Getting Started with  
bigImageLabeler

# About the App

The bigImageLabeler app was designed to facilitate manual, semi-automatic, and automatic labeling of single images too large to fit in memory; these may range up to several gigabytes. Navigation tools are provided to load high-resolution imagery from sub-regions following a selector in an “overview thumbnail.”

The user can annotate the image at either the low-resolution overview level or at the high-resolution sub-image level; annotations are “synchronized” between the two views. Annotations may be either “freehand,” to capture the contours of regions of interest (ROIs), or rectangular, to capture bounding boxes.

Additionally, the user can operate on one or a few small sub-images to tune a labeling function, then subsequently apply the tuned algorithm to all sub-images across the entirety of the big image.

Once the user has completed the annotation (labeling) phase, the bigImageLabeler will automatically extract training images into appropriately (and automatically) labeled folders.

Sessions are saved (automatically or manually) and can be re-loaded.

# Credits

bigImageLabeler was built primarily by Brett Shoelson, PhD ([brett.shoelson@mathworks.com](mailto:brett.shoelson@mathworks.com)). The author gratefully acknowledges the contributions and suggestions of Joyeeta Mukherjee, PhD; Ashish Uthama; Jeff Gruneich, PhD; and Sean de Wolski. (All were employees of MathWorks at the time of their contributions.) All code is Copyright 2018 The MathWorks, Inc.

# Launching the bigImageLabeler

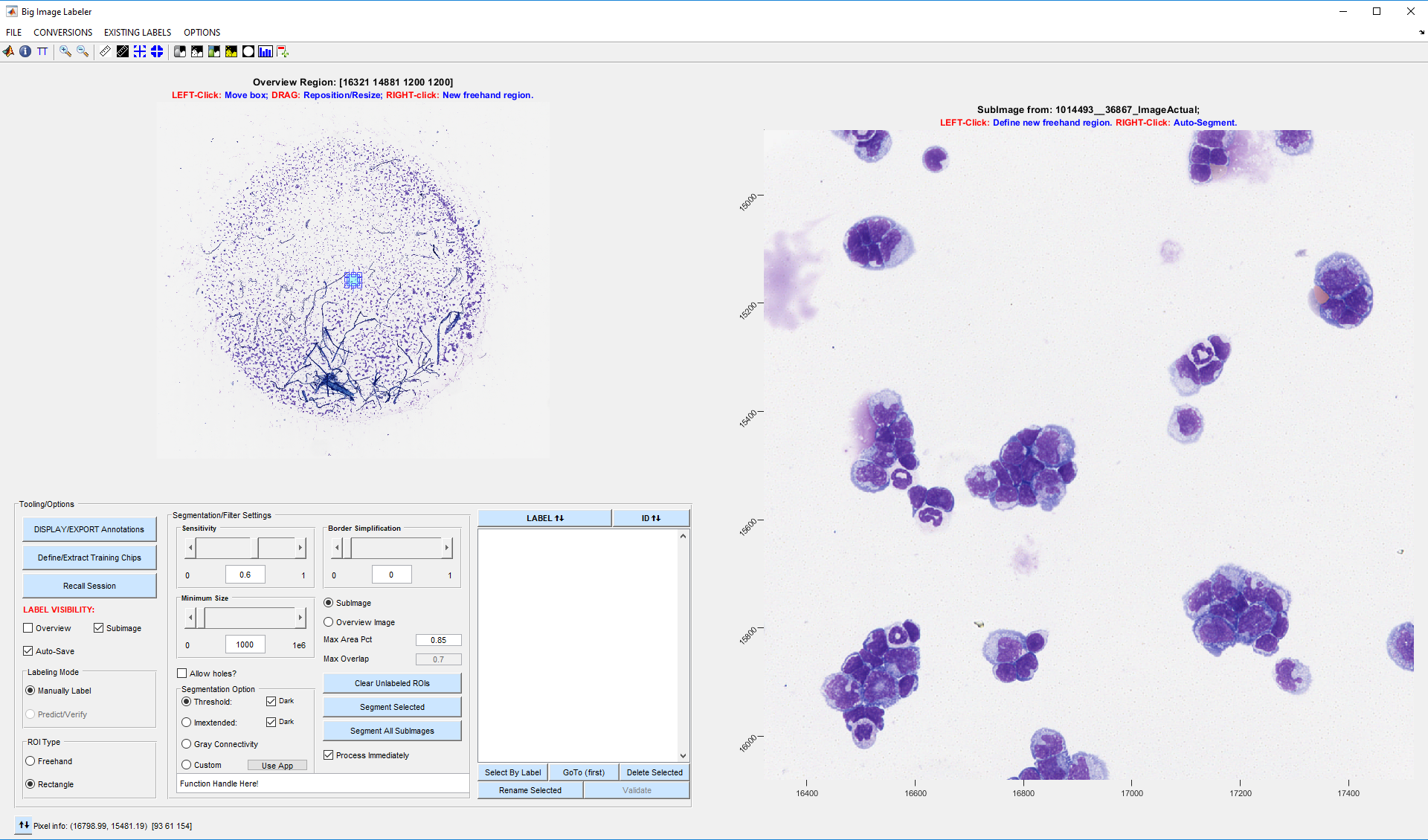
To launch bigImageLabeler, type at the MATLAB Command Line:

bigImageLabeler(nameOfImage)

% for example:

% bigImageLabeler('1014493\_\_36867\_ImageActual.svs');

The app will automatically generate the overview thumbnail, and will display a center-cropped, *whole-tile*, full-resolution image approximately 2k x 2k:



# Navigation

Note that you can “navigate” around the whole image by clicking on the overview image or by dragging the “overview selector box.” The box may also be resized to change the size of the high-resolution sub-image displayed on the right. (The maximum size of the sub-image is capped at 10k x 10k, to maintain an acceptable level of performance.)

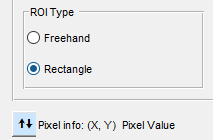
  
**(CLICK TO SHOW NAVIGATION VIDEO)**

# Labeling and Naming Regions of Interest

There are three modes of labeling:

## Manual Labeling

Specify “Freehand” or “Rectangle” on the ROI Type panel (lower left):

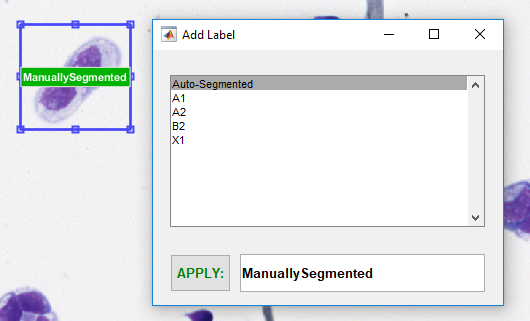


Next, left-click on the sub-image, or right-click on the overview image to add manual ROIs. Note that click behavior is summarized in the image titles:



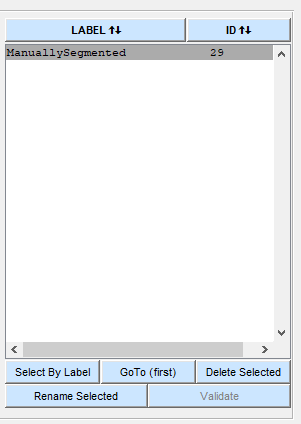


When you manually draw an ROI (freehand or rectangular), the region is automatically named “Manually Segmented.” Clicking on the label opens a dialog box for renaming the ROI:



Type whatever name you want to give the region or select a label from the listbox. Then hit “APPLY.” (Labels are saved across multiple sessions.)

