SAPHIRE

(**S**tochastic **A**nnotation of **PH**enotypic **I**ndividual-Cell **RE**sponses)

Software Demo and Instructions

Table of Contents

User Agreement

This software is covered by the MIT License.

SAPHIRE Demo

The software has been tested on MATLAB version 2015a.

STEP 1: Provided demo data overview and setting software path

The "Software" folder must be set as the current MATLAB path in order to properly read in all data and execute the scripts. The "Software" folder contains the following subfolders and files:

- **data** – the .mat files of the demo cell trajectories and their SAPHIRE models.
    - **DemoData.mat** – time series of actin images and cell body binary masks of four example cells used for the demo.
    - **Demo_SAPHIRE_models.mat** – results of SAPHIRE analysis on the four DemoData.mat cell trajectories.

- **SAPHIRE Package** – contains the scripts for image time series editing with the GUI tool ("SAPHIRE ImageProcess GUI" folder), calculation of shape features from cell object masks ("Feature Scripts" folder), and time series modeling of cell trajectories via Bayesian hidden Markov modeling (HMM).

- **util** – contains various auxiliary and miscellaneous scripts for data manipulation and plotting in the SAPHIRE analysis pipeline.

- **Demo.m** – main script for running the SAPHIRE demo.

- **Features.xlsx** – Excel file with the names of the image-derived features to compute for each cell object. The name of the feature (*) must match the feature label in the feature calculation scripts (wcf_*.m files) in the "SAPHIRE Package\Feature Scripts" folder. Users can incorporate additional image-derived features for cell phenotype description that are relevant for their cell type or biological process of interest.

STEP 2: Demo user settings

At the top of the Demo.m MATLAB script file is a section for editing the following user settings:

- **demoType** –
    - Set this variable to **'QC'** in order to run a demo of the editing GUI tool for an example cellular image time series stack, where you can modify cell segmentation, label phenotype, delete frames, etc. for each frame.
    - Set this variable to **'plot'** to display the results from the SAPHIRE analysis that was run for the four example cell trajectories, which are already

saved in the "data" subfolder (Demo_SAPHIRE_models.mat). This provides a quick way to see what SAPHIRE results and outputs look like.

  - Set the variable to **'analysis'** in order to execute the image feature calculations and time series modeling on the four example trajectories from scratch, given the cell body mask time series in DemoData.mat. NOTE: because time series modeling with SAPHIRE implements MCMC sampling, it can take a long time to run. For the four example cell trajectories in the demo using parallel processing of trajectories in MATLAB can take approximately 10 minutes on a 3.40 GHz, Core i7-4770, 24GB RAM machine with the other user settings left as they are in the demo. A progress bar is shown in the command window as the time series modeling progresses. For longer trajectories, testing larger number of maximum states (**Kmax**, see below), or analyzing many trajectories, may require running the analysis overnight and/or on a computer cluster.

- **trajToAnalyze** – set which trajectory to run the demo for in DemoData.mat. There are four example trajectories in the demo data, so **trajToAnalyze** must be set to 1, 2, 3, or 4 in this demo.

- **pxScale** – set the scale of a pixel in the cell images (e.g. microns per pixel). This is useful for having the cell shape features computed with the desired units (e.g. length in microns as opposed to length in pixels). However, this is not crucial for subsequent analysis steps, such as principal component analysis, because the features are normalized using a z-score across cell objects and will be unaffected by a constant scaling.

- **Kmax** – set the maximum number of underlying states (a positive integer) that you would like to test. These model states emit the observations, or the cellular trajectory PC feature space points, the number and parameters of which must be determined since they are unknown *a priori*. If **Kmax** = 3, for example, then SAPHIRE tests 3 model – one model with only one state, one model with two states, and a third model with three states. Although a key advantage of the Bayesian HMM framework in SAPHIRE is that it models time series without user-defined parameters, setting **Kmax** constrains the number of competing models (each with a different number of states) that are to be tested, and therefore can dramatically decrease runtime. Generally, **Kmax** should be determined empirically for a subset of cell trajectories to get a sense of the maximum number of states that cells explore. This can depend on, among other factors, the cell type, treatment condition/dose, how quickly cells change in their measured features, and how long cells were imaged.

