Supplementary Material For:

Reports

Quantitative analysis of colony morphology in yeast

Pekka Ruusuvuori[1,2], Jake Lin[1,2,3], Adrian C. Scott[4], Zhihao Tan[4,6], Saija Sorsa[1], Aleksi Kallio[5], Matti Nykter[5], Olli Yli-Harja[1,2], Ilya Shmulevich[1,2], and Aimée M. Dudley[4,6]

[1]Department of Signal Processing, Tampere University of Technology, Tampere, Finland, [2]Institute for Systems Biology, Seattle, WA, [3]Luxembourg Centre for Systems Biomedicine, University of Luxembourg, Luxembourg, [4]Pacific Northwest Diabetes Research Institute, Seattle, WA, [5]Institute of Biomedical Technology, University of Tampere, Tampere, Finland, [6]Molecular and Cellular Biology Program, University of Washington, Seattle, WA

BioTechniques 56:18-27 (January 2014) doi 10.2144/000114123

Keywords: colony morphology; image analysis; software; yeast; phenotype; time-lapse

Supplementary Methods

Arraying colonies using a FACS sorter

It has been observed that colony morphologies are affected by the distance to neighboring colonies. In order to standardize the spacing between colonies, single cells were deposited in a grid pattern by using a FACSAria II cell sorter (BD Biosciences) with an Automated Cell Deposition Unit. Cells are arrayed following a 96-well format, but in order to increase the spacing of colonies, alternating wells were skipped. This results in colonies that are spaced approximately 12.7 mm from each other.

The plates used in this experiment are rectangular OmniTrays (Nunc) filled with 30 mL of YPD + agar (2% glucose). For each strain, 48 cells are deposited per plate.

Time-lapse imaging

The 6 plates used in this study were placed in a 2 × 3 grid (Figure 1), face up under a heavy glass plate with their lids removed. Each plate was divided into 16 regions, each containing 3 colonies. At every time point, one image was captured per region. The camera was moved to each region on the plate with a custom-built 2-axis gantry. The gantry has two linear slides, each driven by a leadscrew and stepper motor assembly. The stepper motors were each driven by an EasyDriver, which was in turn controlled by an Arduino UNO microcontroller. Scripted movement commands are sent to the microcontroller via a Java application, which also controls the shutter

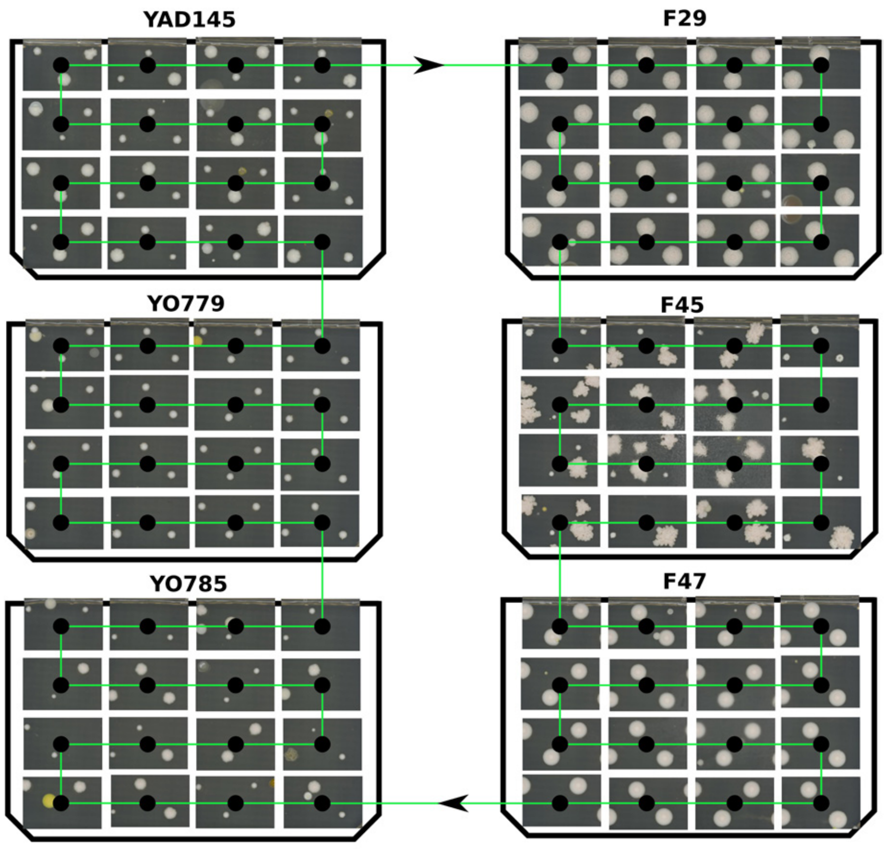


Figure 1. Time-lapse plate layout. OmniTrays with 48 colonies each were placed in a 2 × 3 grid and imaged continuously using a camera mounted to a custom built robotic device. Each plate was divided into 16 regions (centered on the black dots above) and 1 picture was taken of each region for each time point. The camera was then moved to the next region, or the next plate, following the green path. A cycle through all 6 plates was completed in ~14 minutes.

of a Canon 5D Mark II camera. Once all regions on all plates had been imaged, the cycle repeated. The images for each region were registered to reduce positional jitter by using the StackReg plugin (1) for Fiji (2).

