Morphometrics v1.1 User Manual

Tristan Ursell, © 2012-2017.

Table of Contents

The Basic Idea

Morphometrics is a set of MATLAB© scripts, wrapped into a graphical user interface (GUI), that are designed to analyze phase and fluorescence images of cells or other objects to provide quantitative, reproducible segmentations, contours, and meshes under a variety of conditions. This includes, but is not limited to a multitude of different types of cellular images, colony counting, or any other shape analysis of a high contrast image. Morphometrics makes absolutely no assumptions about the shape of the objects being analyzed, which is a doubled-edged sword; it can potentially identify and analyze objects of any shape making it a versatile program, but it will also, at times, combine or split objects that are, to an eye with prior knowledge, obviously supposed to be separate or contiguous, respectively.

The program has a graphical user interface that allows for batch processing and custom configuration. The output data can be used to perform spatio-temporal quantification of: morphology (shape) and size, peripheral and internal fluorescence, segmentation, position tracking, and basic phylogeny (i.e. object lineage through time). Please read this help file in its entirety – it explains what you need to know to use this program. Every button, checkbox, text field and menu option is discussed in this help file. Throughout this help file, the use of the word 'cell' should be taken to mean any object that provides contrast and/or an edge in an image to which a contiguous group of pixels can be assigned.

Segmentation is the act of using an algorithm to examine an image and decide which group of connected pixels in an image are associated with an object of interest, subject to a set of user-defined constraints. A contour is a topologically circular, ordered group of points that defines the boundary between the inside and the outside of an object. A mesh is a sub-division of the internal space of an object that allows for finer quantification and characterization. A Region of Interest (ROI) is a sub-region of an image that is used for various processing purposes.

Throughout this help file, all text in `Courier font` is either MATLAB code that can be entered into the command line, file names, file motifs, or directory names, and all text in Garamond font are either menu items, GUI text, or parameter inputs.

Requirements

This software requires MATLAB© software (R2012b or newer) with the *Image Processing Toolbox*. To determine if your version has this toolbox type

```
license('test','image_toolbox')
```

in the command line and the result should be `1`. Morphometrics automatically checks for the proper toolboxes, and alerts the user if a toolbox is missing. The included analysis scripts `intensitmetrics` additionally requires the *Mapping Toolbox.* To determine if your version has this toolbox type

```
license('test','map_toolbox')
```

in the command line and the result should be `1`.

A command line version of many of these scripts is also included with this software download. A later section addresses how to use them.

Recommendations

The results of this program depend on the quality of the input images. Whichever type of image you use, internal or peripheral fluorescence or phase contrast, ensure that the cells are in focus and that the exposure time is not too short (resulting in decreased signal-to-noise), and not too long (so as to saturate the image). Additionally, with phase contrast images, ensure that the phase contrast optics are properly aligned to maximize contrast. In some cases, the results of this program can be improved by applying a noise reduction filter (e.g. median filter, relative noise filter, or Gaussian blur filter) prior to use in Morphometrics; however Morphometrics already employs a relative noise reduction filter[1], and has options to use a Gaussian blur filter.

For more information about setting up your microscope, see Nikon MicroscopyU:

http://www.microscopyu.com/articles/phasecontrast/

[1] http://www.mathworks.com/matlabcentral/fileexchange/35556-image-noise-reduction-by-local-statistics

http://www.microscopyu.com/tutorials/java/kohler/

Morphometrics only accepts image stacks in an uncompressed, single channel TIF (`*.tif` or `*.tiff`) format. A stack is a group of TIF (uncompressed) image files wrapped into a single file. Many image capture programs output single files for each exposure -- you will need to combine them into a stack -- see the next section on Image Preparation. Some programs produce uncompressed image stacks with other metadata and file extensions like `*.stk`.

Reliably detectable feature sizes are at least an order of magnitude larger than the pixel size in the image, hence ensure that, in addition to good contrast, the dimensions of the object to be segmented and tracked are approximately an order of magnitude (or more) larger than the native pixel size of the image.

Examples

The software comes with examples of microbial segmentation, using phase contrast images of both sparse and dense phase microscopy, and using peripheral fluorescence. Use this help file to learn how to load those example files and their parameter sets, and to examine their outputs.

Software Updates

Morphometrics will automatically check and alert you when a new version of the software is available online. If you wish to turn this feature off, select **Do not Update on Startup** under the **Options** menu. You can also manually check for software updates by selecting **Check for Software Update** under the **Options** menu. Download the new software and unzip the files into a single directory. Removal of older copies of Morphometrics is recommended to avoid software conflicts.

Morphometrics is a software package that requires many sub-files to be accessible and in the same directory. The easiest way to ensure that MATLAB can access all those files is to add the Morphometrics directory to the MATLAB path. This can done by selecting **Add Morphometrics to MATLAB Path** from the **Options** menu. Alternatively, one can 'Add with Subfolders...' to the MATLAB path using the command `pathtool`, or you can also directly add to the path by typing

```
addpath 'C:/somefolder/somefolder2/morphometrics'
```

in the MATLAB command-line.

Image Preparation

Most microscopy software produces individual images associated with a particular channel. For instance, a simple time lapse experiment that takes images in bright-field, green fluorescence, and red fluorescence, might generate a series of images with names that look like:

```
name_BF_0001.tif
name_GFP_0001.tif
name_RFP_0001.tif
```

```
name_BF_0002.tif
name_GFP_0002.tif
name_RFP_0002.tif

...
```

where `name` is the string appended to each image from your imaging software, `BF`, `GFP`, and `RFP` refer to the channel, and the number refers to the order in which they were taken. Morphometrics uses TIF stacks, which are an ordered combination of all the images associated with a single channel combined into a single file, usually with the extension `*.tif`, `*.tiff`, or `*.stk`.

If you are starting from individual images, you will need to combine those images into a stack. Go to the **Data Prep** menu and select **Combine Images into Stack**; this script can be used as a stand-alone program by typing `stack_maker` in the MATLAB command-line. This feature works in two different ways. If you want to create TIF stacks from individual images in a single folder, select the directory, enter a motif that all the images have in common (in the example above a good choice would be `name_*.tif`), and hit **Convert Images to Stacks**. The program will tell you how many images it found that match the motif, and it will create an individual image stack for each channel; you also have the option to delete the original images as the stack is created. The program will also give an estimate of the mean time between frames, which is generally more accurate than the time assigned by the original imaging software. Stacks can also be created in batch; if for instance you collected images at multiple positions you might have folders with names like:

```
Pos1
Pos2
Pos3
...
```

each containing images with names like those above. Selecting **use sub-directories** will calculate the number of folders in the parent directory that meet the sub-directory motif (in this example `Pos*` would be a good sub-directory motif), and selecting **Convert Images to Stacks** will create individual stacks for each channel from each folder, and will append the folder name to the stack name for each channel. For instance, for the file and folder names above, this would produce image stacks with names:

```
Pos1_name_BF_##s.tif
Pos1_name_GFP_##s.tif
Pos1_name_RFP_##s.tif
Pos2_name_BF_##s.tif
Pos2_name_GFP_##s.tif
Pos2_name_RFP_##s.tif
Pos3_name_BF_##s.tif
Pos3_name_GFP_##s.tif
Pos3_name_RFP_##s.tif
```

where the `##s` is the inter-frame time in seconds appended to the file name. Depending on the image type and/or contrast, it may be advantageous to invert the contrast. The output stack can be inverted by selecting **invert images**.

In some cases, time-lapse image data may show significant drift, that is, a translation of the image between frames greater than ~10% the size of the object being segmented. These translations make it very difficult to perform object phylogeny / tracking over time, and hence Morphometrics includes a cross-correlation script to correct for drift. Additionally, optical shifts (translations) between channels should be corrected before alignment and subsequent processing with Morphometrics – this can be done using ImageJ / FIJI. Select To begin, locate the files you want to align, which could be multiple channels and hence multiple stacks that need simultaneous alignment. **Align Stack Images** under the **Data Prep** menu; this script can be used as a stand-alone program by typing `stack_align` in the MATLAB command-line. Enter the motif that all stacks to be aligned share -- from the example above, if Position 1 needed alignment, then the motif could be 'Pos1*.tif'. Then enter the channels you wish to have simultaneously aligned separated by commas (up to four at once). In the example above that could be `_BF`, `_GFP`, and `_RFP`; each of these channels will then become available under the **Master** and **Slave** check boxes on the right. The **Master** channel is used to perform the alignment, and all the **Slave** channels will be translated according to the results from the **Master** channel. In the **Exclude motifs** text box, enter the file extensions of any files that you do not want considered for alignment -- the default settings are `.fig`, `.mat`, and `.txt` which should be sufficient. Finally, the alignment algorithm needs to be told the maximum allowable translation between frames in pixels; keep in mind that the time the alignment operation takes grows quadratically with this number. Selecting **adaptive translation** allows the algorithm to search beyond the **Max image translation** when no suitable translation is found within that range, however this may take longer to compute.

Select **Align Stacks**, and the script will prompt you to pick a directory where the stacks to be aligned are located. The script will alert you if it finds too many stacks that match the **Master** channel and image motifs, or if no files match the criteria. The images can be aligned so that a user-selected point is the center of the newly aligned image stack, by selecting **yes** when prompted if you want to create a center point. The alignment is performed with a user-selected region of interest (ROI) of the image. The key to producing stable, reproducible alignments is to select a ROI that contains features to track over time that remain in the tracking region throughout the stack. Alignment may take a few minutes depending on the stack size. After alignment has finished, the script will create a new, aligned TIF image stack with the text `_aligned` appended to the name of the original stack.

Finally, in many cases, large portions of the image do not contain usable data or are simply void of objects. The computational time required for segmentation grows with image size, and thus much time can be saved by cropping an image stack down to a minimal Region of Interest (ROI). If multiple channels are involved, the same exact cropping should be propagated to all channels.

Settings

Image Attributes

Morphometrics accepts three classes of images: (i) phase contrast images produced using bright-field illumination where the objects of interest are dark against a lighter background, (ii) internal fluorescence images where the internal volume of objects of interest is light against a dark background, and (iii) peripheral fluorescence where the surface of objects of interest is light against a dark background and dark interior. The first setting in **Image Attributes** is to select between these options.

The **Phase**, **Fluorescence (internal)**, and **Fluorescence (peripheral)** check boxes are mutually exclusive. **Phase** indicates that the input images are gray scale phase contrast images where objects of interest are dark, as is often used in microbial microscopy. **Fluorescence (internal)** indicates that the input images are gray scale images that fluoresce from the interior of the cell (e.g. cytoplasmic dye). **Fluorescence (periphery)** indicates that the input images are gray scale fluorescence images that outline the periphery of the object being analyzed (e.g. a membrane dye). In some cases, due to either cell size or poor label specificity, peripheral fluorescence labeling may be better analyzed using **Fluorescence (internal)** or vice versa. These settings can be employed with any intensity image that bares the same intensity features, e.g. having its interior filled bright or filled dark (invert the image), or having its periphery bright or dark (invert the image).

Morphometrics can be used to track cells through time. The **Images are independent** check box tells the program not to attempt to match cells from one frame to the next, or alternatively said, every identified cell in every frame will have a unique cell ID.

The **Cells are in proximity** check box is selected when cells are close enough that their light fields overlap. Functionally, this tells the program to perform two rounds to segmentation: a primary round to identify regions that are clearly not background, but may be composed of adjacent cells; and a secondary round of segmentation that examines only the previously segmented regions to perform further, finer segmentation, to split nearby cells. Often, even when cells are close to each other, a single round of segmentation is sufficient, hence only use this option if deemed necessary. Checking this box enables the entry field titled **Cut distance** which sets the sensitivity of the secondary segmentation; lower values tend to re-segment objects more finely while values greater than the size of the objects tend to leave the original segmentation.

The **Reject false positive** check box selects whether the program applies intensity constraints to the segmented regions to determine if they are false positives, relative to the type of input image. Generally, this should be left on, however there are circumstances where due to irregular intensity patterns in or around the cell, it may be advantageous to keep all segmented regions, with the possibility of filtering them in post-processing. False positive rejection is performed by comparing the mean intensity along the periphery of a segmented region with the mean intensity of the region interior. For most phase and periphery fluorescence images, the periphery should be lighter than the interior, whereas for most interior fluorescence images, the interior should be brighter than the periphery. If Morphometrics senses that these ratios are beyond a user-defined range (see section on **Advanced Options**), they will be rejected.

The **Seed contours** check box segments the first image, fit contours to those objects, and then uses those contours as seeds for each subsequent frame. This necessarily means that the number of segmented objects is set by the first frame, and also means that the lineage parameters (**Frame overlap** and **Fractional overlap** in the **Process Parameters** pane) are not needed as the cell IDs are set by their labels in the first frame. This method is significantly faster than the standard algorithm since segmentation only takes place on the first frame and no phylogeny is required. This method works well when the change in shape and position of the cells between frames is relatively small (e.g. see the included example phase data). This method will not work if there is significant changes in shape or position between frames or if the frames are independent.

The **Track objects manually** check box allows the user to give unique cell IDs to segmented objects by hand; these hand selected cell IDs need not be consecutive, nor span the entire stack. This can be very useful when the image contains a few cells that change shape or position significantly over time. After segmentation, the user will be prompted in a separate window to begin the manual tracking. All operations for manual tracking are controlled by mouse clicks and arrow key strokes. Directions for how to perform this tracking are presented in the tracking window. Manual tracking can be time consuming, thus it is advised only when the number of cells and/or frames is reasonable.

The **Exclude edge objects** check box will remove any object from segmentation that touches the edge of the image. This can useful for excluding objects that are being cut off by the bounds of the image.

When **Color code segmentation** is selected, the segmentation images shown on the Morphometrics axes during processing will be color-coded. Objects that have been identified in the image and meet all of the user criteria are shown in green (these objects are always shown); objects that have been removed due to size constraints are shown in red and are not used in further processing steps; and objects that been removed due to false-positive rejection are shown in yellow and are not used in further processing steps.

For beginners who desire fewer input parameters, unchecking **Advanced mode** removes all but the essential adjustable parameters, and puts default values in for the more advanced parameters.

When **Do not find contours** is selected, segmentation will proceed and segmentation data will be written to the output file, but no contours will be fit to the segmented objects. This can be useful to save time / memory if contours are simply not desired.

The **Save output** check box writes the contour data to a mat-file, where if the input file name is `input_file.tif`, the saved output file name will be `input_file_CONTOURS.mat`; later sections discuss the structure of that output file.

The **Keep temp files** check box saves the most recent TIF stack's temporary segmentation files. These can be useful in analyzing how well the program is working, or why the program is missing certain objects. Note that each of the two temporary files will be as large as the input TIF file, and every time Morphometrics runs, it will erase these temporary files and start writing new ones. Thus if these files are to be kept, the temporary file names must be changed between runs, in particular remove `_Gparent` and `_Gsegt` from the file names.