- **mcmc_params.parallel** – set to **'on'** in order to run parallel processing on the competing models, and to **'off'** to run parallel processing on the cell trajectories. In general, if you are running SAPHIRE on many cell trajectories it makes sense to set this parameter to **'off'** to parallelize the analyses of the multiple trajectories across multiple CPU cores on your machine.

STEP 3: Running the demo

After setting the user settings in Step 2 above, the demo executes differently depending on the **demoType** set by the user. Make sure the file Demo.m is open in MATLAB and press 'Run' on the MATLAB toolbar, or type the following in the command window:

```
>> Demo
```

Below we walk through the outputs of the demo, which highlight the parts of the three major components of the live-cell phenotypic profiling framework provided in the SAPHIRE software package:

I. Cell image time series editing GUI tool
II. Single-cell time series modeling
III. A combined 'phenotypic signature' for a group of cells

I. Cell image time series editing

The image processing GUI is a standalone part of the SAPHIRE framework. It is designed for quality control and editing of time series image stacks of individual cells. The GUI scripts are in the "SAPHIRE ImageProcess GUI" subfolder within the "SAPHIRE Package" folder. The function SAPHIRE_cellTrajQC_main.m is the overhead script to run the GUI, with a structure array input variable **Iarray** that must be supplied by the user. **Iarray** has the following fields:

Iarray.CB – cell array of fluorescence intensity time series images used for cell body delineation.

Iarray.CBMask – cell array of segmentation masks of each cell image in **Iarray.CB**. These are initial segmentation masks for which the user wants to assess the quality of segmentation or to edit the masks.

Iarray.Nuc – cell array of fluorescence intensity time series images of the live-cell nuclear reporter.

Iarray.nucCentMask – cell array of logical (binary) masks of nuclear localizations. In each image there is one pixel that has value 1 that corresponds to the nucleus center, obtained from a prior nuclear segmentation.

Iarray.phenoLabels – cell array of phenotype labels of the cell at each time frame. These labels are usually letters to designate the phenotype, which the user can edit. For example, we use 's' to label cell objects that are isolated cells, 'b' to mark cells that are touching other cells or the image boundary, 'd' to mark dividing cells, and 'a' to mark apoptotic/dying cells.

After adding the "SAPHIRE ImageProcess GUI" subfolder to the MATLAB path, the GUI is run with the following command:

```
>> SAPHIRE_cellTrajQC_main(Iarray)
```

The GUI window appears (Figure 1). The main window shows the time series images of the cell with the red outline around each cell object corresponding to the current segmentation of the cell in that frame. The letter in the top left corner of each frame corresponds to the current phenotype label for the cell in that frame. Time progresses from left to right and then top to bottom (i.e. first across columns and then down the rows). There are seven buttons under the main window. Their functions/usage are listed below:

Edit Seg Thresh – edit the segmentation of the cell in a given frame. After clicking the button a hand pointer will appear. Click on the image frame for which you want to edit the segmentation. After a frame is clicked a second MATLAB window will appear (Figure 2), showing a zoomed-in image of the cell fluorescence image (from **Iarray.CB**), a blue outline of the current segmentation, and a blue dot in the cell where the nucleus center is located (from **Iarray.nucCentMask**). The segmentation is re-initialized using a combination (overlap) of corrected Otsu thresholding and Canny edge detection. Follow the instructions in the MATLAB command window for the options available to edit the segmentation. You can add and remove regions or linear/curved segments of the cell, or change the thresholds for Otsu or Canny for faster modification of the segmentation mask. Press "escape" on the keyboard to complete the segmentation edits for the frame.

Add Seg – add foreground to the cell object(s). When this button is clicked a crosshair will appear. Left-click the starting point of where you want to add foreground (add to the mask) in a given frame, or across multiple frames in the main GUI window. Hold the left mouse button down and draw a region for foreground addition. Release the left mouse button to create a "loop" of the region. Press spacebar to accept, or 'u' on the keyboard to undo, the foreground addition. You should see the red outline added to the frame(s) where you drew the region. If the added region does not touch a cell (i.e. not contiguous with the nucleus) then it will be removed automatically since we are only interested in foreground regions corresponding to cells.

