automatically (Fig. S1 *E* and *F*). These new vertices do not have the original definition of points where three or more cells meet but are represented in the same way in the data structure, and they allow for more complex polygon shapes in the 2D imaging plane. Using this method, the user is able to increase the precision of the membranes until the curvature of the membranes is captured to a satisfactory degree, while still preserving the vertex-based representation.

(iii) Tracking in space and time. Tracking in EDGE is based on the assumption of continuity and therefore, requires sufficiently high spatial and/or temporal resolution in the data. The algorithm works by matching polygons between different images. First, a user-specified ground truth image or reference image is defined. This image is typically an image for which the automatic segmentation in EDGE works very well. If necessary, segmentation errors in this image can be manually corrected using the interface provided by EDGE. This image contains all of the cells that will be reconstructed and analyzed. Starting from this reference image and moving out, polygons from other images are then iteratively matched in both space (if more than one layer has been imaged) and time (for live-imaging data). A polygon in one image is matched to a polygon in the next image if and only if the following criteria are met.

i) Each polygon contains the centroid of the other polygon.
ii) The centroids of the two polygons are within some user-specified maximum distance.

iii) The fractional overlap of the two polygons (defined as the overlapping area between two cells divided by the maximum of the two cell areas) is above some user-specified minimum value.

If a match is found, the new polygon becomes the current polygon, and the process is repeated iteratively. Spatial tracking is performed first and separately at each time point. Then, polygons are tracked over time only at the depth of the reference image. In case a polygon image cannot be matched to any polygon in the following image, matching is attempted in subsequent images up to a user-specified number of images. Optionally, tracking may be improved by linearly extrapolating the position of each polygon to its predicted location at the next depth/time before evaluating the above criteria, but this option is disabled by default. Tracking results in 4D (space plus time) reconstructions of individual cells in terms of stacks of polygons.

(iv) Automatic error correction. EDGE offers a built-in platform for manual correction of segmentation errors, including missing or misplaced vertices, missing edges, and falsely identified vertices, edges, or cells. Manual error correction gives the user full control of the data but can be very time-consuming for datasets containing many images. Therefore, EDGE also offers automatic error correction that tries to correct errors in a given image by using information from other images in the stack. The basic idea is that EDGE attempts to rerun the tracking algorithm, and when tracking fails, it makes small changes to the polygons to continue tracking successfully. EDGE can perform three types of automatic error correction. First, vertex addition is used to improve the approximation of polygons to the cell outline; second, edge addition is used to split two cells that may have been merged, and third, edge removal is used to merge two polygons that may have been split. The vertex-based data structure used by EDGE makes these error correction methods simple and effective.

An example of the automatic error correction process is shown in Fig. S1. Fig. S1*A* shows an image after the membrane detection stage. In this example, the detection makes two errors, one merge (red box) and one split (blue box). The cells shown in yellow are those cells that the program cannot track (i.e., match with cells in the previous image), and thus, these cells become the candidates for automatic error correction. The user first uses the automatic vertex addition (also called edge-splitting) routine to recover vertices that were lost because of errors. Note that the parameters of the edge splitting were chosen such that only relatively large new edges can be created, which allows the user to recover vertices lost because of errors without adding unwanted extra vertices (Fig. S1 $A \rightarrow B$). Note also that, after the edge splitting, the two flanking cells become pink, because they are now tracked, given that they are now sufficiently similar to those cells in the previous image. Next, the split (Fig. S1 $B \rightarrow C$, blue box) and the merge are repaired automatically (Fig. S1 $C \rightarrow D$, red box). At this point, all of the errors are repaired, but the user may then decide that the curvature of the membranes is not sufficiently captured by the polygon approximation (Fig. S1*E*, light green boxes). The automatic split edge function can then be used again but with different parameters, and therefore, small edges can also be created. This results in an improved polygon representation that satisfactorily captures the membrane curvature (Fig. S1 $E \rightarrow F$).

The next three paragraphs explain the implementation details of the three automatic error correction routines.

Automatic vertex addition. The automatic vertex addition algorithm works by considering the addition of, at most, one new vertex per edge. For each edge, the algorithm finds the midpoint of the edge and considers the line perpendicular to the edge that passes through the midpoint. It then follows this line in both directions until one of the following scenarios occurs.

i) It reaches an on pixel in the binary membrane detection image. In this case, the algorithm adds a new vertex if and only if the angle created does not exceed the user-specified maximum angle and the two tentative new edges are not shorter than the user-specified minimum edge length.

ii) The angle between the two vertices that make up the edge and the candidate new vertex becomes smaller than the user-specified minimum angle. In this case, a vertex is not added.

Automatic edge addition. The automatic edge addition algorithm considers only polygons that were not successfully tracked. For each such polygon, the algorithm considers all possible edges that would divide the polygon into two. It then considers only the subset of these edges with addition that would result in two polygons that could both be successfully tracked to polygons in the previous image. For each such edge, it computes the fractional overlap of the two resulting polygons with their corresponding polygons in the previous image. For each of the two candidate new polygons, the smallest interior angle of the polygon is computed. A total score for the potential splitting is then computed by adding the two overlap scores to the two angles (in radians). After all potential edges are considered, the split that results in the highest total score is performed. The extra term involving angles is used to bias the algorithm against the creation of cells with sharp corners, because such cells tend not to occur in most data that we have encountered.

Automatic edge removal. The automatic edge removal algorithm also considers only polygons that were not successfully tracked. For each such polygon, the algorithm considers merging that polygon with each of its neighbors. If, for any such potential merger, the resulting polygon would be successfully tracked to a corresponding polygon in the previous image, the fractional overlap is calculated between the candidate new polygon and its corresponding polygon in the previous image. After all neighbors are considered, the merge that results in the largest overlap is performed.

(v) Measurements of geometric properties. After the cells are reconstructed to a satisfactory degree, EDGE can perform measurements on geometric properties, visualize them, and export them. Measurements and visualizations are based on the polygon stacks representing cell geometries. By default, EDGE can calculate cell volume and length as well as the x- and y-position of centroids and the area of cell outlines at a given depth. In case additional channels are provided, EDGE also calculates, by default, the polygon-averaged labeling intensity at each depth. Additional processing modules (based on Matlab code) can be added in a straightforward way by the user (see the online documentation). For each cell, a list of its nth-order neighboring cells is provided for computing properties that rely on the properties of more than one cell.

Download and Documentation. The EDGE source code, as well as additional documentation, is freely available at http://code.google.com/p/embryo-development-geometry-explorer/. EDGE is implemented using a combination of Matlab and Java. Matlab was used for its image processing features, and its prevalence in the scientific community allows for user-generated measurement scripts to be integrated more easily. Java was used because its more object-oriented nature helped to keep the internal data structures organized and the code more readable.

EDGE Examples. ***(i) Example 1: Importing data and segmenting cells with the EDGE Importer.*** EDGE comes with two main interfaces: the EDGE Importer for importing data and processing data and the EDGE Browser for exploring cell properties. Fig. S2 shows a screenshot of the EDGE Importer. At the upper left, an image frame (grayscale) obtained by live imaging of *Drosophila* is depicted overlaid with the polygon-approximation membranes (pink) and cell centroids (red dots) detected by EDGE. The sliders below the image allow the user to navigate through the different images, both in space (different depths in the stack) and time. The buttons at the upper right allow for manual and automatic error correction of the dataset. The user is able to add, remove, or move vertices, add or remove cells, and add or remove edges. See the EDGE online documentation for more information on the buttons and text boxes shown in the screenshot, which allow the user to control the various image processing parameters.

(ii) Example 2: Basic capabilities of the EDGE Browser. Fig. S3 shows a screenshot of the EDGE Browser highlighting some of the capabilities of the software. As with Example 1, a live-imaging experiment on *Drosophila* ventral furrow formation has been processed by EDGE, and the upper left shows a single-image frame. The cell with a red dot in it is the cell currently selected by the user. The user has also selected neighbors mode (see top right; neighbors order = 2). Therefore, first-order and second-order neighbors are also selected (green and light blue dots, respectively). At the bottom right of the screen, we see a 3D rendering of this group of 20 cells. The red line in the 3D rendering corresponds to the depth slice that is currently being displayed at the upper left.

