An automated cell-counting algorithm for counting both the dead (fluorescently-stained) cells and alive cells in a color image

Julius Wu

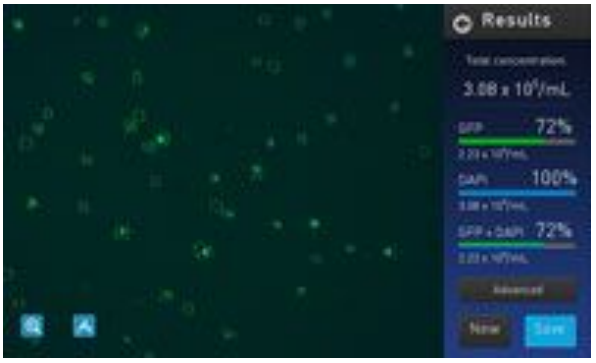
COSC 448A Final Report

Abstract:

A cell-counting algorithm was created to efficiently count both dead and alive cells from a smartphone taken photo. The algorithm works only if dead cells, alive cells and the background are different range of colors. This algorithm works great with counting 90% fo all alive cells. But only calculate 75% of the dead cells due to lighter dead cells blending with the background color.

Background / Introduction:

Traditionally, Scientists and Researchers have to manually count cells under a microscope with a manual hand tally counter. The manual cell counting method is time consuming and sometimes inaccurate due to miscount and/or double count. According to Al-Khazraji et al.'s "An automated cell-counting algorithm for fluorescently-stained cells in migration assays", manual counting method was tested and compared with a cell counting algorithm. 47 cells images was used and it took 2.4 hours to count by manually comparing to 14.5 seconds using an automated cell counting algorithm.



Although, there are lab graded cell counting equipments allowing researchers and scientists to count cells accurately and fast (< 10 seconds) for example, Thermo Fisher's "Fluorescence counting with color options". The equipment could be very expensive and costing upward of CAD 6,000.00 dollars. The goal was to have smartphone cameras and attracted to the Ocular lens of the microscope to take photos of the cells then process them with third party software to cut down expensive equipment's cost.

Our goal was to use Matlab's computer vision and image processing toolbox to build an automated cell- counting algorithm for counting both dead (fluorescently-stained) cells and alive cells from a smartphone's camera. From the sample photos given by Dr. Isaac Li, there are flaws and issues from the photos that needs to be tackled.

Proposed Method:

