ShuttleTracker – User's manual

@version 1.5.3

February 2, 2023

Overview

Overview: Capabilities

ShuttleTracker can detect stained cell nuclei, generate corresponding annular perinuclear extensions, quantify properties of detected objects, and perform frame-to-frame nuclei tracking. Parameters of the image processing algorithms can be changed manually and the effects of these changes are shown immediately overlaid on currently displayed microscopic images. A key capability of ShuttleTracker is that nuclear contours and tracks can be generated automatically and then corrected manually in a WYSIWYG ("what you see is what you get") manner. The tool is scriptable.

Detailed analysis of quantified geometric and photometric properties of tracked nuclei and other objects is out of the scope of the program. Quantifications are exported to plain-text files that can be easily analyzed using external tools. A package of Python scripts and example interactive Python notebooks are distributed together with ShuttleTracker to demonstrate a standard approach to the analysis of features quantified in and exported from ShuttleTracker.

Software homepage: http://pmbm.ippt.pan.pl/software/shuttletracker (permalink), in addition to this manual, contains C++ source code with compilation instructions, precompiled binaries, introductory tutorial, and example inputs.

Launching

Launching with GUI

Initially, ShuttleTracker displays an "empty" window. A directory with images to be analyzed can be selected after clicking menu *File* → *Open directory...* (or by pressing Ctrl+Shift+O). Image files in the selected directory are expected to be properly named, see a further subsection on naming guidelines.

Launching from terminal

When ShuttleTracker is executed from the command line, as the first (positional) argument one may give it a path to a directory with images; ShuttleTracker will then open GUI and immediately load images. If one would like to execute a ShuttleTracker script immediately after loading images, path to a ShuttleTracker script file (*.stscript) should be given after the -s or --script switch. After successful script execution, ShuttleTracker will quit if -q or --quit-after-script-finished has

been given in the command line (or if the script itself contains quit();).

This software has been created with the intent to enable a user to have a good interactive control over the behavior of image processing algorithms, but it is also possible to run ShuttleTracker without displaying GUI (in a so-called headless mode) through a virtual frame buffer such as Xvfb to enable batch processing (e.g., on a cluster). To perform image analysis in the headless mode, one has to provide a ShuttleTracker script and invoke the program from the command line as:

```
xvfb-run ShuttleTracker -q -s PATH_TO_SCRIPT PATH_TO_DIRECTORY
```

(if Xvfb fails to start, just after xvfb-run you may want to add these arguments: --auto-servernum --server-num=1).

Input images

Input images: File names

The following regular expressions are used to fish for images in the selected directory:

- For single time-frame images:
  ^(.+)(?:_?[Cc][Hh]?([0-9]+))\\.(JPE?G|jpe?g|TIFF?|tiff?|PNG|png)$
- For multiple time-frame images (i.e., a time-lapse series), the time-point chunk is: "_?[TtZz]([0-9]+)" (or sk([0-9]+) to accommodate Operetta's Harmony convention), the channel chunk is: "_?[Cc][Hh]?([0-9]+)", and the filename extension chunk is: "\\.(JPE?G|jpe?g|TIFF?|tiff?|PNG|png)".

The time-point chunk and the channel chunk do not have to be zero-padded and may appear in any order, and they can be optionally preceded by some common prefix. Time-point indexing should be zero-based. If, in the case of multiple time-frames, consecutive time points are named using t# (T#), file names should not contain z# (Z#; and vice versa). At least one channel should contain fluorescence microscopy images (as assessed based on pseudo-colors given in metadata).

Prior to version 1.4, file naming was much more restrictive. Currently, because of still scarce test coverage, if possible, it is still recommended to name the files according to the pre-v1.4 stringent convention, that is, in a manner similar to:

```
MyExperiment_t000_ch0.tif
MyExperiment_t000_ch1.tif
```

```
MyExperiment_t000_ch2.tif
MyExperiment_t001_ch0.tif
MyExperiment_t001_ch1.tif
MyExperiment_t001_ch2.tif
```

(and so on).

In the pre-v1.4 convention, common prefix is mandatory, time-point indices are padded with zeroes (in the example it is assumed that more than one hundred but not more than one thousand time points is available); both the index of a time-frame and the channel number are zero-based and both indexes should be used without gaps; filenames of bright-field images, if present, should bear the last (the highest) channel index. So, overall, the pre-v1.4 convention is just a special case of the current more flexible naming convention. If names of your files do not conform to this pattern, and if you are not in an especially adventurous mood today, it is currently recommended to rename your files (Appendix B contains a section with hints on how to rename files in bulk).

Input images: Image formats

ShuttleTracker can read only single-channel (gray-scale) images in the following graphic formats: TIF (*.tif, *.tiff), PNG (*.png), and JPEG (*.jpg, *.jpeg; use of JPEGs in microscopic image analysis is strongly discouraged because they are oftentimes compressed using a lossy algorithm that distorts originally recorded pixel intensities).

Input images: Bit depth

Bit depth of images (maximum-intensity gray-scale value) depends, among others, on the scanner or camera used to record the images. ShuttleTracker learns about bit depth from the metadata associated with a given image set (see a subsection on metadata below). If images have more than 8-bit gray-scale resolution, their bit depth is scaled linearly to 8-bit depth. All image processing and analyses (most importantly, quantifications) are performed using 8-bit images. This setting may crucially affect image preprocessing pipeline.

Input images: Metadata

Metadata are used to:

- establish the mapping between channel number and pseudocolor/LUT,
- declare the bit depth (i.e., gray-scale resolution) of the scanner or camera used to acquire the images,

- get the number of time frames and their relative times (in seconds).

Metadata can be read from the MetaData directory that is exported by Leica LAS software. Information expected to be found in metadata can be also provided manually – by creating a file shuttletracker_metadata.txt in the directory that contains images. In an example meta-file of the following contents:

```
channel 0  Protein1_GFP      green 8
channel 1  Protein2_mCherry  red   8
channel 2  H2B               blue  8
time_interval 120.3
```

first three lines define the mapping from channel number to channel LUT (pseudocolor), provide informative mini-descriptions of image contents, specify bit depth of images; optionally, an additional last column may contain intensity values to be subtracted from image before bit depth conversion is performed (this is absent in the example above). Time interval is expected to be given in seconds. Pseudocolor assignments are used only to display images in specific colors in GUI and may be relevant for generating legible channel overlays (still, within ShuttleTracker all images can be viewed in gray-scale). In the last line, time interval between frames is provided (in seconds). This is a free-format text file: the exact number of white-spaces does not matter; lines can appear in any order.