The 3D rendering is useful to give the user a general intuition about cell shape and what the cells look like. However, EDGE is capable of producing quantitative information as well. In the lower left of the screen, we see a plot with three curves, corresponding to the three colors of selected cells (red, green, and light blue). This plot is of cross-sectional area vs. time at the currently selected depth (the green and light blue curves show averages over their respective cells). This kind of analysis could, for instance, be used to determine whether a cell and its neighbors constrict synchronously.

(iii) Example 3: Integration of multichannel data. EDGE requires images of the cell membranes, which it uses to perform the membrane detection, tracking, and all other functionalities. However, the software is not limited to just single-channel images; along with the membranes, any number of additional channels may be imported into the software (providing information on, for instance, the nucleus, myosin concentration, adherens junctions, etc.). When EDGE segments the membrane image, it identifies which parts of the image belong to which cells, and thus, this information can be combined with information from the other channels. Fig. S4 shows an example of a dataset from a fixed embryo with stained cell nuclei. As before, one image from the dataset is shown at the upper left. The raw data and detected membranes are again shown in grayscale and pink respectively, and the additional channel, the cell nuclei, is now shown in green.

The lower left of Fig. S4 shows a plot of the nucleus stain intensity vs. depth for the selected cell. The stain has been averaged over the cell's cross-section at each depth. The curve shows that the nucleus staining is high within a space of ~15 μm. The black dashed line indicates the slice currently selected by the user (i.e., the one that corresponds to the image in the upper left and the red marking in the lower right).

Unlike the embryos shown in Figs. S2 and S3, the embryo shown in Fig. S4 is fixed. EDGE, therefore, does not display the time-adjustment slider to the right of the image in this screenshot, and it plots measured quantities as a function of cell depth instead of time (as in the lower-left plot of nucleus stain intensity vs. depth).

Accuracy of Segmentation and Tracking in EDGE. ***(i) Accuracy experiments.*** We performed an analysis of the segmentation and tracking errors of EDGE based on the two-photon data used in this study. The accuracy of EDGE was measured by using 100 images at different times (total of 20 min at a depth $z = 20$ μm below the top of the cells and starting at time $t = -13$ min) and separately using 27 images at different depths (total of 27 μm at time $t = 5$ min starting 10 μm below the top of the cells). We prepared ground truth segmentations using the manual error correction features in EDGE, and then, in each case, we compared the ground truth segmentation to both the EDGE segmentation algorithm and the EDGE segmentation followed by EDGE automatic error correction (automatic edge splitting followed by edge addition followed by edge removal). We considered only cell outlines that were present in all images in the ground truth (for time: $N = 48$ per image, total = 4,800; for depth: $N = 56$ per image, total = 1,512). In each case, a reference image was chosen and manually corrected to form a correspondence between the cells in the ground truth and the automatic segmentations. Then, to compute accuracy scores, we considered a cell successfully tracked if it met the following two criteria: (*i*) it overlapped with a ground truth cell in the corresponding image by at least 75% (where overlap fraction was computed as the intersection of the two areas divided by the union of the two areas), and (*ii*) the cell was tracked to the correct cell in the reference image as determined by the correspondence mentioned above. Intuitively, the first criterion checks for accuracy of the segmentation, and the second criterion checks for accuracy of the tracking. Note that the above criteria are stricter than the tracking parameters used in this study, and therefore, this evaluation is conservative.

The analysis shows that EDGE is very effective in segmenting and tracking cells in our data; essentially, all cells were successfully tracked in the considered time (or depth) window. Fig. S5 illustrates the results of this analysis. In Fig. S5, green indicates successful tracking, and red indicates failures. In the spatial experiment, the EDGE segmentation algorithm (Fig. S5*A*) achieved 98.4% accuracy (error rate = 1.6%), and the EDGE segmentation plus automatic error correction (Fig. S5*B*) achieved 99.7% accuracy (error rate = 0.3%). In the temporal experiment, the EDGE segmentation algorithm (Fig. S5*C*) achieved 98.5% accuracy (1.5% error), and the EDGE segmentation plus automatic error correction (Fig. S5*D*) achieved 99.5% accuracy (0.5% error). Fig. S5 shows that, in both experiments, all of these errors are isolated mistakes, and no cells were permanently lost in the window shown. This result is possible because EDGE is capable of skipping over missing cell outlines during tracking (the extent to which this skipping is done is specified by the user) as mentioned above.

(ii) Limitations. Despite the high accuracy of EDGE on the dataset from this study, there are several limitations to the segmentation. First, the signal-to-noise ratio (SNR) of the raw images must be sufficiently high, and therefore, the membrane detection algorithm is able to find the membranes. However, EDGE performs very well even with relatively low SNR, which is shown in Fig. S6. On the high SNR data, EDGE makes very few errors; on the low SNR data, the program still manages to segment most cells, but some errors are apparent. However, as discussed above, these errors are typically isolated failures, and tracking is generally able to skip over these failure cases and still create full 4D cell reconstructions. Noise in the form of other objects in the image can also be problematic, especially if these other objects contain the same fluorescent marker as the membranes.

Other problems can arise, such as in images containing cells with highly nonconvex cross-sections. The algorithms that convert the binary membrane image to the polygon representation assume that, if the vertices are sorted by angle using the cell centroid as the origin, then the sorted vertex list is also the order in which they are connected. Cells with sharp corners tend to violate this rule. The EDGE segmentation also assumes that all cells are contiguous objects within an image. Therefore, EDGE will not work with highly curved cells that have two or more noncontiguous cross-sections within a particular image slice.

Postprocessing and Supplemental Information on Analyses Performed in the Present Study. ***(i) Estimating cell limits at apical and basal ends.*** EDGE provides 4D reconstructions of individual cells in terms of stacks of polygons. Fig. S7*A* shows an example of quantitative measurements obtained from these reconstructions. The areas depicted are the areas of the individual polygons in a given cell as a function of depth and time. Each area was assessed independently. The measurements are consistent throughout the bulk of the cell but become more noisy toward the apical and basal cell limits. This result is mostly caused by relatively weak labeling intensity at the apical surface and basal cell ends (because of the E-cadherin–based marker that was used). In particular, for cells tilted with respect to the axis perpendicular to the image plane, measurements were more difficult. This section and the subsequent sections address this problem.

The first step is to estimate the proper cell limits at each point in time. Our procedure for this step is illustrated in Fig. S7*B*. The idea is to move from the bulk of the cell to the cell limits and monitor changes in membrane labeling intensity. Within the bulk of the cell, the labeling intensity close to the membrane is high but relatively low in the interior of the cell. After the cell limit is reached, the label intensity varies less across space. This transition robustly marks the cell limit.

To implement this method, we first computed the cell axis, a curve running apically–basally through the approximate cell's center at any given depth. At each given depth z_i in the image stack, we calculated the center of mass $\vec{c_i}$ of the five nearest polygon centroid 3D positions. This center, together with the direction $\vec{v_i}$ obtained by the first principle component of the same five centroids, defines a linear regression to the five points in 3D space. The location of the cell axis at depth z_i was defined as the intersection of this regression line with the x,y -plane at depth z_i. The upper- and lowermost three centroids were discarded. Note that this procedure interpolates the centroids inside the bulk of the tracked region and extrapolates beyond it, and therefore, the obtained cell axis runs through all depths in the image stack. After the cell ends are estimated properly, a refined version of this cell axis is computed for additional analysis (see below).