Remove Seg – remove foreground from the cell object(s). The functionality is the same as for **Add Seg** above, but removes a user-defined foreground region from the cell object(s) in the GUI window instead of adding foreground regions.

Delete Frame – mark a frame for deletion. When this button is clicked a hand pointer will appear. Left-click on a frame that you want to mark for deletion. This frame could, for example, have some imaging artifact or other undesirable property that you want to flag. A blue 'X' will appear of the frame you marked for deletion. You can left-click the frame again to un-delete it, which will in turn remove the 'X' over that frame. The deleted frames are marked as 1's in the **dataGUI.deleteFrameMask** cell array, which you can access in the MATLAB workspace when you press the **Finish** button in the GUI.

Delete All – marks all frames in the trajectory for deletion. This is the same action as for the **Delete Frame** button, but for flagging the entire trajectory. Press 'spacebar' on the keyboard to accept the deletion of all frames or 'u' to undo. If accepted, a blue 'X' is placed over each frame. All values in **dataGUI.deleteFrameMask** cell array are set to 1's to designate that all frames will be flagged for deletion. This can be used, for example, if you want to completely remove this trajectory from further processing and analysis.

Pheno Edit – edit the phenotype label of the cell object in a given frame. When this button is clicked an arrow pointer will appear. Hover (do *not* click with the mouse) over an image frame in the GUI window and press a key (e.g. a letter) on the keyboard. The label of the cell object in the upper left corner of the frame will change to the key you pressed. Phenotype labeling is useful, for example, to designate dividing ('d') or dying cells ('a'). The modified labels are stored in the **dataGUI.editedPhenoLabels** cell array, which you can access in the MATLAB workspace when you press the **Finish** button in the GUI.

Finish – complete cell trajectory editing and exit GUI. Click this button when you are finished editing the cell trajectory. The GUI will close and the structure array variable **dataGUI** will appear in the MATLAB workspace, where you can access the modified cell masks, phenotype labels, and frame deletion flags.

Reset – resets all user modifications back to original ones when the GUI loaded. Press this button if you would like to undo all the modifications made to the cell trajectory and start over.

Zoom out – zooms the main GUI image window back to original magnification so that all the cell frames can be seen. You can use the magnifying glass tool in the built-in MATLAB figure toolbox to zoom in and out of the main GUI window. The open hand pointer in the MATLAB figure toolbar enables you to pan around the zoomed-in GUI window to see different image frames.

