Segmentation code (beta-version)

Overview of folder contents:

The primary code for the segmentation gui is found in the SegmenterV1.m file.

To run type SegmenterV1 in the command line within this folder or anywhere once the folder and its contents have been added to your search path. See the link for more on search path

<http://www.mathworks.com/help/matlab/matlab_env/add-remove-or-reorder-folders-on-the-search-path.html>

To run an experiment from the CellaVista it is important to first sort the images into different channel folders using the FileSorterGUI function in the same folder. Type FileSorterGUI in the command line to run the GUI. It is important to run the function from the command line as oppose to double clicking the .fig file. Double clicking will pull up the GUI, but it will not run the initiation function in the GUI where some of the default variables are initiated. This is also true for SegmenterV1.

To use the FileSorterGUI use the top button to select the experimental directory where the images are stored. Fill in the other information. Cellavista's file output is absolutely insane as it puts all the channels and whole well images of each experiment in the same folder brilliantly labeled 1,2,3, and so on depending on the number of imaging sessions run. If there are two channels in the experiment every other image is one channel and the remainder is the other. If there were three channels used in the experiment, every third image belongs to the same channel and so on. To sort the images out the filesorter goes through and labels each image as a nuclear or cytoplasmic channel and then sorts them into the correct folders. To correctly sort out the nuclear label the filesorter needs to know what position the nuclear images are (I.e first of 2 channels). Check to make sure the nuclear channel is correctly selected.

Once the files have been sorted. You can run SegmenterV1 from the command line to pop up the GUI. It is important to have selected the rows and columns correctly or no image will be segmented! Also changing the cidre directory is important to not pull an error. Cidre is an image correction technique we have been using. Here is the link to the paper:

<http://www.nature.com/nmeth/journal/v12/n5/full/nmeth.3323.html>

If nothing seems to be happening check the command window for an error. I haven't had time to go through and put in the error statements for most of the common errors. Another day i'm afraid.

It helps to understand how GUIs pass/store information. Each widget (the control elements) has a callback function which is a function it executes when the user clicks/enters text. To find the callback function for each widget, right click on the SegmenterV1.fig file and select Open in GUIDE. Then right click the widget and select “View callback”. From the GUIDE interface you can edit the properties of all the widgets including default values, position, labeling ect.

To pass functions around the GUI the handles structure stores all the information for the current session. I use the handles function to pass information in between functions. It is not as elegant as a hard core programmer might want, but it works...

SegmenterV1 calls 3 functions. MuliChSegmenterV14GUI, MuliChSegmentNoParallel, and segmenterTestGUI. The first two are the same other than one uses the parfor() function in matlab to parallelize the segmentation and the other just uses a for() loop. Parfor() is roughly 4-6X faster (depending on the number of workers allocated) but can run into some problems if one of the segmentations goes poorly. SegmenterTestGUI is just the function to update the figure with the new segmentation parameters. All three call NaiveSegmentV2 which is where the actuall segmentation takes place.

Note it is important to look at images where you expect no cytoplasmic stain as if you are getting something crazy it will break the parfor loop(). Increase the noise disk will help and optionally I can make the threshold which identifies a blank image higher which should be line 73 in the NaiveSegmentV2 code. I have broken the code multiple times doing this!

The parforsaverGUI and Par functions are used to save the segmentation within a parallel loop. They are adapted from open-source code found at these two sites.

% http://www.mathworks.com/matlabcentral/fileexchange/32101-progress-monitor--progress-bar--that-works-with-parfor

% http://www.mathworks.com/matlabcentral/fileexchange/27472-partictoc/content/Par.m

Finally the BayesClassifier is still under construction. It is leftover from a previous graduate student who wrote the GUI to take segmented images and then allow for manual classification to build a bayesian classifier which is then applied to the whole data set. When it works it is pretty cool, but it is buggy still. The output of this will eventually feed into the tracking code where all the cells being tracked have been labeled as dying, dividing, or debris. Still working on that portion.

I tried commenting the code especially where there are lots of variable names. I know I have sinned egregiously to use variables such as temp and idx (and foo) in my code but I restrict such uses to largely unimportant variables.

If you want to play with the code outside of the GUI, (a good idea to start because GUIs tend to make it very difficult to access any of the data generated) I also included cellaVistaFileSorter\_withoutGUI and SegmenteWithout\_GUI to let you work through the code manually which I would strongly recommend. The GUIs are really nice for allowing easy visualization of different parameters, but make it nearly impossible to understand what is going on “under the hood.”

Next up is to tackle the tracking code and fit it into the output of the segmentation code.