At each depth (inside and outside of the tracked region), we then estimated a polygon used to monitor the labeling intensity at this depth. The polygon was estimated from the union of the four nearest tracked polygons after being shifted to its depth. Polygons are divided into an interior region and a membrane region, each occupying 50% of the polygon area. Marching along the cell axis, the average label intensities inside the two polygon subregions were monitored for each depth. Within the cell limits, the intensity is larger in the membrane than the interior region. Outside the cell, the labeling is similar in the two regions. The label difference Δ is, therefore, large inside the cell and fluctuates around zero outside the cell. The transition points mark the cell limits. These transition points are calculated from the peaks in the derivatives of the label difference Δ with respect to depth (Fig. S7 *C* and *D*). In tests, we found that cell limits obtained by this method deviated, on average, by 1.2 μm from the limits identified by visual inspection (Fig. S7*E*).

(ii) Repairing cell ends. After the cell limits are obtained, the polygon stacks representing individual cells were adjusted. In our analysis, we chose tracking parameters, and therefore, in the majority of cases, EDGE overtracked cells, with polygons reaching beyond the identified cell limits. In this case, we eliminated all polygon pixels beyond a plane that was perpendicular to the cell axis at the cell limit. For cells in which the polygon stack did not cover the total range between cell limits, an automatic reparation of the missing cell parts was performed, which is illustrated in Fig. S8. Starting from the outermost tracked points, we proceeded to the cell limits (Fig. S8*B*) and collected all pixels with high-labeling intensities (Fig. S8*C*) inside polygons, which were computed as in Fig. S7. For the upper end, pixels with intensity below 0.5 of the maximal intensity of all collected pixels were discarded; for the lower end, pixels with intensity below 0.7 of the maximum were discarded (Fig. S8*D*). Each resulting cloud of voxels in 3D was then interpolated by computing its convex hull (Fig. S8*E*). Finally, these volumes were integrated into the cell's polygonal representations.

(iii) Computing cell axis and properties as a function of depth. The cell axis provides a 1D coordinate system in each cell. Based on the improved reconstruction in the previous sections, the cell axis was reestimated for each cell at each time. At each given depth inside the cell, the plane perpendicular to the cell axis was extracted. The center of mass of the cross-section of this plane with the cell was computed. The refined cell axis was obtained by the chain of centers of mass. Through this procedure, small deviations of the axis from the cell's center line (particularly at the cell ends) were reduced. Thus, this adjusted cell axis runs from the apical to the basal end of the cell and passes roughly through the center of the cell's cross-section at any given depth. All depth-dependent measurements are obtained based on this cell axis. Cell length is the length of this axis from the apical to the basal cell limit.

(iv) Postprocessing and error estimation. Fig. S9*A* shows an analysis of the same cell as in Fig. S7*A*. The area depicted is the area of the cross-section perpendicular to the cell axis as a function of cell depth and time. Zero depth corresponds to the apical cell limit. At any given time, values range from the apical to the basal cell limit, and therefore, the depth at the basal limit corresponds to the cell length. The estimated cell length generally increases continuously over time but also, occasionally, shows heavy fluctuations from one point in time to the next. Because the changes between two subsequent time points should be small, these fluctuations were identified as reconstruction errors and discarded from the measurement traces. The changes between two subsequent time points can then also be used to estimate the errors for the measurements.

To discard the time points contaminated by strong reconstruction errors, we first compared the cell lengths in an ensemble of reconstructed cells from a given embryo to their median (Fig. S9*B*). We then discarded all time points in any given cell deviating more than 10 μm from the median (which has been smoothed by local robust regression using weighted linear least squares and a second-degree polynomial model of a length of 13 frames). This procedure eliminated strong outliers (Fig. S9 *C* and *D*, dark blue dots) and is based on the assumption that cell length does not vary substantially across the local cell population being considered. In a second step, we eliminated time points based on individual measurement traces. For this step, we focused on the *z*-position (depth in image stack) of the most apical (Fig. S9*C*) and most basal (Fig. S9*D*) points in a given cell. We computed linearly interpolated low pass-filtered versions of these quantities (again, based on regressions with second-degree polynomials but of a length of 33 frames) (Fig. S9 *C* and *D*, red curves) and discarded all time points with values deviating more than 3 μm from these curves (Fig. S9 *C* and *D*, green dots). Finally, for the case of cell length and cell volume, we eliminated additional outliers based on their fluctuations across subsequent measurements. For any given cells, we smoothed cell length/volume as a function of time by local robust regression with a second-degree polynomial [kernel size = 23/33 frames (length/volume)] and eliminated all time points in the remaining raw trace for which the length/volume deviated more than a given threshold from these smoothed curves; the threshold was chosen as 1.0/0.5 SD of length/volume differences between two subsequent frames. For any given cell and measured property, this procedure resulted in a curve consisting of the subset of measurements from well-segmented cells.

From these curves, we estimated the measurement error for a given property from the fluctuations (SD of differences) between subsequent values. We found that the error for cell length was 1.5 μm and the error for cell volume was 97 μm^3. The relative errors of these properties and others were in the range of 5–10%.

(v) Estimating the time point at which gastrulation starts (t = 0) and aligning different embryos in time. To align different WT embryos in time, we used the drop in apical surface area as an indicator for the beginning of gastrulation (Fig. S10*B*). We fitted the sigmoidal function (Eq. **S2**)

$$f(t) = a_1 \frac{1}{1+\exp((t-a_2)/a_3)} + a_4 \qquad \textbf{[S2]}$$

to the average apical surface area in each embryo (least squares fit). Here, t denotes time, and a_1, a_2, a_3, and a_4 are fit parameters. a_2 approximates the point in time of maximal negative slope. a_3 determines the scale over which the surface area decreases. We defined the beginning of gastrulation as (Eq. **S3**)

$$t = a_2 - 4a_3, \qquad \textbf{[S3]}$$

and called this time point $t = 0$ in the manuscript. As shown in Fig. S10*B*, this time point marks the moment in a given embryo when cells near the midline start constricting apically.

All curves in Fig. S10 are aligned at $t = 0$. *cta; T48* mutants were aligned manually based on the average cell length (Fig. S10*A*). Embryos imaged with a confocal microscope (Fig. 5) were aligned with respect to each other in time using the method described above for the two-photon WT data. To enable a precise comparison of the volume fluxes in the two datasets (Fig. 5*G*), $t = 0$ was shifted by ~30 s in the confocal data. This shift was estimated from matching the two curves in Fig. 5*G* based on visual inspection. The yolk stalk data (Fig. S13) was aligned to the two-photon data based on visual inspection of the onset of lateral movement of cells.

(vi) Cell length inflection point at the beginning of gastrulation. The inflection point in cell length at the transition between cellularization and gastrulation (Figs. 2*D* and 3*A*) was estimated by fitting a linear-constant-linear curve (least squares fit) to cell length as a function of time in each cell (Fig. S11*A*). The constant part was included, because many cells exhibited a near constant length over a brief period in between the two phases of cell elongation (Fig. S11*B*).

The piecewise linear function used for fitting reads (Eq. **S4**) is

$$h(t) = c_0 + c_1[-t + t_0]_+ + c_2[t - (t_0 + \Delta t)]_+, \qquad \textbf{[S4]}$$

where $[\cdot]_+$ denotes rectification ($[z]_+ \equiv \max(0, z)$). The function h exhibits a plateau of value c_0 in the interval $[t_0, t_0 + \Delta t]$. The slopes c_1 (preplateau slope) and c_2 (postplateau slope) account for the time dependence of cell length outside of the plateau region. We provided initial conditions and constraints to the fitting routine depending on the embryo under consideration. For instance, for all cells in the embryo shown in Fig. S11, we used the initial conditions $t_0 = 14$ min and $\Delta t = 0.8$ min, and for c_0, the cell length is at t_0. Furthermore, the initial condition for c_1 was estimated by linear regression of cell length as a function of time over the interval $0\ \text{min} \le t < 11\ \text{min}$, and the initial condition for c_2 was estimated over the interval $19\ \text{min} \le t < 25\ \text{min}$. As constraints, we required $11\ \text{min} < t_0 < 18\ \text{min}$ and $0.4\ \text{min} < \Delta t < 2.4\ \text{min}$.

