How it works

Thursday, January 13, 2022 3:32 PM

General overview:

This algorithm works to segment and label solid fluorescent objects in the cell.

It does this by applying a threshold on each pixel's intensity to be classified as either part of an object or part of the background. It then applies a few filtering algorithms on the objects to fill holes and remove lone islands of objects of specified size. This algorithm is run twice for two levels of thresholds.

Detailed overview:

Step 1

The algorithm will be run in a loop over each image and looped over all samples. Once we load in cell masks, a matrix is created to hold the perimeters of the cell masks. This is for visualizing the clusters later.

Step 2

We need to have a single image matrix to apply the thresholds. For this end, we loop over each z-slice for this image. Inside of this z-slice loop we find the CV of all of the cells in the fluorescent channel of interest. This is so that we can find the best z-slice for each cell. Now we begin to construct our image by looping over each z-slice and all of the cells in the fluorescent channel of interest. If this slice holds the highest CV of a cell then we load in the fluorescent image of the cell to our single image matrix. On this single cell we apply a normalization where we convert the fluorescent values to a double and make the highest pixel value in the cell hold the value of 1 and the lowest pixel value in the cell hold the value of 0. This is so that our single threshold value can be nicely applied to each cell.

Step 3

Now that we have our normalized single fluorescent image, we can apply our thresholding on each pixel. The resulting logical matrices are then filtered by existing matlab algorithms. The first is a filling algorithm, imfill, to fill any holes in the thresholded objects. The second is an erode algorithm, imerode, which fills a disk with the smallest value in the area. This removes small islands of pixels. The last is a dilation algorithm, imdilate, which fills a disk with the highest value in the area. This should restore lost pixels from the previous erosion.

Step 4

The only thing left in the code is creating plots and storing information about the objects. The plots include a histogram of pixel intensities within a cell and reconstructions of the objects within the cells.

Step 5

The information about the objects are just found by looping over each high and low cluster. This will be covered in more detail in the next page since the 'algorithm' or process to do this is very straight forward.

Pseudocode:

Initialize_all_params (…);

```
Loop over thresholding values {

    Loop over samples {

        Loop over images within samples {

            Display time taken and current progress ();

            %% STEP 1
            LcFull = Load in cell segmentation (…);
            ALL_cellPerim = Get cell perimeter (LcFull);
            N = Get number of cells (LcFull);

            %% STEP 2
            all_cv = zero matrix (z_range rows, Cell number columns);
            max_normed = zero_matrix(dimensions(LcFull));
            max_fluor = zero_matrix(dimensions(LcFull));
            Loop over each z_slice {

                FluorImage = Load in fluor image (…);
                Loop over each cell {

                    all_cv(z_slice, cell number) = CV(FluorImage(current cell, current z_slice));
                } End loop over each cell
            } End Loop over each z_slice
            Loop over each z_slice {

                FluorImage = Load in fluor image (…);
                Loop over each cell {

                    If max_index(all_cv) == current z_slice {

                        max_normed = max_normed + normed(FluorImage * LcFull ==
                        current cell);
                        max_fluor = max_fluor + FluorImage * (LcFull == currentcell);
                    } End if best z_slice
                } End loop over each cell
            } End Loop over each z_slice

            %% STEP 3
            T_high_image = threshold(max_normed, high_threshold);
            T_low_image = threshold(max_normed, low_threshold);
            smooth_t_high_image = filters(T_high_image, disk(rad);
            smooth_t_low_image = filters(T_low_image, disk(rad);

            %% STEP 4
            Figure();
            Plot_histogram(max_normed);
            Fit_three_gaussians(max_normed);
            Figure();
            Reconstruct_base_image(LcFull);
            Reconstruct_unfiltered_normed_image(ALL_cellPerim, max_fluor);
            Reconstruct_filtered_normed_image(ALL_cellPerim, max_normed);
```

```
                Save_figures_and_close_them ();

                %% STEP 5
                Save_cluster_data (smooth_t_low_image);
                Save_cluster_data (smooth_t_high_image);

            } End loop over images within samples
        } End loop over samples
} End Loop over thresholding values
```