Viewing

Navigation

After images are loaded, each channel is displayed in a separate pane. Channels are referred to using their assigned pseudocolors. There are seven predefined available pseudocolors: blue, green, red, cyan, magenta, yellow, and bright-field; the software can handle up to seven channels. Default nuclear channel is blue; if absent, then green is the default nuclear channel. ShuttleTracker can compose and display overlays of selected channels.

One may zoom into/out of a region of an image using mouse scroll and move around when the left mouse button is pressed. Fields of view in all channel panes are synchronized to assure that the corresponding image region is displayed in all panes.

Using menu *View* one can hide each pane or black out contained image. Blacking-out is useful when one would like to show only image annotations (called markings). When multiple markings occlude microscopic image in the background, one may also, conversely, hide all markings

to display only the microscopic image (hist: it's more convenient to use the spacebar key instead of selecting hide/unhide in the menu).

Enhanced view

Microscopic images can be displayed (*i*) as originally provided, (*ii*) with pixel intensities normalized, or weakly (*iii*), (*iv*) moderately, or (*v*) strongly auto-leveled. Weak, moderate, strong auto-leveling discards 2%, 4%, 8% of brightest pixels, respectively, and performs normalization of the remaining pixels. Enhancements (*i*)–(*iv*) are available in the saturated (pseudocolor) as well as desaturated (gray-scale) mode. The enhancements affect only how the images are displayed (they do not impact image segmentation nor quantification).

Auto-leveled mode helps to visually discern darker portions of the image (after some time you may find yourself falling into the habit of pressing F1–F5 or F6–F10 just after loading input images). Desaturated images appear brighter on the monitor screen than their saturated counterparts and thus may be more convenient to work on. When you work with desaturated (gray-scale) images, you may want to use green nuclear contours (can be set within the Nuclei Editing toolbox). Of note, switching from-to the normalized view enables assessment whether input images have a reasonable dynamic range.

Toolboxen

All available toolboxen are available from the Toolbox Menu and, additionally, from a dockable tool-bar that can be displayed when selected in a pop-up menu that appears after right-clicking on the main menu bar (under Linux and Windows). You may find it handy to use keyboard shortcuts Ctrl+1, Ctrl+2, and so on, to switch between the toolboxen, especially in the full-screen mode (press F11 or use *Window* menu to enter/exit).

Visibility or "selectability" of objects overlaid over microscopic image depends crucially on the currently active toolbox.

Toolbox: Image masking

Selected areas of an analyzed image can be masked to get rid of the regions in the image that should not be analyzed nor quantified (because, for example, they contain contaminations that would skew analysis). Masked areas are excluded from segmentation (within Nuclei Detection toolbox) and whole-image quantification (within Quantification toolbox).

If one wants to mask some portion of the image, one should activate the Image Masking toolbox, click with right mouse button over the microscopic image to enter the marking mode and while pressing left mouse button mark the area to be masked. Several (non-overlapping) areas can be marked in this way. To clear all markings, one can use button *Reset* in the toolbox. If all desired markings are ready, one can click toolbox button *Mask* to create the mask. Masked areas are persistently blackened out. Masking contours for a current image can be stored and restored using suitable buttons at the bottom of the toolbox pane, labeled *Mask* ↔ *file*. To create masks for a series of images, each image has to be treated separately.

Toolbox: Nuclei detection

Nuclei can be detected based on images with nuclei-specific staining (with the use of, e.g., Hoechst, H2B-GFP) or based on cytoplasmic "counter-staining". In the first case, binarized image resulting from the preprocessing pipeline is locally thresholded to find nuclear contours. In the second case, nuclear contours are found in a preprocessed greyscale image using edge detection.

Image preprocessing

Before nuclear contours are detected, the input image has to be preprocessed. Image preprocessing pipeline consists of the series of filters:

1. **Normalization** linearly stretches the range of intensities of all 8-bit image pixels so that they fill the full available intensity range (0...255).
2. **Denoising** removes salt-and-pepper noise using non-local means denoising. Associated parameter, *strength*, influences jointly several filter properties: size of the scanning window, range of a local scan, and the strength of filter application. This filter is CPU-intensive for larger values of *strength* that imply more non-local scan.
3. **Smoothing** performs bilateral noise filtering. Neighborhood of each pixel is defined based on the filter only parameter, *radius*, and the weights of the photometric and geometric distances are set as linearly proportional to *radius*. This filter may be used as a faster replacement, or as a complement, of the previous denoising step.
4. **Closing** [applicable only in the case of cytoplasmic staining] removes small dark spots and sharpens contours of nuclei by performing morphological dilation followed by erosion of the gray-scale image. A number of consecutive closing rounds can be performed; the sizes of structuring elements for dilation and erosion are forced to be identical not to displace nuclear boundaries.

All these stages (filters) in the pipeline can be individually turned on and off. If the *preview*

stages checkbox has been checked, previews of the effects of the filters will be displayed in separate windows. The filters and their parameters are also described briefly in tooltips that pop up when hovering the mouse cursor over filter names or their parameter labels.

Image preprocessing does not impact image quantification nor the way in which the image is displayed.

Nuclear contour detection

Detection of stained nuclei is performed by searching for blobs in a thresholded and optionally morphologically opened binarized image:

- **Thresholding** [applicable only in the case of nuclear staining] binarizes input image using local adaptive thresholding: for each pixel, a mean intensity of pixels in its neighborhood (of radius controlled by parameter *block size*) is set as the binarization threshold. The threshold can be further adjusted using parameter *base-line*. Local adaptive thresholding handles well images with uneven illumination. If a single, global intensity threshold is desired, one should just increase *block size* to be twice the larger dimension of the image. Optimal values of both the parameters can be found automatically via a naïve grid search (radio button: *Manual* → *Auto*), with resultant nuclear contours solidity being the optimization objective. For some parameter values the algorithm for nuclei detection and splitting may have long completion times, hence it's possible to impose a time-out for testing a single set of parameters (that is carried out in a single thread).
- **Opening** [applicable only in the case of nuclear staining] regularizes silhouettes of nuclei after thresholding and removes small "debris" by performing morphological erosion followed by dilation. A number of consecutive opening rounds can be performed; the sizes of structuring elements for erosion and dilation are advised to be equal to prevent having nuclear contours that would be artificially round or overly tight/loose.