Segmentation

Segmentation of the colony from the background is performed offline using a custom Matlab script [Version 8.1(R2013a), MathWorks Inc., Natick, MA] by thresholding the green channel of the original image using a straightforward thresholding operation, with the option of using histogram stretching to cover the full dynamic range and a sensitivity parameter for adjusting the threshold value. This leads to the following thresholding operation

$$BW(x,y)=\begin{cases}1, if\ I(x,y)>\alpha t\\ 0, otherwise\end{cases}$$

where t is the threshold value from a global thresholding operation (3) and α is a sensitivity parameter for controlling the segmentation.

After thresholding, the binary image is further processed by filling holes and removing objects touching the image borders. Finally, objects <0.1% of the image area are considered as too small for any quantificatio and are therefore excluded. The result is the region of interest for which all subsequent processing is done.

Segmentation of the inner shape in the colonies is performed using band-pass filtering, which is implemented as a Difference of Gaussians. Essentially, the desired content, which in this case is the ruffles and other shapes in the image, is enhanced by band-pass filtering. The enhanced image is obtained as a difference of two low-pass filtered versions of the original image,

$$E = I*H_1 - I*H_2$$

where H_i, with $i \in \{1,2\}$, are 2D Gaussian kernels

$$\frac{1}{2\pi\sigma_i^2} e^{-(x^2+y^2)/(2\sigma_i^2)}$$

with different bandwidths σ^2_i and * is the convolution operator. After filtering, the final binary segmentation result BW_{BP} (where $_{BP}$ refers to band-pass filtering with difference of Gaussians) is obtained by thresholding the enhanced image E.

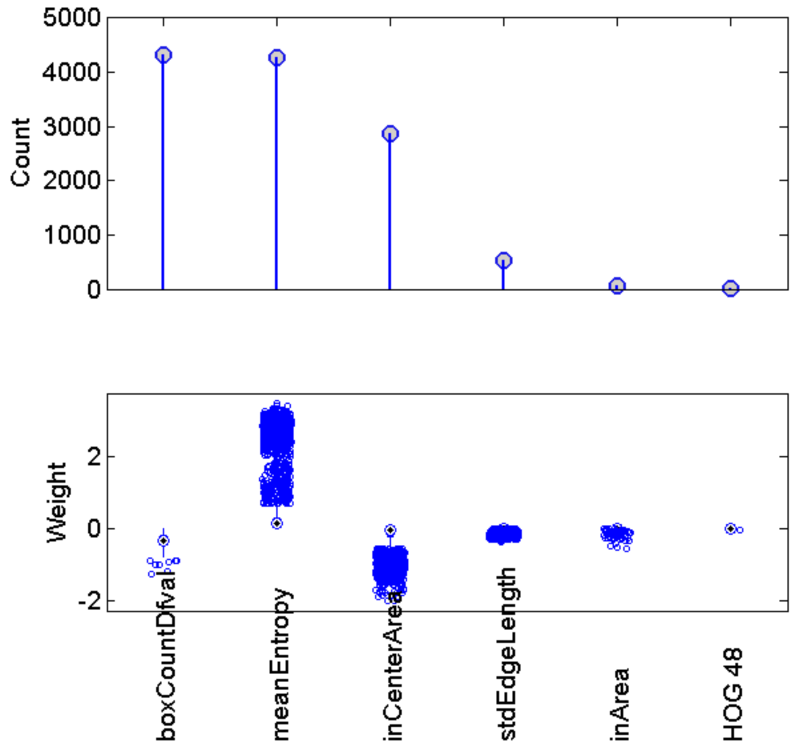


Figure 2. Feature selection counts (top) and logistic regression classifier model weights collected from the 5000 repetitions.

Feature extraction

The features extracted using custom Matlab scripts from the segmented region of interest in the colony images are described in the following table. Some features make use of Matlab toolboxes (Image Processing Toolbox ver. 8.2, Statistics Toolbox ver. 8.2) and publicly available functions/toolboxes (fractal dimension, local binary patterns, histogram of oriented gradients). The total number of features is 427.

Supervised yeast colony classification

After feature extraction, the colonies are represented with a feature vector $\boldsymbol{x}$ where each element of the vector corresponds to a numerical value of the feature. When training samples (i.e., images for which the phenotype/class is known and available), supervised classification methods offer powerful tools for automated phenotype classification. In our case, the feature vector includes hundreds of features, many of which may be redundant because the feature set is not tailored for a single, specific purpose. Thus, we use the logistic regression classifier with regularization due to its capability to produce sparse classification models. We used the implementation from the probabilistic modeling toolkit for Matlab/Octave, http://code.google.com/p/pmtk3/.

The sparsity promoting regularization works efficiently for the colony phenotype classification. The cross-validation case study presented in the article led to 98.79% overall accuracy, and during the 5000 repetitions of the hold-out validation, only 6 features were used. For a more detailed analysis of the features, we collected the model coefficients from the 5000 repetitions of the hold-out classification experiment. One indicator of the importance of a feature is to count how many times it has been selected into the classification model, that is, how many times the corresponding weight value in β is nonzero. The maximum count in this experiment is 5000, which would mean the feature has been used in all repetitions. On the contrary, if the count is zero, the feature did not get selected to the classification model during the cross-validation repetitions. Figure 2 shows the selection counts and feature weights during the 5000 hold-out repetitions.