(vii) Joining segmentations from difference reference layers. One limitation of EDGE is that it is only able to track cells that are present in the reference image. This limitation affects cells that move into the imaging window over depth or time. In the data presented in this study, we encountered this problem while attempting to reconstruct cells over a wide range of movements. Therefore, it was sometimes necessary to use multiple reference images and stitch the reconstructions together. In the analysis presented in this study, we associated two 4D polygon stacks if their centroid–centroid distance was < 1 μm for > 5 depths for some point in time.

Additional Analyses. ***(i) Constriction starts at the apical cell end at the beginning of gastrulation and progresses basally over subsequent minutes.*** Fig. S12 illustrates, for two representative cells, how cell shape changes proceed over time after gastrulation starts. The cross-sectional area as a function of depth and how it changes over time are shown. Shortly after $t = 0$, the beginning of gastrulation, only the uppermost 5–10 μm are affected. This finding is consistent with the model that constriction is driven by a contractile actomyosin meshwork close to the apical surface. Five minutes later, the cross-sectional area displays pronounced changes close to the apical end, and signs of constriction are visible even 20 μm below the apical surface. At the same time, the basal end moves more basally (i.e., the cells elongate). These examples also illustrate why the apical 75% of a cell was used to estimate the apical volume loss (Fig. 4*A*).

(ii) Apical constriction induced leakage at the basal cell end. Our calculations indicated that, at the beginning of gastrulation, not all volume lost apically reappeared basally in the form of cell elongation. Toward the end of cellularization, the furrow canal at the front of the progressing membrane widens, and in each cell, a contractile actomyosin ring subsequently reduces the membrane-free part at the basal end, which is called the yolk stalk. We measured yolk stalk area in myosin-GFP–labeled embryos (Fig. S12*A*). Embryos were aligned in time based on the onset of lateral movement of cells, which coincided well with the onset of apical constriction. We found that, around $t = 0$, the yolk stalk area measured about 75% of the basal surface area, and yolk stalks did not close within the first 10 min into gastrulation (Fig. S12*B*). Thus, cellularization was not completed at the time gastrulation started, and this finding is consistent with the observation that the cell volume continued increasing during this period (Fig. 2*C*).

The presence of an opening at the basal cell end during the early phase of gastrulation when cells already start constricting apically may cause cytoplasm to leak. In fact, for many cells, we observed that strong apical volume loss often coincided with a rapid pronounced drop in total volume (Fig. 4*B* and Fig. S12*C*). Volume rate and apical volume loss showed a strong negative correlation during the early phase of gastrulation (Fig. S12*D*). To estimate the amount of leakage, we identified the subset of cells that showed only little apical volume loss (in general, this subset changed with time). We used the volume rate based on this subset to estimate the true volume change caused by cellularization. As shown in Fig. S12*E*, this volume rate (purple curve) was consistently larger than the volume rate based on all cells (green curve) but similar to the volume rate observed in *cta; T48* mutants (bright green dashed line), in which cells showed only little apical volume loss (Fig. 4*E*). We estimated the leakage rate by the difference between the purple and the solid green curves (orange curve). This leakage only describes the average net in- or outflow per minute. The outflow increased around $t = 0$ and decayed close to zero at the time that the yolk stalk closed. The part of volume flux at the basal end that is translated into lengthening is given by the apical volume loss rate plus the leakage rate. Transforming this volume rate into lengthening results in a smaller rate compared with the rate estimated in Fig. 4*A* (Fig. S12*F*, yellow curve).

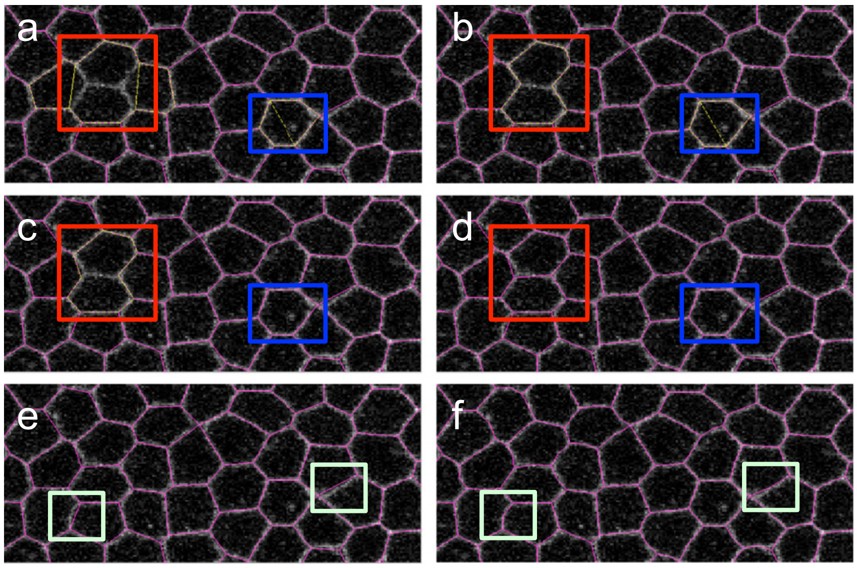
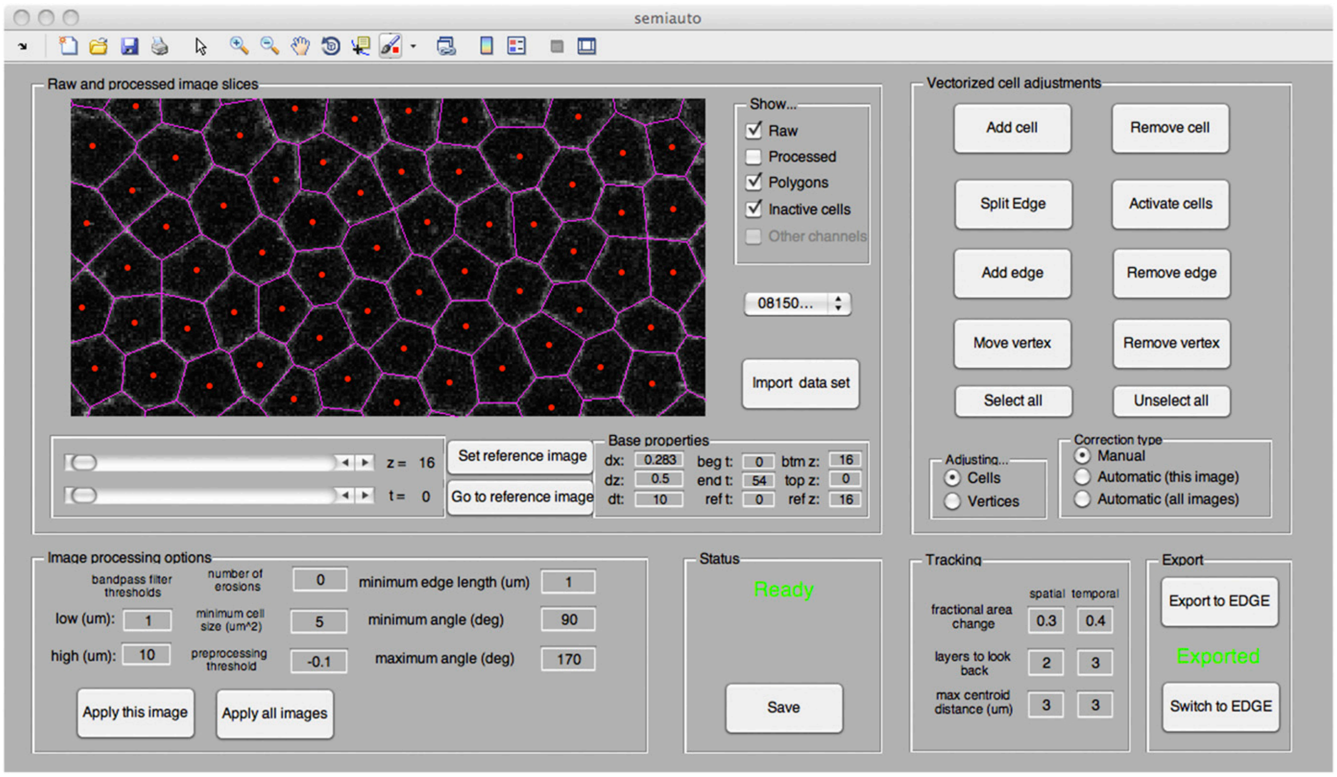