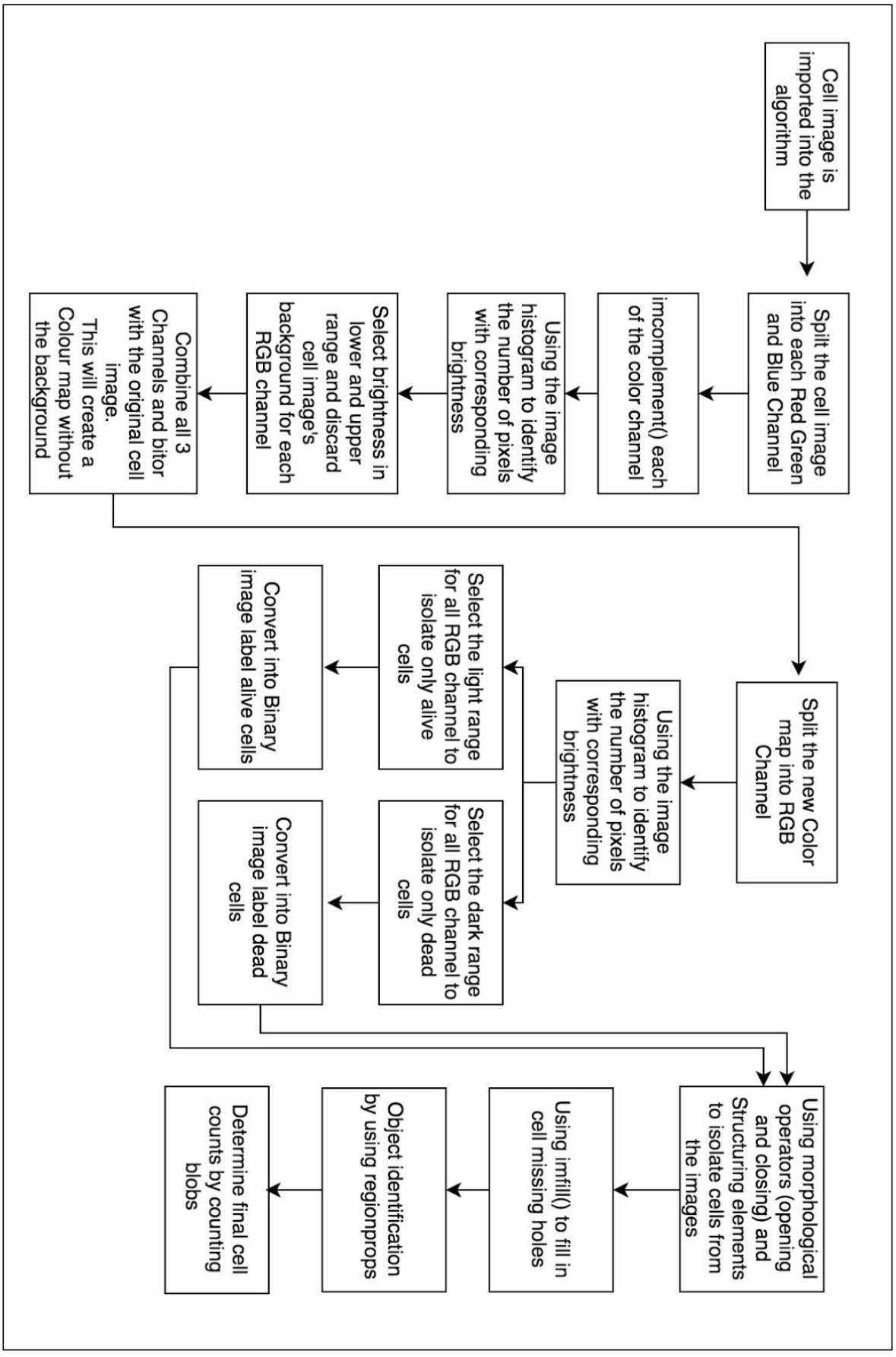


Figure 1) Flowchart outlining the main steps of the proposed cell counting method

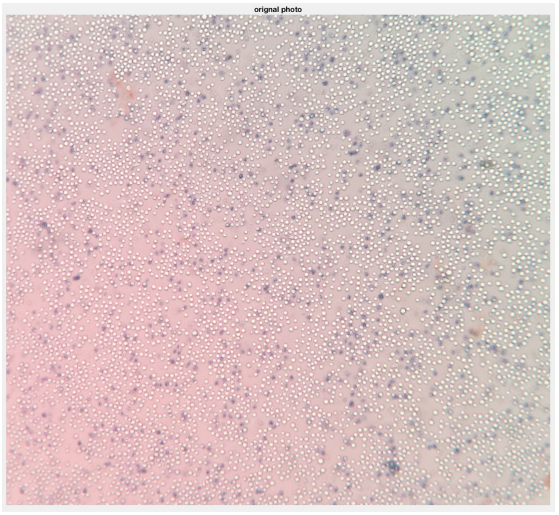


Figure 2) cropped image with adapthisteq() to enhance color balance.