It's also possible to "**uniformize**" nuclei, that is, make the bright nuclei dimmer so that they do not skew computation of statistics of pixel intensities, based on which local thresholds are determined. Uniformization is advised to be used whenever nuclei have widely differing intensities and brighter nuclei appear to cause shift/shrinkage of contours of adjacent darker nuclei.

In the case of cytoplasmic staining, edge detection is performed according to the Canny method. Initial contours are proposed within usual Canny hysteretic thresholds (the *lower* acceptance threshold and the *ratio* of the higher-to-lower threshold are defined by the user) and then subjected to filtering controlled by a single metaparameter *tolerance* that gauges acceptance/rejection of contours with outstanding area, irregular shape, extraordinary intensity, etc. Additional *recovery* mode may be turned on to attempt finding nuclear contours based on

non-closed contours (this is a relatively CPU-intensive and thus multi-threaded procedure).

Nuclei assessment and splitting

Median nuclear contour area serves as the reference contour area and two user-defined parameters, *min area* and *max area*, expressed as fractions of the reference median area, are used to classify all contours as: (*i*) too small to be a nucleus, (*ii*) too large to be a single nucleus, or (*iii*) having area close enough to the reference nucleus and thus being single-nucleus contours. Too small contours are called *debris*. Contours that are too large can be split (declumped) depending on their solidity defined as the surface area of the contour divided by the surface area of its convex hull. If a too large contour has solidity higher that user-defined *min solidity*, it is deemed non-splittable. However, when solidity of a too large contour is lower, then it will be subjected to splitting based on convexity defects called *indents* (this approach appears in many cases more robust than standard watershed-based segmentation and is especially well suited to split contours of nuclei determined based on cytoplasmic staining). Indents that can be used for splitting should be sufficiently large, which is decided based on the user-defined *min indent* parameter. A split can be attempted only when a contour has two or more indents. Splitting works recursively as long as there are indents that can be used to obtain contours of nuclei of areas close to the reference area. Contours that result from splitting but according to the *min area* parameter are too small are called *split orphans*. Descriptions of parameters involved in nuclei assessment are shown in tooltips that pop up when hovering the mouse pointer over parameter labels.

Nuclear contours are drawn over microscopic images in colors associated with their origin in the above-described segmentation algorithm: typically sized contours that do not result from splitting (denoted N in the message bar and log window) are white, typically sized contours resulting from splitting (SpN) – green, large non-splittable (nSp) – red, split orphans (relatively rare, denoted SpO) – yellow, debris (D) – violet. Obtained contours are scored according to the variance of their solidities: the lower is variability, the higher is the score. Scoring is used currently for automated optimization of thresholding parameters at the image preprocessing stage.

If the checkbox *auto-click* has been checked, then one can adjust image preprocessing and nuclei assessment parameters and nearly instantaneously observe their impact on nuclei detection (without clicking *Detect* button after each parameter adjustment). Two CPU-intensive operations are *Smoothing* in the image preprocessing pipeline and *reconstruct* in contour detection (in case of cytoplasmic staining).

All parameters of image preprocessing, contour detection, and nuclei assessment may be as-

signed within a script or saved to/loaded from a text file nuclei_detection.ini using buttons labeled *Settings ↔ file*.

Toolbox: Nuclei editing

It is possible to correct automatically detected nuclei one-by-one by deleting existing nuclear contours and adding new ones manually:

- **To remove a nucleus**: select the unwanted nucleus left-clicking on it and then make a right-click.
- **To add a nucleus**: after right-clicking in a free space, press left mouse button and use mouse pointer to draw the contour of a new nucleus.
- **To split a nucleus**: select the nucleus to be split by left-lick; then press the left mouse button and when keeping it pressed click with the right mouse button to start drawing a demarcation line.

All nuclei are numbered sequentially starting from 1. Use checkbox *numbers* in box *Annotations* to show/hide the numbers. After a nucleus of a given number is removed, all higher nuclei numbers are decreased by one. After a new nuclear contour has been drawn, it is assigned the smallest available number (equal to the current number of all nuclear contours). A contour can be redrawn, that is, replaced without changing its number, only in the tracks editing toolbox.

Double-left-click on a contour can be used to manually toggle a binary state of a nucleus called *toggled*. This state is saved in both the coordinates and the quantification files, and thus can be used during later stages of analysis. One can toggle nuclei that lie on the image border using a suitable button in the box *Editing by location*. Using other buttons in this box one can remove such border touching contours as well as contours classified as debris or split orphans. One may externally provide a list of numbers of nuclei to be highlighted in ShuttleTracker. A path to the file with nuclei numbers, listed one by line, is specified by the setting path_to_file_with_nuclei_to_be_highlighted of application preferences (see section Preferences).

Coordinates of nuclear contours can be save to/loaded from a text file using button labeled *Nuclei ↔ file*. The order of contours in the file corresponds to the numbering displayed on the screen.

When all nuclei in a frame are removed (which takes place also upon nuclei contours file loading), all perinuclei and all tracks are removed as well.

Toolbox: Perinuclei derivation

Finding cell boundary in fluorescence microscopy images can be very hard or even unfeasible. Thus, instead, one can mark perinuclear annulus that serves as a proxy of the cytoplasmic region. Determining such proxy is much easier than finding cell boundaries and may be sufficient to capture mean or median pixel intensities in the cytoplasm. ShuttleTracker derives perinuclear annuli (called just perinuclei) based on nuclear contours and two principal user-defined parameters: *inner offset* that sets the distance from a nuclear contour to the inner boundary of a corresponding perinucleus (in pixels), and *ring width* that tells how thick the perinucleus should be (in pixels). Perinuclei that would overlap are eroded at the potential overlap site; erosion strength is controlled by parameter *overlap avoid*. Perinuclei that do not overlap but nevertheless are expected to be placed safely apart from their neighbors may be locally abraded with the strength controlled by parameter *neighbor avoid*. Parameter *min area* is used to weed out perinuclei that are too small (and thus likely non-representative).

In many cell lines, boundary of the cell (visible in the cytoplasmic staining channel) may lie very close to the boundary of the nucleus (visible and determined in the nuclear staining channel). In such case, a perinuclear contour that would go around the boundary of the nucleus would erroneously include regions of the image in which there is no cytoplasm. One may mitigate this issue by excluding background from perinuclei using *background avoid* option. It is assumed that background is sampled in cytoplasmic staining channel using Regions Editing toolbox. Based on currently marked regions, mean background intensity (μ) and standard deviation of background intensity (σ) are computed. Parameter *expansion* is used to set the effective background threshold as μ + expansion × σ. This threshold value discriminates background vs. non-background (think: non-cell vs. cell, respectively) and is applied to obtain a background mask. Background mask results from thresholding of the image in the specified cytoplasmic channel. Smoothing, performed by dilation of the background mask with a kernel of specified radius (parameter *smoothing*), eliminates potential “spottyness” of the mask.

