

# Installation and Usage instructions for the ImageJ plugin growRois

Thomas Braschler, Zahra Sadat Ghazali, June 2020

## Content

Content .....	1
Installation .....	1
Prerequisites .....	1
Download .....	1
Plugin installation .....	2
Mode of operation .....	3
Example 1: Cell segmentation using a the nuclei and a binary cell area mask .....	3
Image preparation .....	3
Plugin launch .....	6
Plugin options .....	7
Plugin output .....	8
Algorithm and Background .....	8
Cell segmentation algorithms .....	10
Open questions and issues .....	12
Bibliography .....	13
License .....	14

## Installation

### Prerequisites

This plugin requires a functional ImageJ[1] or Fiji[2] installation. We developed it on a 2.0.0-rc-44/1.50g ImageJ implementation as part of a Fiji installation on Mac OSX but it should work on other operating systems as well. While we hope this plugin is useful, we cannot guarantee any functionality or any fitness for any particular purpose.

### Download



**Figure 1.** Download calciumImaging\_.jar from <https://github.com/tbgitoo/calculusImaging>

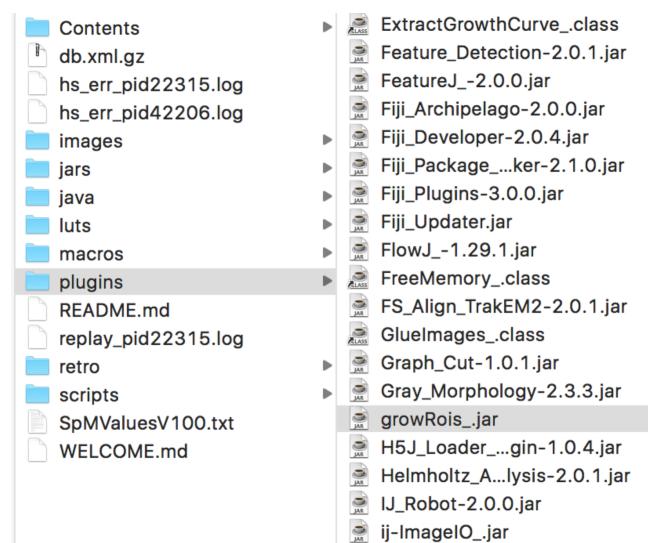
The first step in installation of the calciumImaging plugin is downloading the full .jar file <https://github.com/tbgitoo/growRois>. The “growRois\_.jar” file is the relevant binary (Figure 1), it can be downloaded navigating to growRois, then click on growRois\_.jar followed by clicking the “Download” button.

Alternatively, a direct link is:

[https://github.com/tbgitoo/growRois/blob/master/growRois\\_.jar](https://github.com/tbgitoo/growRois/blob/master/growRois_.jar)

## Plugin installation

The growRois plugin is installed like any other ImageJ plugin, namely by placing the downloaded “growRois\_.jar” file into the “plugins” folder of the ImageJ installation. The location of the plugins folder varies, depending both on operating systems and user choices during installation. Typical locations for ImageJ installations on Mac OSX, are /Applications/Fiji or /Applications/ImageJ. Windows locations could be C:\\Program Files(x86)\\ImageJ respectively Fiji, or also C:\\\\Program Files\\ImageJ. In these locations, there is the “plugins” folder where the “growRois\_.jar” file should be put.



**Figure 2.** Installation in the plugin folder (here, of Fiji)

If running, ImageJ needs to be restarted after placing the “growRois\_.jar” file in the plugins folder of your ImageJ/Fiji installation. The plugins are loaded into memory during startup of ImageJ/Fiji.

## Mode of operation

The plugin takes predefined ROI (regions of interest) as listed in ImageJ’s ROI Manager. In its simplest implementation, the ROI are each grown dilation. As a minimum requirement, the plugin therefore needs ROIs defined in the ROI manager and an open image (from which it will use the image processing capacities and also the overall width and height).

In addition to this basic behavior, the plugin disposes of the following non-exclusive options:

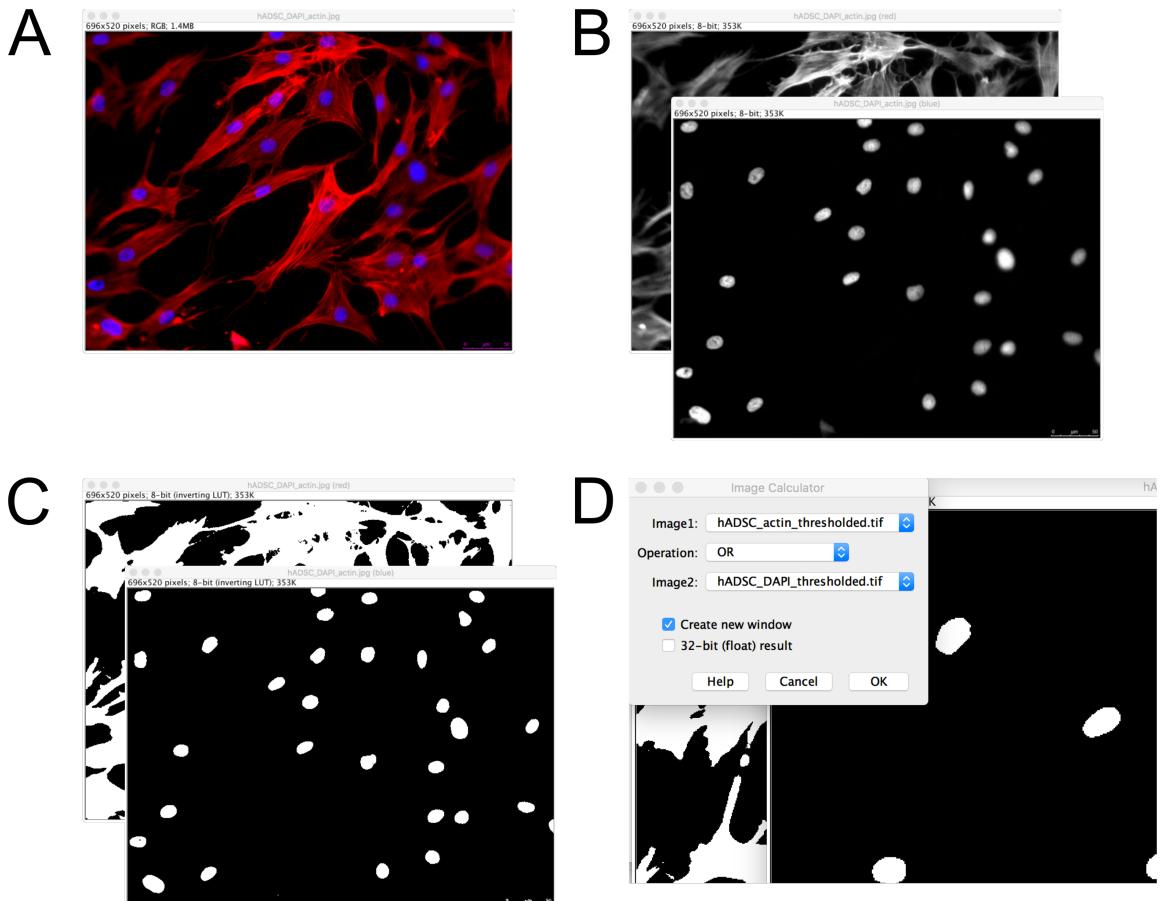
- Option 1:** Avoiding overlap between neighboring ROIs by preventing expansion into areas already occupied by other ROIs
- Option 2:** Using a binary mask of accessible (value 255) vs. inaccessible (value 0)
- Option 3:** Using of greyscale image of preferentials (low values) to non-preferential expansion areas (as in watershedding).

The options are available through the plugin dialog after launching the plugin.

## Example 1: Cell segmentation using a the nuclei and a binary cell area mask

In this example, we analyze a fluorescence image, from which the nuclei and total cell area are defined. This makes use of the options 1 (avoiding overlap) and 2 (binary mask), but not the grey-scale guide image.

### Image preparation



**Figure 3. Preparation for growRois.** A) Input image (here, fluorescence image of phalloidin- and dapi-labelled cells in culture, nuclei in blue, cytoskeleton in red). B) Separation of channels. C) Thresholding. D) Combination of the nuclei and cytoplasma channel to obtain an image where all cell pixels (nuclear or cytoplasmic) are bright (255 pixel value) and all non-cell pixels black (0 pixel value).

The growROI plugin is intended to work on an image where particles are already selected as ROI (regions of interest) in ImageJ's ROI Manager; using a binary mask requires the binary mask to be defined before plugin launch. The workflow leading to this starting situation is shown in Fig. 3. Here the starting image, shown in Fig. 3a, is fluorescence image of cells in culture, labeled with DAPI to define the nuclei, and with phalloidin to outline the cytoskeleton. To identify nuclear and cytoplasmic pixels, the image is split into its individual fluorescence colors (Fig. 3B). Both channels are thresholded to define binary (on-off) pixel values for nuclei and cytoplasm (Fig. 3C). In order to guide the ROI dilatation algorithm in the growROI plugin, we also produce an image where all cellular pixels are bright (255 pixel value) and all background pixels are black (0 pixel value). For this, we combine the nuclear and cytoplasmic thresholded pictures by an OR operation, as both nuclear and cytoplasmic pixels are part of the cells.

Of the various pictures produced, two are important for the subsequent analysis:

**Image#1:** The thresholded picture with the nuclei

**Image#2:** The combined picture with cellular pixels (nuclear and cytoplasmic) bright, background dark.

ImageJ has a bit of a complicated handling of black and white in pictures since it can also apply inverting look-up tables. This is particularly important for Image#2: the

background needs to have a pixel value of 0, whereas the cellular pixels must have a pixel value of 255. If in doubt, in ImageJ, the pixel values can be read by letting the mouse hover over the pixel and looking at status bar. This is shown in Figure 4.