The **Open single stack** button prompts the user to select a single TIF stack for processing, whose file name, once selected, will be displayed in the text box below the button. The first image in the stack will be displayed on the program axes, and the image size, bit depth, and minimum and maximum intensity values will be displayed in the **Image data** pane on the bottom left of the Morphometrics window.

The **Image data** pane displays the width, height, bit depth, and intensity range of the current image, and warns the user if image saturation, which degrades segmentation and contour fitting quality, is detected.

The **Open multiple stacks** button brings up a GUI window to select a directory, inside of which all files that meet the search string `*motif*.tif` will be processed serially. Enter the `*motif*.tif` name motif before selecting **Open multiple stacks**. The number of files to be processed and their names will appear in the text box below the **Open single stack** button.

Segmentation

Morphometrics offers four different algorithms for segmentation. Depending on the contrast level and the density of cells one method of segmentation may work better than another – thus choosing the best algorithm for a particular data set is a bit of a black art. These algorithms only affect which regions are segmented, but do not affect how the contour fitting is performed.   Not all **Process Parameters** are compatible with all segmentation methods, and those parameters that are not compatible with a given method will be disabled by Morphometrics when that segmentation method is in use. Additionally, the **Laplacian Segmentation** and **Adaptive Threshold Segmentation** methods do not work with **Fluorescence (peripheral)** image types.

The segmentation algorithms are some of the most intricate internal aspects of Morphometrics, and as such their detailed workings will not be discussed here. However, the internal code is open source, and interested users can study the internal code in the `simply_segment.m` file to better understand these algorithms. Despite these complexities, below are guidelines for the usage of each segmentation method.

Gradient Segmentation looks for a contiguous region bounded by a region of high image gradient magnitude to watershed segment regions for further processing. This is the work horse method that tends to work well on a broad class of input images, but can fail to separate large groups of densely packed cells. All of the **Process Parameters** are employed by this algorithm.

Laplacian Segmentation is similar to **Gradient Segmentation** in that it looks for maxima in the image gradients, however in this case the algorithm looks directly at the zero crossings of the image Laplacian and includes a number of other morphological steps to arrive at the segmentation. This method often works well with dense aggregates of cells (see the second detailed example in this help file). All of the **Process Parameters** are employed by this algorithm.

Threshold Segmentation requires the user to manually select an ROI in the image stack which is then defined as the stack background. From this background region a mean background level and background variance are calculated for each frame. Using these reference values the segmentation algorithm looks for contiguous regions with pixel intensities that lie **Intensity threshold** (**Process Parameters** pane) number of standard deviations above (**Fluorescence (internal)**) or below (**Phase**) the background variation. This algorithm is arguably the simplest to understand and gives robust results on images where the background intensity is flat. The **persistent bckgrd ROI** check box keeps the user-selected background ROI in memory for future use; this box should be unchecked if a new background ROI is desired. A persistent background ROI cannot be used in batch stack processing with **Open multiple stacks**. Selecting **simple threshold** uses a user-specified intensity threshold, as opposed to a threshold measured in standard deviations of the background fluctuation.

Canny Segmentation uses the classic Canny edge detection algorithm to find object edges for segmentation. The Canny edge detection algorithm requires that a lower and upper threshold for edge detection be specified in addition to a 'sigma' smoothing factor, and thus does not require a **Primary Threshold**. The meaning and usage of these parameters is discussed extensively in the image processing literature, for instance see:

http://www.mathworks.com/help/images/ref/edge.html
http://en.wikipedia.org/wiki/Canny_edge_detector

The Canny method finds edges by looking for local maxima of the gradient of the input image. The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be fooled by noise, and more likely to detect true weak edges. The sigma factor sets the standard deviation of the Gaussian filter, thus larger values tend to produce smoother segmentations at the cost of losing weaker edges. [ref: MATLAB `help edge`] Canny segmentation is not compatible with the false-positive detection algorithm. Like the **Gradient segmentation** method, this tried and true method of edge detection can be used with a broad class of input images, however it does not work with **Fluorescence (peripheral)** type images.

Process Parameters

Image scaling scales the image in both X and Y by a factor ranging from 0.1 to 10 of its input size. This can be useful in cases where the feature size is only slightly larger than the pixel size -- the segmentation algorithm will perform better if features are on the order of 10 pixels. Keep in mind that larger images take longer to process than smaller images, and the scaling process itself adds time to the calculations. The values of all subsequent parameters are in reference to the *unscaled* image. This allows the user to adjust the image scaling without having to adjust other parameters (like minimum and maximum object size constraints) to suit the image scaling.
Back. rmv. size stands for 'background removal size' and sets the standard deviation, in pixels, of the Gaussian blurring filter that is used to remove background from input images. This value should greater than the length scale of objects to be segmented; for instance, if cells are 20 pixels across, the background removal size should be set to at least 30, and more likely 60 or 80 pixels. Setting this value to zero turns off background removal altogether. In many instances where either contrast in the original image is low or there is enough signal that local adjustment of the intensities can sharpen object boundaries, removing the spatially varying background may be advantageous. The size of the Gaussian blurring filter scales with this parameter and hence removing background at a larger spatial scale takes more time. To keep the filter properly centered and not induce a shift in the images this filter size will always be an odd number.

Histogram limit is the percentage at the top and bottom end of the image intensity histogram that will be saturated. This is useful when there is a particularly light spot in an image that would otherwise distort the segmentation thresholding. For larger images, it is helpful to set the value higher, for instance, as large as 0.01 for a 1 megapixel image.

Min region size sets the minimum size of a contiguous block of segmented pixels to which a contour can be fit, or said differently, this is the minimum object size that will be segmented. This number must be a positive integer. A good estimate is to set this value to 1/10 the estimated object size. Often if a small region seems like it should be easily segmented by the program but it is not, you may want to decrease this value, to allow smaller objects to be found.

Max region size sets the maximum size of a contiguous block of segmented pixels to which a contour can be fit. This number must be an integer greater than **Min region size**, and less than the size of the image itself. A good rule of thumb is to set this value to be at least 4 times the estimated size of the objects to be segmented. If a large region seems like it should be easily segmented by the program but is not, you may want to increase this value, allowing larger objects to be found.

Image smoothing is a positive value for the size, in pixels, of a Gaussian smoothing filter applied to the input images. Smaller values (near 1) tend to preserve details, making for closer and generally more detailed fits to the cell contour, but will have higher noise and may result in spurious regions being segmented. Larger values will create a smoother contour at the cost of losing details and possibly joining objects that are close together, but fewer spurious regions will be segmented. Setting this parameter to 0 turns off the smoothing altogether, and in many instances no smoothing is necessary.

Primary threshold is a positive value between 0 and 1 that sets the significance of intensity extrema that will be segmented. Lower values are more inclusive, and may split objects that are close together, at the cost of including spurious regions or dividing regions that should not be divided. Higher values can be used when cells are better separated and/or when the image has higher native contrast. Thus it is always preferable to collect images with the highest genuine (i.e. not saturated) contrast.

Cut distance is a positive value greater than zero that is only active when **Cells are in proximity** is selected. The **Cells are in proximity** box is selected when groups of cells are close enough that their light fields overlap. After the primary segmentation, the **Cut distance** value sets the significance of extrema within the already segmented regions. Higher values tend to keep objects together, while lower values allow objects to split apart. Thus cells that are close together can often be well-resolved by such two-step segmentation. The value of this parameter is approximately the narrowest approach of two sides of the segmented region across which secondary segmentation will cut. For instance, if two cells, each 20 pixels across, are joined by a narrow pixel region of 5 pixels, setting **Cut distance** to 5 pixels will tend to split such objects.
Force smoothing is a positive value that smoothes the pseudo-energy landscape used to fit the contour. This parameter makes fine adjustments to the contour shape, but does not have an effect on which regions the algorithm chooses to fit a contour. Smaller values tend to put the contour more toward the outer edge of the cell, while larger values tend to put the contour toward the inner edge of the cell. Larger values will also create smoother contours at the expense of degrading shape information along the contour. Leaving this value at 1 is usually a reasonable balance between spatial detail and noise.

Fractional overlap is a positive value between 0 and 1 that is only active when **Images are independent**, **Seed contours**, or **Track objects manually** are *not* selected. This value sets the minimum fractional area overlap between two objects for them to be considered the same object between different frames. The number of frames into the past and future over which objects will be compared is set by **Frame overlap**. The value that one chooses depends on how much the cells move and/or grow between frames. If cells are well separated, then the unique cell IDs are relatively robust to changes in this value. If movement of the cells between frames is large and/or the cells are very close together, you will have to play around with this parameter to achieve consistent cell IDs across many frames. If the value is set below 0.5 there is the possibility that two distinct cells may be identified with the same cell ID in the same frame, thus it is generally recommended that this value be greater than 0.5.

Frame overlap, as discussed above, is an integer greater than zero that sets how many frames into the past and future Morphometrics looks for cells that might overlap sufficiently with the current cell to be considered the same actual cell. If the algorithm loses track of a cell in a single frame then setting this value greater than one will allow the algorithm to give a cell the same ID even if it did not appear in consecutive frames, but met the overlap condition.

Testing and Processing

The **Run Analysis** button will use the current stack(s) and parameters to find unique cells and their contours. If **Save output** is selected, the results will be written to an output mat-file after each file has finished processing. One may wish to run the analysis without saving just to be sure that the current parameters perform well across the entire data set, and in that case, simply press **Run Analysis** with **Save output** unselected.

Pressing **Save parameters** takes all of the currently entered parameters, check box values, and advanced parameters and writes them to a mat-file with a user-specified name. Saving a parameter mat-file in the Morphometrics folder with the name `Morphometrics_prefs.mat` makes those values the default values to which the program will initialize. The program also has a set of internal defaults that it will use in the absence of an input parameter file. While not generally advisable, these defaults can be adjusted in the `morphometrics.m` file. Every saved output mat-file created by **Run analysis** also includes the parameters used to create it, and hence ever output mat-file is also a parameter file.

Pressing **Load parameters** allows the user to load a particular configuration of check boxes and parameter values from a chosen mat-file. The mat-file can either be created by **Save parameters**, or any mat-file created by **Run analysis** also contains all the parameters used to create it, thus loading a `*_CONTOURS.mat` file as a parameter file will configure the program in the way used to generate that mat-file.

The **Test parameters** button is useful for testing the current set of parameters against the current image, without investing the time required to process the full image stack. Thus one can choose a **Frame** to analyze and test as many parameters sets as one wishes against the current image before committing to processing. When batch processing, this function will only test parameters on images in the first stack of the batch. The **Help estimate primary threshold** check box will bring up an additional window during testing showing the histogram of image intensities in such a way as to guide the user to approximate an appropriate **Primary threshold**.

The **save test image** check box saves a TIF image of the current test frame with the contours overlaid, and also creates a vector graphic output of the current frame with contours overlaid and a snap shot of the program state at the moment of testing.

The **Select freehand ROI** allows the user to draw a region of interest by hand over the image; only cells contained within this region will be processed when **Run analysis** is pressed. This ROI will be persistent on the current stack until either a new stack is loaded or until **Delete ROI** is pressed.

The **Stop** button stops any currently running operation and saves no data. If processing multiple stacks in batch, the **Stop** button will stop the current stack computation, and move on to the next stack in the batch.

The **Help** button opens this help file in the native PDF program (that's probably how you got here).

The **Exit** button closes the program and erases all temporary program variables; you can also close the program with the 'X' button at the top of the program window.

Meshing

The **Calculate meshes** button uses the X-Y coordinates of the object contours from an output mat-file (i.e. a file produced by **Run Analysis**) to calculate an internal cell mesh and cell neutral axis or branch network. There are two kinds of meshing algorithms from which to choose. The **full mesh & branching** option creates data structures for an internal cellular mesh and branch network of each cell in the image. All attributes of the mesh and branches are unambiguously specified by the contour points, and hence no parameters are required for this analysis. The advantage to this type of mesh is that it can mesh the internal cellular space and create a branched network for *any* shape it is given, so it is very general. Examples of branching and non-branching cells with this type of meshing can be seen in the included file `full_mesh_example.pdf`. The output file will append `full_MESH` into the input mat-file name. The **pill mesh** option creates data structures for a 1D internal mesh and single internal branch of each cell in the image. These types of meshes are used when the object being examined is clearly elongated in one direction, such as a rod-shaped bacterium, and you wish to parse its shape along that direction / axis. Selecting **Use motif** will prompt the user to enter a search string motif and a directory that contains contour files to be processed. The program will then batch process all the mat-files that match the name motif in the current directory. More details about what exactly is calculated by each of these meshing algorithms is included in output description under the section **Output Data Structures**.

Advanced Options

Under the **Options** menus selecting **Advanced Options** allows the user to change a series of internal control variables. Generally, these do not need to be changed and default values are set in the `morphometrics.m` m-file. However, segmentation with certain images may benefit from changes to these parameters.

The first parameter sets the radius in pixels of the noise reduction filter (`relnoise.m` filter) used to remove noise from the input images. Setting this parameter to 1 turns off this feature completely.
The second parameter adjusts the false positive ratio parameter, which sets the threshold for false positive detection; setting this parameter less than 0 will encourage false positives, setting this parameter between 0 and 1 will weakly reject false positives, setting this parameter equal to 1 should be a decent choice to reject actual false positives, and setting this parameter greater than 1 will tend to create false negatives.

The third parameter sets the fraction of the intensity histogram to be used as a simple threshold for a mock segmentation. Morphometrics uses this value to determine the level for the threshold, then segments the image based on this threshold and examines the maximum object size from this segmentation. If the maximum size is less than the minimum area requirement for segmentation, then no segmentation is performed. This is to prevent Morphometrics from attempting to segment objects from images that lack significant image features. Setting this value to 0.5 (above the recommended maximum of 0.5) will allow segmentation to proceed on any input image, regardless of the features present (or not present). Values closer to zero impose a more stringent requirement for segmentation to begin.

The fourth parameter sets the minimum contour length that can be processed. Morphometrics calculates contour features, such as curvature, that require a minimum number of neighboring contour points, thus all usable contours must be at least 4 points, and 10 points is a good minimum to ensure that Morphometrics is fitting a contour to a usable object.

Morphometrics automatically removes kinks and acute angles in the fitted contours. The fifth parameter sets the maximum allowed contour angle. Setting this value to 1 softens any contour angles more acute than 90 degrees, setting it less than 1 allows for angles more acute than 90 degrees, and setting it greater than 1 will soften angles more obtuse than 90 degrees. The default value is 1.

Morphometrics automatically determines which cells are next to which in each frame, and this sixth parameter sets the distance in pixels below which two cells are considered proximal.