Please note that a single perinucleus can comprise several disjoint contours.

One cannot edit perinuclei; when having a given set of nuclear contours along with specific derivation settings, one may faithfully derive a set of corresponding perinuclei. Perinuclei “derivation” parameters can be assigned within a script or saved to/loaded from a file perinuclei_derivation.ini using buttons labeled *Settings* ↔ *file*.

Toolbox: Regions editing

Within Regions Editing toolbox, arbitrary regions may be marked on the images. They may be used, e.g., to manually mark cell boundaries or background regions (to correct for background intensity or characterize noise in further analysis).

To enter marking mode, press right mouse button. Then, using left mouse button, mark the region (you don't have to draw a closed loop – the contour will be closed automatically). Regions are annotated with ordinal numbers prefixed with R (use checkbox *numbers* in box *Annotations* to show/hide the annotations).

The toolbox can suggest regions of the image in which there are likely no objects of potential interest (intensity is low), which can be useful for finding background regions. First, images in all channels *that are included in overlays* are decomposed into rectangular tiles. Then, for each channel, tiles are rank-ordered based on their average intensity. Finally, each tile is scored based on the product of its ranks in all channels. Adjacent background tiles may be merged.

To obtain a very rough segmentation of the image into regions adjacent to the nuclei, one can automatically tesselate the image into Voronoi "cells". Geometric centers of the nuclei serve as dual Delaunay triangulation points. All Voronoi "cells" may be scaled homothetically and then eroded (strong erosion should be used with care as it may wipe out small Voronoi "cells", and as a result the number of obtained regions will not match the number of nuclei, which may not be the desired outcome).

Region contours can be saved to/loaded from a file using buttons labeled *Regions* ↔ *file*.

Toolbox: Quantification

Areas enclosed in nuclear contours, perinuclear contours, and region contours can be quantified w.r. to surface area and fluorescence intensity sum, 1^{st} quartile, median, 3^{rd} quartile, mean, standard deviation, min, and max, and other features, in each channel separately. The same values can be calculated for the whole image. Nuclei are additionally characterized by their eccentricity and "toggled" status (described in the section on Nuclei Editing toolbox). A complete list of quantified features is provided in Appendix B.

Channels of interest, in which the contours are to be quantified, can be individually selected and named in the box *Channel descriptions*. The names cannot contain whitespaces nor commas as they are used as prefixes of data columns in exported files. Quantifications for the current time frame in selected channels are exported to the following comma-separated value (CSV) files:

- file named *-nuc_quant.csv contains quantifications of all nuclei;

- file named *-per_quant.csv contains quantifications of all perinuclei, ordered accordingly to their corresponding nuclei;
- file named *-reg_quant.csv contains quantifications of all regions;
- file named *-reg_quant.csv contains quantifications of all halos, ordered accordingly to their corresponding nuclei;
- file named *-img_quant.csv contains quantification of the whole image.

Quantification files should be easy to read-in and analyze in external tools. See section Further analysis to learn about auxiliary analysis tools that are deployed with ShuttleTracker. In addition to CSV files, ShuttleTracker can export a PNG image with all contours marked. To add/remove contour labels in the exported image, use checkboxes in *Annotations* boxes in the Nuclei Editing and Regions Editing toolboxen.

Toolbox: Tracking

To perform tracking, nuclei from adjacent time frames are matched using a greedy algorithm that takes into account predicted nuclei positions and their geometric and photometric characteristics.

Prediction of nuclei positions

The next position of a nucleus is computed by extrapolating nucleus displacements observed in several previous time frames. The number of previous displacements used for extrapolation is determined by parameter *memory*. Previous displacements are weighted according to the *conservation* parameter. The weights are proportional to $\exp(-k/H \times (1 - c) \times \lambda)$, where k is the zero-based index of the displacement (counting starts from the track end), M is the *memory*, c is the *conservation*, and λ is an arbitrary scaling constant (hardcoded as equal 5). If *contribution* parameter is 1, the extrapolated position is taken as the final predicted nucleus position. If one would like to use a linear combination of the last position and the extrapolated position, one can set the *contribution* parameter between 0 and 1. Parameter *contribution* equal 0 means that extrapolation is not performed and simply the last position is used (the same effect can be obtained by just unchecking the *Position prediction* checkbox).

Nuclei matching algorithm

Nuclei matching is performed in three phases:

1. **Nuclei similarity scoring**. Similarity in all nuclei pairs, in which one nucleus comes from to the previous time frame and the other nucleus belongs to the current time frame, is estimated. Nuclei similarity score is computed based on the following features:

- *proximity* – calculated based on the distance of predicted nuclei positions,
- *surface area*,
- *eccentricity* – calculated based on how different is the nucleus shape from circle,
- *orientation* – calculated using the angle of the first principal axis
- *sum of pixel intensities* – in the nuclear staining channel,
- *inertia of pixel intensities* – in the nuclear staining channel (quantifies how peripheral *vs.* central is the distribution of pixel intensities in a quantified contour),
- *distribution of pixel intensities* – the Kolmogorov–Smirnov distance between two distributions of nuclear intensities is computed.

Importantly, each feature forms an independent order statistics. Similarity scores, computed as weighted linear combinations of order statistics (plural), form a matrix, where the number of rows and the number of columns are equal to the number of nuclei in the previous and the current time frame, respectively. Relative weights of the order statistics (plural) are defined by the user.

2. **Internuclear distance-dependent similarity scaling**. Each estimated similarity score is scaled depending on the predicted internuclear distance. Parameter *similarity scaling* is the exponent for center-of-the-mass (COM)-dependent scaling of inter-nuclear distances in the feature space (dissimilarities): scaled dissimilarity is proportional to dissimilarity × (predicted distance)^(*similarity scaling*).

3. **Extension of tracks based on max similarity**. A predicted distances-scaled similarity score of maximum value is sought in the scaled similarity matrix. When found, two corresponding nuclei are considered as matched. The procedure of finding maximum similarity is repeated for the still unmatched nuclei until the predicted distance of nuclei in a potential match is larger than the allowed inter-nuclear spacing, set as parameter *Δxy cutoff* expressed in median inter-nuclear distances.