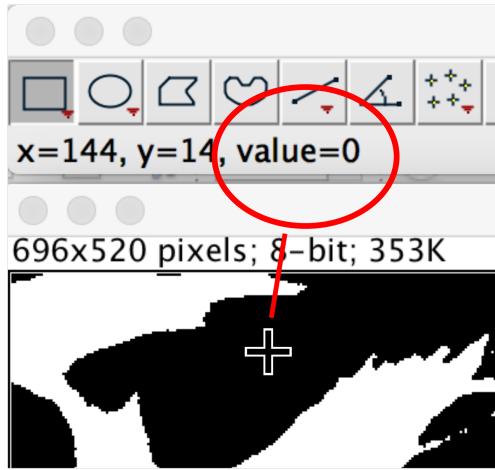
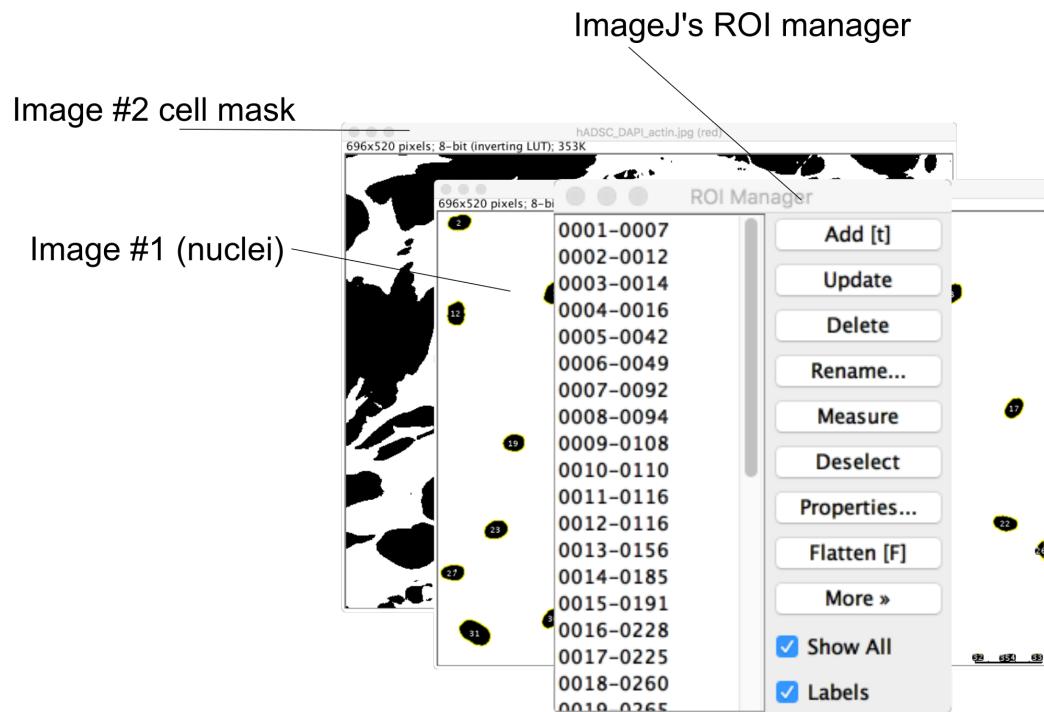


Figure 4. Evaluation of the pixel value (as opposed to color) by hovering with the mouse over a pixel. Here, the lookup table (LUT) is non-inverting, and so black corresponds to 0.

From the nuclear thresholded image, ImageJ's built-in particle analyzer is used to identify the nuclei (Fig. 4C, we also inverted the nuclear image since the particle analyzer excepts dark, rather than bright particles).

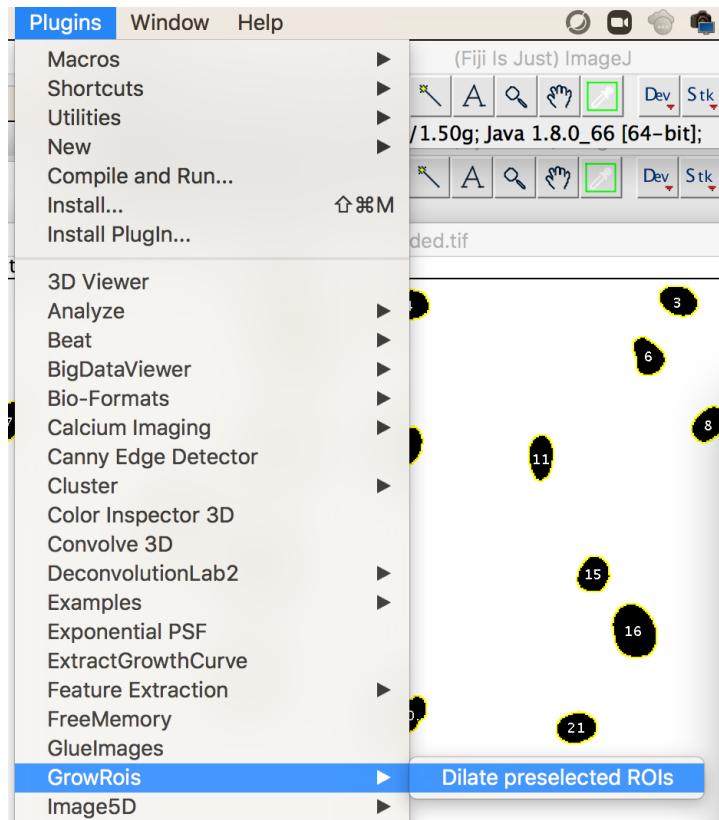
The growRoi plugin is intended to work on already pre-selected regions of interest ROIs. In the example, we use the nuclei as the seeds for defining the cell outlines. So the next step in the preparation is to run ImageJ's particle analyzer on the nuclear image (possibly using inversion between black and white to ensure selection of the nuclei rather than background; ImageJ's particle analyzer is launched via "Analyze->Analyze Particles..."). This sets the stage for the growRoi plugin: The nuclear image (image#1) with the nuclei defined by the ROIs in the RoiManager and the cell outline image (image#2) both loaded. This starting situation is shown in Fig. 5. Image #2 is technically optional, but it is goal of the growRoi plugin to use the cytoplasmic information to better define the outlines of cells.