Since the curvature is a second derivative of the contour position, it is particularly beholden to small fluctuations in contour point positions, and hence to produce better results Morphometrics automatically smoothes the measured contour curvature. The final parameter sets the size, in number of neighboring contour points, of the Gaussian filter used to smooth the contour curvature measurement. Larger values tend to produce smoother profiles of contour curvature at the cost of reduced spatial resolution of curvature along the contour.

Enable Buttons
To prevent accidental execution of commands while Morphometrics is in the midst of other calculations, the program automatically disables all the execution buttons during computation. After the computation is finished, the buttons are then re-enabled. However, should Morphometrics experience an error during computation, the buttons will not be automatically re-enabled and the user must select **Enable Buttons** from the **Options** menu, or press the **Stop** button, before further computation can commence.

Viewing Contours

Under the **Analysis** menu, selecting **View Contours and Stats** launches a script that allows the user to view and manipulate previously calculated contours with their matched image stacks, and to calculate basic statistical properties of contours.

Visualize Contours

Press **Load Data** and first select the image stack that you want to view and then select the contour mat-file that corresponds to that image stack. The script will then tell you how many frames there are to work with and which files were selected. Choose a frame number next to **view frame** and press **View Frame** -- this displays that frame from the image stack with the contours overlaid in green. Selecting **uniquely color each cell** gives each contour a unique random color. Selecting **uniquely number each cell** plots the cell ID over the contour of that cell. Selecting **show cell poles** plots the first and last point of the contour in green and red, respectively, at one pole, and will mark the opposing pole with a red 'x'.

If a `pill_mesh` is provided in the loaded mat-file, selecting **show shape classification** plots the mesh over the cell contour color-coded by whether it is: a bulging region (red) defined by regions with opposing contour curvatures that are both positive, a bending region (green) defined by regions with opposing contour curvatures of opposite sign, or a constricting region (blue) defined by regions with opposing contour curvatures that are both negative. The opposing contour curvatures are given by `frame(i).object(j).side1_kappa` and `frame(i).object(j).side2_kappa`. This options is mutually exclusive with **uniquely color each cell**.

In some cases, the optics of the microscope will cause there to be a slight image shift between channels, and you may want to shift where the contours sit on the image, using the **X shift** and **Y shift** input parameters. If mesh data is available (i.e. after you have used the **Calculate meshes** feature), selecting **show mesh** will overlay the cell mesh with the contour. Selecting **save TIF stack** and pressing **View All and Process** will create a TIF stack with the current figure frames as images. Pressing **Save Frame EPS** saves the current frame with the selected visualization options as a vector graphic file, suitable for figures and presentations. The rate at which frames switch when **View All and Process** is pressed is set by the **frame delay** in seconds. Alternatively, by checking **progress via arrows** and pressing **View All and Process**, the user can skim through the stack with the selected visualization options one frame at a time, using the right and left arrow keys.

Contour Statistics and Plots

In addition to viewing contours, one may want to analyze basic features of object shape, including object widths and lengths, across many objects and/or through time. If the selected mat-file contains pill mesh data, then selecting **Make Plot** will create one of the four selected plot types indicated immediately to the right of the button. The object length is defined by the length of the centerline of the object's pill mesh and the mean length is calculated over all valid objects in a single frame. The width varies along the centerline and is defined at each centerline point by length of a line normal to the centerline at that point that intersects the each side of the contour. However, in many instances (e.g. bacteria), valid values of the width should be filtered based on the curvature of the contour. Thus the **curvature cutoff (1/um)** option allows the user to exclude a width measurement if the contour is has a curvature that is above the curvature cutoff. This is useful, for instance, in ignoring the width of polar regions in pill-shaped cells. If no curvature cut-off is desired, simply enter **na**. The mean width of an object is then the mean length of all such intersecting lines that meet the curvature criteria. For stacks of independent objects one can create histograms of cell length and width using the appropriate checkboxes. The **bin number** sets the number of bins in these histograms. For stacks with persistent objects, one can create plots of cell length and/or width as a function of frame number (time), and may be plotted with the standard deviation of width / length at each time point by selecting **w/ std**. These plots require that the time between frames be entered in **seconds / frame**. In all of these cases, the pixel calibration must be entered in **nm / pixel** and selecting **exclude proximal cells** will only calculate the desired statistics for cells that are not immediately adjacent to other cells, that is, those objects for which `frame(i).object(j).proxID` is an empty vector.

Creating a Cell ID Table

If images in a stack are not independent and hence have objects that are persistent through time, creating a cell ID table can be very useful for analysis. Under **Analysis**, select **Create Cell ID Table**, and a window will appear prompting the user to select a contour mat-file. The script will automatically check that the necessary structure field `frame(i).object(j).cellID` is present in the given mat-file. The script will add a new structure to the mat-file called `cells`, containing two sub-fields `frame` and `object`. If Morphometrics identified a number of objects as being the same between frames then each such persistent object will have an entry in `cells`, for instance the first two cells of many in 10 frames might look like

```
cells(1).frame = [1 2 3 4 6 8 9 10];
cells(1).object = [2 2 3 3 2 2 3 5];

cells(2).frame = [1 3 4 5 6 7 8 9 10];
cells(2).object = [3 3 4 4 1 1 2 8];
```

Note that the frames are not necessarily consecutive and the object numbers are not necessarily the same in each frame. In this example the persistent object with cell ID = 1 appeared in frames 1, 2, 3, 4, 6, 8, 9, and 10, and in each of those frames had the object label 2, 2, 3, 3, 2, 2, 3, and 5, respectively; likewise for cell ID = 2. The total number of cells is `length(cells)`, and for any particular cell `n`, the number of frames in which it appears is `length(cells(n).frame)`. Said differently, if one wants to look up all the data associated with cell ID = `n`, one need only cycle through an index `k` in

```
frame(cells(n).frame(k)).object(cells(n).object(k))
```

and

```
frame(cells(n).frame(k)).object(cells(n).object(k)).cellID
```

should be the number `n` for all valid values of `k`.

Analyzing Object Intensity

Selecting **Analyze Object Intensities** under the **Analysis** menu opens the contour and internal intensity analysis script `intensimetrics`, which is used to compute the intensity along object contours and, if pill mesh data is included, for computing intensity along object centerlines. First use the **Select *.mat file** and **Select image stack** buttons to select mat-files produced by Morphometrics and a matching image stack, respectively. The image stack need not be the same image stack used by Morphometrics to create the mat-file, for instance maybe phase contrast was used by Morphometrics to create the contours and now you want to use those contours to calculate the intensity of a fluorescent protein within the cell or along the contour.

Selecting **process all cells** will analyze the intensity of all objects in all frames, whereas selecting **process cell** and then entering a valid cell ID number will only process that cell ID as a persistent object; both options will create additional fields in the `frame` data structure, discussed below. Selecting **exclude proximal cells** will exclude immediately adjacent cells from the **process all cells** intensity analysis, which is useful because cells that are immediately adjacent often have overlapping intensity fields. Selecting **calculate background** will display a stack projection image and prompt the user to select a rectangular sub-region in the image stack that will be defined as the background for all frames in the stack. It is important to choose a rectangular sub-region that does not contain any objects on any frame. The script will prompt the user to enter a 'channel' name before entering the background data (or any other intensity data) into the `frame` data structure. For example, if you were analyzing a GFP image stack, you might enter 'GFP' into the channel name field. If data with that channel name already exists, you will be prompted if you want to overwrite this data. If data with that channel name does not exist, a new `frame.background` sub-structure will be added to `frame`.

For each frame `i`, the following background data will be added to the `frame` data structure

`frame(i).background(n).name` is the name of the channel for which the `n`-th background statistics were calculated.

`frame(i).background(n).mean` is the mean of the selected sub-region background intensities for each frame `i`.

`frame(i).background(n).median` is the median of the selected sub-region background intensities for each frame `i`.

`frame(i).background(n).std` is the standard deviation of the selected sub-region background intensities for each frame `i`.

Selecting **subtract background** will subtract `frame(i).background(n).mean` from all calculated intensity profiles in frame `i`, using the background data associated with `frame(i).background(n).name` channel, where `n` identifies the individual intensity data channels. If the background in a particular image stack is not even, then additional pre-processing steps (such as median filtering or high-pass FFT filtering) must be applied to the image stack. These methods are not included in the Morphometrics software package, but are part of NIH's free ImageJ software package.

In some cases, the optics of the microscope will cause there to be a slight image shift between channels, and you may want to shift where the contours sit on the image, using the **X** and **Y Contour offset** parameters.

Selecting **remove noise** applies the relative noise filter employed throughout Morphometrics to all contour, pill mesh, and background intensity calculations. Under the **Options** menu, selecting **Noise Filter Parameters** allows the user to adjust the **Filter size** in pixels and the strength of the filter **Sigma**. If changed, the new parameters will persist until the script is launched anew.

Selecting **contour intensity** computes the intensity of the given channel along object contours. Under the **Options** menu, selecting **Contour Parameters** allows the user to adjust the **Line width** in pixels used to compute the intensity at every point on the contour, the **Resolution** which sets the number of intensity samples per pixel along the **Line width**, and the **Offset** which adjusts how the sampling points are laid down with respect to the contour, where 0 means that the sampling points will start at the contour and sample outward, 0.5 means half the points will sample inside the contour and half outside, and 1 means the points will start at the contour and sample inside the contour. Selecting **smooth profile** applies a polynomial smoothing algorithm to the contour intensity data using the point **span** and polynomial **degree** specified.

Selecting **pill mesh intensity** calculates the integrated intensity of the mesh element associated with each point of the object centerline. It is important to note that mesh elements that contain more pixels will, all else being equal, have higher integrated intensities, thus selecting **normalize by mesh area** will divide the total integrated intensity at each centerline point by the area of the associated mesh element, thus giving the mean intensity of the pixels within the given mesh elements. Note that calculating the pill mesh intensities is computationally laborious because mesh elements cover fractions of pixels and thus many intersections of pixels and mesh elements must be calculated, and thus this operation can take a long time.

Selecting **interior intensity** calculates the mean integrated intensity of the originally segmented pixels as well as the corresponding pixel area. The mean intensity will include any adjustments due to background subtraction or noise removal.

Selecting **show figures** shows the current frame, highlights the object being processed, and shows the intensity profiles being calculated, however this generally slows down the processing.

Visual Quality Check

Under the **Analysis** menu, selecting **Visual Quality Check** launches a script for visually assessing contour quality. First use the **Select *.mat file** and **Select image stack** buttons to select mat-files produced by Morphometrics and a matching image stack, respectively. The image stack need not be the same image stack used by Morphometrics to create the mat-file. For instance, one may want to determine the translation between distinct imaging channels.

Selecting **process all cells** will display the images and corresponding contours of all objects in all frames, one-by-one, whereas selecting **process cell** and then entering a valid cell ID number will only display that cell ID as a persistent object; both options will create additional fields in the `frame` data structure, discussed below. Selecting **exclude proximal cells** will exclude immediately adjacent cells from the **process all cells** visual assessment, which is useful because cells that are immediately adjacent often have distorted contours, due to overlapping intensity fields. In some cases, the optics of the microscope will cause there to be a slight image shift between channels, and you may want to shift where the contours sit on the image, using the **X** and **Y Contour offset** parameters before making your visual judgment.

After the files have been loaded and the **Quality Check** button selected, the script will display a cropped image of each identified object and its contour overlaid. Pressing the right-arrow key both advances to the next object in the frame (or the first object in the following frame) and writes to the current data structure a positive affirmation of contour quality

```
Frame(i).object(j).quality = 1
```

When on any image and corresponding contour the down-arrow is pressed, a negative affirmation will be written to the current data structure

`Frame(i).object(j).quality = 0`

At any time, the delete / backspace key may be pressed to save the current data and end the quality assessment process.

Command Line Scripts

For advanced users who wish to perform processing through a command line (CL) / terminal interface, the CL version of the Morphometrics scripts are included with the software release in the `Morphometrics_CL` folder. These scripts use precisely the same algorithms as the GUI version, but a parameter mat-file will need to be created in order to run the scripts. The parameter testing platform, keeping temporary files, getting help with the threshold estimation, manual tracking, and freehand ROI selection are not available in the CL version of Morphometrics, because they all require graphical user interaction. A convenient way to create a CL parameter file is simply to save a parameter file from the GUI version of Morphometrics. Special care should be taken when creating parameter files for CL use as the fail safes, parameter checking, and warnings built into the GUI are not present in the CL version of Morphometrics.

Morphometrics GUI parameters have internal representation that either begins with `f_` or `v_`. Below is a list of all internal variables and to which parameters in the GUI they map:

Image Attributes:
`v_imtype <-->` 1 = Phase; 2 = Fluorescence (internal); 3 = Fluorescence (peripheral)
`v_method <-->` 1 = Gradient Segmentation; 2 = Laplacian Segmentation; 3 = Adaptive Threshold Segmentation; 4 = Canny Segmentation

Process Parameters:
`f_resize <-->` Image scaling
`f_back <-->` Back. rmv. size
`f_histlim <-->` Histogram limit
`f_areamin <-->` Min region size
`f_areamax <-->` Max region size
`f_gstd <-->` Image smoothing
`f_hmin <-->` Primary threshold
`f_hmin_split <-->` Cut distance
`f_r_int <-->` Force smoothing
`f_pert_same <-->` Fractional overlap
`f_frame_diff <-->` Frame overlap

Canny segmentation:
`f_canny_th1 <-->` Canny lower threshold
`f_canny_th2 <-->` Canny upper threshold
`f_canny_sig <-->` Canny sigma smoothing factor

Advanced Options:
`f_relsz <-->` Diameter of relative noise filter
`f_int_rej <-->` False positive rejection parameter

`f_seg_min <-->` Intensity histogram tail weight for segmentation to occur
`f_contmin <-->` Minimum contour length
`f_acute <-->` Contour acute angle parameter
`f_pixelprox  <-->` Cell proximity cutoff
`f_curve_std <-->` Standard deviation curvature smoothing

Internal controls:
`f_Nit  <-->` number of iterations for contour fitting algorithm
`f_kint <-->` rate of point movement during contour fitting

Check boxes:
`v_indpt <-->` Images are Independent
`v_prox <-->` Cells are in proximity
`v_falsepos <-->` Reject false positives
`v_save <-->` Save output
`v_mm_mesh <-->` full mesh & branching
`v_mt_mesh <-->` pill mesh
`v_seed  <-->` seed contours

After a parameter mat-file is made, locate the full path of the image stack to be processed and the parameter file, then enter:

```
morphometrics_cl(stackname,paramname)
```

in the MATLAB command line, where `stackname` is the full path to the image stack and `paramname` is the full path to the parameter mat-file. When the script is finished running a new output mat-file will be created in the same directory as the image stack. The script can also be called with two additional inputs

```
morphometrics_cl(stackname,paramname,rect1)
```

where `rect1` is a vector [X0,Y0,width,height] which specifies the coordinates of a box for background determination during Adaptive Threshold Segmentation, and

```
morphometrics_cl(stackname,paramname,rect1,meshq)
```

where `meshq` is a binary variable to tell Morphometrics to perform meshing. Setting `rect1=0` tells Morphometrics to ignore the background rectangle input.