One can require that a track is not extended if an extension would result in a significant relative reduction of the area of the nucleus.

Currently, tracking must be gapless, that is, the outline of the nucleus has to be marked in every consecutive track frame. Joining tracks interrupted by gaps is delegated to the post-processing stage.

In some frames, nuclear status can be set to “toggled” so that it is possible to exclude, e.g., incomplete nuclei in further analysis (resulting in, for example, discontinuity in a track). The “toggled” status of nuclei does not influence nuclei matching.

Tracking settings can be saved to/loaded from file tracking.ini using buttons labeled *Settings ↔ file*.

Toolbox: Tracks editing

Tracking results are displayed with *segments* – straight lines that connect centers of the mass of matching nuclei. If no matching nuclei were found for a given nucleus in the previous and in the next time frame, a *stub* is created and shown in the center of the mass of the derelict nucleus.

Tracks are listed in a tabular form, sorted by their lengths; tracks of the same length are ordered by a descending confidence score. Each track can be selected by clicking on any of its segments drawn over the image or by selecting an entry on the tabular tracks view. After visual inspection, tracks can be annotated as *revised* using checkboxes in the last column of the tabular tracks view.

It is possible to remove a track by right-clicking on the corresponding row of the tabular tracks view. It is also possible to correct erroneous tracks manually: segments can be added and removed with mouse:

- **To add a segment**, select one segment or (a stub) with mouse left-click, press Ctrl key (Command key on the macOS), and select another segment (or a stub) with the left-click. As tracks are assumed to be continuous, only the tracks that end/begin at consecutive time frames can be joined.
- **To remove a segment**, select it with the mouse left-click and then right-click on the selection. If segment removal splits a track, two resulting tracks will be listed as two last entries in the tabular tracks view.

Nuclei can be also edited as follows:

- **To add a new nucleus** (when having tracks), switch to the Nuclei Editing toolbox, draw a new contour, then switch back to the Tracks Editing toolbox and click with mouse middle button on the new contour to create a new stub. Stubs can be merged into segments, as described previously.
- **To correct a nucleus within a track**, first: select it with the mouse left-click, second: while pressing right mouse button make the left-click, third: draw the new contour (by pressing left mouse button), or left-click in any empty space to abandon the correction/replacement mode.

Having tracks, one cannot remove any existing nuclei; however, the nucleus “toggled” status can be toggled and used during further data analysis to exclude parts of a track.

Tracks are saved to files in a simple textual format. First column is the frame number (zero-

based); the second column is the ordinal number of the nucleus (nuclei numbers do not have to be consistent along the track). Nuclei numbering is consistent with the nuclei numbering in the nuclei contours file and in the nuclei quantification file exported for a given time frame. Nuclei numbers start from 1.

Please note that when all nuclei in a frame are removed (which takes place, e.g., upon nuclei contours file loading), tracks are removed as well.

Scripting

ShuttleTracker ships an embedded ECMAScript (JavaScript) interpreter to enable automation. The interpreter has been expanded with several syntactic extensions (e.g., to facilitate looping over all time frames). Scripts are edited and executed in Script Editor (menu *Script* → *Edit*...). Actions and parameters that can be called and set programmatically are listed in a side panel of the Script Editor; the panel appears after clicking on *Show API*; additionally, Appendix A of this manual contains a documentation of the programming interface with actions and parameters grouped by toolbox. Several parameters have long (yet descriptive) names, so the editor has been equipped with auto-completion. Simple debugging capabilities are provided: error messages and problematic script line numbers are displayed upon script failure at the bottom of the editor.

The scripts are an integral part of the software tool: several key functionalities of ShuttleTracker are provided through scripts (for example, a script is used to play the sequence of images as a movie) and operations to be performed in the headless mode have to be expressed in scripts. The scripts can be launched directly from the *Scripts* menu (also via keyboard shortcuts; shortcuts are assigned after listing script file names in the lexicographic order). Newly added scripts, saved as *.stscript files, become immediately available in the menu.

On Linux, script files are installed in *INSTALL_PREFIX*/share/shuttletracker/scripts. On Windows, scripts are stored as plain text files in user's Document/STScripts directory. On macOS, scripts are stored in ShuttleTracker.app/Contents/Resources/shuttletracker/scripts. On all systems, the exact location can be checked at the bottom of the menu *Help* → *About* window (Ctrl+Shift+V).

Preferences

Several preferences can be set or unset in the Preferences menu:

- *Over-8-bit images: normalize to camera bit depth if known*. This preference affects the way in which 16-bit images are loaded.
- *Memory: Load and store original images*. If enabled, all original images (after conversion to 8 bit depth) may be stored in RAM for fast access or, if disabled, accessed from disk on-demand, which is slower but saves RAM.
- *Memory: Precompute and store overlay images*. When original images are stored in RAM, one can also store pre-computed image overlays.
- *Memory: Cache images in fast display buffers*. This preference affects the way in which images are prepared for display just after they are loaded. Cached images occupy more space in RAM but are displayed faster.
- *Bright-field images: Skip when loading*. This preference affects the way in which images are loaded. If BF images are not to be used, skipping them will save some memory.
- *Bright-field images: Hide even when available*. Auto-hide BF panel even when BF images were loaded.
- *Bright-field images: Exclude from initial overlays*. BF images overlaid on fluorescence visually images attenuate (pseudo)colors, thus it may be preferred to not use BF images for overlays. It may be actively overridden by manually selecting overlay components in menu *View*.
- *View: Retain channel panes and overlay composition*. Use current view settings (channel panes, channels included in overlays, and zoom) for the next loaded image series. This settings is allowed to override the setting 'Bright-field images: Exclude from initial overlays'.
- *Upon startup: check for updates*. Connect to the server pmbm.ippt.pan.pl to learn about the latest available ShuttleTracker release.

When exiting, ShuttleTracker stores these preferences and some other settings to restore them when launched again. Stored settings include: widths of contours, ranges of parameters for performing nuclei detection parameters scan, nuclei detection scoring criterion, window geometry, the location of the last opened directory, exported quantifications, etc. On Linux, preferences and settings are saved to an INI-file (which is placed in *INSTALL_PREFIX*/shared/shuttletracker/config). On macOS, settings.ini is located in ShuttleTracker.app/Contents/Resources/shuttletracker/config. Under Windows, preferences are saved to the system registry (don't worry: upon deinstallation, the registry is purged of all entries created by ShuttleTracker).

