**ShootingStar Manual**

**V1.0 rev 10.24.16**

**Contents**

1. **Installation**
2. **Parameter Adjustment**
3. **Running the Application**
4. **Lineage Editing**
5. **Ablation Configuration**
6. **Installation**
   1. **System Requirements**

ShootingStar is, in principle, platform-independent but has only been extensively tested on Windows-based systems. Given the need for high-throughput analysis, a fast multi-core CPU is essential. While not essential, we strongly recommend:

* quad core CPU
* 8 GB of RAM
* Solid-State Drive

For analyzing very large images or multiple stage positions in parallel we recommend:

* Dual quad or hex core CPU’s (8-12 physical cores in total) with >3 GHz max clock rate
* 16 GB of RAM or more
* Solid-State Drive

For extremely large images where disk throughput might be a limiting factor (ie much larger than 512×512×35 voxel images) we recommend using multiple SSD’s in a RAID 0 array or a PCIe SSD.

In order to run ShootingStar, two third party software packages are required:

* MATLAB 2013B (newer versions should work fine but have not been tested)
  + Image Processing Toolbox
  + Parallel Processing Toolbox
  + Signal Processing Toolbox
  + Statistics and Machine Learning Toolbox
* Oracle Java JRE v1.8

ShootingStar currently only supports images saved as multipage TIF stacks. With the exception of two-channel images in which both channels are saved side-by-side in each frame, ShootingStar currently does not support the display of multi-channel images. If the nuclear channel is saved as its own series of TIF stacks, these can still be read and processed by ShootingStar.

**1.2 Configuration**

MATLAB installs with its own JRE, so the default JRE that it runs in must be changed to the Oracle 1.8 distribution, mostly in order to support execution of the some of the third party Java libraries the data curation interface uses. In our experience this hasn’t affected core MATLAB functionality although we do notice user interface glitches in MATLAB only if ShootingStar has previously been run in that instance of MATLAB. To point MATLAB to our Oracle JRE, a user variable must be created in windows. The easiest way to accomplish this is to search for “system variable” in the Windows start menu which should suggest the option “Edit the System Environment Variables”. In the window that opens, click on the button labeled “Environment Variables” and create a new system variable. For Variable name type “MATLAB\_JAVA” and for Variable Value type the path to the Oracle JRE you’ve installed on your system. For example on ours this is “C:\Program Files\Java\jre1.8.0\_66”

The final configuration step is to add all of the relevant jars for the user interface to MATLAB’s Java path and all of ShootingStar’s m scripts to the MATLAB path. On our system we’ve copied the SN and AT directories containing the executables, jars and m scripts into the local user’s MATLAB directory so the additions to the MATLAB Java path look like this:

C:\users\UserName\Documents\MATLAB\AT\JarName.jar

Where the underlined portions will depend on your system and the specific jar file. Using the following MATLAB command:

javaaddpath(path,’-end’)

You must add each of the following jar files into the MATLAB javaclasspath:

/AT/AceTree.jar

And all of the jars contained in /AT/jars/

You can confirm that this process was successful by executing the javaclasspath command in MATLAB and checking to see that all of the added paths are now present in the java path.

1. **Adjusting ShootingStar Parameters**

There are generally only 4 basic parameters that need to be adjusted for the vast majority of samples: a threshold, the expected diameter of nuclei in the first timepoint, the voxel size of the image (in micrometers) and the number of slices per volume. Included in the file distribution is a sample parameter file titled “ShootingStarParams.txt”. This table contains all of the basic parameter values that should be adjusted when analyzing data acquired on a new system or a new sample

**Table 1. Essential ShootingStar Parameters**

|  |  |
| --- | --- |
| Parameter Name | Description |
| firsttimestepdiam | The average diameter of nuclei in the first timepoint (in pixels) |
| firsttimestepnumcells | If using time-staged parameters, the number of cells in the first volume of the series |
| xyres | The size of each pixel in micrometers |
| zres | The spacing between slices in micrometers |
| slices | The number of slices per stack |
| parameters.intensitythreshold | Adjust the multiplier if the sample is globally brighter or dimmer |

Some of these parameters, particularly the intensity threshold can be “staged”, meaning for sample where the intensity of the nuclei varies in a predictable pattern over the course of the experiment, the intensity threshold can be adjusted using the number of cells present as a marker to select the appropriate threshold. For samples where the intensity of the nuclei is fixed, the array can simply be deleted and replaced with a single value along with replacing the value for parameters.staging with a number larger than the expected total number of cells per volume.