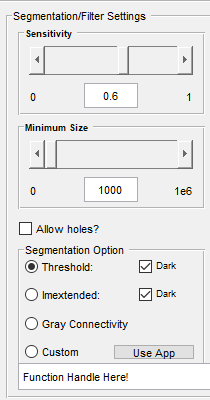
Note that all labels for a labeling session are stored in the ROI Listbox:



Each label will be assigned a unique ID. You may sort (ascending or descending) by label name or by ID by clicking on the buttons above the columns. You may also select a single ROI or multiple ROIs from the listbox or using the “Select by Label” button. Once you have made a selection, you may delete or rename selected labels.

## Semi-automated Labeling

The bigImageLabeler app has several built-in algorithms for semi-automatically labeling ROIs. You may specify the “Sensitivity” of the algorithm and the “Minimum Size” (in pixels) for acceptable ROIs. The Sensitivity option works differently for the “Threshold,” “Imextended,” and “Gray Connectivity” algorithms. Generally, increasing the Sensitivity parameter increase the likelihood of detecting ROIs. The Minimum Size parameter facilitates ignoring ROIs smaller than a user-specified number of pixels.



**Threshold** implements [imbinarization](https://www.mathworks.com/help/images/ref/imbinarize.html) using an automatically selected threshold. (The Sensitivity parameter adjusts that threshold.)

**Imextended** implements [imextendedmax()](https://www.mathworks.com/help/images/ref/imextendedmax.html) or [imextendedmin()](https://www.mathworks.com/help/images/ref/imextendedmin.html) algorithms.

*Both of these functions are applied to the sub-image currently displayed in the sub-image axes.*

Trigger the algorithm by either right-clicking on the sub-image, re-selecting the radio button, or pressing the “Segment Selected” pushbutton.

(Note that you change the polarity of the algorithm using checkboxes to detect dark objects against a lighter background, or lighter objects against a darker background.)

The **Gray Connectivity** algorithm allows you to click on a particular ROI to interactively label it.



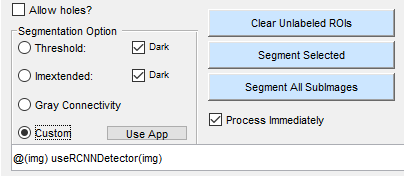
## About the Labels

Note that all automatically labeled regions are named “InitialAutoSegmentation.” ***ROIs of this label are automatically discarded if you run multiple algorithms on the same sub-image!*** (This allows you to try different algorithms and to change parameters to “tune” the segmentation without getting many duplicate detections.) When you move to another region, the current “InitialAutoSegmentation” labels are re-coded as “Auto-Segmented.” (Alternatively, you may click on an individual label, or select one or more ROIs from the ROI Listbox and rename them *en masse*.)

**IMPORTANT:** Please limit region names to 28 characters, and do not use spaces in the names of regions. (Underscores are permitted.)

## Using the Custom function

The “Custom” radio button and the edit box below it (“Function Handle Here!”) allows you to call a custom algorithm (like a pre-trained neural network, for instance) and apply it to the current sub-image. Before you select the “Custom” radio button, enter a [function handle](https://www.mathworks.com/help/matlab/matlab_prog/creating-a-function-handle.html) to call the custom algorithm. For example:



where function useRCNNDetector() is a custom-written algorithm on the MATLAB path:

function outputStruct = useRCNNDetector(img)

persistent detector

if isempty(detector)

detector = load('newdetector.mat');

detector = detector.detector;

end

tic;

[outputStruct.Boxes,outputStruct.Scores,outputStruct.Labels] = ...

detector.detect(img);

toc

end

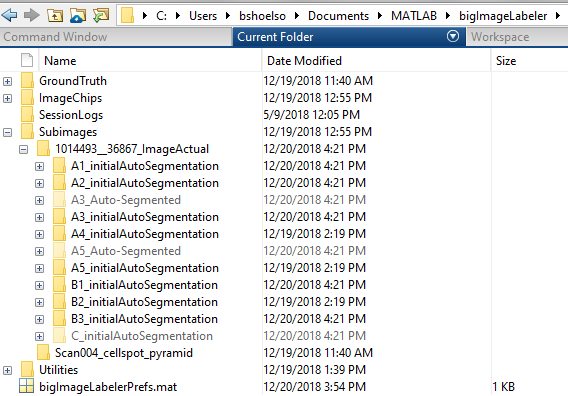
# Applying an Algorithm to the Entire Image