Spatiotemporal profiling

The spatiotemporal profile is constructed for each time point T by taking a cumulative sum of the segmented colony shapes obtained through band-pass filtering and thresholding, as explained above. Thus, the profile

$$M(T)=\sum_{t=1}^{T} BW_{BP}(t)$$

Feature	Description
'Area'	Colony area (pixels).
'MajorAxisLength'	Length of longer axis in ellipse fitted to colony mask.
'MinorAxisLength'	Length of shorter axis in ellipse fitted to colony mask.
'ConvexArea'	Area of convex hull of the colony mask.
'EquivDiameter'	Diameter of the equivalent circular object.
'Solidity'	Ratio of area and convex hull area.
'Extent'	Ratio of object area to area of bounding box.
'Perimeter'	Colony mask perimeter length.
'MeanIntensity'	Average intensity within the colony area.
'Contrast'	Intensity contrast between a pixel and its neighbor calculated from gray level co-occurrence matrix. (4)
'Correlation'	Correlation of a pixel to its neighbor calculated from gray level co-occurrence matrix.
'Energy'	Sum of squared elements calculated from gray level co-occurrence matrix.
'Homogeneity'	Closeness of the distribution of elements in the gray level co-occurrence matrix to the co-occurrence matrix diagonal.
'edgePixels'	Number of edge pixels detected using Sobel operator inside the whole colony area.
'edgePixelsCentre'	Number of detected edge pixels inside the center of the colony.
'meanEdgeLength'	Average length of detected edges.
'stdEdgeLength'	Deviation of edge lengths.
'numEdges'	Number of detected edges.
'inArea'	Total area of the colony structure B_{BP} obtained from band-pass filtering.
'inCenterArea'	Area of the colony structure BW_{BP} in the center of the colony mask.
'inBorderArea'	Area of the colony structure BW_{BP} in the borders of the colony mask.
'meanInSize'	Average area of a connected component in B_{BP}.
'stdInSize'	Deviation of the areas in BW_{BP}.
'numInObj'	Number of objects in BW_{BP}.
'meanLPResid'	Average residual calculated from difference of two low-pass filtered images (Gaussian LPF, smaller σ).
'stdLPResid'	Standard deviation of difference of original and low-pass filtered images (Gaussian LPF, smaller σ).
'meanLP2Resid'	Average residual calculated from difference of two low-pass filtered images (Gaussian LPF, larger σ).
'stdLP2Resid'	Standard deviation of difference of original and low-pass filtered images (Gaussian LPF, larger σ).
'innerObj'	Number of separate connected components in the colony shape segmentation result (DoG segmentation).
'innerBrachPoints'	Number of branch points in the skeleton of the inner segmentation result.
'innerEnd points'	Number of end points in the skeleton of the inner segmentation result.
'boxCountDfval'	Fractal dimension determined as a box count slope (for details, see http://www.fast.u-psud.fr/~moisy/ml/boxcount/html/demo.html)
'boxCountDfS'	Standard deviation of box count slope.
'innerSkeletonMeanLength'	Average length of skeleton in binary image BW_{BP}.
'innerSkeletonMaxLength'	Maximum skeleton length in binary image BW_{BP}
'meanEntropy'	Average entropy texture measure within colony area.
'stdEntropy'	Standard deviation of entropy texture measure within colony area.
'meanStd'	Average standard deviation texture measure within colony area.
'stdStd'	Deviation of standard deviation texture measure within colony area.
'stdInt'	Standard deviation of intensities.
'iqrInt'	Intequartile range of intensities.
'skewnessInt'	Skewness of intensity values.
'kurtosisInt'	Kurtosis of intensity values.
'Percentile 1'	1st intensity percentile within colony area.
'Percentile 4'	4th intensity percentile within colony area.
'Percentile 7'	7th intensity percentile within colony area.
…	…
'Percentile 100'	100th Intensity percentile within colony area.
'meanSig'	Average boundary signature distance from centroid.
'stdSig'	Deviation of boundary signature distance.
'iqrSig'	Interquartile range of boundary perimeter distance.
'absdiffSig'	Sum of absolute deviation from mean boundary distance.
'signPrc10Int'	10th percentile of boundary signature distance.
'signPrc90Int'	90th percentile of boundary signature distance.
'Lowpass 5'	Average difference between original image and Gaussian low-pass filtered image ($\sigma = 5$)
'Lowpass 11'	$\sigma = 11$
'Lowpass 19'	$\sigma = 19$
'Lowpass 27'	$\sigma = 27$
'Lowpass 35'	$\sigma = 35$
'LBP 1'	Local binary pattern coefficient 1. (5)
'LBP 2'	Local binary pattern coefficient 2.
'LBP 3'	Local binary pattern coefficient 3.
…	…
'LBP 258'	Local binary patter coefficient 258.
'HOG 1'	Histogram of oriented gradients coefficient 1. (6)
'HOG 2'	Histogram of oriented gradients coefficient 2.
'HOG 3'	Histogram of oriented gradients coefficient 3.
…	…
'HOG 81'	Histogram of oriented gradients coefficient 81.