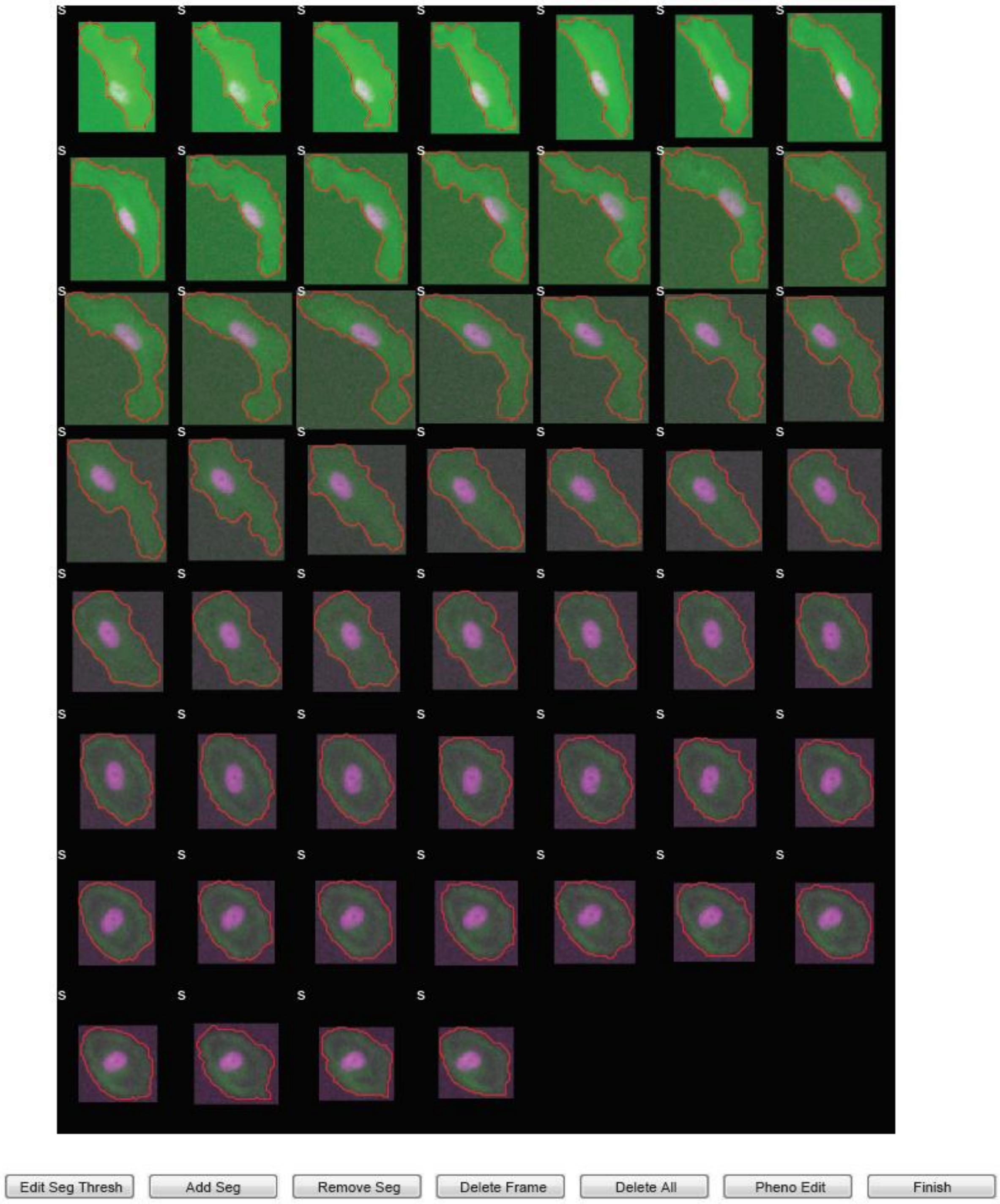

Figure 1. Screen snapshot of the live-cell trajectory editing GUI tool window in SAPHIRE. A live-cell trajectory with 53 time points (18 hours) is shown. The green fluorescence images are LifeAct-eGFP (actin) reporter overlaid with Histone 2B-mCherry (nuclear) reporter in pink. The red outlines around the cell bodies in each frame are the segmentation mask boundaries. The buttons for executing various actions to modify the segmentation masks, re-label phenotypes, and flag frames for deletion are shown under the trajectory image GUI window.

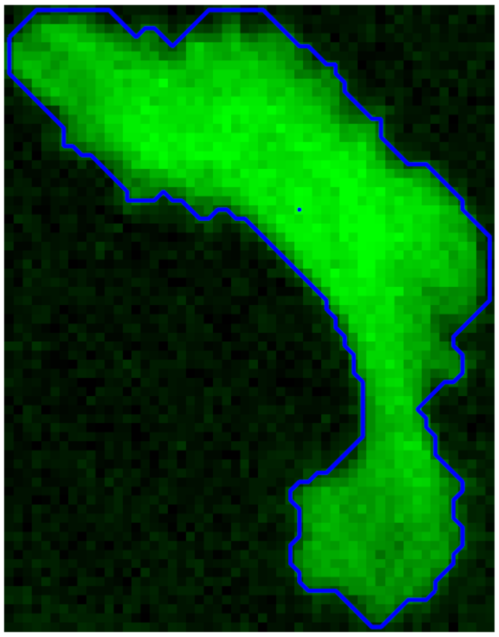

Figure 2. Screen snapshot of a live-cell image frame from the GUI figure window that appears when the **Edit Seg Thresh** button is pressed. The green fluorescence is the LifeAct-eGFP (actin) reporter signal. The intensity histogram of the image is rescaled automatically (*imadjust* function in MATLAB) for better contrast. The blue outline is the segmentation mask for the cell object and the blue point within the cell is the single-pixel nucleus center (obtained from the nucleus mask). The user can modify the blue outline region manually or automatically by changing the segmentation thresholds using the mouse and keyboard.