Fig. S1. Steps of automatic error correction. (*A*) Original image after membrane detection stage. Two errors occurred in the processing. In the red box, two neighboring cells are merged into one. In the blue box, a cell was split into two. With an appropriate set of tracking parameters, the program detects that these cells cannot be tracked (i.e., matched with those cells at the previous depth), and thus, it draws the cells in yellow. (*B*) Additional vertices are added so that cell polygons more closely match the detected boundaries in binary image. This step is important for correcting the merge error in the red box, because in addition to losing an edge, the error also causes the two vertices to be lost. (*C*) The split error in the blue box is repaired automatically by removing an edge, resulting in improved correspondence with the previous image. (*D*) After additional vertices are added, the merge error in the red box is fixed automatically by adding an edge between two vertices. The edge is added in the location that maximizes correspondence with the previous image. (*E*) The same image as in *D* but with different highlighting. The light green boxes show two edges with curvatures that are not captured well by the polygon approximation. (*F*) The automatic edge splitting function is used again, but for this step, error correction parameters are set differently to allow more splits. One additional vertex is automatically added in each of the light green boxes, resulting in polygons that form better approximations of the curved borders. Note that the edge splitting functionality (used in generating *B* and *F*) is based on the raw (prepolygon) binary segmentation of this image, unlike the other steps that are based on forming a correspondence between this image and other images in the stack (i.e., tracking).



Show...
- [✓] Raw
- [ ] Processed
- [✓] Polygons
- [✓] Inactive cells
- [ ] Other channels

08150...

z = 16
t = 0

Base properties

dx:	0.283	beg t:	0	btm z:	16
dz:	0.5	end t:	54	top z:	0
dt:	10	ref t:	0	ref z:	16

Adjusting...
- (●) Cells
- ( ) Vertices

Correction type
- (●) Manual
- ( ) Automatic (this image)
- ( ) Automatic (all images)

Image processing options

bandpass filter thresholds		number of erosions	0	minimum edge length (um)	1
low (um):	1	minimum cell size (um^2)	5	minimum angle (deg)	90
high (um):	10	preprocessing threshold	-0.1	maximum angle (deg)	170

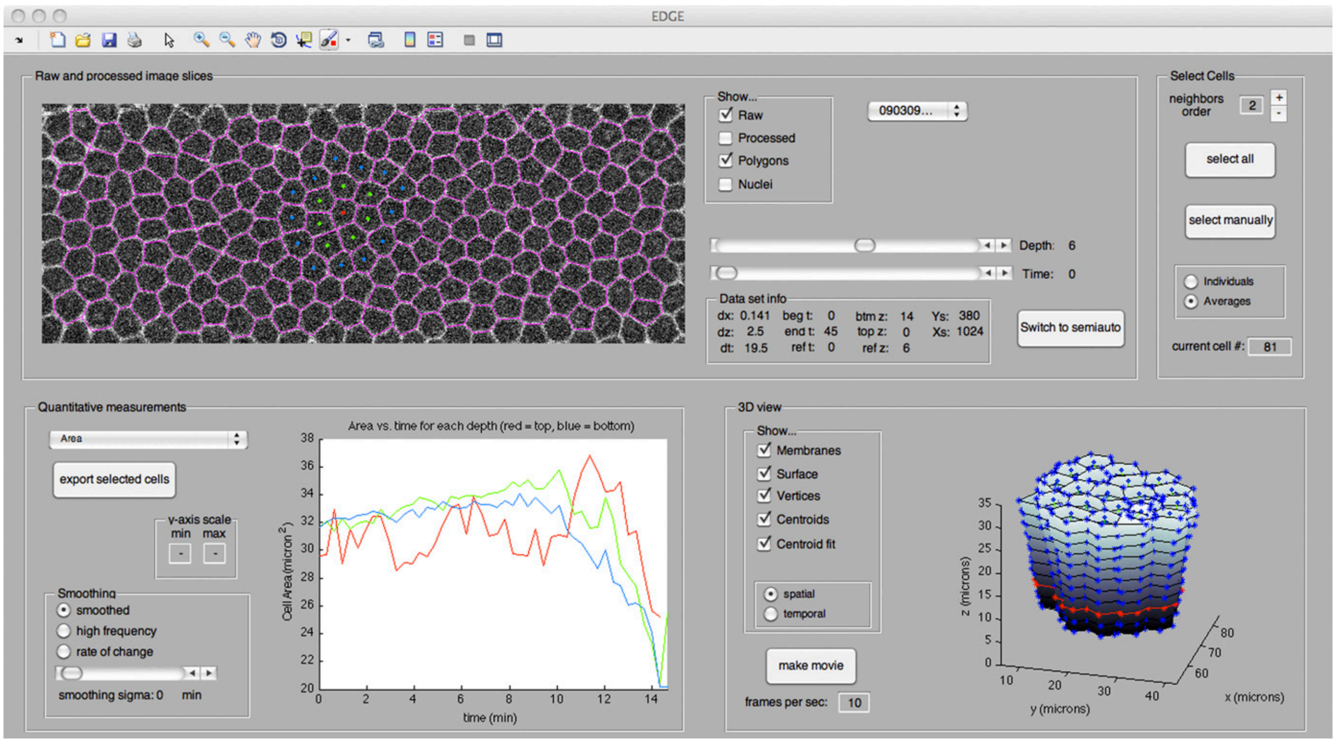
Status
Ready

Tracking

	spatial	temporal
fractional area change	0.3	0.4
layers to look back	2	3
max centroid distance (um)	3	3

Export
Exported

Fig. S2. A screenshot of the EDGE Importer showing the polygon membranes (pink) overlaid over the raw image (grayscale). The red dots show the centroids of all of the identified cells. The buttons on the right can be used to fix any errors, and the many other settings allow for control over other aspects of the processing.



Show...
- [✓] Raw
- [ ] Processed
- [✓] Polygons
- [ ] Nuclei

090309...

Depth: 6
Time: 0

Data set info
```
dx: 0.141   beg t:  0   btm z: 14   Ys:  380
dz:   2.5   end t: 45   top z:  0   Xs: 1024
dt:  19.5   ref t:  0   ref z:  6
```