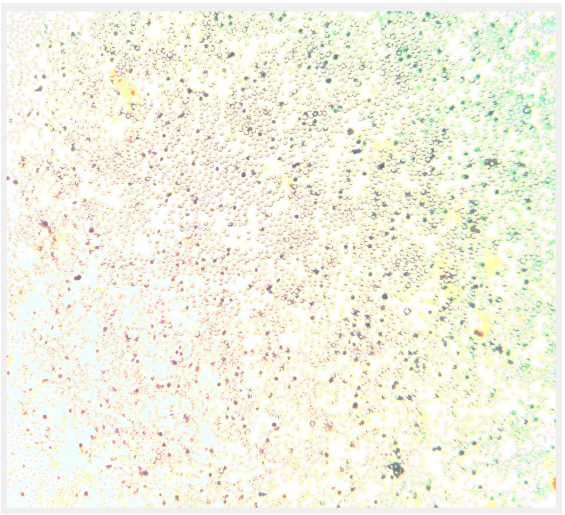


Figure 3) Color map image without the background

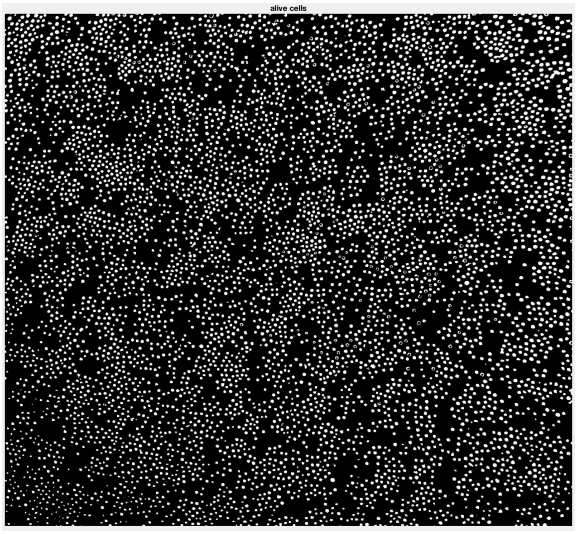


Figure 4) Binary image of all detected alive cells

