**File Format**

This code analyzes multi-channel images saved in the .tif format. Your image folders can come in two different flavors:

1. A separate .tif file for each slice and each color channel. In this case, each file name should end with some identifier for both the slice number and the channel. For example “imageZ02c3,” might indicate the second slice of an image in channel 3
2. A 3-channel .tif file for each slice. This is a common file format which the code will automatically separate into is constituent channels. In this case, just give each file name an identifier for the slice number, e.g. “ ImageZ03” would indicate the third slice in the image

In either case, you should put all your images in their own folder. You will have to point to this folder in the code in order to automatically open the images

**~~Running the 3D Code~~**

There are two main analysis scripts included in this package. **seg3D.m** and **seg2D\_DIC.m**, which are used for analyzing 3D and 2D images, respectively. A 3D image is one where the cells are randomly aligned in 3D and occupy many different z slices, rather than just cells lying flat on a 2D surface.

**Running the 3D Code**

Some lines you should change in seg3D.m

1. **Line 27**: Enter the location of the directory containing the image you wish to analyze. This code was written on a Mac. If you’re running it on a PC, you may need to change the direction of the slashes
2. **Line 9:** Change “pix\_size” to the size of your camera’s pixels, in microns. Our camera has 130 x 130 nm pixels, so “pix\_size” is set to .130
3. **Line 11:** “int\_thresh” represents the intensity threshold for the initial segmentation. The analysis is robust to this parameter. We recommend between .0001 and .000001.
4. **Line 13:** Adjust “ee\_thresh.” This parameter controls the how strict the shape analysis is for judging single cells. A higher value means more objects will be considered single cells. The default value is set at 1.5(1.22?), and we recommend between 1.25 and 1.75.
5. **Line 17**: “pix\_neigh” controls the neighborhood size for the local thresholding. It should be set to around half the length of the average cell. So we recommend setting “pix\_neigh” to between 8 and 12. Alternatively, you can let the code automatically set this parameter by uncommenting the line in line 17.
6. **Line 18:** “volume\_thresh” controls the minimum volume cutoff (in voxels) of a 3D cell. The default value is 500, and we recommend between 200 and 500
7. **Line 19:** “slice\_thresh” sets a cutoff for the minimum number of slices a 3D cell must occupy. The default value is 3, and we recommend between 3 and 5.

If you want to check the effect of these parameters without running the entire analysis, I would recommend the following.

* 1. Comment out (with a % sign) line 67, and uncomment line 68. You can set “g = “ to whatever slice number you want to check.
  2. Set a breakpoint at line 99 (by clicking on the dash at the left of the code). When the code stops, you can type in the console : imshow(objects) . This will show the codes initial attempt at segmenting the image. Don’t worry if cells are combined, the automatic splitting will take care of that. Just see that it got all the cells you would expect and didn’t pick up noise.
  3. If you are satisfied with the results, you can exit out of “debug mode” (click the red square in the top right of MATLAB), comment out line 68, uncomment line 67, and the code is ready to run.
  4. Now, run the code all the way through

**Running the 2D Code (seg2D\_DIC.m)**

To run the 2D analysis code, which is optimized for datasets where cells are lying flat on, for example, a coverslip, change the following lines:

* + 1. **Line 6 :** To automatically save your workspace after the analysis, you can change “filename” to whatever you would like.
    2. **Line 9 :** “ref\_channel” to whatever channel segmentation should be based on. This would be the channel which gives some consistent signal through the whole cell, rather than a spotty signal. **NOTE** : This code is optimized for segmenting based on the DIC image. If you choose to segment based on the DIC image (recommended), changing this line is unimportant
    3. **Line 10**, “ref\_slice,” to whatever slice is most in-focus. Same **NOTE** as in (2)
    4. **Line 14** : “int\_thresh” controls the intensity threshold for the initial segmentation. The code is robust to this parameter. I recommend just keeping it at the default value, .0001, but it can be changed up to .000001
    5. **Line 16 :** “ee\_thresh” controls the strictness of the single cell shape analysis. A higher value means objects will be considered single cells more easily. The default value is set at 1.5, but for DIC images with some noise, the value can be set as high as 3, perhaps at the expense of picking up some extra objects
    6. **Line 29** : “filepath” should point to the directory with your images
    7. **Line 30** : “filepath\_dic” should point directly to your DIC image, if you are segmenting based on the DIC Image

**Viewing your data**

1. To visualize your code, I have included a separate script called “cellDraw3D.m”. Simply run this code after the analysis is complete, and a 3D representation of the segmentation will appear.
2. The file titled “part4” in the workspace contains the data for the segmentation.
3. The file itself should be saved and the data can be exported to another program for further analysis.

Once everything is saved, the workspace should be cleared twice and the next image is ready for segmentation.

**Manual Fixation (manualFixation.m)**

After running your segmentation code of choice, you can check out the 3D results. If you are satisfied with the results, carry on about your business. If not, you can go slice-by-slice through the data and fix it by hand with the script “manualFixation.m”. To work the manual fixation:

1. Each slice will pop up with its current attempt at a 2D segmentation.
2. If you are satisfied, type “y”, then press enter. If not, type “n”, then press enter
3. If you typed “n”, another figure will pop up. The blue outlines represent selected objects. If you want to delete those objects, hold your mouse over one and press “d”. If you want to draw a new object, press “w”, then click, hold, and draw a new border of a cell, then release the mouse.
4. When you are done making corrections to that slice, press “enter”.
5. Repeat for the next slice

When you are done with the manual fixation, you will get the same output as the original segmentation code. part4 represents the data, and you can run “cellDraw3D.m” again to visualize the new 3D cells

**Real Data Parameter Extraction**

To extract parameter values for your own system, use “RealParameterExtraction.m” One example dataset is provided on the GitHub folder, called “cell1”. You will obviously need to provide your own data, which should be well-isolated, single cells.

**Note** : We expect our default parameters should be pretty robust for *E.*coli, and we suspect the same should hold true for *Salmonella*, so this step is optional.

1. **Line 7** : You will need to change “cell\_num” to whatever number of cells you have isolated
2. This script is a loop that will run through individual folders of single-cell z-stacks labeled, with each folder labeled “cell1” through “celln” for n folders. If you have a different naming scheme, change “filepath” appropriately in **Line 33**
3. Arrays of the seven input parameters (2D-Shape Error, Conc, Dcenter, 3D-Shape, Thetaz, Volume, and Slices) are output in the file “FinalStruct” in the workspace. You can make histograms of these values to figure out appropriate parameter values.