To determine the correct intensity threshold, an empirical approach is often sufficient to quickly reach a nearly optimal value. If a particular threshold results in many false positives, increase the threshold until this is minimized. If a threshold results in many false negatives, decrease the threshold until all or most cells are detected and few or no false positives are detected.

1. **Running ShootingStar**

To run ShootingStar, one simply calls the lineage\_launcher\_REALTIME function from the MATLAB command line. This will launch the initialization GUI:

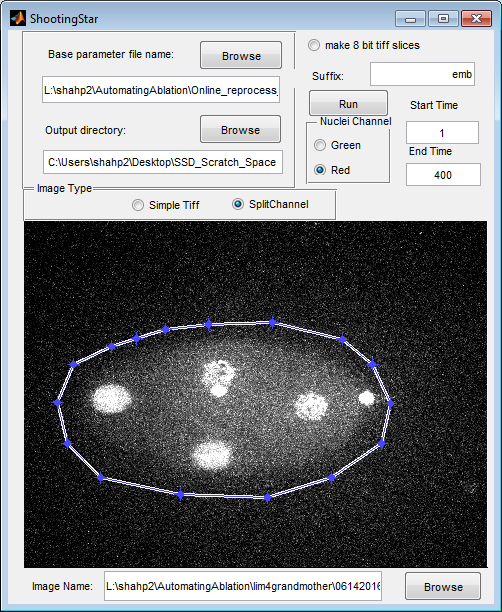


Figure 1. Screenshot of ShootingStar initialization GUI

To initialize ShootingStar to run on a dataset, the following parameters must be set:

**Base parameter file name** should be pointed to the tracking parameter file (an example version is provided with the source code).

**Output directory** is the directory in which results will be saved.

**Image Type** should be set to “Simple Tiff” if each volume is written as a single multipage tiff image with 1 channel per image. If two channels are saved side-by-side, select “SplitChannel”.

**Make 8-bit tiff slices** generates 8-bit copies of the raw data to speed data review after acquisition. This option is unnecessary on most modern computers, but can help with data throughput issues if images are typically stored on a networked server instead of locally.

**Suffix** can be used to affix a custom label to the end of output filenames.

**Start time** is currently ignored.

**End time** can be set to the maximum number of time points ShootingStar needs to process. Normally this is the total number of time points that will be imaged.

**Nuclei Channel** changes the false coloring of nuclei in “Simple Tiff” images. For “SplitChannel” this instructs ShootingStar to look in the left half of the frame (red) or the right half of the frame (green) for the nuclear channel.

**Image Name** is used to load volumes from the image series. ShootingStar assumes a namnig pattern where all image names end in tX where X represents the unpadded integer time point number.

After loading the first image of the series, a max projection of the volume will be displayed in the window. An ROI must be drawn around the portion of the frame to be processed.

Once all initialization is complete, clicking the “Run” button will launch the automated processing pipeline as well as the user interface for curation.

1. **Editing Tracking Results**

Editing tracking results makes use of the curation interface which launches two windows by default: the image window and the tree window.