**Figure 5.** Starting situation for the growRois plugin: Image #1 provides the seeds for the ROI growth process, selected as ROIs in the ROI manager. Image #2 provides the bounds for ROI growth (optional). In the typical use of cell segmentation, image #1 provides the nuclear signal, image #2 the total cell pixel signal (cytoplasmic + nuclear), and the ROI manager contains the information about the nuclei identified as particles.

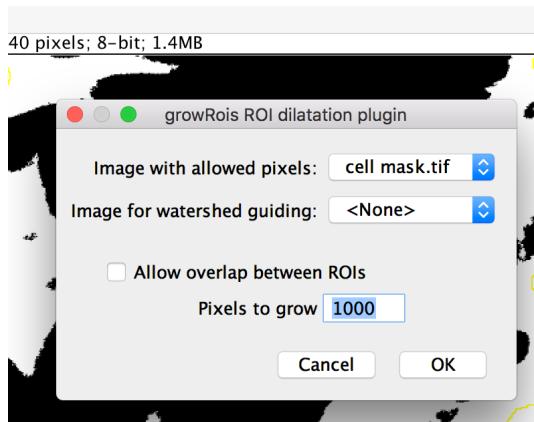
### Plugin launch

With images loaded and particles analyzed as shown in Fig. 5, the growRois plugin is launched. This is done by choosing “Plugins>GrowRois>Dilate preselected Rois” as shown in Fig. 5.



**Figure 6. Plugin launch.** GrowRois is launched via plugins > GrowRois> Dilate preselected ROIs as shown in the screenshot.

## Plugin options

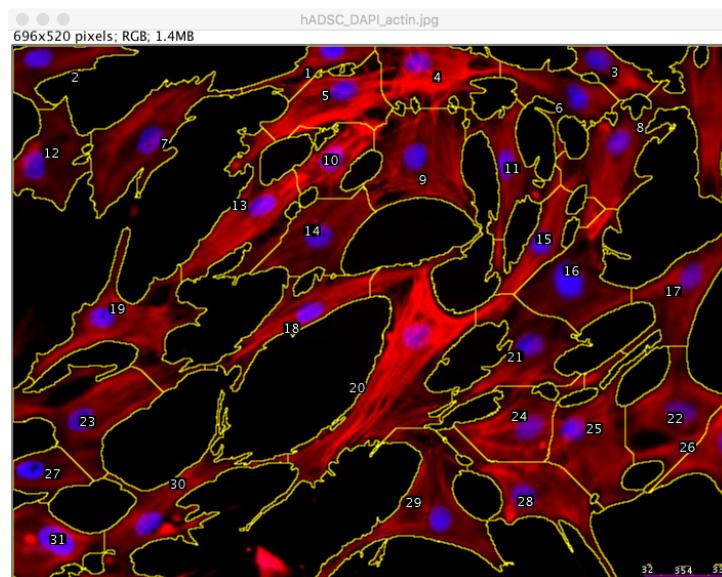


**Figure 7. Options for the growRois plugin.** In this example, we choose an image defining the allowed pixels for ROI dilatation from the dropdown menu. No image for greyscale watershed guiding is chose. Further, the "Allow overlap between ROIs" checkbox allows to select whether ROIs can grow over each other, while the "Pixels to grow" field defines how far the ROIs can grow maximally (if permitted in the allowed pixels mask).

After plugin launch, the option dialog shown in Fig. 7 is displayed. Here, the pertinent image for limiting ROI dilatation can be selected (in the example of cell segmentation, this would be image #2). One can further specify whether ROIs can spatially overlap, and also by how much the ROIs should be grown.