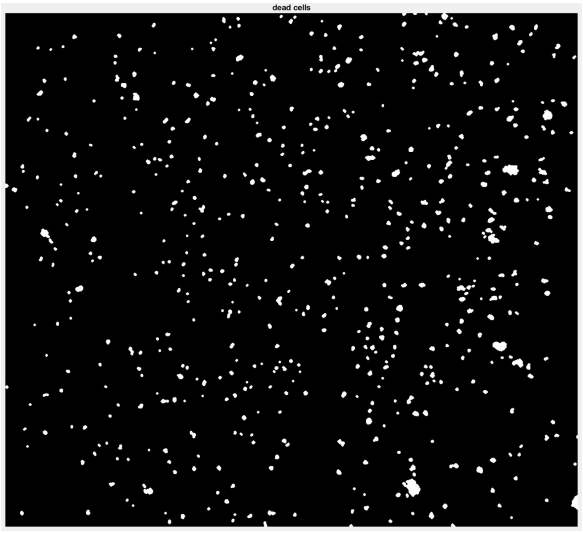


Figure 5) Binary image of all detected dead cells

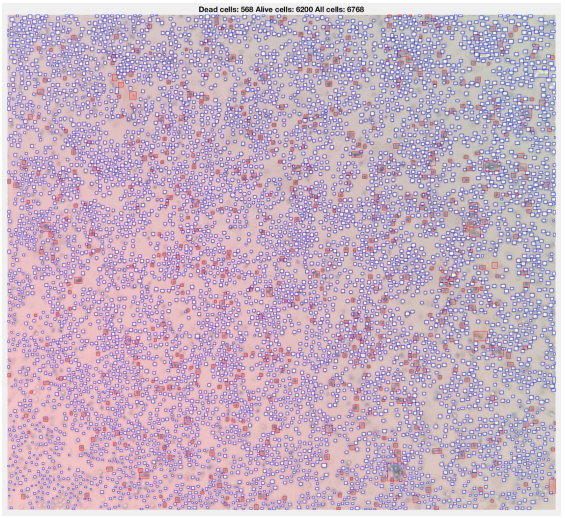
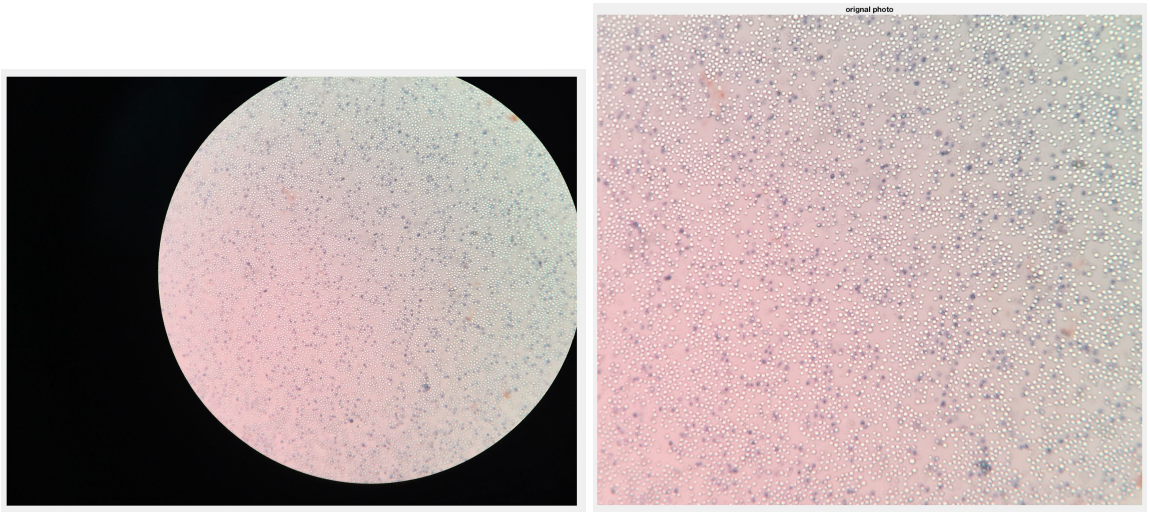
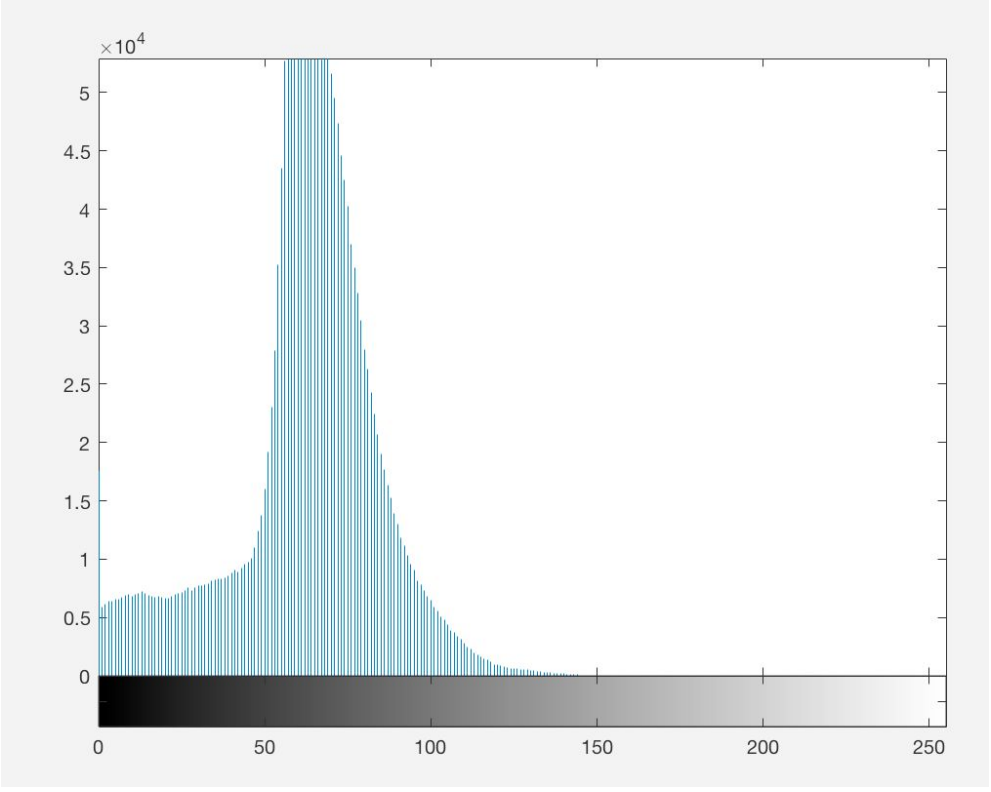