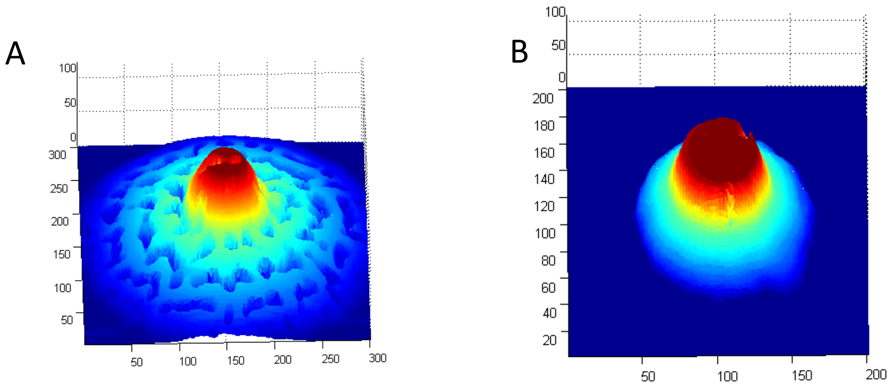


Figure 3. Spatiotemporal profiles of A) fluffy and B) smooth colonies.

where BW_{BP} is the binary colony structure at time t can be plotted over time as a growing and evolving 3-D surface (Supplementary Movies S1 & S2). Example images of spatiotemporal profiles for fluffy and smooth colonies measured over time are shown in Figure 3.

YIMAA Web application

YIMAA is an open source web application for displaying the results of quantitative analysis of yeast colony pattern formation. Using modern browsers, investigators are able to easily explore the original time series images in conjunction with the phenotypic signatures and principal component analysis results. Dynamic interactive charts are plotted for the selected replicates, and gallery panels can be used to compare images, raw and segmented, for all the time points. YIMAA is designed for yeast in this colony morphology study, but the methods and YIMAA web template are applicable to other organisms and associated imaging phenotypic quantitative research efforts.

YIMAA, built with jQuery and other open source libraries, is free for non-commercial and non-profit use. For more information, see the YIMAA user guide.

Availability

The YIMAA web application is available at http://yimaa.cs.tut.fi. The source codes of both the web application and the image and data processing pipeline are available at http://code.google.com/p/yimaa/. The MATLAB code includes dependencies to third party toolboxes, which need to be available when running the analysis pipeline.

References

1. **Thévenaz, P., U.E. Ruttimann, and M. Unser.** 1998. A pyramid approach to subpixel registration based on intensity. IEEE Trans. Image Process. *7*:27-41.
2. **Schindelin, J., I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, et al.** 2012. Fiji: an open-source platform for biological-image analysis. Nat. Methods *9*:676-682.
3. **Otsu, N.** 1979. A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybern. *9*:62-66.
4. **Haralick, R.M., K. Shanmugam, and I. Dinstein.** 1973. Textural Features for Image Classification IEEE Trans. Syst. Man Cybern. *SMC-3*:610-621.
5. **Ojala, T., M. Pietikainen, and T. Maenpaa.** 2002. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. *24*:971-987.
6. **Ludwig, O., D. Delgado, V. Gonçalves, and U. Nunes.** 2009. Trainable classifier-fusion schemes: An application to pedestrian detection. 12th International IEEE Conference on Intelligent Transportation Systems, ITSC '09. p1-6.

YIMAA User Guide

Contacts: Jake Lin (jake.lin@uni.lu) and Pekka Ruusuvuori (pekka.ruusuvuori@tut.fi)
Code Source and other information: http://code.google.com/p/yimaa/
Web address: http://yimaa.cs.tut.fi
Updated August 19th, 2013