neighbors order: 2
Individuals [✓ Averages]
current cell #: 81

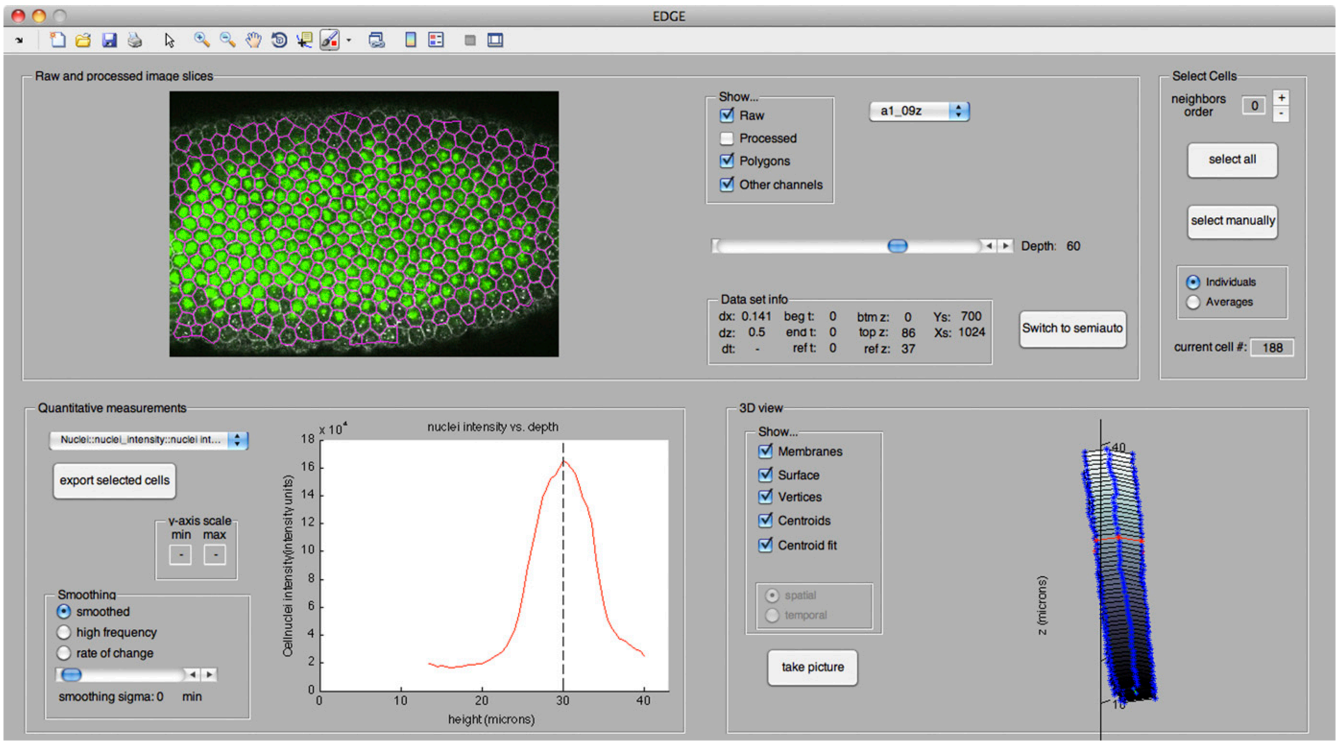
Area

Smoothing
[✓ smoothed] high frequency rate of change
smoothing sigma: 0 min

3D view — Show...
- [✓] Membranes
- [✓] Surface
- [✓] Vertices
- [✓] Centroids
- [✓] Centroid fit

[✓ spatial] temporal
frames per sec: 10

Fig. S3. A screenshot of the EDGE Browser showing its basic capabilities. The polygons in the upper left were produced with minimal manual error correction. The lower left plot shows area vs. depth as a function of time for the selected cell (red dot) and its neighbors (green and light blue dots). A 3D rendering of the selected cell and its neighbors at a selected time point is shown in the lower right.



Show...
- [✓] Raw
- [ ] Processed
- [✓] Polygons
- [✓] Other channels

a1_09z

Depth: 60

Data set info
```
dx: 0.141   beg t:  0   btm z:  0   Ys:  700
dz:   0.5   end t:  0   top z: 86   Xs: 1024
dt:    -    ref t:  0   ref z: 37
```

neighbors order: 0
[✓ Individuals] Averages
current cell #: 188

Nuclei::nuclei_intensity::nuclei int...

Smoothing
[✓ smoothed] high frequency rate of change
smoothing sigma: 0 min

3D view — Show...
- [✓] Membranes
- [✓] Surface
- [✓] Vertices
- [✓] Centroids
- [✓] Centroid fit

spatial temporal

Fig. S4. Another screenshot of the EDGE Browser. This fixed dataset has been stained with two markers, one for cell membranes and another for the nucleus. The nuclei (green) and the polygon membranes (pink) are overlaid on top of the raw data (grayscale). The bottom plot shows the polygon-averaged intensity of the nucleus channel for the selected cell as a function of stack depth. The intensity curve exhibits a clear peak, revealing the vertical position of the nucleus inside the cell.

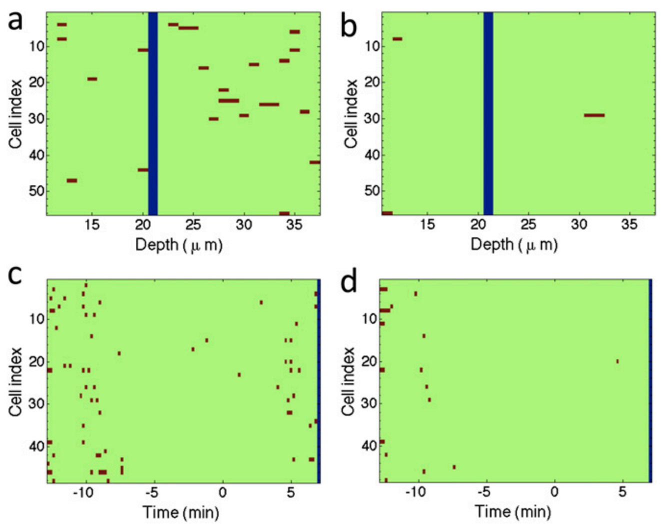


Fig. S5. Results of the experiment showing the accuracy of the EDGE segmentation and tracking across space for segmentation only (*A*), space for segmentation followed by automatic error correction (*B*), time for segmentation only (*C*), and time for segmentation followed by automatic error correction (*D*). The reference image, which is used to make correspondences between cells in the different segmentations, is shown in blue in each case. Green corresponds to successfully identified cells, and red corresponds to failures. The figures show that, in all cases, although EDGE occasionally fails to identify cells at isolated depths or times, no cells are lost entirely over the window because of the robustness of the tracking algorithm. Comparing *B* and *D* with *A* and *C* shows that the error rate is reduced substantially (from 1.6% to 0.3% in the spatial experiment and from 1.5% to 0.5% in the temporal experiment) by the use of automatic error correction.

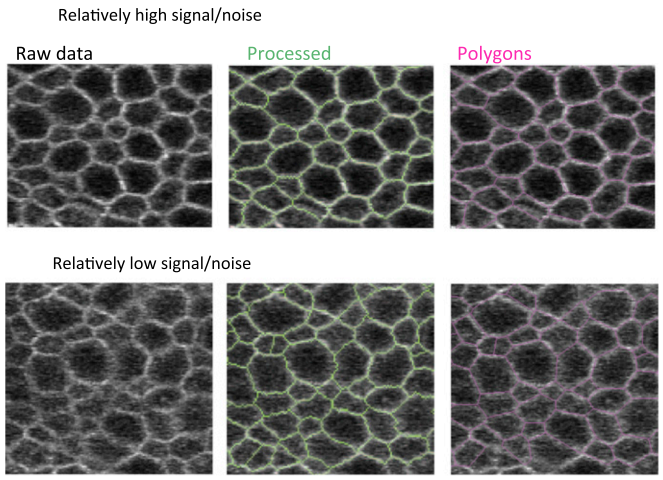


Fig. S6. Estimated cell outlines (green) and polygons (pink) for a representative image with relatively high SNR (*Upper*) and low SNR (*Lower*). Images were obtained by two-photon microscopy, with cell membranes labeled by E-Cadherin-GFP (*Materials and Methods*).