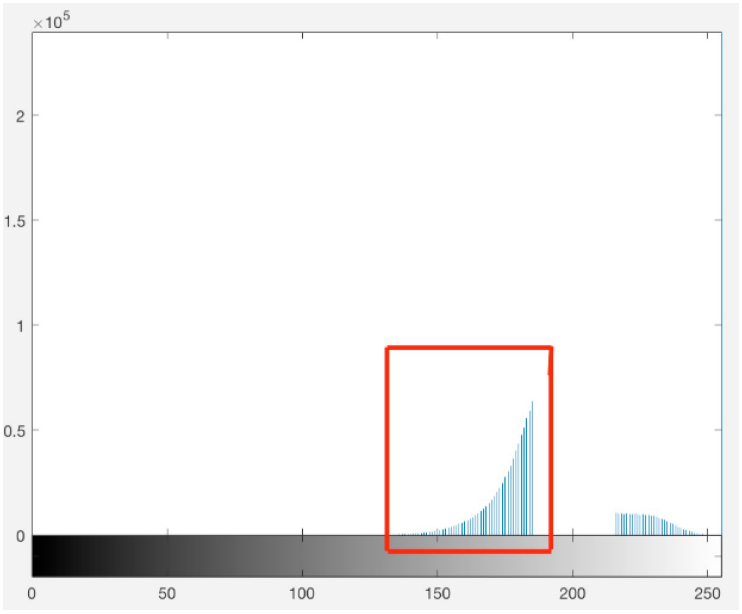
Figure 6) Final Image of the resulting cell counts "Dead cells: 568, Alive cells: 6200, All cells: 6768"



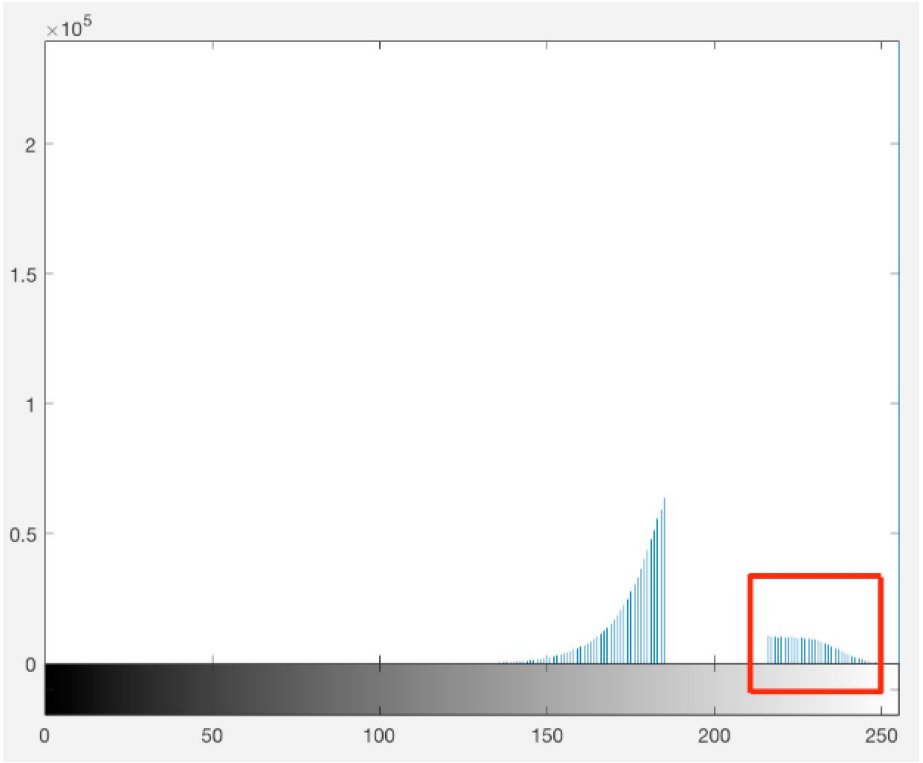
1) Pre-processing the image: Original image were read into the Matlab program. The algorithm asked the user to crop out selecting area for analyzation. Adapthisteq() method is used for each color channel to enhances the contrast of the image. Allowing the image to be sharper and clearer for analyzing and to fixed smartphone photo's color balance.



2) Removing image's background: Separate the cropped image into each Red Blue and Green channel. This allows the program to work with each of the color channel. To invert the color by using imcomplement() method which allows to product pixel histograms for each color channel using imhist() method. From each color channel select only the upper range and lower range of the alive and dead cells and discard the background. Using bitor() operator, invert all three of the image and Concatenate all 3 of the channels back into one image. This will create a color map image without the background color (figure 3).