Once a parameter file has been saved, single variables within the parameter file can be changed in the command line by

```
params=matfile(paramname,'Writable',true);
```

then

```
params.parametername=value;
```

For instance, if one wanted to use seeded contours, type the following in the MATLAB command line

```
params=matfile(paramname,'Writable',true);
params.v_seed=1;
```

Output Data Structures

Run Analysis produces an output mat-file containing the structure array `frame`. Running **Calculate meshes** with **full mesh & branching** adds the sub-structures `mesh` and `branch` to the `frame.object` structure. Running **Calculate meshes** with **pill mesh** will add a series of sub-fields to `frame.object` that are discussed below. An exhaustive list of definitions for each field in the output structures is given below:

`Ncell` is the number of identified cells in the stack.

`outname` is the full path of the output mat-file, this can be useful to determine the original directory and file name from which a mat-file was generated.

`length(frame)` is the number of frames processed, starting from the first frame in the stack.

`frame(i).num_objs` is the number of segmented cells or objects in frame `i`.

`frame(i).object(j)` refers to the segmented object `j` in frame `i`, where `j` *is not* the unique ID number of the cell, but *is* a unique label in frame `i`. The value of `j` is the non-unique label number given to the segmented region `j` in frame `i` by any `bw` operation in MATLAB (e.g. `bwmorph`, `bwlabel`, `regionprops`).

Selecting **Create Cell ID Table** under the **Analysis** menu transforms the cell lineage data stored in an output mat-file into an ordered list of the frames and object numbers in which a particular cell can be found. For example, if `frame(1).object(1).cellID=1` and `frame(2).object(3).cellID=1`, this means that Morphometrics determined that object 1 in frame 1 is the same cell as object 3 in frame 2. **Create Cell ID Table** creates a data structure that lists the frames and object numbers that correspond to each unique cell ID. For this example, calling `cell(1).frame=[1,2]` and calling `cell(1).object=[1,3].`

`frame.object` has subfields:

`frame(i).object(j).X/Ycent` are the X/Y coordinates of the object centroid as defined by the pixels of the segmented region.

`frame(i).object(j).area` is the area in square pixels of the region enclosed by the contour.

`frame(i).object(j).circularity` is the contour length squared divided by $4\pi\pi$ times the enclosed area – this is a dimensionless measure of circularity where a circle is defined as 1, and increasing values are less circular.

`frame(i).object(j).theta` is the object orientation in radians, with respect to the image origin, as defined by the segmented region.

`frame(i).object(j).X/Yperim` are the X/Y coordinates of the pixels that form the boundary of the segmented region, with these values a full reconstruction of the original segmentation can be created. For instance, to reconstruct the original image segmentation of frame `i`, execute

```
im_name='original_image_stack.tif';
Im0=imread(im_name,1);
Im1=zeros(size(Im0));

for j=1:frame(i).num_objs
      for k=1:length(frame(i).object(j).Yperim)
            Im1(frame(i).object(j).Yperim(k), frame(i).object(j).Xperim(k))=1;
      end
end

Im2=imfill(Im1,'holes');

figure;
imagesc(Im2)
xlabel('X')
ylabel('Y')
title([im_name ', frame ' num2str(i) ...
', ' num2str(frame(i).num_objs) ' segmented objects'])
box on
axis equal tight
```

`frame(i).object(j).bwlabel` is the unique MATLAB `bw` operations label of object `j` in frame `i`; this value is unique within a frame but not between frames.

`frame(i).object(j).cellID` is the unique cell ID number of object `j` in frame `i`; this number will (hopefully) be the same for the objects across frames that refer to the same actual cell.

`frame(i).object(j).proxID` is a vector with unique cell ID's of any cell in frame `i` that lies within the pixel distance specified by **Cell proximity cutoff** (under **Advanced Options**) of the current object `j`. If no cells are nearby, this will be an empty vector.

`frame(i).object(j).arc_length(k)` is a vector of distances between the `k` and `k+1` points of the contour. The last distance is between the last and first points. Generally, these values should all be equal and close to 1 pixel.

`sum(frame(i).object(j).arc_length)` is the total arc length of the contour in pixels.

`frame(i).object(j).pole1` is the contour coordinate that Morphometrics estimates to be at the sharpest positive curvature region of the cell; for cells like bacteria, this would be a pole.

`frame(i).object(j).pole2` is the contour coordinate that Morphometrics estimates to be the second sharpest positive curvature region of the cell; for cells like bacteria, this would be the opposing pole to `pole1`. The contour will be oriented around one of these two poles, and hence either `pole1` or `pole2` will be equal to 1.

`frame(i).object(j).kappa_raw` is the three-point curvature vector (i.e. 1 / radius of curvature) of the contour, measured in inverse pixels. If the pixel calibration is known, then the actual curvature is the pixel curvature divided by the pixel calibration (nm / pixel), i.e. you should end up with units of inverse length.

`frame(i).object(j).kappa_smooth` is the smoothed curvature vector of the contour. The smoothing performs a moving Gaussian average over a multi-point range whose standard deviation is given by the parameter **Standard deviation of curvature smoothing** (under **Advanced Options**).

`frame(i).object(j).X/Ycont` are the X/Y coordinates of the object contour in vector format. Many of the other outputs are calculated from these two vectors (e.g. `kappa_raw, kappa_smooth, area, theta_cont, arc_length, Xcent_cont, num_pts, etc.`)

`frame(i).object(j).num_pts` is the number of points in the contour of object `j` in frame `i`, and correspondingly, in the outputs `arc_length`, `kappa_raw`, `kappa_smooth`, `mesh`, etc.

`frame(i).object(j).X/Ycent_cont` are the X/Y coordinates of the object centroid as defined by the average position of the contour.

`frame(i).object(j).theta_cont` is the object orientation in radians, with respect to the image origin, as defined by the object contour.

After running **full mesh & branching** in the **Meshing** pane, the following fields will be added to existing data structures under the sub-structures `mesh` and `branch`, and then saved to a new mat-file.

`frame(i).object(j).mesh` has subfields:

`frame(i).object(j).mesh(k).X/Y_pts` are the X/Y coordinates of the polygon that specifies the `k`th mesh element. Each mesh element is uniquely associated with a contour point (i.e. contour point `k` is associated with mesh element `k`).

`frame(i).object(j).mesh(k).area` is the area of the current mesh polygon.

`frame(i).object(j).mesh(k).width` is the approximate width of the current mesh polygon, measured from the current neutral axis to the contour.

`frame(i).object(j).mesh(k).perim` is the perimeter length of the current mesh polygon.

`frame(i).object(j).mesh(k).centX/Y` are the X/Y centroid coordinates of the current mesh polygon.

`frame(i).object(j).mesh(k).neighbors` is a vector indicating which other mesh elements in the current cell and frame share a side with the current mesh polygon.

`frame(i).object(j).branch` has subfields:

`length(frame(i).object(j).branch)` is the number of unique internal neutral axes within the current cell.

`length(frame(i).object(j).branch(k).degree)` is the number of points that make up branch `k`.

`frame(i).object(j).branch(k).degree(m)` is a vector in `m` whose values indicate the degree of connection of point `m` in branch `k`. For instance, if the degree is 1, this means that this point only connects to one other point, and hence must be at the end of a branch; if the degree is 2, this means the point connects to two other points within the current branch, and hence must lie within the interior of the branch; if the degree is 3 (or more), this means the point connects to 3 (or more) other points, and must be a node in the branch network where other branches meet the current branch.

`frame(i).object(j).branch(k).X/Ypos` are the X/Y coordinates of all the points in the current branch.

`frame(i).object(j).branch(k).neighbors_left/right` are the branches that meet the current branch on either the left or right (left/right) side. If a branch has no entries on a side, then it must be at the edge of the branch network.

`frame(i).object(j).branch(k).length` is the length in pixels of the current branch. Longer branches usually correspond to the main branches of the mesh, i.e. those branches that span long distances within the cell.

After running **pill mesh** in the **Meshing** pane, the following fields will be added to existing data structures, and then saved to a new mat-file:

`frame(i).object(j).pill_mesh` is a list of X-Y points that specify line segments across the cell, where the midpoint of those line segments defines the centerline of the cell (see below). Each row of this matrix is interpreted as [X1 Y1 X2 Y2] and hence defines a line segment from one side of the contour, through the neutral axis, to the other side of the contour.

`frame(i).object(j).centerline` is a list of X-Y points that specify the centerline of the cell, roughly determined by a single branch Voronoi tessellation. (see below)

`frame(i).object(j).length` is the length of the centerline, and hence the approximate length of the cell.

`frame(i).object(j).width` is vector specifying the width in pixels of the cell contour at each point along the centerline. The width is determined by the length of vectors approximately normal to the centerline that intersect the contour on each side, i.e. the length of line segments specified in `frame(i).object(j).pill_mesh`.

`frame(i).object(j).cent_kappa` is a vector specifying the raw, three-point curvature at each point of the centerline.

`frame(i).object(j).side1_kappa` and `frame(i).object(j).side2_kappa` are the contour curvatures on opposing sides of the cell matched by the line segments of `frame(i).object(j).pill_mesh`, in other words, these vectors allow one to compare how curvature on one side of the cell compares with curvature on the other side of the cell, and hence one can detect overall bending (one curvature is positive the other negative), bulging (both curvatures are positive), or constriction (both curvatures are negative). Similarly, polar regions of bacterial cells can then be defined as the contiguous terminal bulging regions of the cell, with no additional parameters.

`frame(i).object(j).kappa_p1` is the absolute magnitude of the first principal curvature of the cell under the assumption that the cell is locally azimuthally symmetric. This vector is the same as the absolute value of `frame(i).object(j).kappa_raw`.

`frame(i).object(j).kappa_p2` is the second principal curvature found with the radius defined by the intersection of the local contour normal vector and the cell centerline. This value will always be positive.

For a better understanding of these quantities, try making the following figure by executing the code below in the MATLAB command window. First load a `*CONTOURS_pill_MESH.mat` file. The red lines are the line segments from `pill_mesh` and define the width along the length of the cell. The black outline is the contour of the cell. The blue line is the centerline of the cell. Regions of cell bending are labeled with black squares, regions of bulging with blue circles, and regions of constriction with red diamonds.

```
mat_name='my_CONTOURS_pill_MESH.mat';
load(mat_name);

i= "a frame in your stack"
j= "an object in your frame"

figure;
hold on

%plot the contour itself in black
plot(frame(i).object(j).Xcont,frame(i).object(j).Ycont,'k-','linewidth',2)

%plot the centerline in blue
plot(frame(i).object(j).centerline(:,1),frame(i).object(j).centerline(:,2),'b','lin
ewidth',2)

for k=1:size(frame(i).object(j).centerline,1)

      %plot the pill mesh over the cell
      plot(frame(i).object(j).pill_mesh(k,1:2:3),frame(i).object(j).pill_mesh(k,2:2
:4),'r')

      %retrieve the matched contour curvatures
      kappa1=frame(i).object(j).side1_kappa(k);
      kappa2=frame(i).object(j).side2_kappa(k);

      %determine if the point is bending, color mesh element black
     if or(and(kappa1<0,kappa2>0),and(kappa1>0,kappa2<0))

plot(frame(i).object(j).pill_mesh(k,1:2:3),frame(i).object(j).pill_mesh(k,2:2:4),'k
s')
     end

      %determine if the point is bulging, color mesh element blue
     if and(kappa1>0,kappa2>0)
```

```
plot(frame(i).object(j).pill_mesh(k,1:2:3),frame(i).object(j).pill_mesh(k,2:2:4),'b
o')
     end

      %determine if the point is constricting, color mesh element red
    if and(kappa1<0,kappa2<0)

plot(frame(i).object(j).pill_mesh(k,1:2:3),frame(i).object(j).pill_mesh(k,2:2:4),'r
d')
    end
end

%make the plot pretty and label the axes
axis equal
xlim([min(frame(i).object(j).Xcont)-10, max(frame(i).object(j).Xcont)+10])
ylim([min(frame(i).object(j).Ycont)-10, max(frame(i).object(j).Ycont)+10])
xlabel('X')
ylabel('Y')
```

After running **Analyze** in the `intensimetrics` script (**Analyze Object Intensities** under **Analysis**), the following fields will be added to existing data structures, and then saved to the exisitng mat-file:

`length(frame(i).object(j).channel)` is the number of channels for which intensity data has been calculated.

`frame(i).object(j).channel(m).name` is the user-defined name of the `m`-th channel.

`frame(i).object(j).channel(m).back_type` is `none` if no background was subtracted, and `subtracted` if the background was subtracted.

`frame(i).object(j).channel(m).contour_int` is the mean intensity of the `m`-th channel at each point along the object contour, subject to the specified parameters, which may include subtraction of the background if selected by the user.

`frame(i).object(j).channel(m).internal_int` is the mean intensity of the `m`-th channel at each point along the object centerline via the pill mesh, which may include subtraction of the background if selected by the user and may be normalized by the mesh element area.

`frame(i).object(j).channel(m).internal_norm` is `normalized` if the sum of pixel intensities in each mesh element was normalized by the area that element, thereby giving the mean intensity for the mesh element, and `unnormalized` if it is the sum of pixel intensities in the mesh element.

`frame(i).object(j).channel(m).pill_mesh_area` is the area of each mesh element in square pixels; note that it is possible for this value to be fractional and less than one since the mesh exists in a continuous coordinate system.

Detailed Examples

Launch Morphometrics by typing

```
morphometrics
```

in the MATLAB command line. If you chose *not* to add the Morphometrics directory to the MATLAB path, then navigate to this directory using the 'Current Folder' tab in MATLAB before attempting to launch the program.