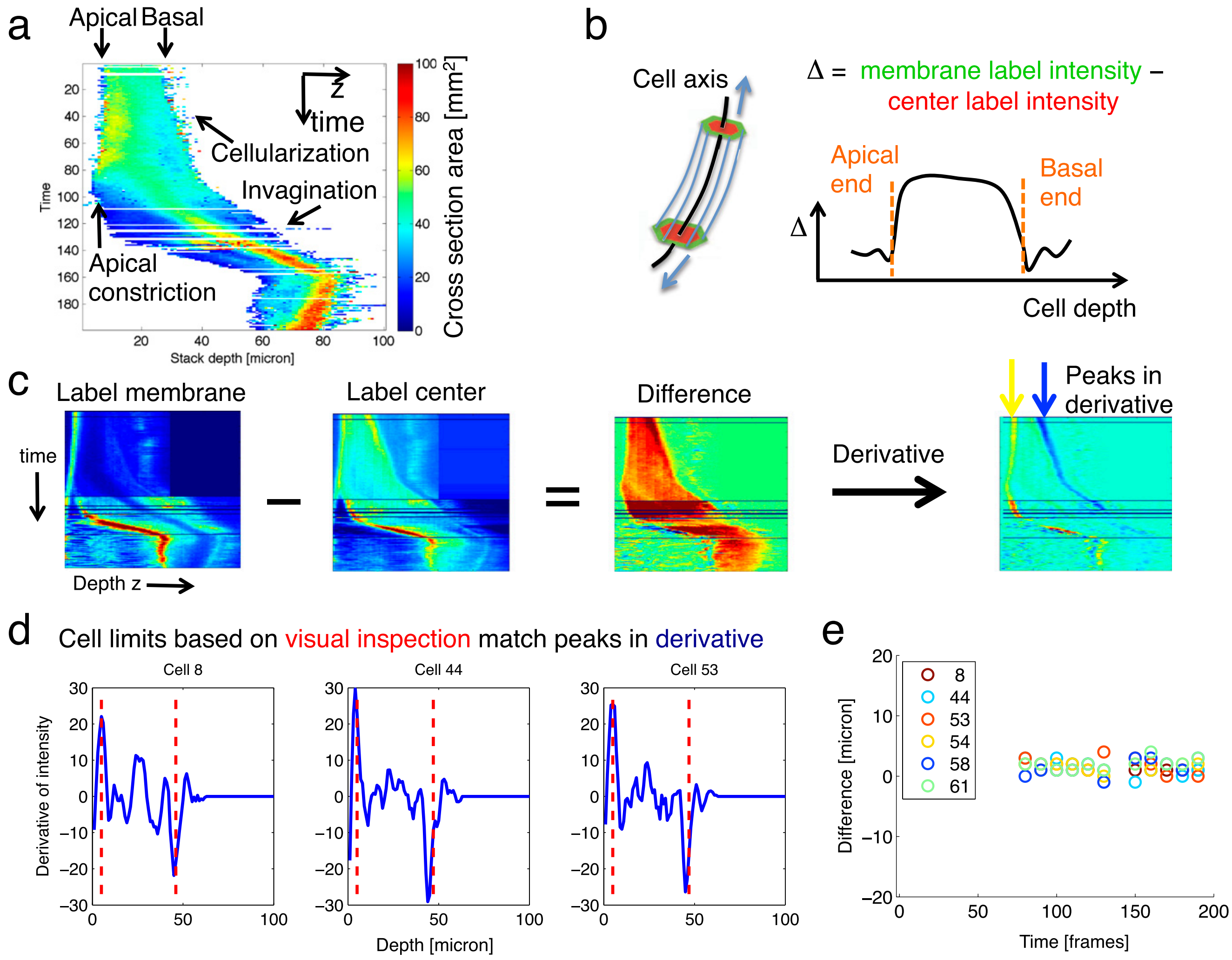


Fig. S7. Estimating cell limits at apical and basal end. (*A*) An example output of EDGE: a measurement of the cross-sectional area as a function of depth (layer in the image stack) and time for one representative cell. Note that each pixel represents an independent measurement. The measurement is consistent throughout the bulk of the cell but more noisy close to its apical and basal end. (*B*) Estimating the cell limits (schematic): a cell axis is estimated by linear inter- and extrapolation of polygon centroids (in the text). Polygons at different depths are divided into a center region (red) and a membrane region (green), each occupying 50% of the polygon area. Polygons inside and outside the bulk region are constructed by averaging over the four nearest tracked polygons. Traversing the cell axis, the average label intensities in both the interior and border regions are monitored at each depth. The label difference Δ is large inside the cell and fluctuates around zero outside the cell. The two transition points mark the cell limits. (*C*) The label intensities and their difference for the cell in *A*. The derivative with respect to stack depth shows peaks at the two transition points in labeling difference. (*D*) Derivative as a function of stack depth at a given time point for the cell in *A* and two other cells (blue solid line). Locations of peaks match well with the cell limits identified by visual inspection in the same cells (red dashed lines). (*E*) Comparison of cell limits estimated by this derivative method, and visual inspection for six cells at 10 time points. Only data for the lower basal end are shown. The mean deviation for these cells averaged over all time points and the basal and apical ends was 1.2 μm.

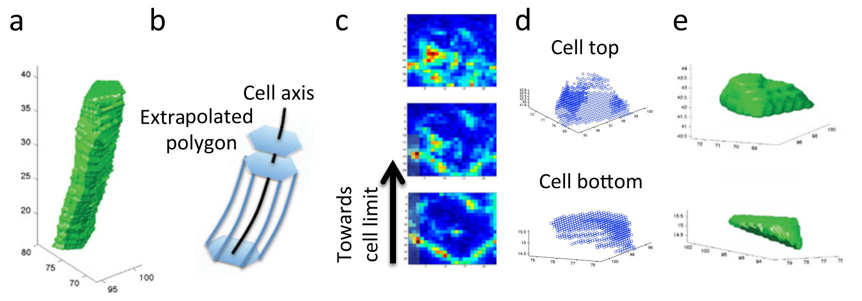


Fig. S8. Automatic reparation of cell ends. (*A*) Output of EDGE, a stack of polygons representing a given cell at a given time. Parts of the upper and lower ends have not been tracked by EDGE in this case, resulting in the cell appearing chopped off. (*B*) Extrapolating cell shapes (schematic) is based on extrapolations of the cell axis and the polygon outside the tracked cell region (both estimated as in Fig. S7). (*C*) The label intensity inside the polygon is collected, while traversing the extrapolated cell axis until the cell limits (estimated in Fig. S7) are reached. Shown are intensities at extrapolation distances 1, 2, and 3 μm. (*D*) The set of voxels with intensity above 0.5 times the maximal intensity in the whole extrapolated region for the apical part (*Upper*). For the basal part (*Lower*), a threshold of 0.7 times the maximum was used. (*E*) Convex hulls of voxels from *D*. Integrating them into *A* gives rise to a representation of the entire cell.

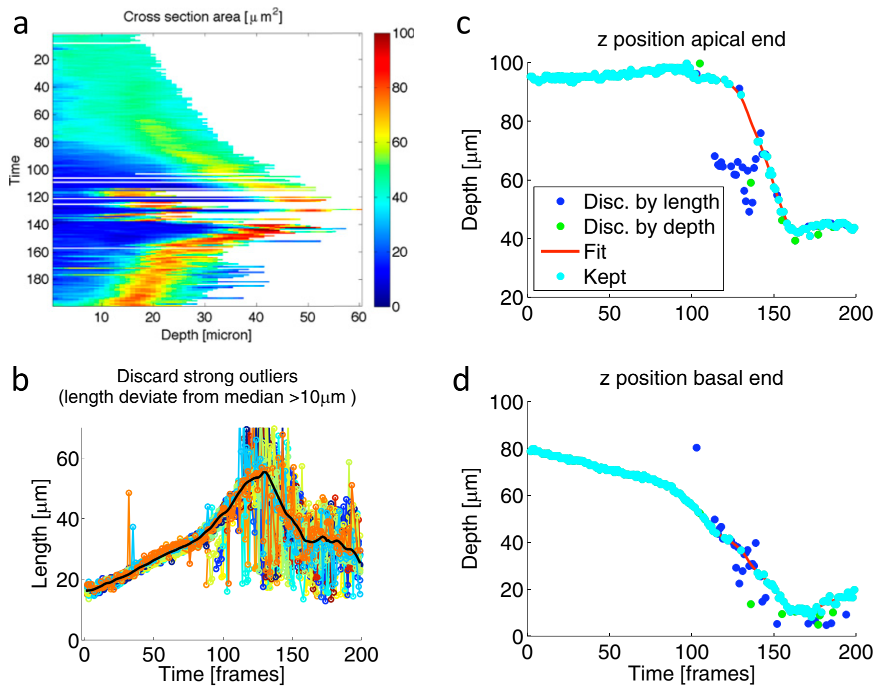


Fig. S9. Discarding outliers in measurements based on continuity. (*A*) Perpendicular cross-sectional area as a function of depth (distance from the apical surface along the cell axis) and time for the same cell as in Fig. S7*A*. Zero depth is defined by the apical cell limit. Depth is measured along the cell axis, a curve running from the most apical to the most basal part of the cell through the approximate centers of mass of a perpendicular cross-section at any given depth. Cross-sectional areas are calculated for sections perpendicular to the cell axis at steps of 1 μm. (*B*) Cell lengths as a function of time for 59 cells from one embryo. Individual cells' traces are represented by different colors, and the median (smoothed; in the text) is represented by the black curve. For a given cell, its length is defined by the length of the cell axis from its most apical to most basal point. Note the often abrupt changes in estimated length for some of the cells at some time points. (*C* and *D*) *z*-position at apical (*C*) and basal (*D*) cell ends as a function of time for the cell shown in *A*. Time points showing strong deviations in length from the average cell length at a given time were discarded from the data (dark blue dots). In a second step, time points showing exceedingly large changes in depth were excluded (details in the text).