Contents

1. General purpose

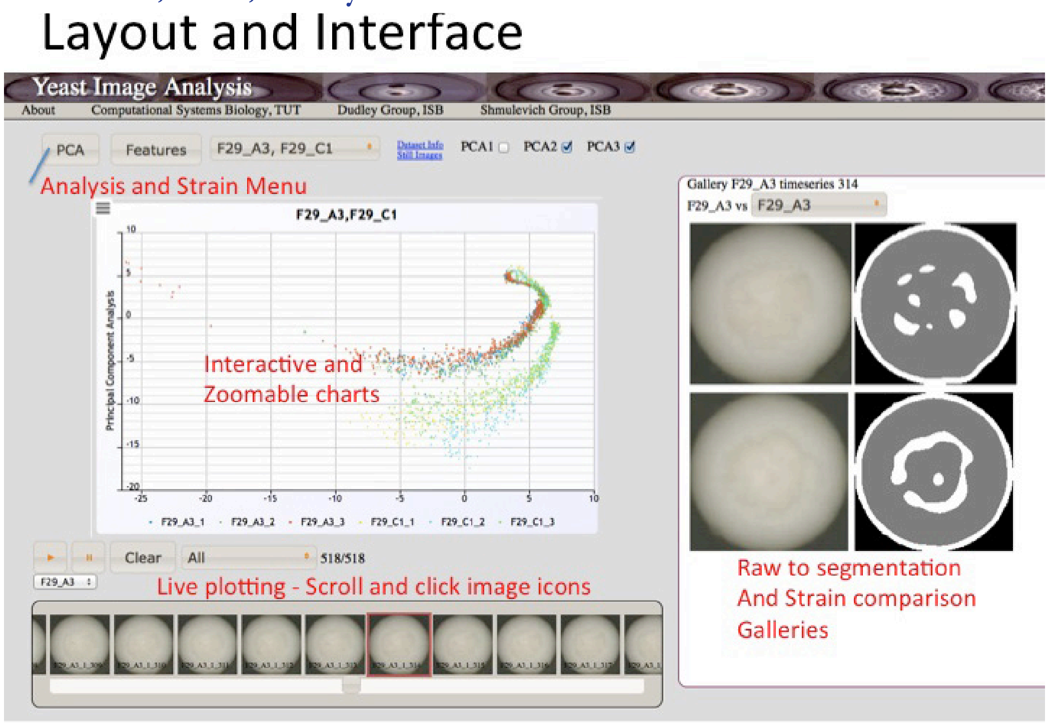
YIMAA is an open source web application for displaying the results of quantitative analysis of yeast colony pattern formation. Using modern browsers, investigators are able to easily explore the original time series images in conjunction with the phenotypic signatures and principal component analysis results. Dynamic interactive charts are plotted for the selected replicates, and gallery panels can be used to compare images, raw and segmented, for all the time points. YIMAA is designed for yeast in this colony morphology study, but the methods and YIMAA web template are applicable to other organisms and associated imaging phenotypic quantitative research efforts.

YIMAA, built with jQuery and other open sourced libraries, is free for non-commercial and non-profit use.

2. Browser recommendations

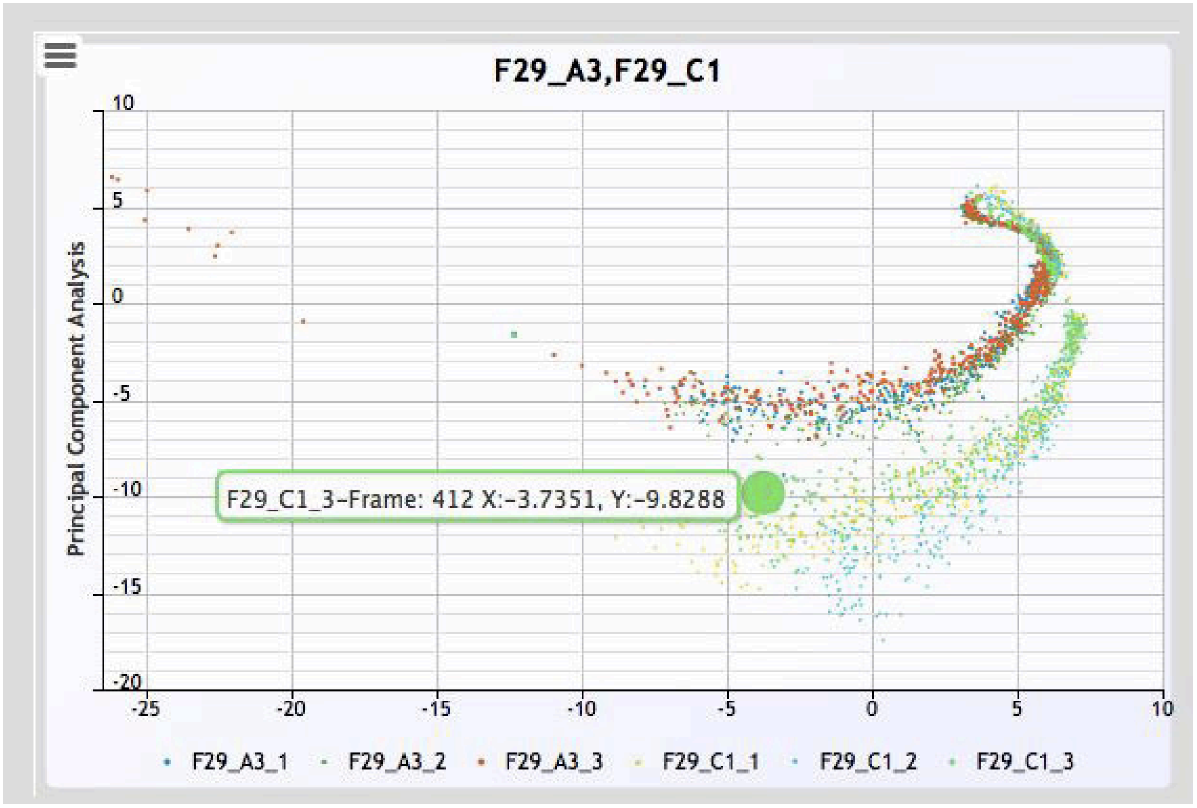
YIMAA is compatible with all modern browsers. We recommend using Firefox and Chrome since the majority of the testing has been on those two browsers.

