Supporting Information

Xie et al. 10.1073/pnas.1018383108

SI Text

The time-lapse images were processed in three stages: segmentation, aggregate linkage, and data analysis. The image processing was performed with Matlab using the Image Processing Toolbox (IPT) and in-house functions.

Image Segmentation. In the segmentation stage, the movie is processed frame by frame. Each frame image is segmented into foreground (aggregates) and background. The segmentation process has five steps: preprocessing (lighting correction), first Markov random field (MRF)-based segmentation, morphological operation (open), second MRF segmentation, and final adjustment. A workflow illustration is shown in Fig. S2, and each step is explained in detail in the following subsections.

Lighting illumination correction. The segmentation stage aims to separate aggregates from one another and from the interaggregate background based on grayscale intensities. The processing starts with illumination correction to make the background intensity even throughout the image. We assume that the illumination distortion generates a smooth surface. Therefore, the idea of illumination correction is to fit a smooth surface on the background pixels and then, subtract this illumination surface from the original image to get the illumination-corrected data.

The entire image is 1,200 × 1,600 pixels. First, divide the whole image into small patches of 200 × 200 pixels. On each patch, use the IPT function graythresh to find a threshold and then, increase the threshold by 20% to reduce the contribution of segmentation errors to the background. Next, the resulting threshold intensity is used to separate the image patch into foreground and background. A background mask is then obtained from the combination of all of the patches and used to extract background pixels from the whole image. A smooth 2D surface is then fit to the resulting background intensity with the third-party Matlab function package gridfit (1). This surface is, thereafter, subtracted from the original image.

First MRF segmentation. The MRF-based segmentation method is applied to segment the aggregates from the surrounding area. Markov random field models provide a robust and unified framework for segmentation problems. The segmentation is a labeling process in which a label $f_p \in L$ is assigned to each pixel $p \in P$. It is assumed that the labels should vary smoothly almost everywhere but may change dramatically at some places such as object boundaries. The quality of labeling is given by an energy function that will be minimized. Finding a labeling with minimum energy corresponds to the maximum a posteriori probability optimization problem. To approximately solve this optimization problem, the fast belief propagation approach (2) is applied to define the MRF energy function. As a result, an approximate labeling with a minimum cost of the energy function is obtained. The segmentation algorithm is implemented in Matlab. To accelerate the process, we again divide the illumination-corrected image into small patches of 200 × 200 pixels, apply graythresh to find a threshold, and use this threshold to get an initial binary segmentation of aggregates and background. This initial segmentation is used as the initial condition for MRF and applied to each patch.

Morphological opening operation. The initial segmentation by MRF contains multiple aggregates that are very small as well as aggregates that are connected by narrow segments. To improve uniformity, imcomplement is used to make the aggregates white for the foreground, and then, IMT function imopen(bw,ones(11)) is used to remove connections less than 11 pixels across, bwareaopen(bw,300) is used to ignore aggregate regions with fewer than 300 pixels, and imcomplement is used to reverse the image. bwareaopen(bw,50) is applied to ignore background regions with fewer than 50 pixels. As a result, false detection and false connections are removed.

Second MRF segmentation and final adjustment. Because these operations may also remove the true targets, we used the resulting segmentation as an initial condition and performed a second-iteration MRF segmentation. The final adjustment is also a morphological operation step, with bwareaopen(bw,300) used on both background and foreground to remove small objects.

We found that such two-step processing reliably detects aggregates at the late stages of aggregation. To ensure that segmentation errors do not significantly affect our analysis, a starting frame was chosen so that the variation in the numbers of aggregates was limited frame to frame (the running variance with 15 frames was no more than five aggregates). The chosen time of the starting frame corresponds to ~13.5 h development and depicts a time at which quasistable aggregates reliably detectable by the proposed algorithm are formed.

Link aggregates frame by frame. The aggregate linkage stage aims to track aggregate movement, shrinkage, and expansion frame by frame. The linkage algorithm is developed based on two observations.

First, the initial aggregates can merge, split, or disperse, and new aggregates might appear. Second, aggregates do not move or only move a little bit from one frame to the next (5 min real time).