Further analysis

Quantifications and tracks generated with ShuttleTracker as saved to plain-text files in the CSV format next to the image files. The output files contain comma-separated columns of numeric values and should be thus easily parseable by external data analysis tools and scripts. A Python module, **shuttletracker**, shipped in *INSTALL_PREFIX*/share/shuttletracker/analysis, can be used to join tracks and respective quantifications in order to, e.g., characterize temporal changes in subcellular location of fluorescently labeled proteins or infer cell division events. A demonstration how one can use the Python module within a Jupyter notebook is provided in *INSTALL_PREFIX*/share/shuttletracker/docs.

Getting help

If you have questions, bug reports, or feature requests, please feel free to send them to shuttletracker.software@gmail.com .

Credits

Development of ShuttleTracker has been significantly influenced by suggestions and feature requests of Maciej Dobrzynski (U Bern), Frederic Grabowski (U Warsaw) and Karolina Tudelska (U Lancaster/U Warsaw). The author is also grateful to Piotr Korczyk (IPPT PAN), Zbyszek Korwek (IPPT PAN), Tomek Lipniacki (IPPT PAN), Joasia Markiewicz (IPPT PAN), Paweł Nałęcz-Jawecki (IPPT PAN), and Wiktor Prus (IPPT PAN) for testing and feedback, and to Nont Kosaisawe (UC Davis) for discussions.

The author is also indebted to the creators of and contributors to the OpenCV libraries and Qt framework, which helped to equip ShuttleTracker with essential image processing capabilities and interactive user interface.

Appendix A: Programming interface

Actions that can be triggered by clicking buttons and parameters that can be set using spinboxes or textfields can all be triggered or set programmatically. Denotational convention used below is as follows:

– numeric argument or return value;
$ – string argument or return value;
% – boolean argument or return value;
@ – predefined argument;
* – wildcard for # or $ or @.

General actions

open_directory($); – same as File → *Open directory…*;
set(@,*); – sets a value (2nd argument) of a predefined parameter (1st argument);
$ get(@); – returns a value of a predefined parameter;
echo(*); – displays a passed value in status bar of the main window;
panic($); – stops script execution and displays a message;
quit(); – closes application;
# frames_count();
# current_frame();
go_to_frame(#);
repaint();
millisleep(#);

Masking toolbox

Actions:

```
apply_masking_contours();
# save_masking_contours();
# load_masking_contours();
% has_masking_contours_file();
```

Nuclei Detection toolbox

Actions:

```
    detect_nuclei();
    detect_nuclei_with_parameter_scan();
  # nuclei_count();
    save_nuclei_detection_settings();
    load_nuclei_detection_settings();
  % has_nuclei_detection_settings_file();
```

Parameters:

```
    nuclei_detection_channel
    nuclei_detection_staining_nuclear
    nuclei_detection_normalization
    nuclei_detection_denoising
    nuclei_detection_denoising_strength
    nuclei_detection_smoothing
    nuclei_detection_smoothing_radius
    nuclei_detection_thresholding
    nuclei_detection_thresholding_block_size
    nuclei_detection_thresholding_baseline_offset
    nuclei_detection_thresholding_auto
    nuclei_detection_thresholding_baseline_offset_auto_from
    nuclei_detection_thresholding_baseline_offset_auto_upto
    nuclei_detection_thresholding_baseline_offset_auto_step
    nuclei_detection_thresholding_block_size_auto_from
    nuclei_detection_thresholding_block_size_auto_upto
    nuclei_detection_thresholding_block_size_auto_steps
    nuclei_detection_uniformization
    nuclei_detection_uniformization_brightness_threshold
    nuclei_detection_opening
    nuclei_detection_opening_repeats
    nuclei_detection_opening_erosion_base_size
    nuclei_detection_opening_dilation_base_size
    nuclei_detection_closing
    nuclei_detection_closing_repeats
    nuclei_detection_closing_dilation_base_size
```

```
nuclei_detection_closing_erosion_base_size
nuclei_detection_contour_detection_threshold_low
nuclei_detection_contour_detection_thresholds_ratio
nuclei_detection_contour_detection_tolerance
nuclei_detection_contour_detection_reconstruct
nuclei_detection_nuclei_assessment_min_solidity
nuclei_detection_nuclei_assessment_min_area_as_median_fraction
nuclei_detection_nuclei_assessment_max_area_as_median_fraction
nuclei_detection_nuclei_splitting_adjacent
nuclei_detection_nuclei_splitting_min_pocket_area
```

Nuclei Edit toolbox

Actions:

```
remove_nuclei();
remove_toggled_nuclei();
remove_debris();
remove_border_nuclei();
remove_interior_debris();
remove_split_orphans();
toggle_all_nuclei();
toggle_border_nuclei();
# save_nuclei_contours();
# load_nuclei_contours();
% has_nuclei_contours_file();
highlight_nuclei_from_file();
toggle_overlay_channel(@);
toggle_overlay_channel($);
```

Parameters: nuclei_editing_nuccut_channel_main nuclei_editing_nuccut_channel_aux1 nuclei_editing_nuccut_channel_aux2 nuclei_editing_nuccut_channel_main_weight nuclei_editing_nuccut_channel_aux1_weight nuclei_editing_nuccut_channel_aux2_weight nuclei_editing_nuccut_color_conservativeness nuclei_editing_nuccut_edge_impassability nuclei_editing_nuccut_exclude_toggled nuclei_editing_nuclei_labels nuclei_labels

Perinuclei Derivation toolbox

Actions:

```
    derive_perinuclei();
    remove_perinuclei();
    save_perinuclei_settings();
    load_perinuclei_settings();
  % has_perinuclei_settings_file();
```

Parameters:

```
    perinuclei_inner_offset
    perinuclei_width
    perinuclei_neighbor_avoidance
    perinuclei_overlap_avoidance
    perinuclei_min_part_area
    perinuclei_background_avoidance
    perinuclei_background_avoidance_channel
    perinuclei_background_avoidance_expansion
    perinuclei_background_avoidance_smoothing
```

Regions toolbox

Actions:

```
    sample_background();
    tesselate();
    remove_regions();
  # regions_count();
  # save_regions_contours();
  # load_regions_contours();
  % has_regions_contours_file();
```

Parameters:

```
    regions_background_subdivs
    regions_background_count
    regions_background_merge_adjacent_tiles
    regions_voronoi_scaling
    regions_voronoi_erosion
```

```
regions_voronoi_exporting_neighborhoods
regions_labels
```