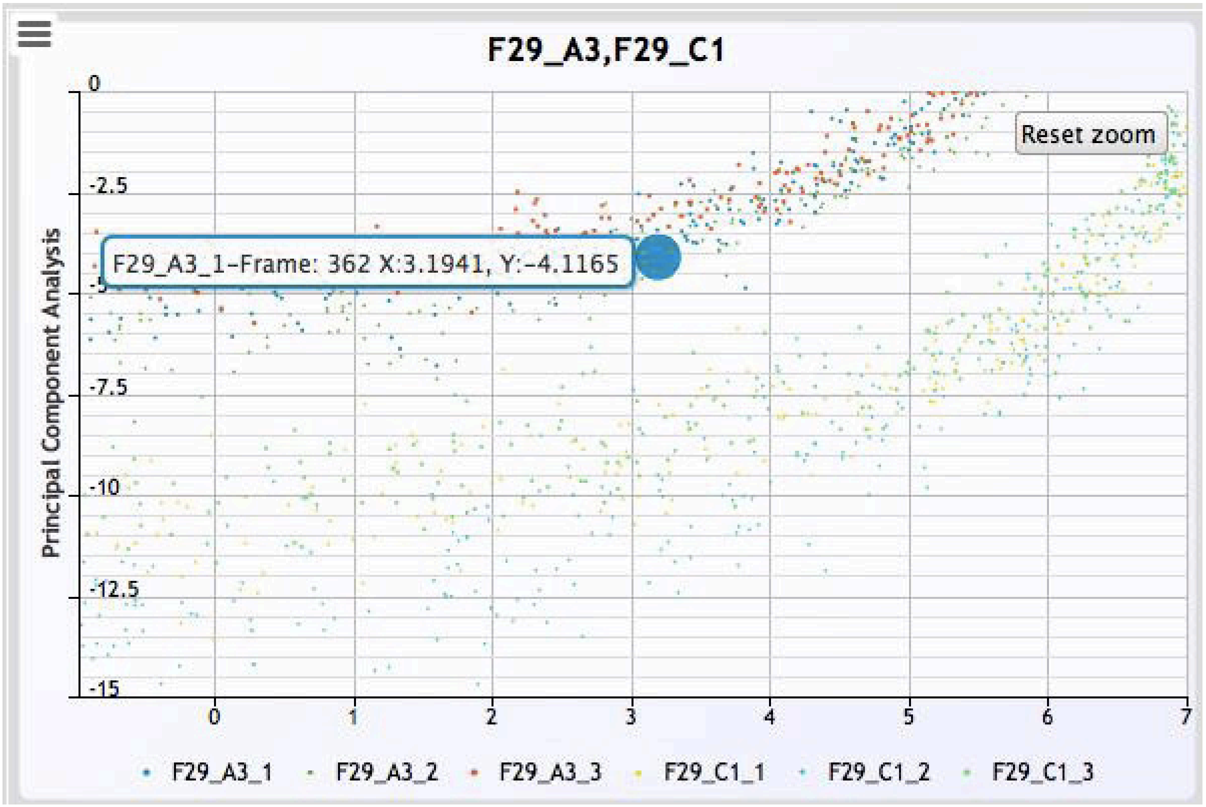
3. Interface, views, and layout



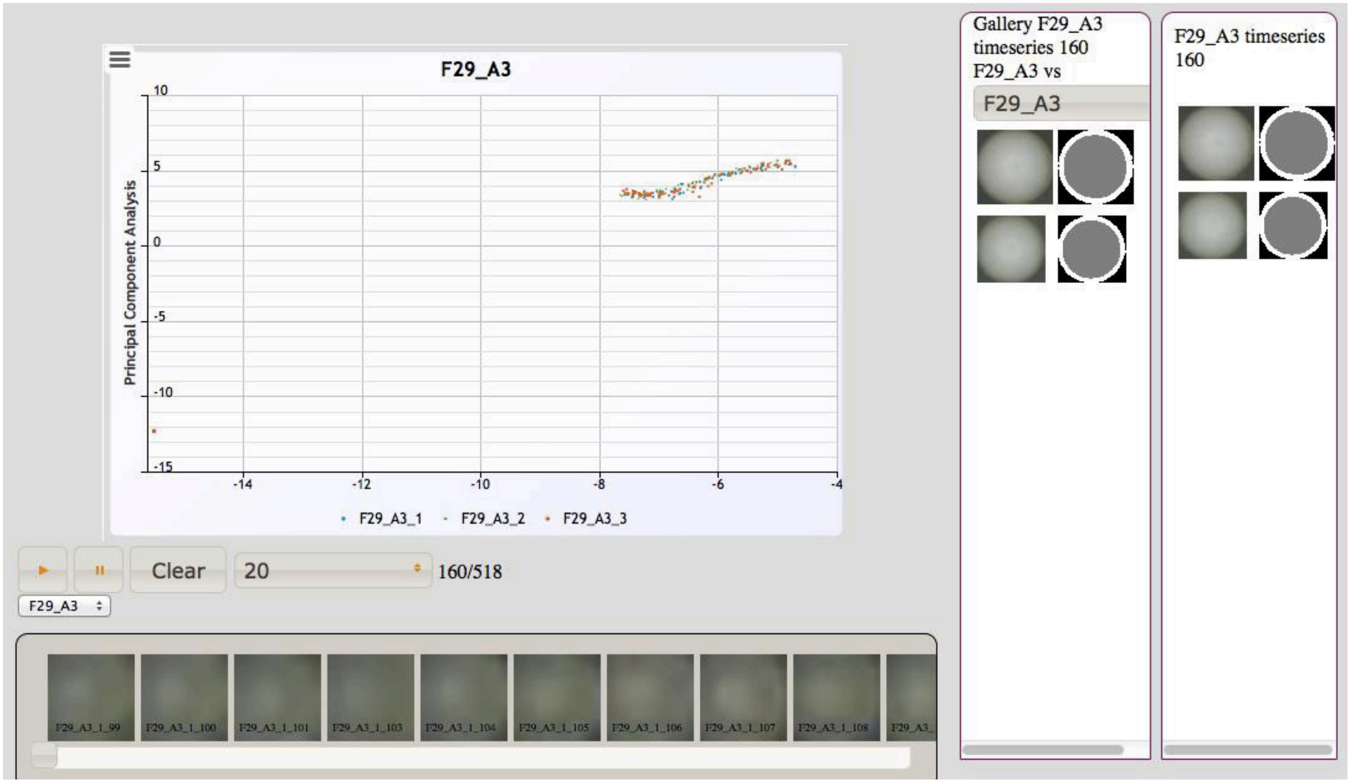
Using modern web technologies and design concepts, images and analysis results are integrated seamlessly with mouseover and zoom functions. Time series data for strains can be compared to each other, along with the original and segmented images. Red text overlaid in the above figure highlights different functionalities of the web interface