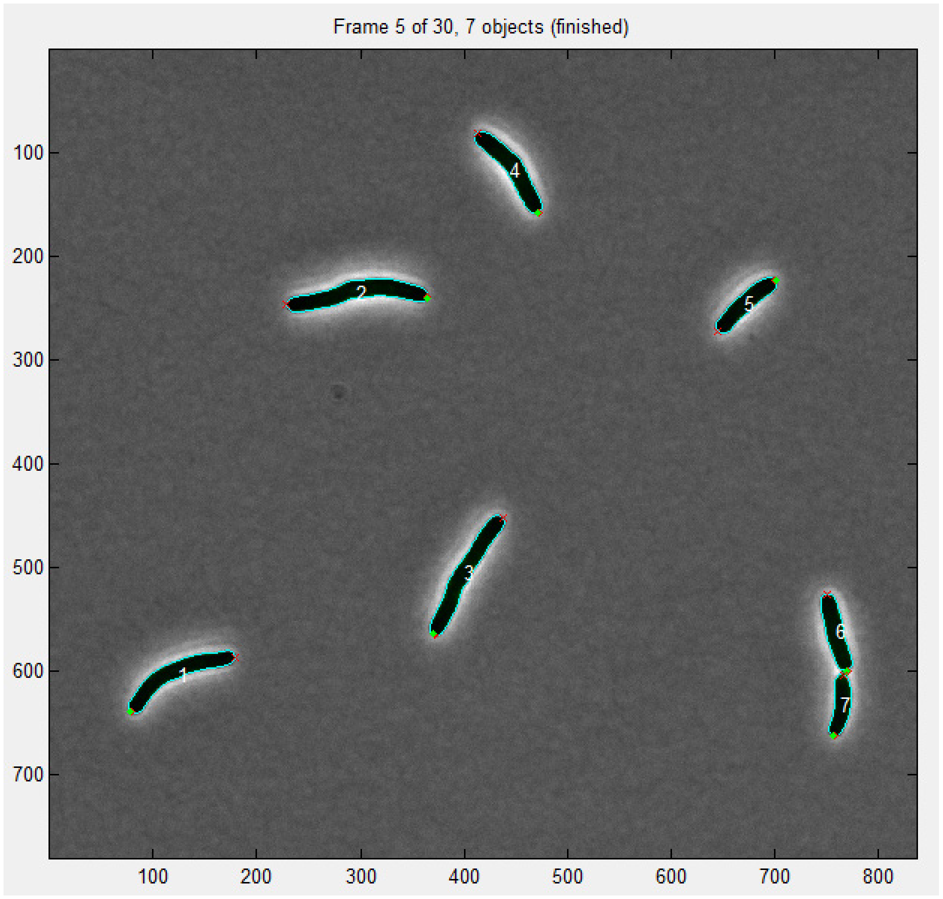
Sparse Phase Data

Press **Load parameters**, navigate to the folder `example_phase_data` and select the parameter mat-file called `Morphometrics_prefs_easy_phase.mat`. This sets all parameters to values appropriate for this example.

Press **Open single stack**, navigate to the folder `example_phase_data` and select the TIF image stack called `easy_test_images_Phase.tif`. This loads the stack into Morphometrics for further processing.

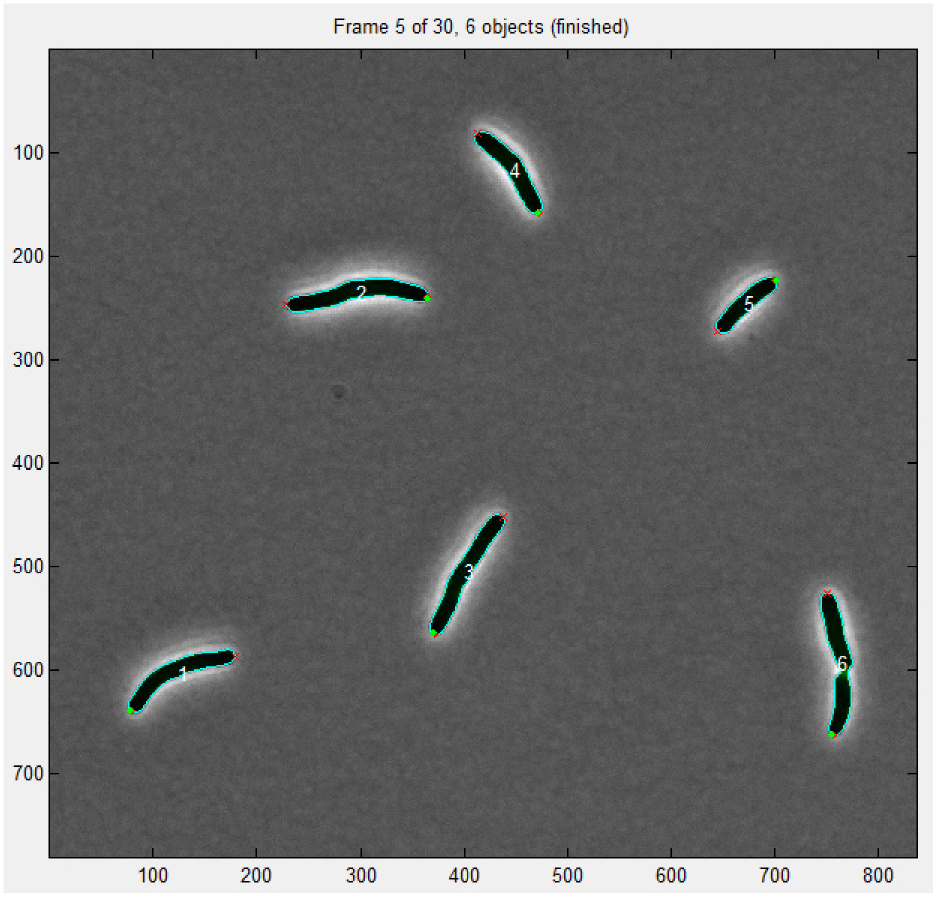
Set the **Frame** parameter under **Test parameters** to 5, to manipulate the 5th frame in the stack for testing.

Press **Test parameters**, immediately the cells should turn temporarily green and then each cell should show a cyan contour, the primary pole marked by red and green dots, the secondary pole marked by a red 'x', and a white number indicating the cell ID. In this image, there should be 7 cells shown

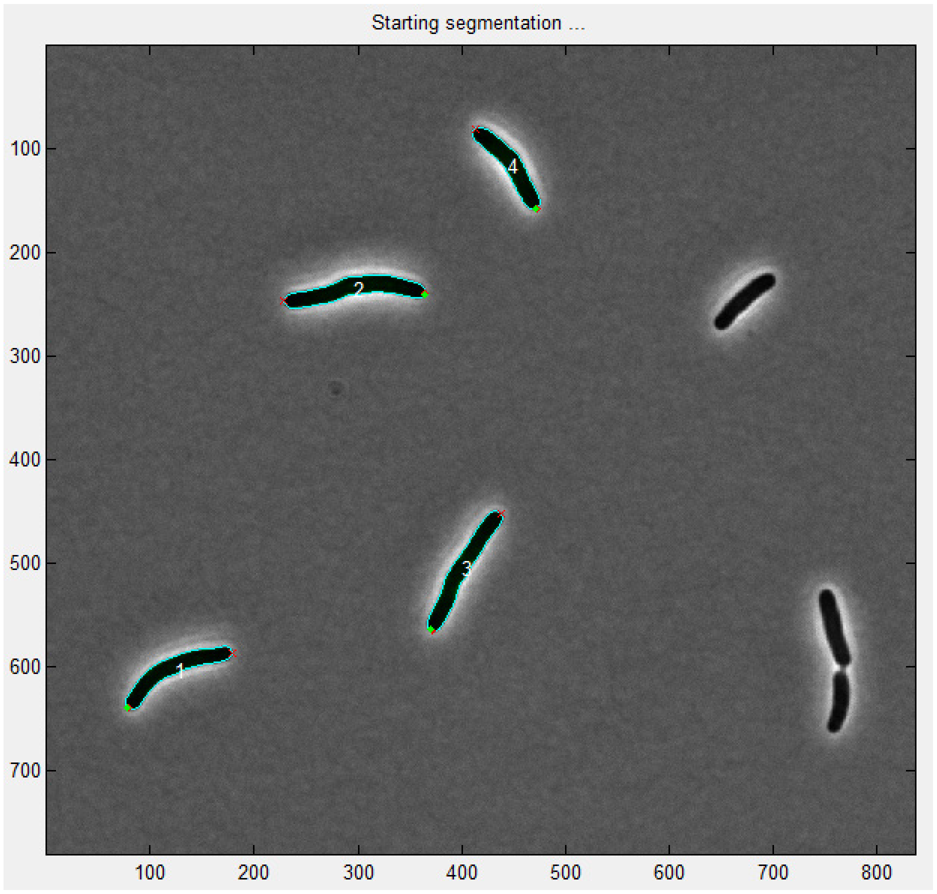


Now let's adjust parameters to see how things change. First uncheck **Cells are in proximity** and press **Test**

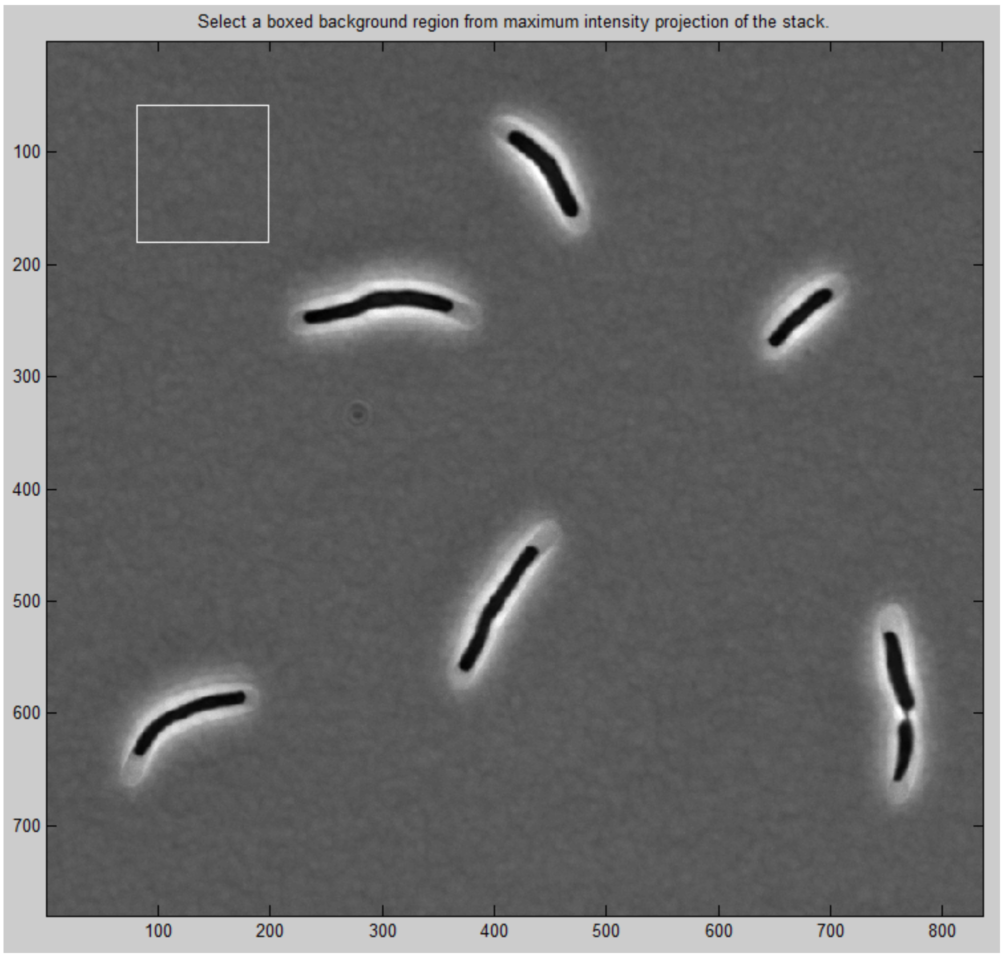
parameters again. Now there should be six labeled cells, since the program no longer uses secondary segmentation to split the cells labeled '6' and '7' above.



Now check **Cells are in proximity**, and set the **Min region size** to 1200 pixels, and press **Test parameters** again. The three cells on the right hand side should momentarily flash red, indicating that they were identified as objects, but were rejected due to the object size constraints, now only four objects should be labeled.

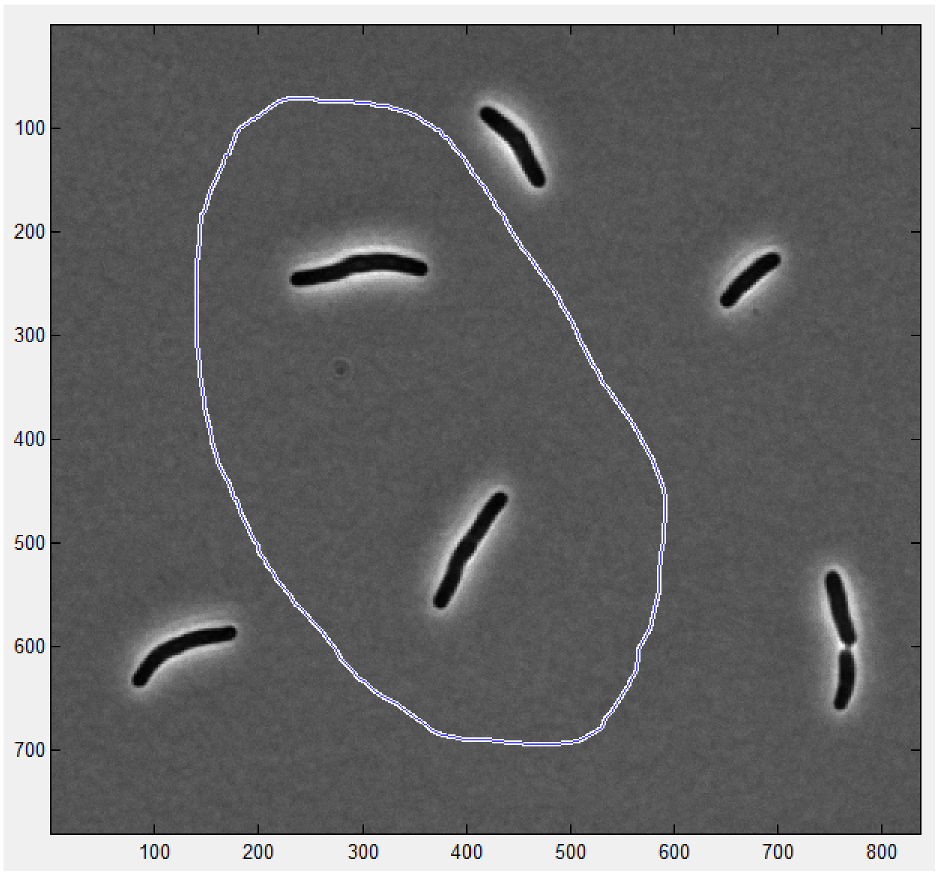


Set the **Min region size** back to 200 pixels, and now select **Threshold segmentation** from the **Segmentation** pane, and select **persistent bckgrd ROI**. Then press **Test parameters** and the following window should pop-up