Each aggregate in the starting frame has been numbered, and the corresponding aggregates on the following frames are traced and recorded until they disperse or until the end of the movie. The linkage method is based on the overlap and the distance between the centroids of aggregates in different frames. The logic is briefly summarized as follows.

i) If an aggregate has less than 10% overlap with any aggregate in the previous frame, a newly appeared aggregate is assumed and added to the list.

ii) If an aggregate overlaps with some aggregates in the previous frame, then one of the following three scenarios occurs.

  a) If the distance between the centroids of this aggregate and the overlapped aggregates in the previous frame is less than a threshold of 23 μm or 10 pixels, then this aggregate is assumed to be the same aggregate as the one in the previous frame, and it is recorded at a new time index as the same one with an updated centroid location and area size.

  b) If the distance is larger than the threshold, splitting or merging is assumed to occur. In this case, a new merged or two new split aggregates are added to the list at this time index with their centroids, area size, and a pointer directing to the original aggregates from the initial list.

  c) The original aggregates at this time index are also updated to indicate merging if the number of overlaps is one or splitting if the number of overlaps is more than one.

Finally, based on the linkage map, all aggregates on the starting frame are labeled as dispersing, merging, or splitting. A segmentation result with a color-coded label is shown in Fig. S3. Red indicates steady aggregates, yellow indicates dispersing aggregates, blue indicates merging aggregates, and cyan indicates splitting aggregates. The linkage is manually reviewed and curated to ensure that the algorithms function properly, especially for relatively rare events of splitting and merging.

Feature Definitions. After segmentation and linkage, aggregates on a specific frame are labeled as dispersing, merging, splitting, or stable. Feature extraction is performed on the 2D black and white image. Matlab IPT function regionprops measures a set of properties for each connected component in the binary image. Each segmented aggregate is a connected component, and regionprops is used to extract image features for aggregate fate analysis.

The following is a list of eight features (the feature index number is the same as that in Table 1) from the regionprops properties.

1) Perimeter: scalar; the distance around the boundary of the region (i.e., a segmented aggregate).
2) Equivdiameter: scalar that specifies the diameter of a circle with the same area as the region.
3) Area: scalar; the actual number of pixels in the region.
4) Orientation: scalar; the angle between the x axis and the major axis of the ellipse that has the same second moments as the region.
5) Equivalent diameter/perimeter.
6) Solidity: scalar specifying the proportion of the pixels in the convex hull that are also in the region.
7) Eccentricity: scalar that specifies the eccentricity of the ellipse that has the same second moments as the region. The value is between zero and one.
8) Minor axis length/major axis length: minor or major axis length is a scalar specifying the length (in pixels) of the minor or major axis of the ellipse that has the same normalized second central moments as the region.

Regionprops property centroid provides a vector that specifies the center of mass of the region (i.e., an aggregate). The displacement relationship of all aggregates in the field can be extracted based on their centroids. Then, the nearest neighbor (NN) of a target aggregate and immediate neighbors at various directions around the target aggregate can both be extracted. Fig. S4 illustrates the relationship of the target aggregate (TA), the nearest neighbor (NNA), and the immediate neighbors at various directions around the target (1, 2 … 5).

The following is a list of 25 features (the feature index number is the same as that in Table 1) calculated based on the NN or immediate neighbors.

1) Distance to the NN/average weighted area of neighbors (i.e., weighted sum of all areas of neighbors within a circular neighborhood with one/distance being a weight).
2) Average distance to neighbors.
3) Distance to the NN.
4) Equivalent diameter/distance to the NN.
5) Area/area of the NN.
6) Area/neighboring area.
7) Area/sum of the area of neighbors.
8) Equivalent diameter/average weighted area of neighbors.
9) Area/maximal area of neighbors.
10) Area/median area of neighbors.
11) Area/average area of neighbors.
12) Area/minimal area of neighbors.
13) Distance to the NN/equivalent diameter.
14) Neighboring area (area of circular neighborhood filled with aggregates).
15) Sum of distance to neighbors.
16) Weighted sum area of neighbors.
17) Sum of equivalent diameter of neighbors.
18) Sum of area of neighbors.
19) Area of the NN.
20) Minimal area of neighbors.
21) Average weighted area of neighbors.
22) Median area of neighbors.
23) Maximal area of neighbors.
24) Average equivalent diameter of neighbors.
25) Average area of neighbors.