I. Cell phenotype time series modeling

Following cell body segmentation and any user editing of the cell trajectory using the GUI tool, image-based features are extracted from each frame that capture various aspects of phenotype that we want to model over time. Using the segmentation masks of the actin reporter, we are interested in temporally modeling overall cell morphology phenotype, for instance. The image-based shape features that we compute for the cell at each point in time are in the Features.xlsx file (see Step 1).

Following feature extraction, principal component analysis (PCA) is applied to the features from all the cells in the experiment (four cells in the demo) so that they are projected to the same principal component (PC) axes. We project the image feature data to two PCs since we found that for our imaging experiments they account for over 80% of the variance. In principle, users can increase the number of components used to describe the image feature space, but the time series modeling will run more slowly due to the larger number of dimensions and therefore a larger number of parameters in the Gaussian state model (e.g. bivariate Gaussian state(s) when using two PCs). This generates cell trajectories in PC space, which, in the case of two PCs, are (PC1,PC2) coordinates that evolve over time.

Time series modeling of the PC space coordinates using SAPHIRE applies hidden Markov modeling (HMM) with Bayesian model selection to penalize model complexity in order to infer the number of states in the PC coordinate trajectory without overfitting. The time series model of a given cell trajectory (dxT numeric matrix variable called **traj**, where d is the number of dimensions in PC space, and T is the number of time points in the trajectory) using the parameters set in Step 1 can be run and obtained with the following command:

```
>> [SAPHIRE_Results.PrM,...
    SAPHIRE_Results.ML_states,...
    SAPHIRE_Results.ML_params,...
    SAPHIRE_Results.full_results,..
    ~,...
    SAPHIRE_Results.logI]...
    = hmm_process_cell_trajectory(traj,Kmax,mcmc_params);
```

The resulting model is a structure array variable called **SAPHIRE_Results** with the following output fields:

PrM

Normalized model probabilities. Each competing model has a different number of Gaussian underlying states (e.g. 1, 2, 3, etc.) that emit the trajectory coordinates (**traj**) in PC space. The most likely model (highest **PrM** value) is chosen to describe the states of the cell trajectory. **PrM** output is useful for assessing how well other models (i.e. different numbers of states) describe the cell phenotype trajectory data.

ML_states

The maximum likelihood (ML) state sequence determined by the Viterbi algorithm in the HMM framework. ML_states is a vector with each index a number from 1 to *K*, where *K* is the total number of states in the most likely model (model with the largest **PrM** value). These indices correspond to the most likely sequence of states that the cell explores over time.

ML_params

Maximum likelihood parameters of the most likely (largest **PrM** value) state model. These are used to assess where in PC space the states are centered (state means) and how spread the states are (state standard deviations). Dynamic parameters from the HMM for the cell trajectory are also reported here.

ML_params.p_start

Probabilities of starting in a given state (one of *K* states) of the most likely model.

ML_params.p_trans

HMM transition probability matrix that is size [*K*,*K*] for the total number of states *K* of the most likely model. Each element in the matrix is the probability of transitioning from the given state row to the given state column. For instance, in a 3-state model, the element in the second row and second column in the 3x3 matrix is the probability of the cell staying in state 2 at the next time step given that it is in state 2 in the current time step. Note that this matrix describes the underlying dynamic model of the cell, which can generate an infinite number of state sequences, the most likely of which is the **ML_states** sequence output from the Viterbi algorithm.

ML_params.mu_emit

Matrix of inferred state means corresponding to the centers of each state in PC space. Rows are the individual axes PC1, PC2, etc. and columns correspond to the coordinates of the means along the axes.

ML_params.sigma_emit

Vector of inferred state standard deviations in PC space. Since the Gaussian states are spherical, having a diagonal covariance matrix with the same values along the diagonal, there is only one standard deviation parameter per state.

full_results

Contains the results (parameters, ML state sequences) for all the models tested, in addition to the most likely one.

logl

Log-likelihood of each model tested. These can be interpreted as unnormalized model probabilities of how well the PC trajectory data fits each model.