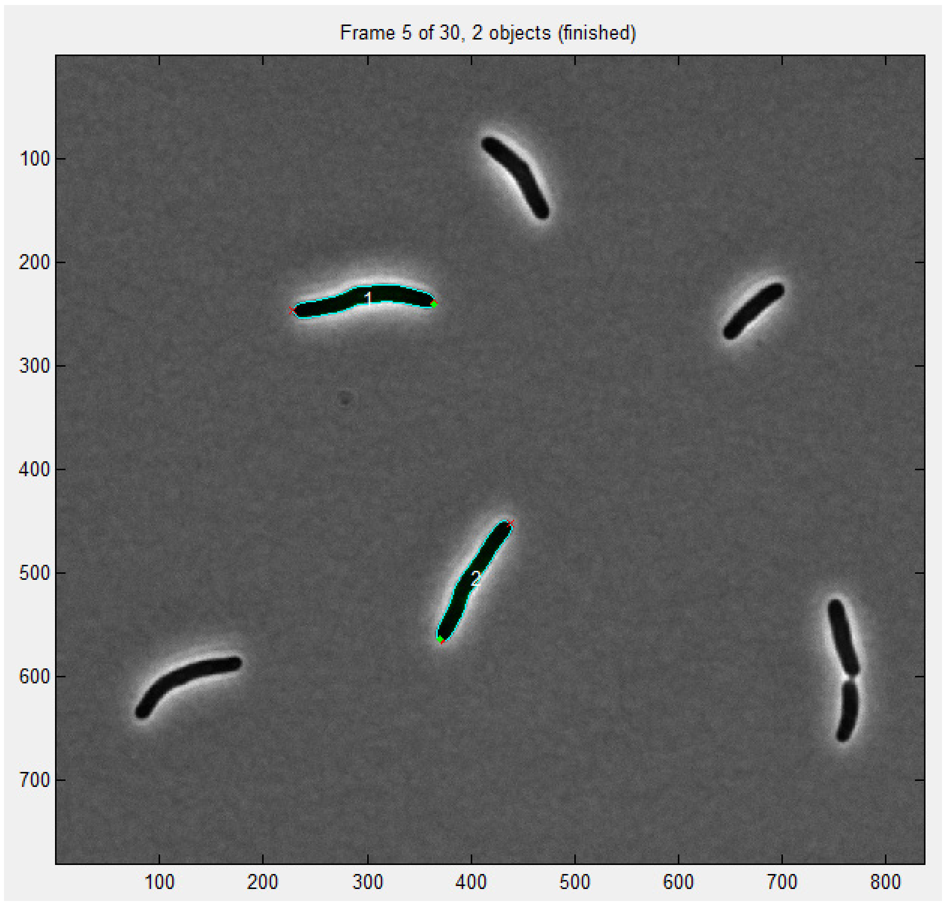

Select a boxed background region from maximum intensity projection of the stack.
100
200
300
400
500
600
700
800

This image is a 'maximum intensity projection' of all the images in the TIF stack; it shows in a single image everywhere that a cell touches during the stack as light and dark regions. Regions that remain background throughout the stack (i.e. regions into which no cells enter) retain the appearance of background (gray in this case). Use the mouse to drag and drop a rectangle that defines a region of background from which Morphometrics will calculate the mean and variance of the image background for each frame (an example rectangle is shown above in white). Once a background ROI has been selected, Morphometrics will use Threshold segmentation to find objects in the image. Like the first case, the segmentation finds seven objects.

Now check **Gradient segmentation**, press **Select freehand ROI**, and then use the mouse to draw a region around cells of interest – only these cells will be processed. An example might look like:



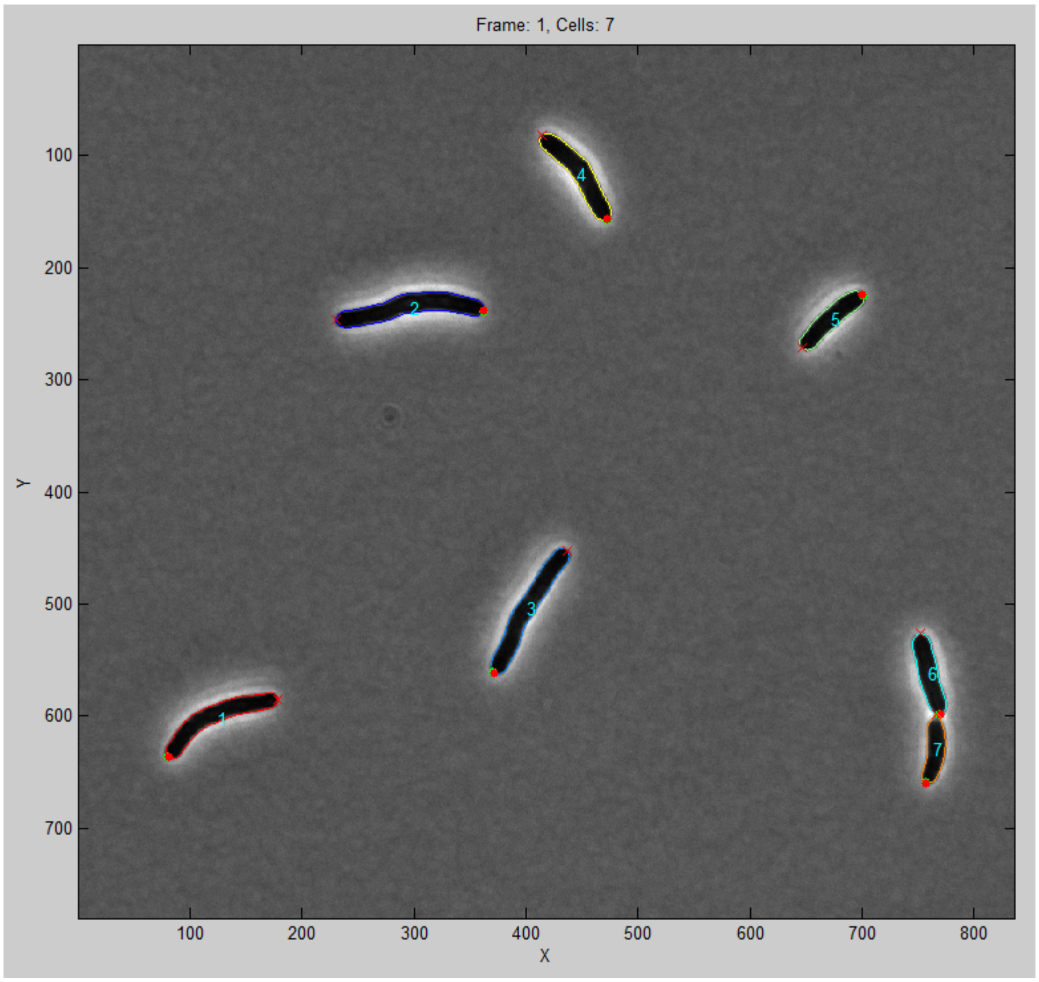
Then press **Test parameters** and note that now only objects enclosed by the ROI are processed, for the case above that would look like



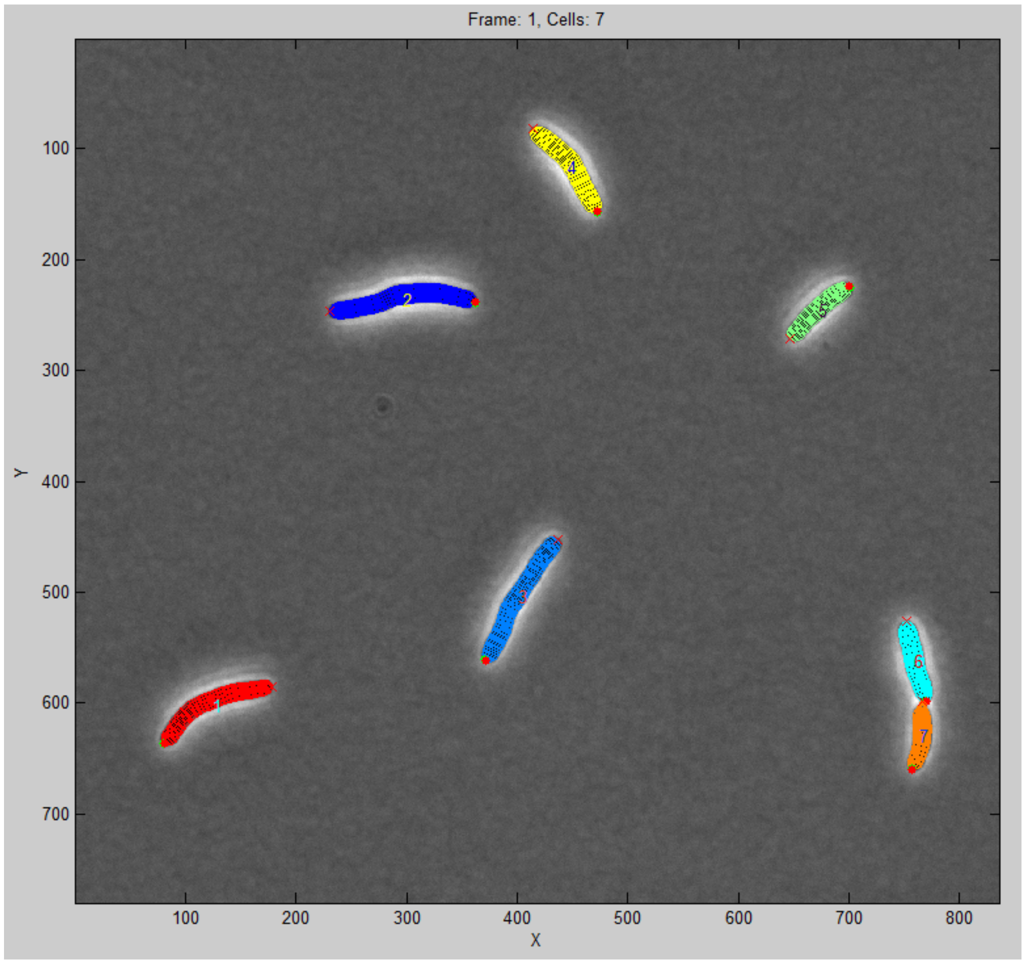
Now press **Delete ROI** to allow all objects to be found, and press **Run analysis**. Morphometrics will apply the segmentation algorithm to each frame in the TIF stack, then use those segmentations to determine which objects persistently appear from one frame to the next. Morphometrics will write all of the contour data to an output mat-file with the name `easy_test_images_Phase_<date>_CONTOURS.mat`.

Now check **pill mesh** in the **Meshing** pane, and press **Calculate meshes**. Select the mat-file that Morphometrics just created. The MATLAB command line will display the progress of the meshing algorithm, and when completed, Morphometrics will write a new mat-file that contains all of the original data plus all of the meshing data, with the name `easy_test_images_Phase_<date>_CONTOURS_pill_MESH.mat`.

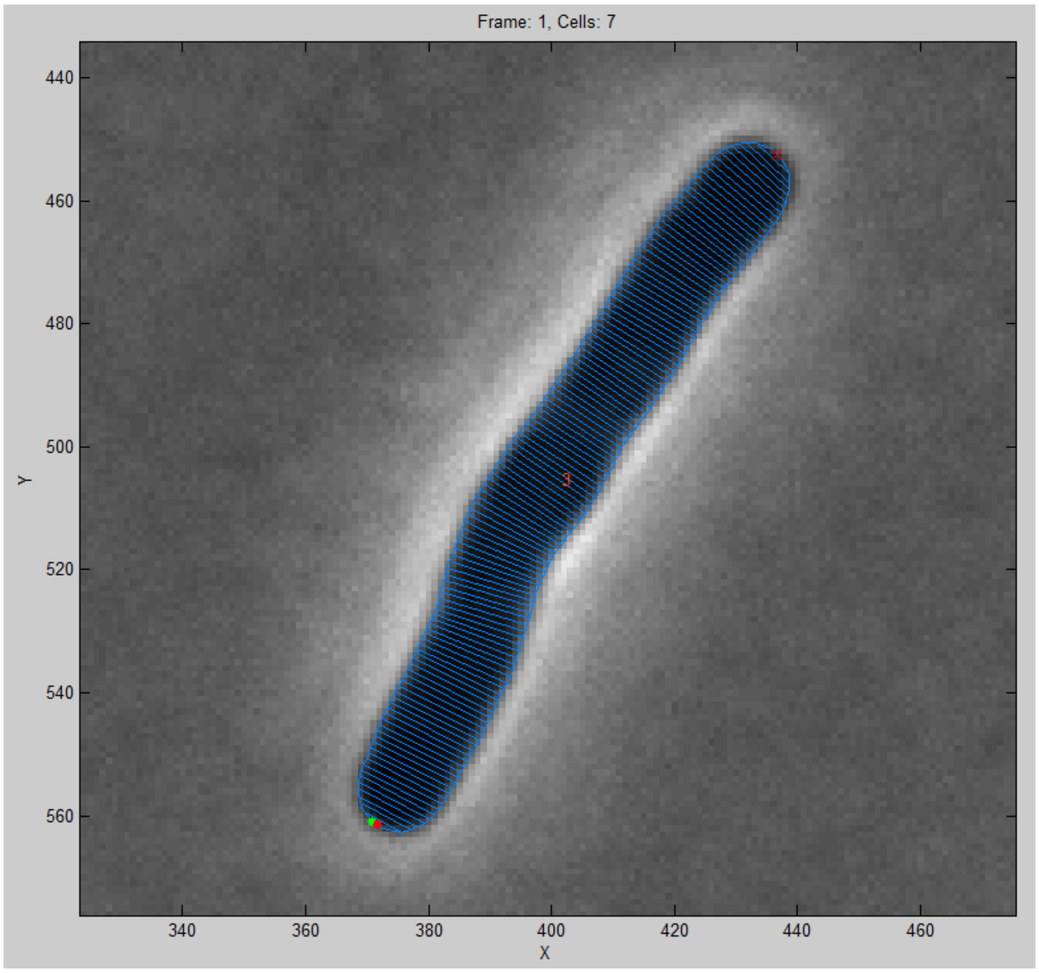
From the **Analysis** menu, select **View Contours**. Press **Load Data** and select the `easy_test_images_Phase.tif` stack and the `easy_test_images_Phase_<date>_CONTOURS_pill_MESH.mat` output file. Now select **uniquely color each cell, uniquely number each cell,** and **show cell poles** and press **View Frame**. This should produce the following image:



This shows each cell labeled with its unique cell ID, labeled poles, and each cell uniquely colored. Now check **show mesh** (note that this option only functions with mat-files that contain mesh data), and press **View Frame**, which produces this image:



Using the zoom tool in the MATLAB figure window, we can see the detailed mesh that Morphometrics created.



The figure window can be saved in various formats, including uncompressed TIF and vector graphic EPS, using the **File** menu, see the pre-saved file `easy_test_images_Phase_frame1_cell3.pdf`. Now press **Save Frame EPS**, which creates a vector graphics file directly from the specified visualization options for the frame number in **view frame**. A PDF version of this output from the example stack is given in `easy_test_images_Phase_02-Jul-2014_frame_1.pdf`. Finally, uncheck **show mesh** and check **save TIF stack**, make sure **progress via arrows** in unchecked, and then press **View All and Process**; this will display all of the frames sequentially in the MATLAB figure window, and will write each frame at full resolution to a TIF stack named `<date of this stack>_easy_test_images_<date of original contour processing>_CONTOURS_pill_MESH.tif`. You can view the included example output TIF stack in an image program like ImageJ (http://fiji.sc/Downloads).