## Plugin output

The output of the plugin consists in redefined ROIs in the ROI Manager. These ROIs can be shown and used on any image. For visualization, the “Show all” checkbox in the ROI Manager, as shown in Fig. 5, is useful. It may be necessary to untick and tick it to force display of the ROI overlay. Fig. 8 shows the ROIs selected obtained by the workflow given in this example, on the original image for comparison with the actual cell morphology.

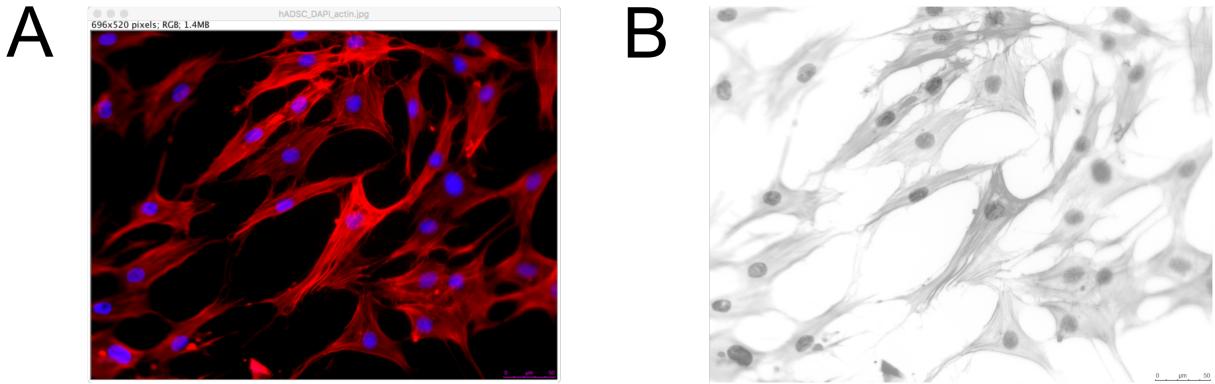


**Figure 8. Plugin output.** The plugin output consists in redefined ROIs in the ROI manager. They are initially shown overlaid over the image that was in the foreground when launching the plugin, but can be applied to any other image (here, the original image for comparison with the actual cell location).

## Example 2: Greyscale guidance

In this example, we analyze the same fluorescence image as in Example 1, reusing the mask on nuclear areas. In addition, we use the greyscale levels of the image as a watershed guide.

### Image preparation

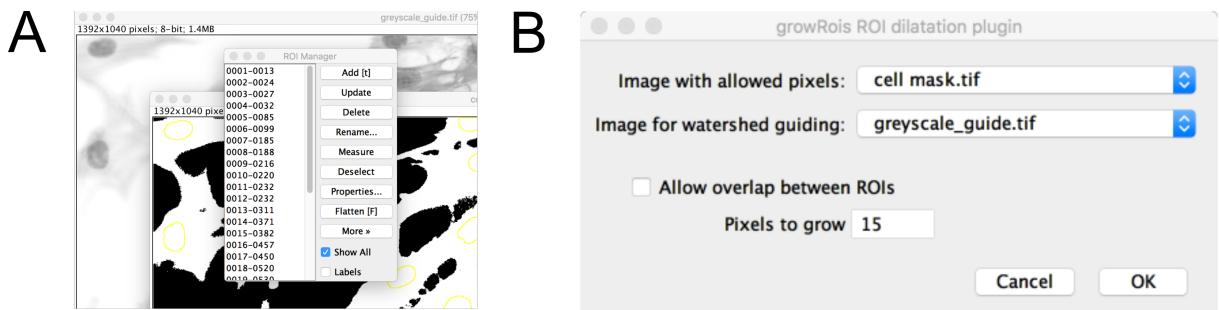


**Figure 9. Plugin output.** The plugin output consists in redefined ROIs in the ROI manager. They are initially shown overlaid over the image that was in the foreground when launching the plugin, but can be applied to any other image (here, the original image for comparison with the actual cell location).

Fig. 9 shows the preparation of the greyscale watershed guide mask. For this, the color image is converted to greyscale. The lower value pixels (darker in Fig. 9B) will be used preferentially for filling the ROIs. In addition to the greyscale guide, we will also make use of the binary mask for the total cell area defined in example 1 (Fig. 5, image #2 cell mask) and nuclear regions obtained by particle analysis in example 1 (Fig. 5, listed in the ROI Manager window).

### Plugin launch

With images loaded and particles analyzed as shown in Fig. 5, the growRois plugin is launched. This is done by choosing “Plugins>GrowRois>Dilate preselected Rois” as shown in Fig. 5.



**Figure 10. Plugin launch for greyscale watershedding.** A) The inputs are a greyscale watershedding guide, a maximal extension mask, and the nuclear ROIs selected in the ROI manager. B) Sample options, the cell mask is selected to find the allowed pixels, the greyscale image is selected for watershed guiding. The “Pixels to grow” parameter has a different meaning when the greyscale image is selected: it indicates the number of pixel growth step per greyscale level; high numbers (don’t exaggerate, it will take a very long time otherwise) indicate complete filling of each grey level, while with lower numbers, the segmentation will remain more geometrical like in Example 1.