NOTE: C libraries for Windows and OS X are provided in the SAPHIRE Package folder. If you want to compile them yourself, the C source is provided. It can be compiled by changing the MATLAB path to the package folder and running:

```
>> mex hmm_forward_entry.c hmm_forward.c -output hmm_forward
```

Below is a list of supported compilers for each platform:

http://www.mathworks.com/support/compilers/R2015a/index.html

Plotting SAPHIRE modeling results

After running SAPHIRE time series modeling the user can plot various useful cell trajectory temporal data and model results. To generate the plots from a given structure array model (**SAPHIRE_Results**, see above), run the following command:

```
>> plot_model_results(SAPHIRE_Results,traj,I,mask)
```

The variable **traj** is the dxT PC space trajectory of the cell (d is the PC trajectory dimension and T is the trajectory length), **I** is the cell array of fluorescence images, and **mask** is the cell array of binary masks of the frames in **I**.

The first plot (Figure 3) shows the temporal evolution of trajectory (**traj**) points in PC space. To its right is the PC space plot with the maximum likelihood state annotations from the model. Points in time are colored by the annotations in the maximum likelihood state sequence found by the Viteri algorithm.

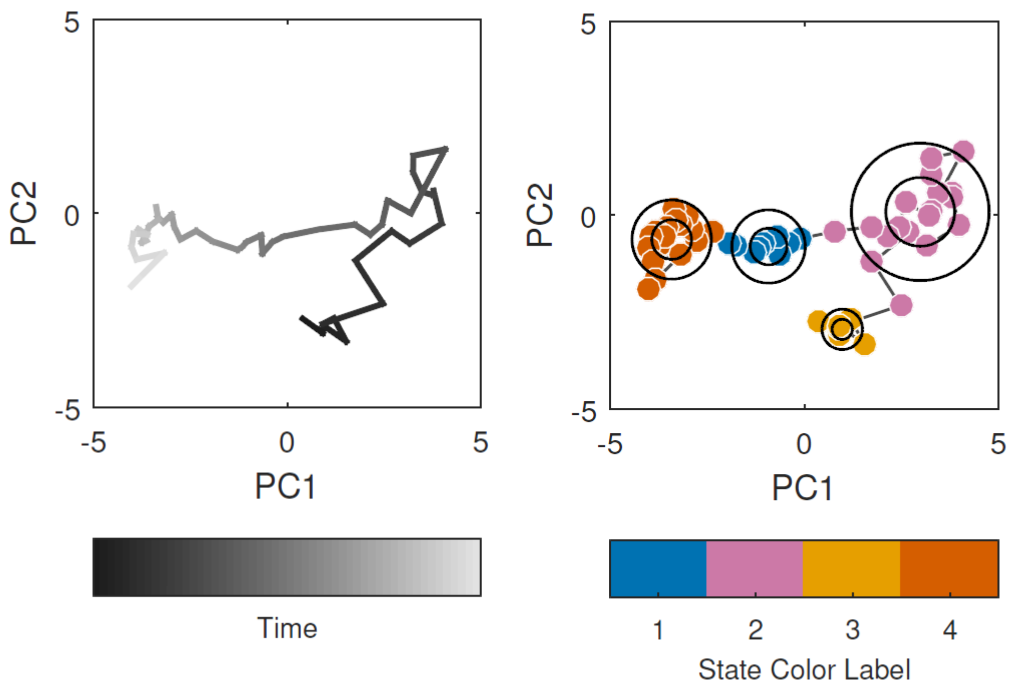

Figure 3. Cell trajectory temporal evolution visualization and state sequence annotation from a SAPHIRE analysis. Left, evolution of the cell trajectory through PC space over time. Right, maximum likelihood state annotations of the time points from the most likely model. In this example, the temporal trajectory is best fit by a 4-state model as found by Bayesian HMM in SAPHIRE.

The second plot (Figure 4) provides the visualization of the fluorescence images and the segmentation outlines colored with the maximum likelihood states. This provides a visual/physical assessment of what the model-derived states in PC space look like in terms of the fluorescence reporter images and cell phenotype. The state numbers correspond to the state annotations in Figure 3.