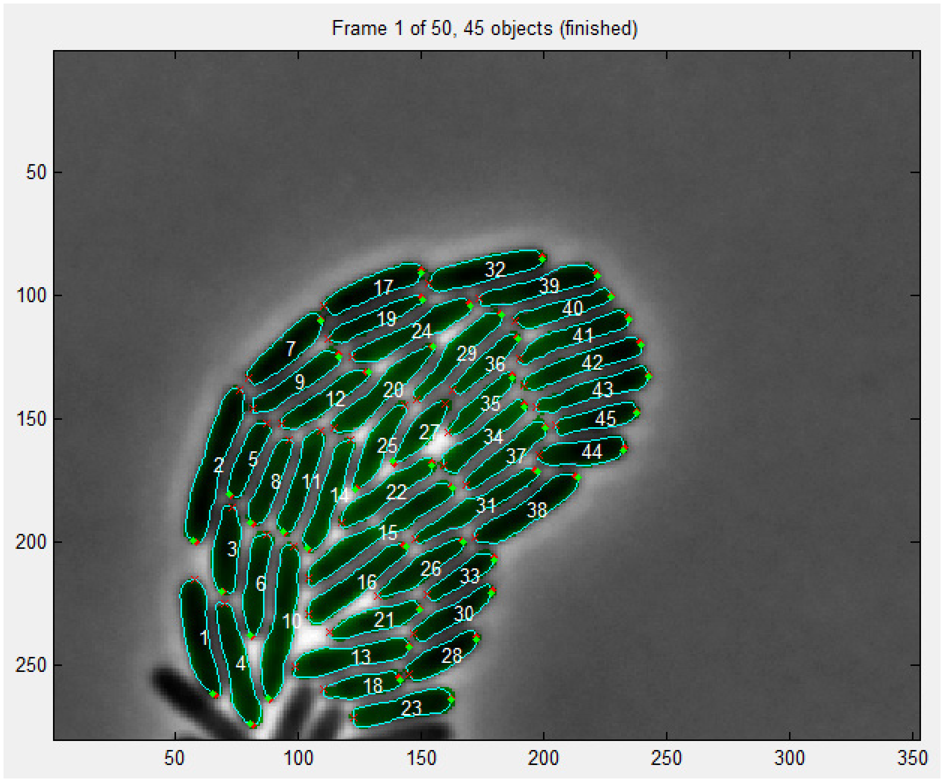
Dense Phase Data

Building on the detailed example above, we will explore a case where cells are closely packed in bright-field phase microscopy. We will not explore as many separate options, but this example should give you an idea for a different segmentation method and more advanced options.

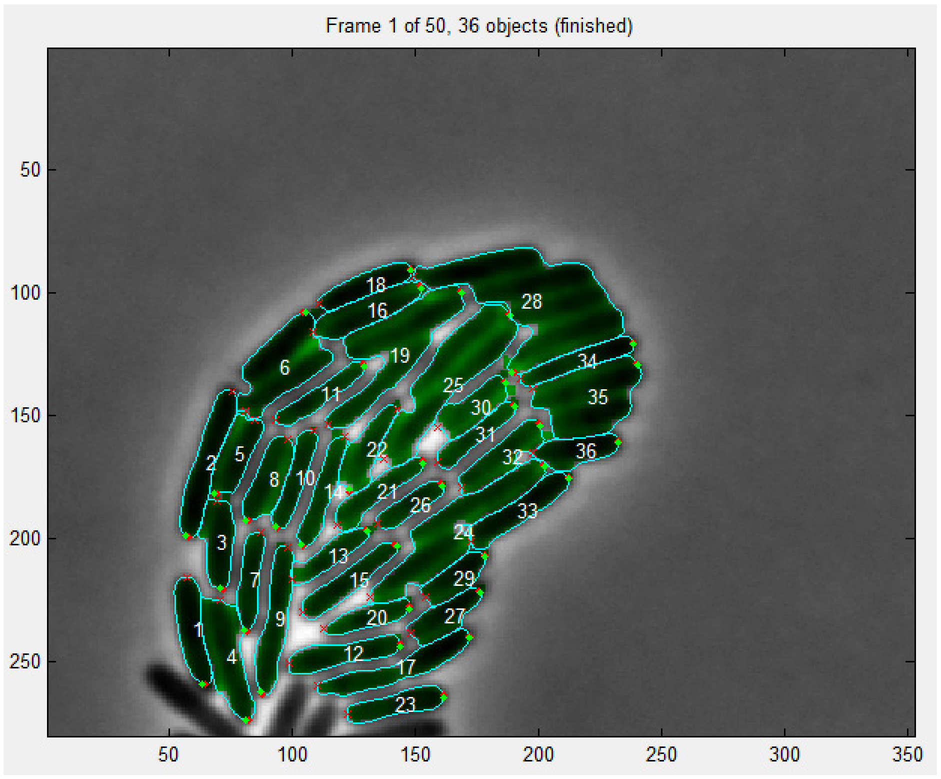
Press **Load parameters**, navigate to the folder `example_phase_data` and select the parameter mat-file called `Morphometrics_prefs_dense_phase.mat`. This sets all of the parameter values to values appropriate for this example. Note that this example operates in **Advanced mode**.

Press **Open single stack**, navigate to the folder `example_phase_data` and select the TIF image stack called `dense_test_images_Phase.tif`. This loads the stack into Morphometrics for further processing.

Laplacian segmentation often works well for densely packed cells and this is the method used in this example. Press Test parameters, and Morphometrics will display a segmentation of 45 cells from the group of cells in the image. Note that cells partially cut off by the image boundary are not segmented by Morphometrics.

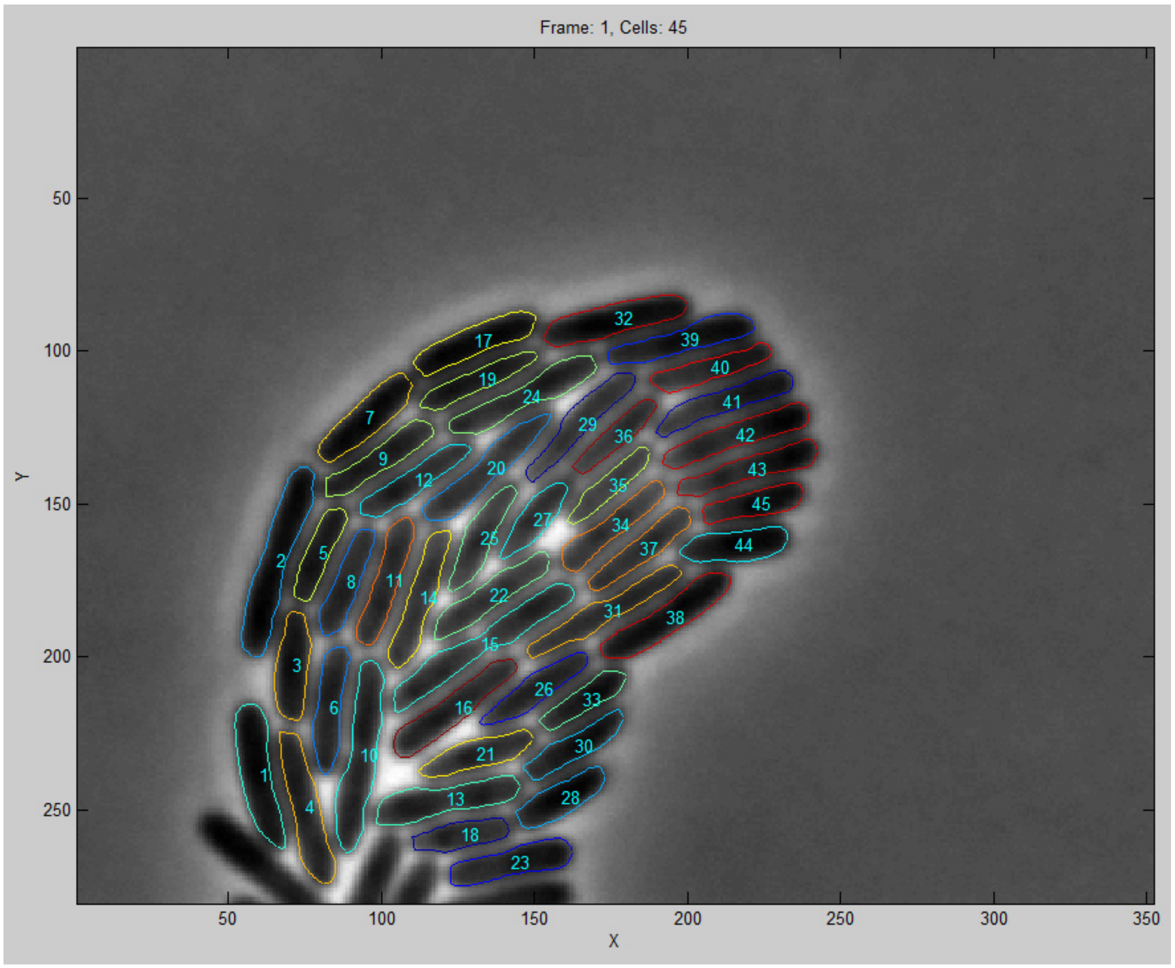


For comparison, check **Gradient segmentation** and press **Test parameters** – note that regions ostensibly composed of multiple cells are segmented instead of many individual cells.



Go back to **Laplacian segmentation** and then press **Run analysis**, this creates the output mat-file `dense_test_images_Phase_<date>_CONTOURS.mat`.

Under **Options**, select **View Contours**. Press **Load Data** and select `dense_test_images_Phase.tif` for the image stack and `dense_test_images_Phase_<date>_CONTOURS.mat` for the output mat-file. Now check **uniquely color each cell**, **uniquely number each cell**, and **progress via arrows**, and then press **View All and Process**. This will display the cells with unique colors and labels, and you can use the arrow keys (or mouse click) to move through the stack and view the contours; use the 'escape' key to exit. Below is what frame 1 should look like:



Now uncheck **progress via arrows**, check **save TIF stack.**, and press **View All and Process**; this saves an output TIF stack of the labeled cells with contours overlaid. This was done to generate the included stack <> in the `example_phase_data` folder.

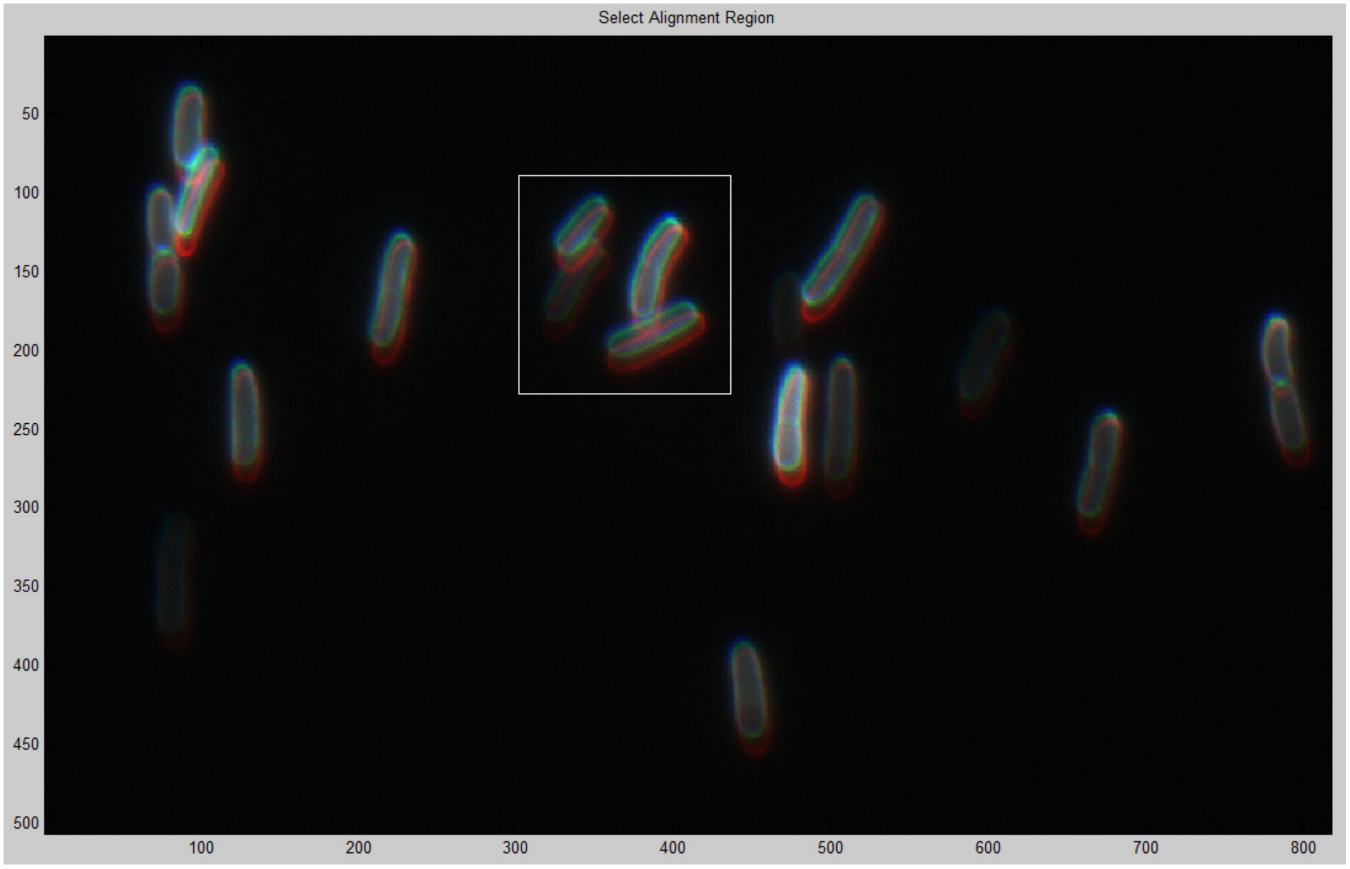
Try this example again, except now with **Seed contours** checked in the **Image Attributes** pane – this reduces the computational time, since only the first image is segmented and there is no object phylogeny calculated, but also means that no additional cells will be identified as the frame processing progresses. Note that as the contour fitting progresses, the contours tend to be misshapen as time progresses, because any errors in the contour relative to the cell shape are propagated through time. Seeding contours works best when the difference between cell position, rotation, and shape between frames is very small.

The included output mat-file `dense_test_images_Phase_02-Jul-2014_seeded_CONTOURS.mat` and accompanying output TIF stack `dense_test_images_Phase_02-Jul-2014_seeded_CONTOURS.tif`, were created using seeded contours.