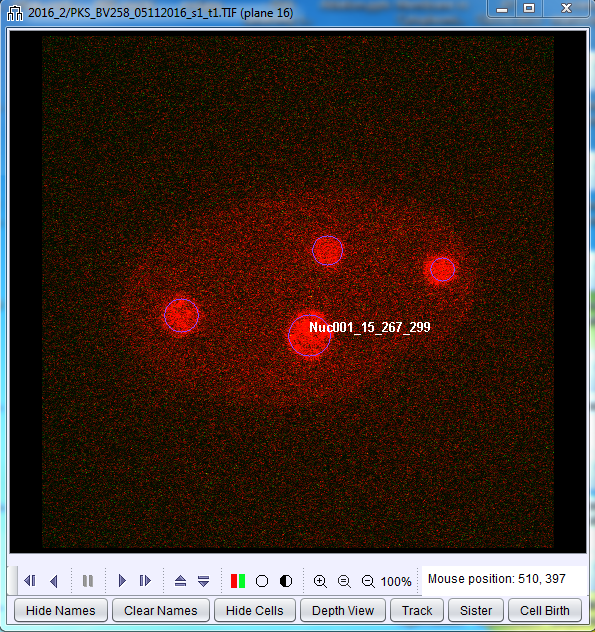
.

Figure 2. Image Window

The Image Window, when active, simplifies exploring the 4D dataset to explore the images. The up/down arrow keys will move the active plane up and down in the volume while the left and right arrow keys will move forward and backwards in time.

Detected cells (shown by default with blue circles) can be selected by right clicking. Once selected the circle will turn white and moving forward/backwards in time will automatically follow the selected cell in z. This behavior can be turned off by clicking the “Track” button.

The names of individual cells can be displayed or hidden by left clicking on the cell. Clicking the “Hide Names” button will disable displaying any cells name while the “Clear Names” button will remove displayed names from all cells other than the active cell.

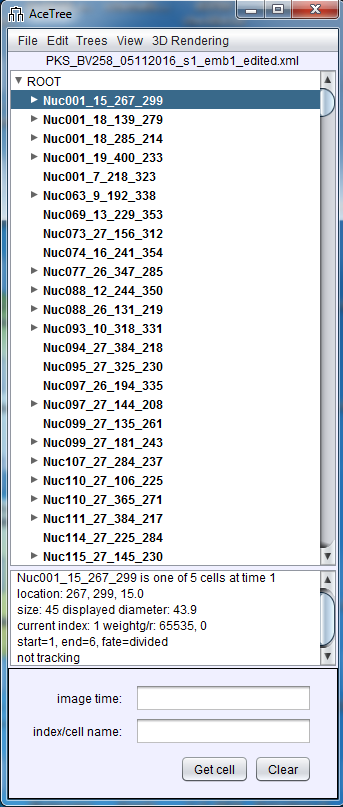


Figure 3. Tree Window

The Tree Window displays all detected cells in a nested list format where daughters of a cell’s division are hidden as collapsed lists. These can be expanded or collapsed again by clicking the triangle next to a cells name. By default the window will generate arbitrary names for all detected cells.

The Tree Window will also display key parameters of the cell including the number of cells present during a time point, the X, Y and Z coordinate of the selected cell, its size and the range of time points during which that cell exists.

Left clicking on an entry in the list will make the cell active and the Image Window will update to the first time point that cell was detected. Right clicking will move the Image window to the last time point that cell was detected, making it easier to jump to cell divisions or track truncations.