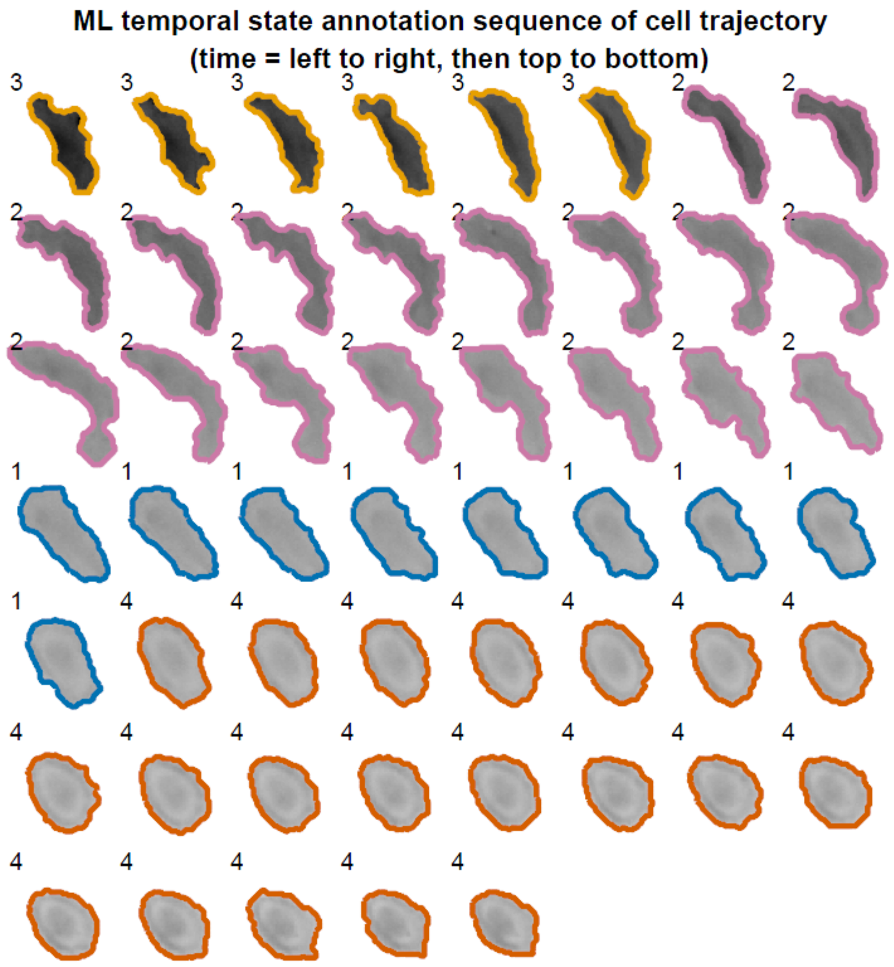

Figure 4. Visualization of fluorescence image time series with SAPHIRE analysis state annotations. The cell was imaged over 53 time points (18 hours), with the gray intensity corresponding to inverted images of the LifeAct-eGFP (actin) live-cell reporter. Cell outlines correspond to boundaries of segmentation masks used for shape representation, colored by the maximum likelihood state annotations inferred by SAPHIRE. The numbers in the upper left corners of each frame correspond to the state indices in the SAPHIRE analysis results structure array.

The final plot (Figure 5) shows the state transition diagram for the modeled cell trajectory. The phenotype dynamics are represented by a network diagram, in which nodes are the inferred states and the arrows are transitions between states from the maximum likelihood state annotation sequence. Normalized state dwell time (orange color grading) is the amount of time a cell spends in a given state divided by the trajectory length. The normalized state transition dwell time (blue color grading) is the amount of time a cell spends in the state the arrow points to given that it transitioned from the state where the arrow originates.