a. Principal component analysis (PCA)—Quantitative feature representation of yeast colonies can be viewed in reduced dimensionality using PCA. User can select which principal components are shown in the graph (1st, 2nd, 3rd). Multiple strains can be selected for dynamic comprehensive plotting. The figure below depicts the tool tip as the mouse cursor activates detailed information about each time point and selected regions can be highlighted and magnified.



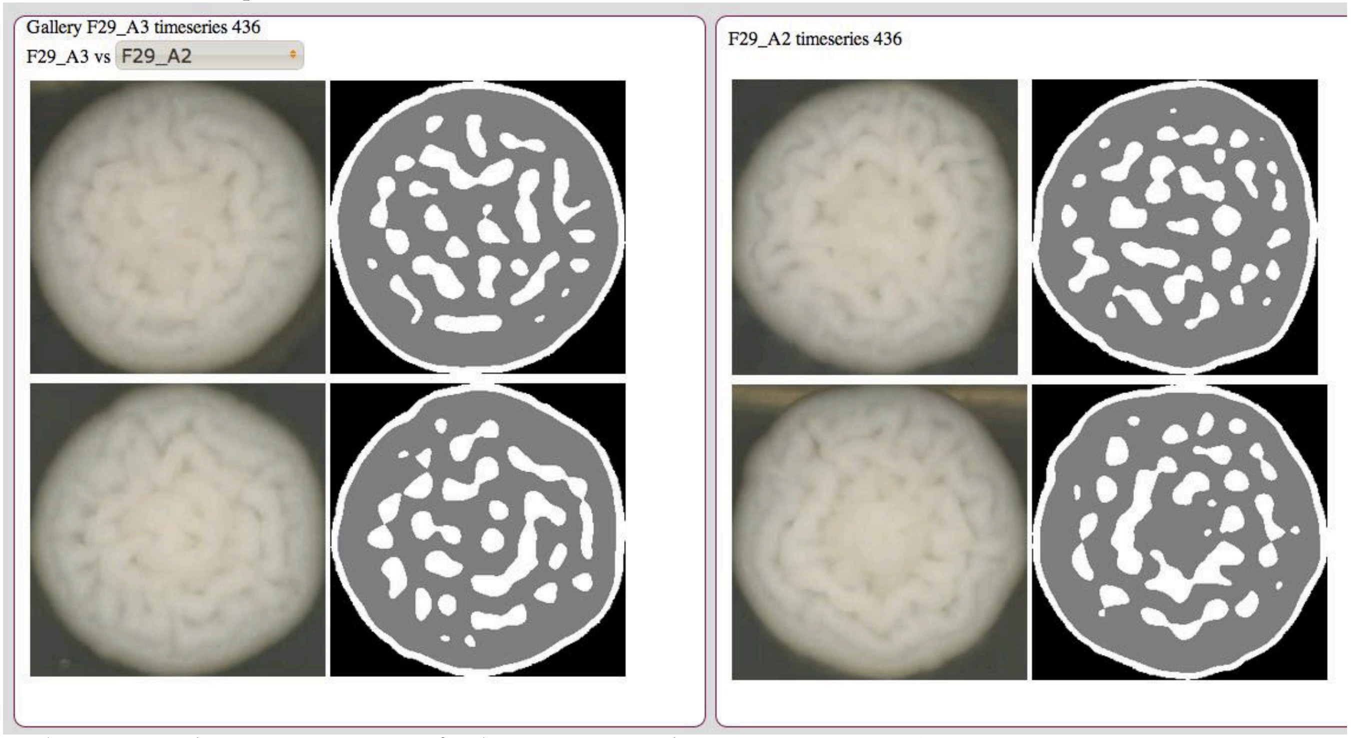


b. Live series plotting—Users can view the incremental progress of PCA plots by selecting the number of time points. The plot and image gallery will dynamically render the iterations. A pause option is provided for flexible control.