Fig. 10A shows the conditions for launch for greyscale-assisted segmenting. In addition to the greyscale image prepared in Fig. 9, we use the cell mask and the nuclear regions from example 1. Fig. 10B shows the launch options: the cell mask defines the maximal extent of the ROIs after enlargement, the greyscale image the watershed guidance. The “Pixels to grow” parameter in this setting indicates how many pixels the ROIs can grow per greyscale level. Since there are 255 grey levels, the maximum growth is multiplied

by 255, and so generally, one will have to indicate smaller number than in Example 1 where there is only one distinct growth round.

## Plugin output

The output is similar to example 1, consisting in individual ROIs in the ROI manager. The use of the greyscale guide, particularly if many pixels are added per grey-level, tends to push the cell boundaries towards lower intensity areas.

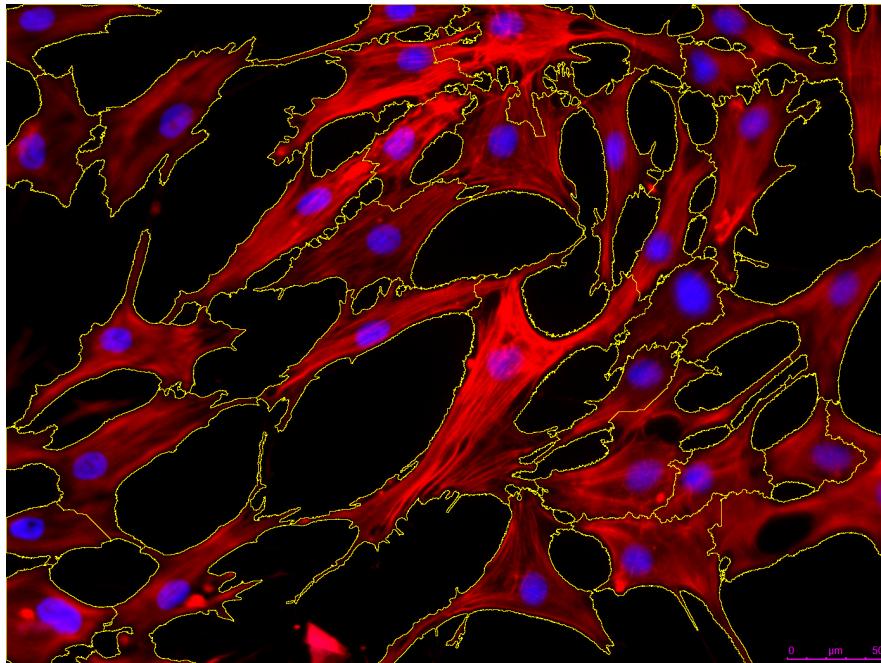


Figure 11. Plugin output (example 2). The plugin output consists in redefined ROIs in the ROI manager, as in example 1, and can be overlaid on any image (here, the original color image). Compared to example 1, the greyscale guiding results in a slight displacement of the boundaries towards lower

## Algorithm and Background

### Cell segmentation algorithms

There are many plugins and algorithms for particle segmentation in general and cell identification and segmentation in particular[3]–[7]. These algorithms and software implementations differ widely, in relation with the multitude of related, but ultimately different segmentation challenges (i.e. imaging modes, 2D vs. 3D, knowledge of cell boundaries, markers, nuclei and so forth).

For cell segmentation in particular, the technique of “seeded watershed” is very often employed[8]. Since the nuclei are a relatively easily identified unique feature of the cells, additionally often reflecting the rough shape of the cell[9], they are used as seed areas which are then progressively enlarged until reaching some neighbors, giving rise to a watershed type segmentation. The way the seeded areas are let to grow differs between different implementations. For example, in its basic installation, Fiji[2] includes a balloon-inflation based method[4] (i.e Plugins > Segmentation > Balloon), probably

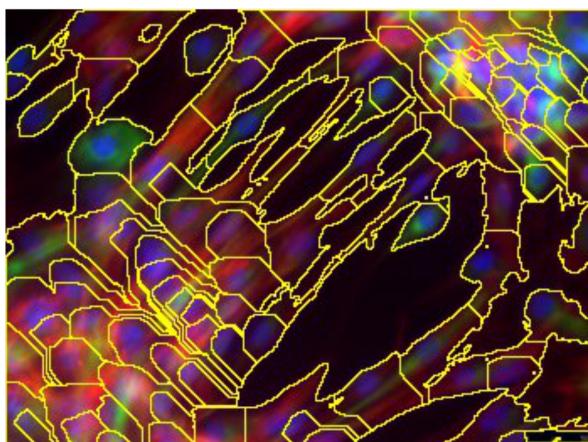
particularly well adapted to plant cells since it is based on emulated pressure-driven inflation[4]. Many variants of prototypical watershedding algorithms including seeded ones are implemented in Fiji[2] (for example, Process > Binary > Watershed), and the installable MorpholibJ library offers an extensive collection of watershed algorithms, including seeded ones[10].