Peripheral Fluorescence Data

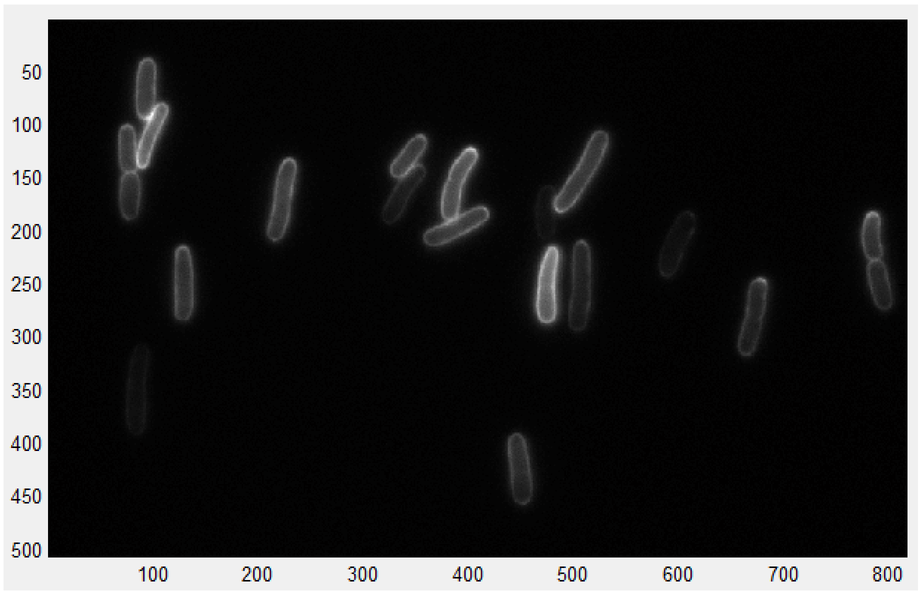
In this example we begin by aligning a small image stack that contains image drift. Under **Options** select **Align Images**. We will work with the file `easy_test_images_flr_periph.tif`. Since there is only one channel and one image file we could enter the same text for both **Channel motifs** and **Image motif**, but instead, enter `easy_test*.tif` in **Image motif** and `_flr_periph` in **Channel motifs**; the **Exclude motifs** do not need to be changed. After entering these motifs, the GUI will show one **Master** channel called **flrperiph**; click on this **Master** channel. Below the **Align Stacks** button, the GUI will indicate how many image files match the motif criteria.

Press **Align Stacks**, and select the master image stack, in this case `easy_test_images_flr_periph.tif`, from the `example_peripheral_data` folder. The GUI will then display a figure window showing the first (red), middle (green), and last (blue) frame in the stack. This allows the user to estimate the drift trajectory through the image stack. Using the mouse, select a rectangular region that contains alignment features, such as cells. The master alignment image and an example alignment region (white box) are shown below.

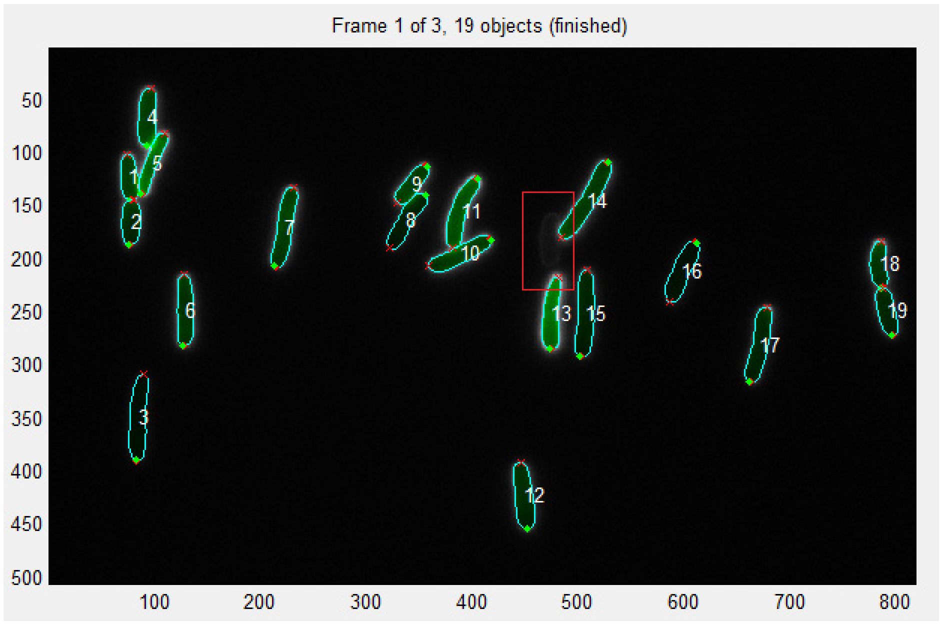


The GUI will then ask if you want to select a center-point, that is, to shift all images in the stack so that a particular point is at the center of image frame. In most cases, this is not necessary, select **No**. The alignment script will produce a new, aligned image stack with the name `easy_test_images_flr_periph_aligned.tif`; this aligned stack is now ready for contour analysis.

From Morphometrics, press **Load parameters**, navigate to the `example_peripheral_data` folder, and select the parameter file `Morphometrics_prefs_easy_peripheral.mat`, then press **Open single stack**, and select `easy_test_images_flr_periph_aligned.tif`. This displays the following image:

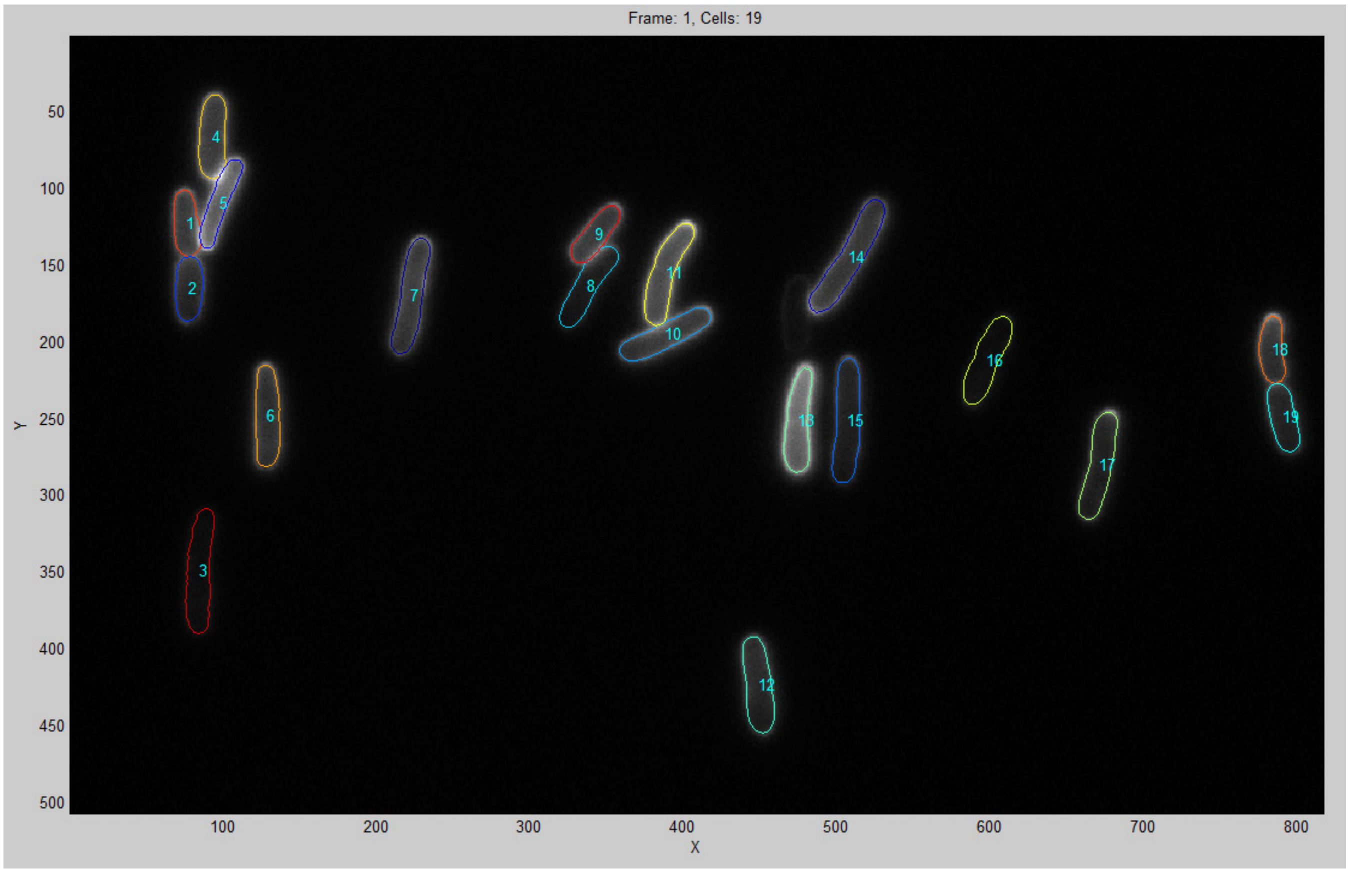


There are, depending on how one chooses to divide the cells, 20 cells; note the wide range of cellular intensities found in this image. For this example, we will **Seed contours** and use **Gradient segmentation**, and note that **Fluorescence (peripheral)** is selected. Press the **Test parameters** button; this should produce the image below. The segmentation algorithm misses one faint cell, herein highlighted in red by the author.

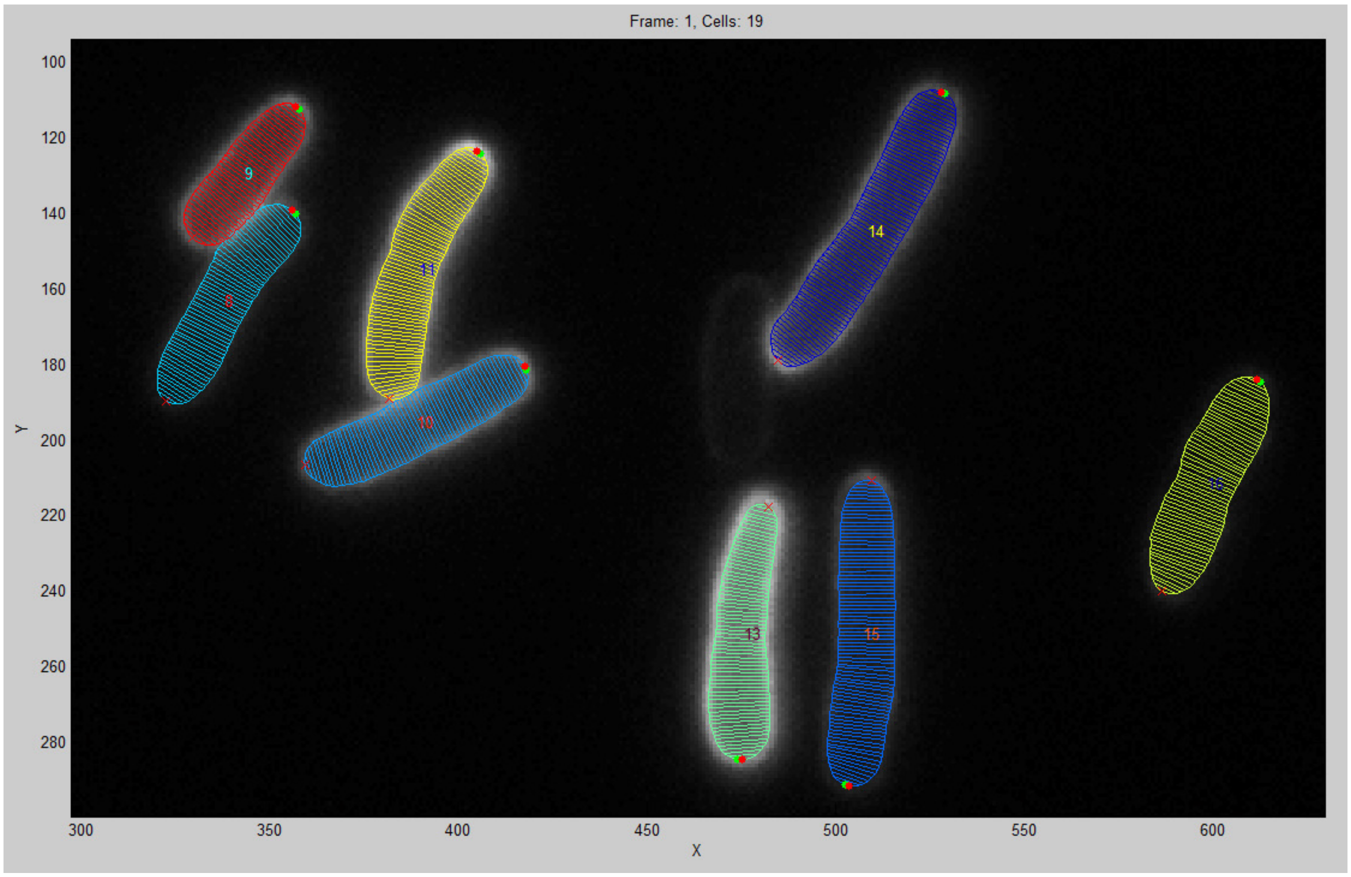


Press **Run analysis** and Morphometrics will save the output mat-file `easy_test_images_flr_periph_aligned_<date>_CONTOURS.mat`. In the **Meshing** pane, select **pill mesh**, then press **Calculate meshes**, and select the mat-file you just created. After the meshing algorithm has finished, a new mat-file with the name `easy_test_images_flr_periph_aligned_<date>_CONTOURS_pill_MESH.mat` will be created. Under the **Analysis** menu, select **View Contours**. Press **Load Data** and navigate to the mat-file you just created using **Calculate meshes**. Select **uniquely color each cell** and **uniquely number each cell** and press **View Frame**;

this should produce similar to the image below.



Now select **show mesh** and press **View Frame**, use the zoom tool in the figure window to see the internal cell meshes, your image should look similar to the one below.



Acknowledgments

Dr. Timothy Lee for significant code speed upgrades and improvements.
Dr. Carolina Tropini for help with the initial GUI.
Prof. K.C. Huang and members of the Huang Lab at Stanford University for discussion and feedback.
Dr. Benjamin Bratton for various code improvements and suggestions.

Legal Agreement

The program Morphometrics and all of its attendant scripts and functions are provided free of charge, and you may use, distribute and modify the program so long as you properly cite usage, and forward substantive changes to Tristan Ursell 'tsu@uoregon.edu'. This software is provided with no warranty or assurances of safety, functionality, correctness, or accuracy of any kind. You use this software completely at your own risk.