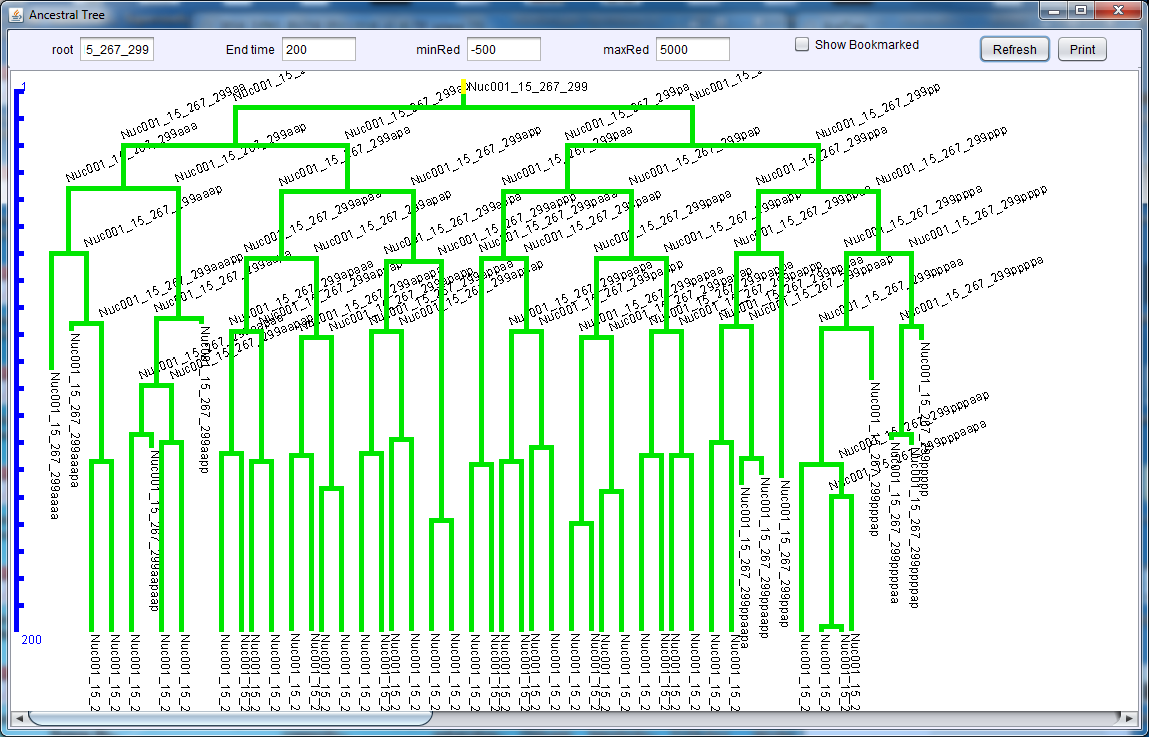


Figure 4. Interactive Lineage Display

The interactive lineage display can be launched from the “Trees” menu in the Tree Window. It is a powerful tool for tracing a sub-lineage that will contain the target cell of interest. The root of the tree can be set to any detected cell and the time range to display can be customized as well. Left clicking on any branch in the tree will select that cell and move the Image Window to the corresponding time point and z-plane of where the tree was clicked while right clicking will move the Image Window to the end of that cell’s track.

Multiple interactive lineages can be generated and set to distinct roots for applications where it is necessary to track multiple cells.

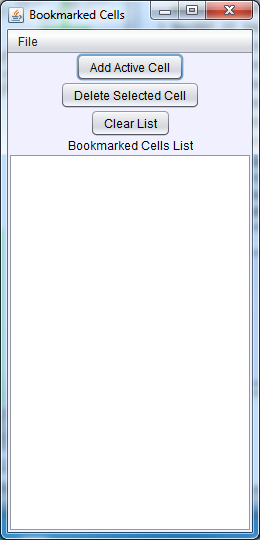


Figure 5. Bookmark Window

The Bookmark window can also aid in tracking multiple cells during complex experiments, it can be launched from the “File” menu in the Tree Window. “Add Active Cell” will add the currently active cell to the bookmark list, clicking its entry will re-activate it and move the image window to that cell. Lists of bookmarks can be saved to an xml file and re-read, a feature that is particularly useful when repeating an experiment where the same cells need to be tracked. Bookmarked cells can also be set to display in a different color in the Image Window by clicking on the display options button (the open circle) and can be highlighted in the interactive lineage display by checking the “Show Bookmarked” checkbox at the top of the window.

Tracking Results can be modified using the two edit windows which are launched by clicking on the “Edit Tools” option in the “Edit” menu in the tree window.