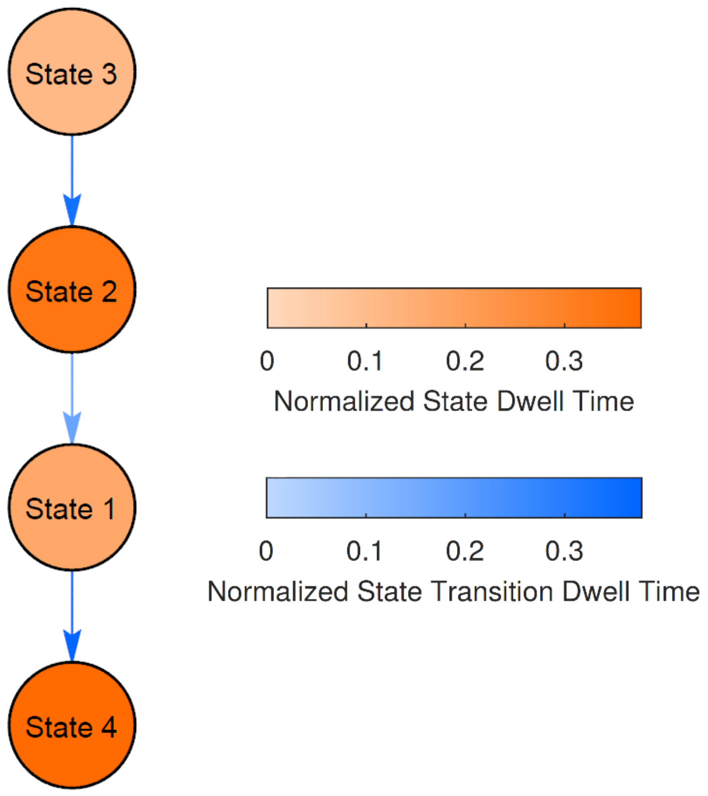
State 3
State 2
State 1
State 4
0
0.1
0.2
0.3
Normalized State Dwell Time
0
0.1
0.2
0.3
Normalized State Transition Dwell Time

Figure 5. Maximum likelihood state sequence transition diagram of a cell trajectory. Circles correspond to states explored by a cell over time as inferred by SAPHIRE analysis. Arrows correspond to transitions between states. The cell shown here moved progressively from state 3 to 2 to 1 to 4 without returning to any given state after switching to another state over 18 hours of imaging. Deeper orange color of a node means the cell spent a longer amount of time in that state. Deeper blue arrow color means the cell spent a longer amount of time in the target state (where the arrow terminates) following transition to that state from the source state (where the arrow originates).

III. Dynamic phenotypic profiles for groups of cells

A principal use of SAPHIRE is in live-cell phenotypic profiling of drugs or other perturbations in high-content imaging screens. We next discuss how the SAPHIRE time series models derived for a group of cells from the same treatment condition can be used to generate a phenotypic profile for the treatment condition. The profile is a vector that combines four pieces of information derived from SAPHIRE models of multiple cells. The four dynamic features are: the state locations in PC space, normalized state dwell times, state transition direction magnitudes, and state transition dwell times, the second and fourth of which correspond to the orange and blue color grading, respectively, for a given cell in Figure 5. Please see Materials and Methods section of the paper for more details of how these dynamic features are computed. To derive a profile for a group of cells, place the SAPHIRE models from the individual cells from a given treatment condition into a cell array, as is done in the demo for the four cells whose cell array of models are stored in the Demo_SAPHIRE_models.mat file in the “data” subfolder of the software package. To generate the phenotypic profile for the group of cells in the cell array use the following command:

```
>> [feats,sig] = compute_pheno_signature(SAPHIRE_Models);
```

The input cell array of models is **SAPHIRE_Models** for the group of cells, the variable **feats** (called **phenoComboDynamicFeats** in the Demo.m file) contains the four dynamic features discussed above for a group of cells, and the variable **sig** (called **phenoComboDynamicSig** in the Demo.m file) is a simple concatenation of the four features (**feats**) into a single phenotypic profile (a numerical vector). A profile can be computed for each treatment condition in an experiment. Profiles from different treatments can be compared to each other using, for example, hierarchical clustering or other approaches, depending on what conclusions are sought about the treatment condition comparisons. Clustering of the profiles, for instance, may be used to assess similarity in live-cell phenotypic responses between treatment conditions.