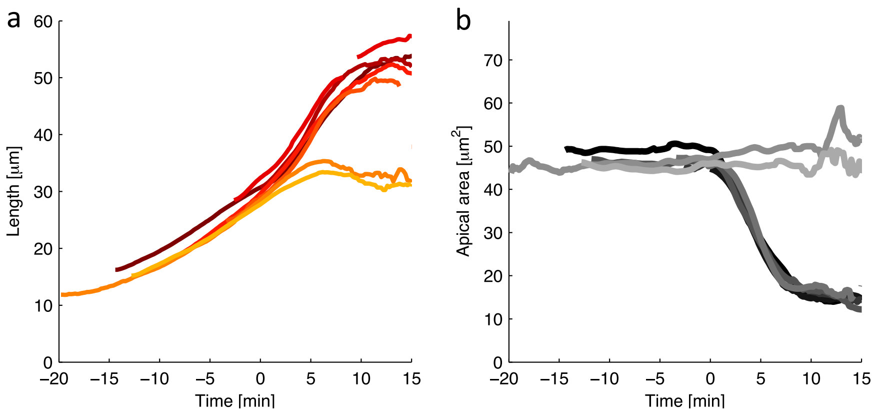


Fig. S10. Estimating the time at which gastrulation begins. (*A*) Average cell length in cells within 20 μm from the midline in the five WT embryos (red) and the two *cta; T48* embryos (orange) used in this study. (*B*) Average apical surface area in the same cells (dark gray, WT; light gray, *cta; T48*).

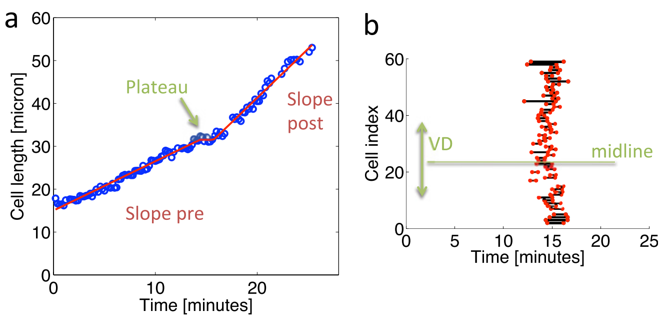


Fig. S11. Estimating the inflection point in cell length at the beginning of gastrulation. (*A*) Length of one cell as a function of time (blue circles) together with fits using a linear-constant-linear function (red lines). (*B*) Start and end point (red dots) of plateaus (black lines) for 59 cells from one embryo. Cells are sorted by the distance from their apical ends to the midline.

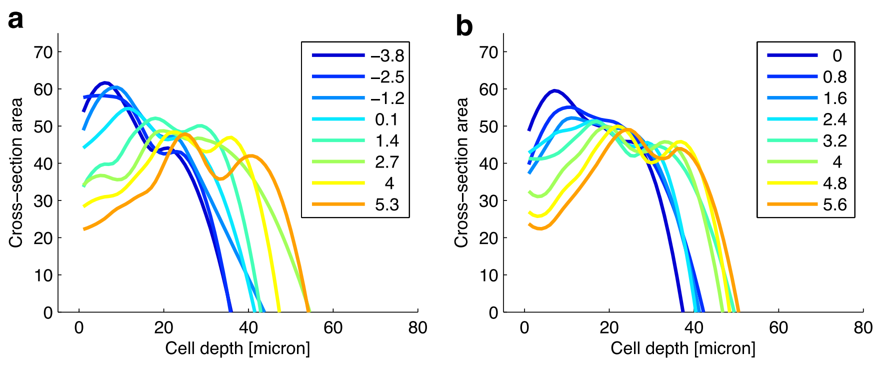


Fig. S12. Constriction initiates apically and proceeds more basally over the first few minutes of gastrulation. (*A*) Cross-sectional area as a function of depth (distance from the apical end) in a representative cell. Different curves correspond to different times as indicated (numbers denote minutes from $t = 0$). Curves were smoothed for illustration purposes. (*B*) The same as in *A* for a different cell.

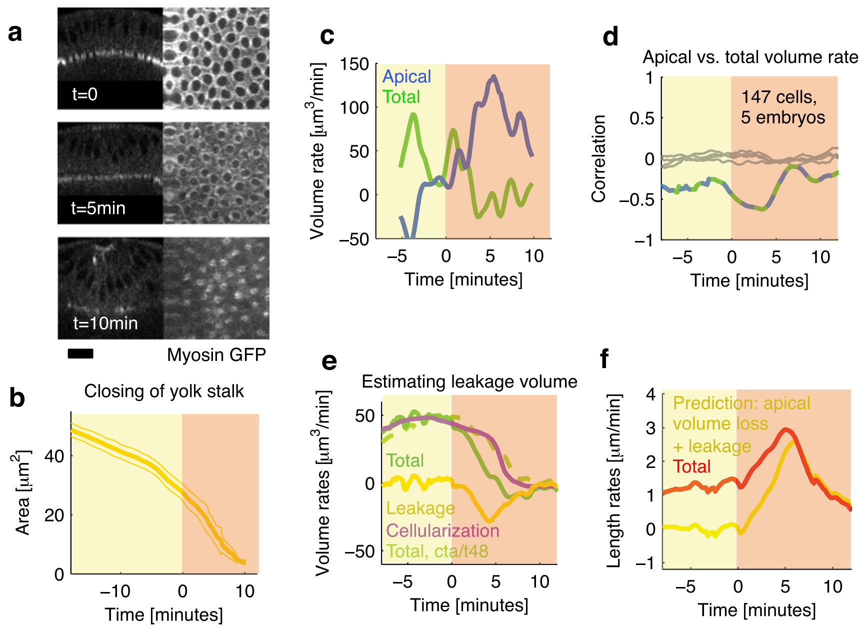


Fig. S13. Apical constriction induced leakage at basal cell end. (*A*), Myosin-GFP images obtained by two-photon microscopy to visualize yolk stalks (*Left*, *yz* cross-sections; *Right*, maximum projections of the basal *z*-slices). (Scale bar: 10 μm.) (*B*) Yolk stalk area in *A* estimated by EDGE (thick line, mean; thin lines, ±SD). (*C*) Apical volume loss rate (blue) and the rate of change in total volume (green) for a representative cell. (*D*) Cross-correlation (zero lag) of the two quantities in a sliding window of length of 6 min (147 cells, five embryos; gray lines, control: data randomly shifted in time). (*E*) Estimated true rate of change of volume caused by cellularization (purple), the measured volume rate (green), their difference, and the estimated volume leakage rate (orange; in the text). The measured volume rate in *cta; T48* mutants is shown in dashed light green. (*F*) Elongation rate predicted by apical volume loss taking basal leakage into account (yellow) compared with the measured elongation rate (red).