Quantification toolbox

Actions:

```
export_all_quantifications();
export_nuclei_quantifications();
export_perinuclei_quantifications();
export_regions_quantifications();
export_image_quantifications();
```

Parameters:

```
quantify_channel_green
quantify_channel_red
quantify_channel_blue
quantify_channel_cyan
quantify_channel_magenta
quantify_channel_yellow
channel_green_description
channel_red_description
channel_blue_description
channel_cyan_description
channel_magenta_description
channel_yellow_description
quantify_image_to_csv_file
quantify_image_only_nonmasked
quantify_nuclei_to_csv_file
quantify_nuclei_to_png_file
quantify_perinuclei_to_csv_file
quantify_regions_to_csv_file
quantify_regions_to_png_file
```

Tracking toolbox

Actions:

```
  track_nuclei();
# save_tracking_settings();
# load_tracking_settings();
% has_tracking_settings_file();
```

Parameters:

```
  tracking_prediction
  tracking_prediction_memory
  tracking_prediction_conservation
  tracking_prediction_contribution
  tracking_similarity_weight_proximity
  tracking_similarity_weight_area
  tracking_similarity_weight_eccentricity
  tracking_similarity_weight_orientation
  tracking_similarity_weight_intensity_sum
  tracking_similarity_weight_intensity_inertia
  tracking_similarity_weight_intensity_distribution
  tracking_feature_distance_scaling
  tracking_displacement_cutoff
  tracking_nucleus_area_drop
  tracking_nucleus_area_drop_threshold
```

Tracks Editing toolbox

Actions:

```
# tracks_count();
  remove_tracks();
  remove_tracks_nonrevised();
# save_tracks();
# load_tracks();
```

Parameters:

```
  tracks_save_only_revised
```

Appendix B: Quantified features

Nuclei, perinuclei, regions

Geometric:

- area
- area_masked (only nuclei)
- eccentricity (only nuclei)
- center_x (only nuclei)
- center_y (only nuclei)

Photometric:

- intensity_min
- intensity_max
- intensity_median
- intensity_sum
- intensity_quartile1
- intensity_quartile3
- intensity_mean
- intensity_stddev
- intensity_inertia (only nuclei and regions)
- intensity_80_mean (mean computed after dropping 10% of brightest and 10% of dimmest pixels)
- intensity_80_stddev (standard deviation computed after dropping 10% of brightest and 10% of dimmest pixels)
- intensity_h{1,2}_mean
- intensity_t{1,2,3}_mean
- intensity_q{1,2,3,4}_mean
- linear_{025,050,075,100,200,300,400,500} (only halos)
- sigmoidal_{10,15,20}_{100,125,150,175,200} (only halos)

Bookkeeping:

- nucleus_id (only nuclei)
- corresp_nucleus_id (only perinuclei)
- region_id (only regions)
- is_toggled (only nuclei)

Image

Geometric:

- area
- area_masked

Photometric features of the image same as photometric features of nuclei.

Appendix C: Hints and troubleshooting

Batch file renaming

Swapping parts of individual file names. Under Linux, on Debian and derivatives, there is a Perl script (originating from https://metacpan.org/release/File-Rename and installable via apt-get install rename) that is a very convenient tool for renaming multiple files. For example, if you want to rename a bunch of files named as follows:

```
MyExperiment_c0_t00.tif
MyExperiment_c1_t00.tif
MyExperiment_c2_t00.tif
MyExperiment_c0_t01.tif
MyExperiment_c1_t01.tif
MyExperiment_c2_t01.tif
```

it is sufficient to type:

```
rename 's/_c(\d)_t(\d+)/_t\2_c\1/' *.tif
```

(\1 and \2 are ordered references to two groups of digits captured within parentheses). If channel indices do not begin at 0 or their numbering is discontinuous, you should treat each channel individually:

```
rename 's/_c1_t(\d+)/_t\1_c0/' *.tif
rename 's/_c2_t(\d+)/_t\1_c1/' *.tif
rename 's/_c4_t(\d+)/_t\1_c2/' *.tif
```

The same rename script can be installed under macOS (e.g., with brew install rename); analogous tools exist for Windows (e.g., RegexRenamer).

If time-point indices do not begin at 0, as a workaround you may duplicate time frame for time=1 and name it appropriately for time=0 (do not forget to account for this quick fix in your further analyses).

Upon frame extraction and/or bit depth conversion. Proper naming of files can be often assured during conversion: for example, assume that you have a two-channel movie saved as two multi-layer TIFF files. In the command line under Linux or macOS, having bash shell and ImageMagick installed, you can use the following method to generate a series of properly named input images:

```
export NFRAMES=100
```

```
for i in $(seq -w 0 $(($NFRAMES - 1)); do
  convert  MyMovie_Channel1.tif[${i}]  MyMMovie_t${i}_ch0.tif
  convert  MyMovie_Channel2.tif[${i}]  MyMMovie_t${i}_ch1.tif
done
```

Bit depth conversion

Using tiff_expander.py. You can perform bit depth conversions with a featuring standalone script tiff_expander.py that allows for converting images of any over-8-bit depth to 8-bit depth with or without "clipping" of the over-the-range values (clipping can be advantageous when, for example, a 14-bit image, that has very few pixels of intensity equal or higher than 2^12, is converted to an 8-bit image by treating it as a 12-bit image; in this way, more "resolution" in lower-value pixels will be preserved than in the case of 14-to-8 bit conversion). To learn how to use the script, invoke it with argument -h (or check an example invocation in the docstring of the script file). The script is usually placed in *INSTALL_PREFIX*/shared/shuttletracker/support, requires python3-pil and python3-numpy, and works on multi-frame TIFF files. Currently it is assumed that each channel is stored in a separate multi-frame TIFF file.

Using ImageMagick. Alternatively, you can perform bit depth conversions using ImageMagick by passing an appropriate convert parameter: -evaluate multiply *A_MULTIPLE_OF_TWO* (in addition to -depth 8). This may be done simultaneously with image extraction from multi-frame TIFFs:

```
export NFRAMES=100
for i in $(seq -w 0 $(($NFRAMES - 1)); do
  convert  Expt_Ch1.tif[${i}]  -evaluate multiply 4  -depth 8  Expt_t${i}_ch0.tif
  convert  Expt_Ch2.tif[${i}]  -evaluate multiply 4  -depth 8  Expt_t${i}_ch1.tif
done
```

Of note, both tiff_expander.py and ImageMagick (convert) will not process TIFF files that are larger than 4 GB. Such files may be processed by ImageJ.