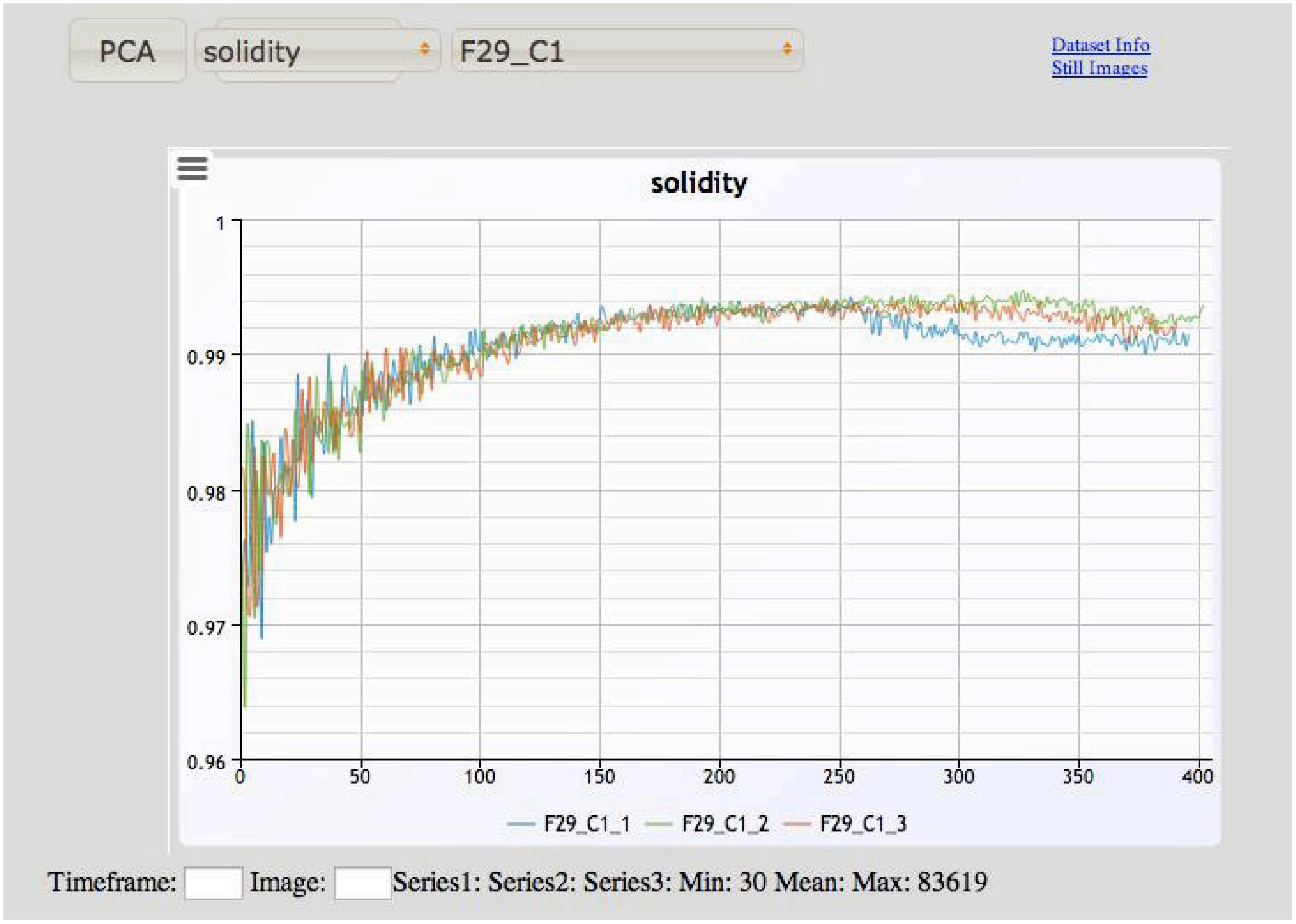
Time point 160

c. Image comparisons—For every strain and every time point, investigators can choose to compare the raw and segmented images between the selected strain and others in the study. This enables quick comparisons of phenotype properties across replicates or different strains at a chosen time point.



Looking at raw and segmentation images of replicates F29_A3 and F29_A2 at time point 436

d. Quantitative features—427 specific phenotypic features such as area, solidity (see figure below), and different statistical measures are available as dynamic plots. Interactive web app features like mouse over and zoom capabilities are fully functional. In addition, the user can jump to certain time frames and hide/show series data sets. See the complete list of image features in Supplemental material Table 1.



Timeframe: Image: Series1: Series2: Series3: Min: 30 Mean: Max: 83619

4. Data retrieval and object construction

The YIMAA web application uses AJAX for data retrieval. On applicable events, an asynchronous event along with required parameters is submitted to a server, and then the data are fetched and returned as JSON array objects. The data are stored as files organized by strains and series.