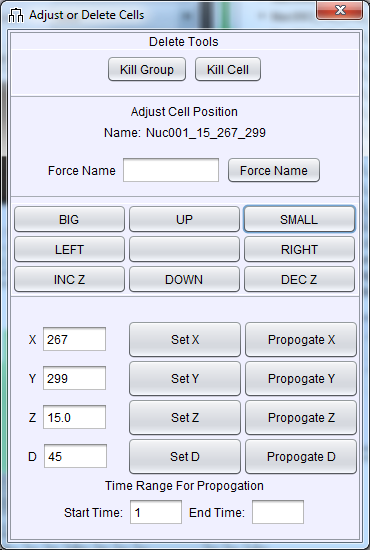


Figure 6. Adjust or Delete Cells Window

The first edit tool allows an individual cell to be deleted, moved, resized or renamed.

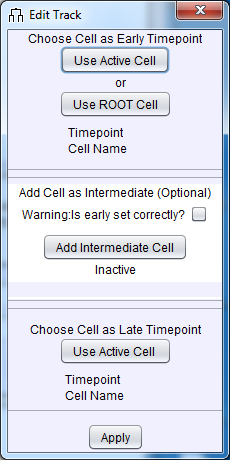


Figure 7. Edit Track Window

The second, and most often used, is the Edit Track window which allows tracking results to be modified. Linking two cells can be accomplished by activating the first cell in an earlier time point and clicking “Use Active Cell” in the early time point portion of the Edit Track Window, then activating the second cell in a later time point and clicking “Use Active Cell” in the late time point portion of the Edit Track Window. Clicking “Apply” then sets the second cell as a successor of the first. If the first cell already has a successor, this is now treated as a cell division, if not the second cell is treated as a continuation of the first. If there are multiple time points separating the two cells, the interface will automatically fill the gap with new cells at cells at each intervening time point positioned linearly between the two endpoints.

For manual tracking or for cases where a cell is not detected for long periods of time, intermediate positions for the cell can be added manually using the “Add Intermediate Cell” button. If a cell is to be added *de novo* first click the “Use ROOT Cell” button to indicate that the cell should be added to the root of the tree, check the warning box and click “Add Intermediate Cell” to activate the tool. Clicking in the Image Window will now create a new cell at the position clicked. Alternatively, a cell’s track can be extended in this manner by setting it as the Early Timepoint cell and then clicking to add intermediate cells in subsequent timepoints.

1. **Ablation Configuration**

Ablation parameters are currently configured in two places, the ShootingStar main script and in the listener implemented in the microscope control software. For the purposes of this tutorial I will describe how this is configured in the case of laser cell ablation using a MicroPoint dye laser controlled using MetaMorph Journals. A similar approach can be adapted to any platform that supports scripting-based control over hardware.

On the ShootingStar side a block of code at the end of the detect\_track\_driver\_allmatlab\_REALTIME script defines the targeting control (see lines 199-261). In the ablation case this consists of a few steps:

1. Reading the target cell names from a text file
2. Checking to see whether the target cell(s) exist in the current time point
3. Incrementing a counter which is compared against a persistence threshold
4. If any of the target cells exceed the persistence threshold, ablation is initiated by writing an ini file specifying the X, Y and Z coordinates of the cell and setting a status flag to tell the listener that the file has been updated
5. Incrementing a “killed” counter which is compared against a threshold to determine the number of iterations of illumination that should be carried out on that cell
6. Incrementing the persistence threshold by the desired interval between iterations
7. Setting a “used” flag which limits perturbation to a single cell per time point to ensure that acquisition of the next time point is not delayed

The companion listener (implemented as a metamorph journal) checks the ini file written by MATLAB for the status flag which, if set to “Ready”, will trigger the listener to read the X, Y and Z coordinate, move to the Z position corresponding to the plane number, draw a region of interest centered at the X and Y coordinate and trigger the micropoint using a for loop the control the number of pulses and the delay between pulses. The “AutomatedAblation.jnl” provides our implementation of this which may be readily ported to other systems using similar hardware.

While the parameters are commented such that they should be relatively easily adjustable by any user, more significant modification to the strategy used for selecting target cells or for the activation of illumination from custom hardware will likely require some degree of familiarity with programming.