Feature Clustering. A set of 33 features encompassing multiple aspects of each aggregate was automatically detected for more than 150 aggregates from the last frame of a time-lapse movie. The Matlab statistics toolbox (ST) function corr (Spearman type) is used to compute a correlation coefficient matrix between feature values. The Spearman rank correlation assesses how well the relationship between two variables can be described using a monotonic function, linear or nonlinear, to capture the interdependence of two different features such as area and equivalent diameter.

The ST function linkage creates an agglomerative hierarchical cluster tree from the correlation coefficient matrix. We use linkage(CCoeff,'average','euclidean') to get four major classes and apply the ST function dendrogram to draw the clustering tree shown in Fig. 2. The features divide into four major clusters representing features associated with the proximity of the aggregate to neighbors (1–3), various size parameters (4–15), parameters of the aggregate's neighbors (17–28), and image shape and topology (30–33) (Fig. 2).

Mutual Information and Support Vector Machine Analysis. Having feature and fate labels for all aggregates, we used information theory and machine learning approaches to connect features and fate (3). Normalized mutual information, $NMI = I(D,F_i)/H(D)$, was used to find single features that correlate with aggregate fate (Eqs. **1** and **2**). To estimate mutual information, the probability distribution of aggregate dispersal $p(d)$, a probability distribution of the given feature ($i = 1 \ldots 33$) $p(f^i)$, and the joint probability distribution between the dispersal and a given feature $p(d,f^i)$ are all estimated based on 1D and 2D histograms using Matlab function histc.

Support Vector Machine (SVM) analysis was used to test whether multiple feature combinations can better predict the fate of an aggregate (4). SVMs have been widely suggested for binary classification. The linear SVM defines a hyperplane in the feature space, which separates the training examples of the two classes. The problem of determining the hyperplane can be formulated as a convex quadratic programming problem. If the classes are not linearly separable, by relaxing the constraints and introducing a slack parameter, a similar quadratic programming problem can be formulated and solved. The SVM analysis is based on Matlab bioinformatics toolbox (BT) functions svmtrain and svmclassify.

In the analyzed movie, there are 160 examples of steady aggregates and 91 examples of dispersing aggregates available for training and testing. Each example is represented by a multidimensional (up to 33D) feature vector. In each case, 75% of the data is used to train SVM, whereas the remaining 25% is used to estimate the prediction error rate. The data index is randomly permutated, and the experiments have been repeated 30 times using different combinations of training and testing data to reduce the training bias and estimate SE. The error rate is defined as the percentage of cases (both false positive and false negative) where dispersal is not correctly predicted by a threshold model.

Any single feature or feature combination vector can be used to run the SVM to test whether they predict the fate of transitional aggregates. Because all combinations of the 33 features are too many to test, only a selected set of combinations are tested in this analysis. First, every single feature is tested and the features are sorted based on their average prediction error from the lowest to the highest. The size related feature generates the lowest error rate, consistent with the mutual information analysis. Then, the combination of the top one to k ($k = 1 \ldots n$) features based on the sorting are tested, and the results are shown in Fig. 3*C*.

1. D'Errico J (2005) Surface Fitting using gridfit, available at Matlab Central, http://www.mathworks.com/matlabcentral/fileexchange/8998.
2. Felzenszwalb PF, Huttenlocher DP (2006) Efficient belief propagation for early vision. *Int J Comput Vis* 70:41–54.
3. Wireman JW, Dworkin M (1975) Morphogenesis and developmental interactions in myxobacteria. *Science* 189:516–523.
4. Nariya H, Inouye M (2008) MazF, an mRNA interferase, mediates programmed cell death during multicellular Myxococcus development. *Cell* 132:55–66.

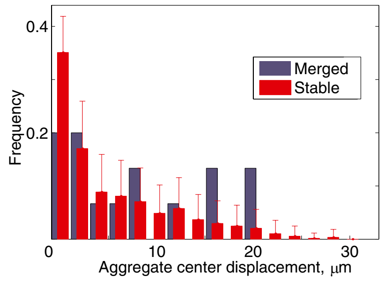


Fig. S1. Displacement distribution of merging aggregates and stable aggregates. For each merging aggregate, the displacement of its center from the starting frame (13.5 h) to the frame preceding merger is recorded. For each merged aggregate, displacement of the randomly selected stable aggregate center for the same time interval is recorded. The histogram of the resulting displacements is then computed for merged and stable aggregates. For stable aggregates, the sampling is repeated 10,000 times to compute mean histogram, and SDs are shown as error bars. For the overwhelming majority of samples (~90%), distributions of displacements of stable and merged aggregates are equivalent based on Kolmogorov–Smirnov test with $P = 0.05$.