Once you have tuned your detection algorithm or implemented and selected a custom detection algorithm, you can apply it across all sub-images in the image. Simply press the “Segment All SubImages” pushbutton and the bigImageLabeler will do the rest! (Note that this may take a little while, depending on the size of the big image and the complexity of the detection algorithm.)

ROIs labeled using the whole-image labeling approach include the word “Auto” in their names to differentiate them from manually drawn or validated ROIs.

# Extracting Training Images

After you have generated all the labels you want to extract from the overall image, use the “Define/Extract Training Chips” button to automatically generate a collection of rectangular sub-images suitable for training machine learning algorithms.



# Additional Features/Capabilities

There are many additional features already built into the bigImageLabeler that are not described in this “Getting Started” document. Please explore by clicking around, but be aware that you might encounter bugs here and there. Please note and report any bugs (with reproduction steps, if possible) to me so that I may fix them!

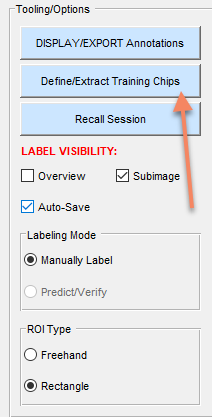
# Comments/Suggestions

This is a work in progress. Please comment and critique liberally. Note what you like about the app and what you don’t like. And note whatever enhancements you think of that would make the onerous labeling task easier and less tedious.

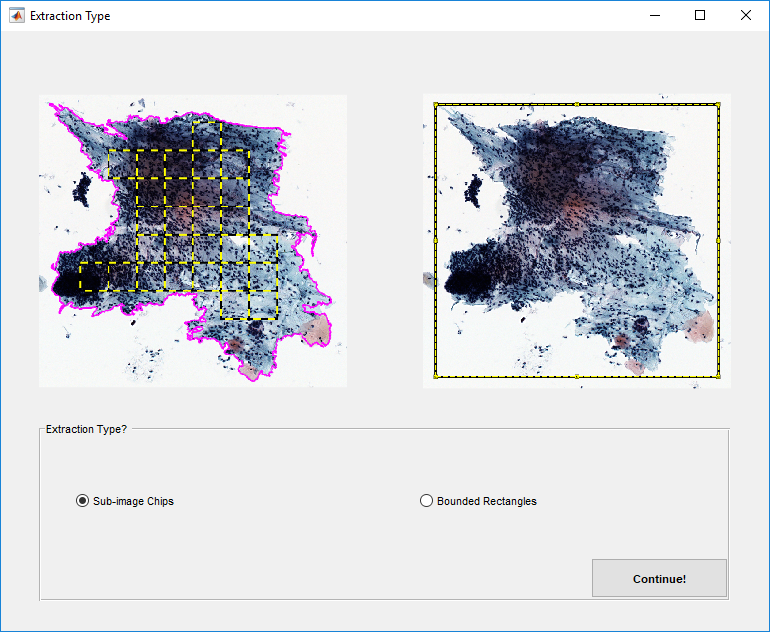
Thank you, and happy labeling!

Brett Shoelson

brett.shoelson@mathworks.com



You will be prompted to select “Sub-image Chips” or “Bounded Rectangles” in a pop-up window:



**Sub-image Chips** are auto-generated rectangular regions *entirely within the bounds of the ROI*. The user will be prompted to select a size (typically, 100 x 100) for these chips. This is useful for characterizing the texture of a particular type of ROI.

**Bounded Rectangles** are defined by the bounding box of the ROI. (Even for freehand regions.)

Each extracted image will be automatically stored in a folder specific to the exact label name associated with the ROI.