The difference between a ballooning approach and watershedding lies in the focus of the objects: in ballooning[4], the focus lies on the objects and their physical properties during the inflation process; in watershedding, the primary unit is the pixel and like rain drops, they are ultimately assigned to the draining basin[10].

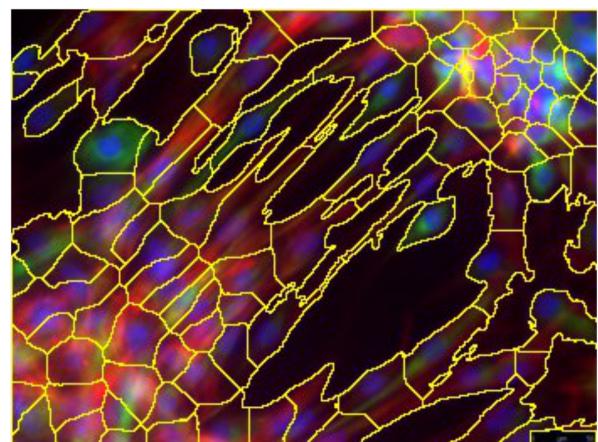
The algorithm here is intermediate in the sense that it grows individual regions, but uses neighborhood connectivity like a watershed algorithm to do so. The result is, at least visually, somewhat intermediate between ballooning and watershedding: like in watershedding, there is no “resistance” keeping cells from filling very intricate shapes, but due to the order of pixel assignment around growing regions in turn (rather than systematically running through the pixels), it is more unlikely to find small areas partitioned between two cells or also to find cells with very narrow necks extending into a large region. By the object-centered design of the algorithm, it is typically easier to implement object-level characteristics: for example, in the plugin, the growth of the seeded regions can be limited to for example define perinuclear regions. Pixel-centered watershedding algorithms such as implemented in MorpholibJ[10] more easily accommodate for pixel-level properties such as gray levels or gradients.

The difference between region-based (growRois plugin) and pixel-based watershedding (Morpholib) is illustrated in Fig. 12, in the context of an approach based on a binary mask only (as in Example 1). Within the chosen context of segmentation based on binary masks for both the nuclei (thresholded from the blue channel) and the total cell area (thresholded from grey-value averaged over all color channels), the pixel-wise growth shows an ordering artifact in the forms of a preference for  $45^\circ$ -lines (Fig. 12A, particularly lower left corner) unrelated to the image. This reflects the systematic bias for the  $45^\circ$  descending diagonal arising from systemic analysis of the pixels per round of area expansion. This particular artifact is absent in region-wise dilation (Fig. 12B). With greyscale guiding, the magnitude of the effect decreases but can still be seen in places.

A) Pixel-wise (morpholibJ)



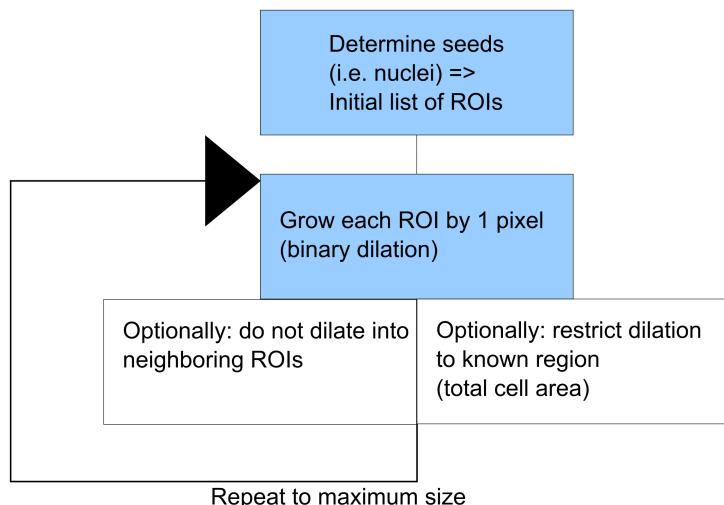
B) Region-wise (growRois)



**Figure 12. Comparison of segmentation using MorpholibJ[10] (pixel-wise watershedding) and the growRois plugin here (region-wise watershedding).** The results are globally similar, but the pixel-wise segmentation leads sometimes to characteristic 45°-preferential separation lines for the pixel-wise approach (particularly, compare the lower left regions of the images). The segmentation is carried out here on binary mask only (nuclei thresholded from blue channel, overall outline thresholded from overall grey level) including for MorpholibJ. The case is admittedly a bit selected to outline the orientation-bias inherent in a systematic pixel-wise evaluation, as general rule, the effect is more subtle but existent.

A further practical difference concerns the natural output of the algorithms. The region-based approach outputs regions of interest in the form of ImageJ ROI-Manager selections. This is what one would get naturally from a ballooning-type of algorithm[4], as opposed to label-colored images in watershedding[10]. The two are fundamentally equivalent, although conversion tends to be bit cumbersome (ROIs -> label image: this can be done by the ROIMap plugin from LOCI; label image -> ROIs. Thresholding for each level, conversion to binary mask, selection from mask).