Interoperability with ImageJ

If your ImageJ stack exported to a multiple TIFFs cannot be read by ShuttleTracker, make a copy of the original multi-frame TIFF file (it will be overwritten) and process it within ImageJ/Fiji as follows:

1. Open the multi-frame TIFF file using menu *File* → *Open*...
2. Press Shift+Z and in the pop-up window select Grayscale.

3. Overwrite the multi-frame TIFF file using menu *File* → *Save*.
4. Using menu *Plugins* → *Bio-formats* → *Bio-formats Exporter*, dump the stack to individual TIFF files so that each time point and each channel end up in a separate image file of names in which time points are padded with zeroes (to this end, place ticks in three appropriate checkboxes); you may use LZW compression.

Resulting files may require renaming to conform to the expected naming convention.

Alternatively, you may use the ImageJ macro MasterTiffToAtomicTiffs.ijm deployed in directory *INSTALL_PREFIX*/share/shuttletracker/support. This macro performs processing that is analogous to the above-described procedure and names the exported files according to an expected naming convention.

High-resolution displays

If you use a monitor display with increased pixel density, such as Retina, and graphical user interface of ShuttleTracker looks uncomfortably small, you may scale up all graphical widgets by setting the environmental variable QT_SCALE_FACTOR; for example, to enlarge widgets to 150% of their original size, set QT_SCALE_FACTOR to value 1.5. In Linux or macOS terminal, the variable can be set in the bash shell with:

```
export QT_SCALE_FACTOR=1.5
```

before launching ShuttleTracker. Note that altering this setting globally may impact graphical user interfaces of other applications that also use Qt libraries. In the command line under Linux and macOS, you may set the scaling only for the current launch of ShuttleTracker (without impacting other applications) by prepending the path to the executable binary by QT_SCALE_FACTOR=1.5.

If you work on Windows and various graphical widgets of ShuttleTracker are scaled disproportionately on your high-DPI display, you can create file qt.conf with the following contents:

```
[Platforms]
WindowsArguments = dpiawareness=0
```

and place this file next to the binary executable (ShuttleTracker.exe).

Multi-threaded image processing

The number of threads used by several image processing algorithms can be controlled by the environmental variable SHUTTLETRACKER_OPENCV_NUM_THREADS. In Linux or macOS terminal, the variable can be set in the bash shell with:

```
export SHUTTLETRACKER_OPENCV_NUM_THREADS=2
```

before launching ShuttleTracker. Setting this number to values higher than 2 would result in a marked speed-up for images significantly larger than 1000×1000 pixels.

Appendix D: Version history

Version 1.6.0 (UPCOMING)

- Input: can parse names of files exported by Operetta's Harmony.
- Rendering: contours of nuclei can be white.
- Export: contours drawn over an exported image are antialiased.
- Preferences: Add a preference to retain channel view across image directory loadings.
- Quantification: relative inertia is computed non-canonically (intensity is squared).
- Input: image paths containing a channel-like chunk are interpreted correctly.
- Script: disallow a script command to run in parallel in another thread.
- Quantification: prevent manual quantification in the overlay pane.
- Pixmap cache: pixmaps are generated in parallel (if images in RAM).

Version 1.5.0 (2021-05-23)

- Nuclei detection: bright nuclei can be dimmed ("uniformization") before thresholding.
- Can interactively quantify parts of the image in any selectable contour.
- More quantifiables: for quarters, thirds, and halfs of intensity distributions.
- Can quantify halos in finely adjusted Voronoi cells around nuclei.
- Quantifiable features of nuclear and region contours include relative inertia.
- Z-order added to facilitate manual contour selection.
- Nuclei editing: can split touching or overlapping nuclei with mouse.
- Nuclei editing: can center on a selected nuclear contour.
- Nuclei editing: can highlight externally specified nuclear contours.
- Regions editing: adjacent background tiles can be merged.
- Menu View: can display images with intermediate auto-leveling.
- Command line: more flexibility in parsing.
- No more spurious nuclear contours when thresholding baseline > 0.
- Contours of large perinuclei are no longer rhomboidal.
- Fixed a subtle bug that led to omission of minor C-shaped perinuclei.
- Fixed a subtle bug that gave rise to duplicate split orphan contours.
- Script-set channel descriptions propagate to the very bottom of data structures.
- Memory leaks due to Qt detachment have been removed.
- Scripting: the for-each-frame construct performs "refresh".
- Can read a sequence of images with initial time index > 0.
- Regions editing: background sampling faster by an O(magnitude).

- Joining of tracks and quant-files in Python module faster by an O(magnitude).
- Reorganized keyboard shortcuts.
- Nuclei ”incompleteness” status is now called just ”toggled”.
- Regions editing: removed subshifting of a putative background tile.
- Added an installer for the executable compiled with g++ under MinGW-w64.

Version 1.4.0 (2019-11-09)

- Input image files can be named according to more than one convention.
- Original images can be either stored in RAM or loaded on demand from disk.
- Image overlays can be optionally generated on demand to save RAM.
- Conversion to 8-bit may have non-integer resolution of camera bit depth.
- Conversion to 8-bit may globally subtract a given (background) intensity.
- Basic UNIX signal handling.
- Changed names of files in which toolboxen settings are stored.

Version 1.3.0 (2019-10-03)

- Perinuclei can avoid image background.
- Quantifications include intensity std dev, and 1st and 3rd quartile.
- Menu View shows channel descriptions, if available.
- Manual window has scalable font size.
- Background sampling is performed out of image masks.
- Script execution does not freeze GUI.
- Stop script execution upon failure (”panic()”).
- Checking latest available version upon startup is configurable.
- Nuclei matching for tracking is done using single-channel images.
- Log window displays all messages.
- Container recipe can be used by the latest Singularity.
- Can build source code distribution.
- Python module can load tracks partially.

Version 1.2.0 (2019-07-02)

- Can detect nuclei based on cytoplasmic staining.
- CVS-ize tesselation neighbors output file.
- Speed-up in-contour quantification ~10x.
- Add example standalone nuclei detection programs.

Version 1.1.0 (2019-06-02)

- Can create Voronoi cells around nuclei.
- Can propose background regions.
- Change format of region contours file.
- Label regions using numbers.
- Display log also in terminal.

Version 1.0.0 (2019-01-29)

- Include manual in PDF format.
- Promote to release version 1.0.0.