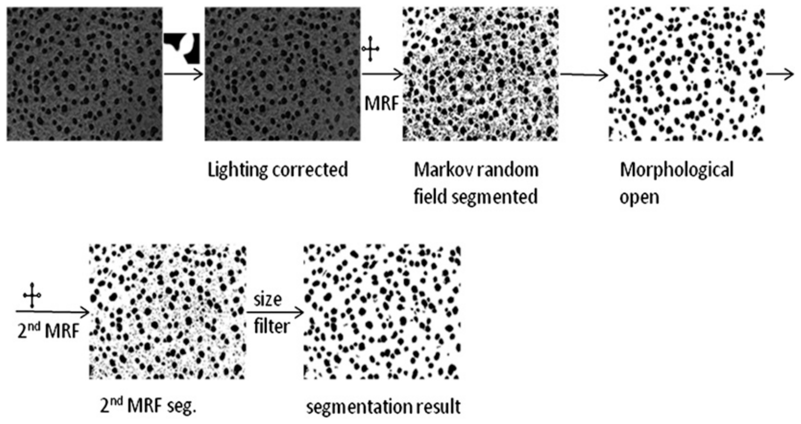


Fig. S2. The workflow illustration of the two-stage MRF-based image segmentation process.

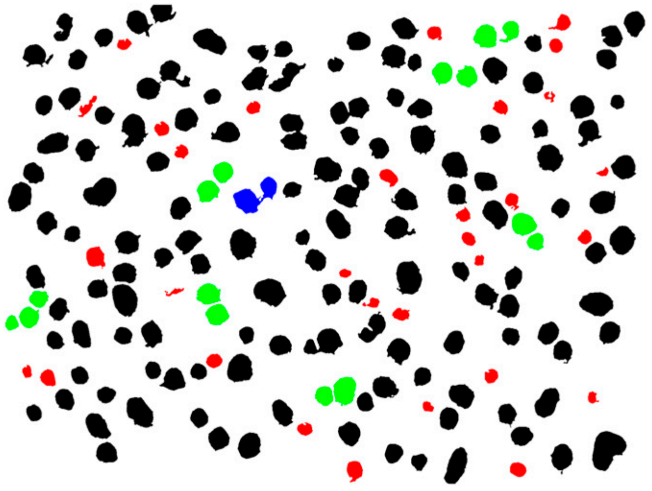


Fig. S3. A segmentation result of an aggregation image at 13.5 h with color-coded coloring (black, stable aggregates; red, dispersing aggregates; green, merging aggregates; blue, splitting aggregates).

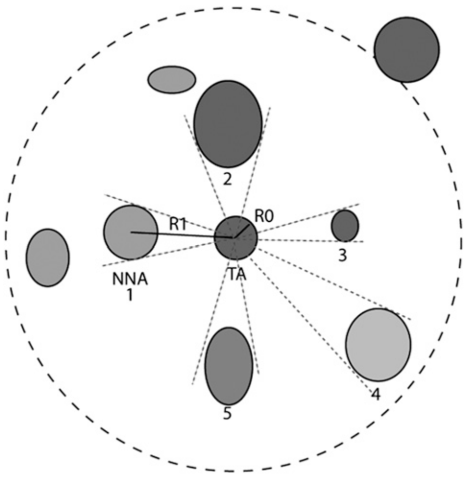
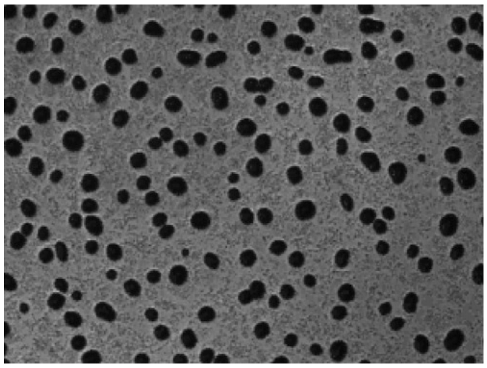


Fig. S4. The target aggregate (TA), the nearest neighbor (NNA), and the immediate neighbors (1–5) at various directions around the target. Note that, in each direction, we only consider the immediate neighbor.



Movie S1. A sample of the developmental aggregation movie used in our analysis.

Movie S1