Fig. 12 outlines the algorithm used.



**Figure 13. Basic algorithm**

## Open questions and issues

At present, the growROIs plugin is fairly rudimentary as it needs manual preparation of the images, and does not directly handle colored images or analysis based on color-similarity. The aim of making the code publicly available is also that if pieces of ROI-centered operations can be re-used in related applications, this should be done without hesitation.

The plugin works well with scattered and semi-dense cells (as shown in the examples). At strong confluence, finding the cell boundaries is difficult including for a human observer without taking care to use special labeling and so if the cells are too dense, it is probably not advisable to use this plugin. If there is strong cell overlap (as in over-confluent cell culture imaged 2D), segmentation becomes very difficult, and this plugin

does not solve this issue. The greyscale guiding permits to some extend to take into account the internal structures of cells and thus to better account for the internal structure. As there are other artifacts arising through greyscale watershedding (such as extensive growth of one region through a narrow high-intensity access with ensuing improper erosion of a neighboring area), it is sometimes a matter of judgement to which extent binary masking and watershedding are used. This can be adjusted with the “Pixels to grow” in the plugin dialog when a greyscale guide is selected. This comes at the cost of some arbitrariness, but this is already inherent in thresholding and nuclei particle analysis upstream of the plugin.

Another limitation inherent in the present implementation is the use of polygons to define the regions of interest. As implemented, the ROIs cannot have holes and so will be solid objects only, the plugin wouldn’t work on annual objects for example.

If you are looking for ways to segment cells or other objects in various settings, it is certainly worth to also take a look at the much more extensive library MorpholibJ[10], and at other built-in methods such as the Ballooning[4] mentioned above. We wrote this plugin for the particular application of determination of cell orientation and elongation, where the orientation bias shown in the extreme case (Fig. 12) is an issue, and make it available in the hope that it can be suitable in other applications as well.

## Bibliography

- [1] C. A. Schneider, W. S. Rasband, and K. W. Eliceiri, “NIH Image to ImageJ: 25 years of image analysis,” *Nat Methods*, vol. 9, no. 7, pp. 671–675, Jul. 2012, doi: 10.1038/nmeth.2089.
- [2] J. Schindelin *et al.*, “Fiji: an open-source platform for biological-image analysis,” *Nat Methods*, vol. 9, no. 7, pp. 676–682, Jul. 2012, doi: 10.1038/nmeth.2019.
- [3] O. Sarrafzadeh and A. M. Dehnavi, “Nucleus and cytoplasm segmentation in microscopic images using K-means clustering and region growing,” *Adv Biomed Res*, vol. 4, Aug. 2015, doi: 10.4103/2277-9175.163998.
- [4] F. Federici, L. Dupuy, L. Laplaze, M. Heisler, and J. Haseloff, “Integrated genetic and computation methods for in planta cytometry,” *Nature Methods*, vol. 9, no. 5, pp. 483–485, May 2012, doi: 10.1038/nmeth.1940.
- [5] C. Wählby, J. Lindblad, M. Vondrus, E. Bengtsson, and L. Björkesten, “Algorithms for Cytoplasm Segmentation of Fluorescence Labelled Cells,” *Anal Cell Pathol*, vol. 24, no. 2–3, pp. 101–111, 2002, doi: 10.1155/2002/821782.
- [6] Y. al-kofahi, A. Zaltsman, R. Graves, W. Marshall, and M. Rusu, “A deep learning-based algorithm for 2-D cell segmentation in microscopy images,” *BMC Bioinformatics*, vol. 19, Dec. 2018, doi: 10.1186/s12859-018-2375-z.
- [7] H. Irshad, A. Veillard, L. Roux, and D. Racoceanu, “Methods for Nuclei Detection, Segmentation, and Classification in Digital Histopathology: A Review—Current Status and Future Potential,” *IEEE Reviews in Biomedical Engineering*, vol. 7, pp. 97–114, 2014, doi: 10.1109/RBME.2013.2295804.
- [8] L. P. Coelho, E. Glory-Afshar, J. Kangas, S. Quinn, A. Shariff, and R. Murphy, “Principles of Bioimage Informatics: Focus on Machine Learning of Cell Patterns,” in *Lecture Notes Comput. Sci.*, vol. 6004, 2010, pp. 8–18.
- [9] M. Versaevel, T. Grevesse, and S. Gabriele, “Spatial coordination between cell and

nuclear shape within micropatterned endothelial cells," *Nature Communications*, vol. 3, no. 1, p. 671, Feb. 2012, doi: 10.1038/ncomms1668.

[10] D. Legland, I. Arganda-Carreras, and P. Andrey, "MorphoLibJ: integrated library and plugins for mathematical morphology with ImageJ," *Bioinformatics*, vol. 32, no. 22, pp. 3532–3534, Nov. 2016, doi: 10.1093/bioinformatics/btw413.

## License

GNU General Public License. We adhere to the usual license text :

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, see <<http://www.gnu.org/licenses/>>.