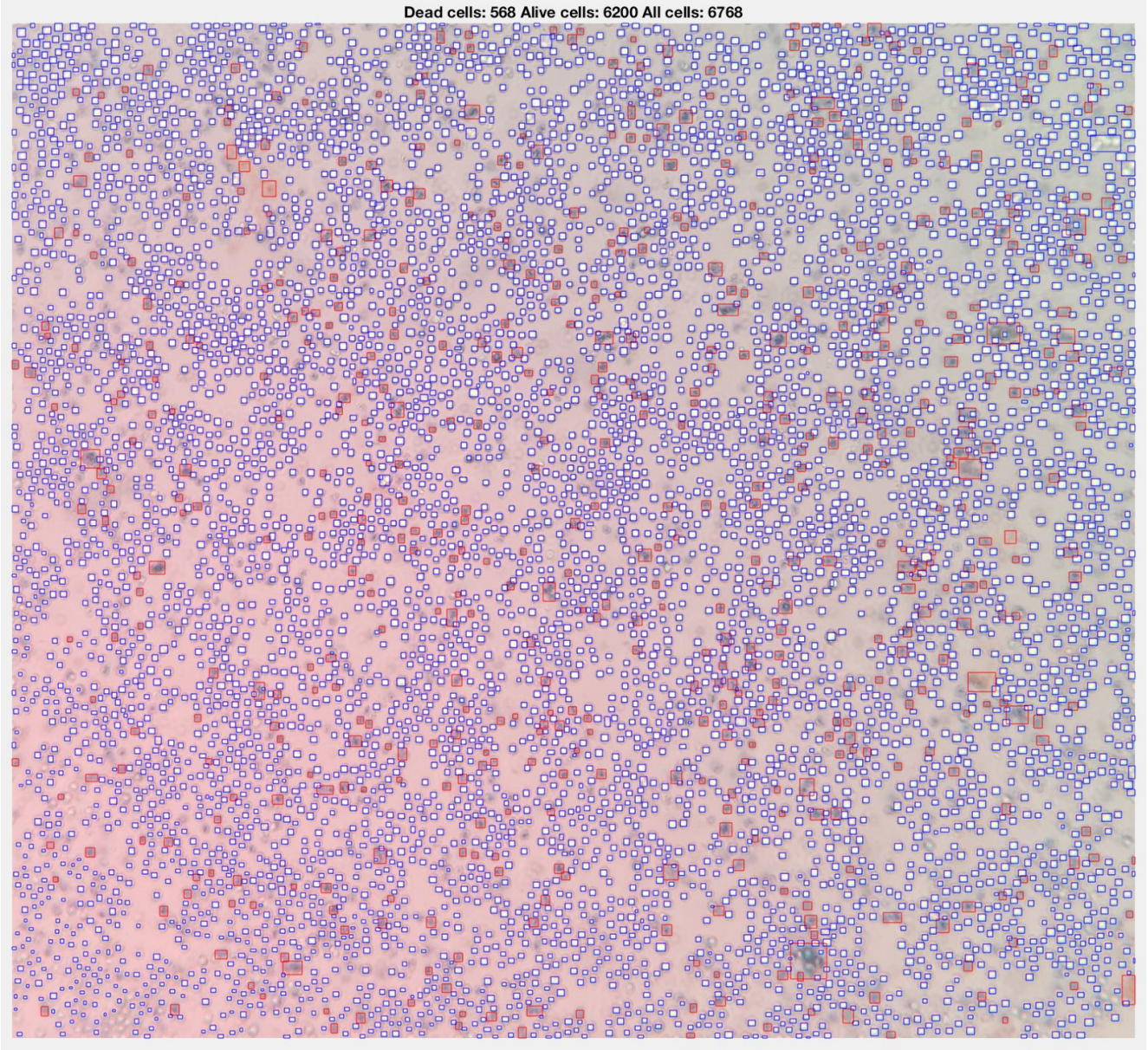
3) Identifying dead cell's color range: Once again, separate the color map image into each red, blue, green channel. Using imhist() to limit out the lower light range of the all three color channel. This will isolate only dead cells from the color map image. Create a binary image of the dead cells by using OR operators the channels. Figure 5 shows a binary image of all detected dead cells.



4) Identifying alive cell's color range: Using imhist() to limit out the upper light range of the all three color channel. This will isolate only the alive cells from the color map image.Create a binary image of the alvie cells by using OR operators the channels. Figure 4 shows a binary image of all detected alive cells.

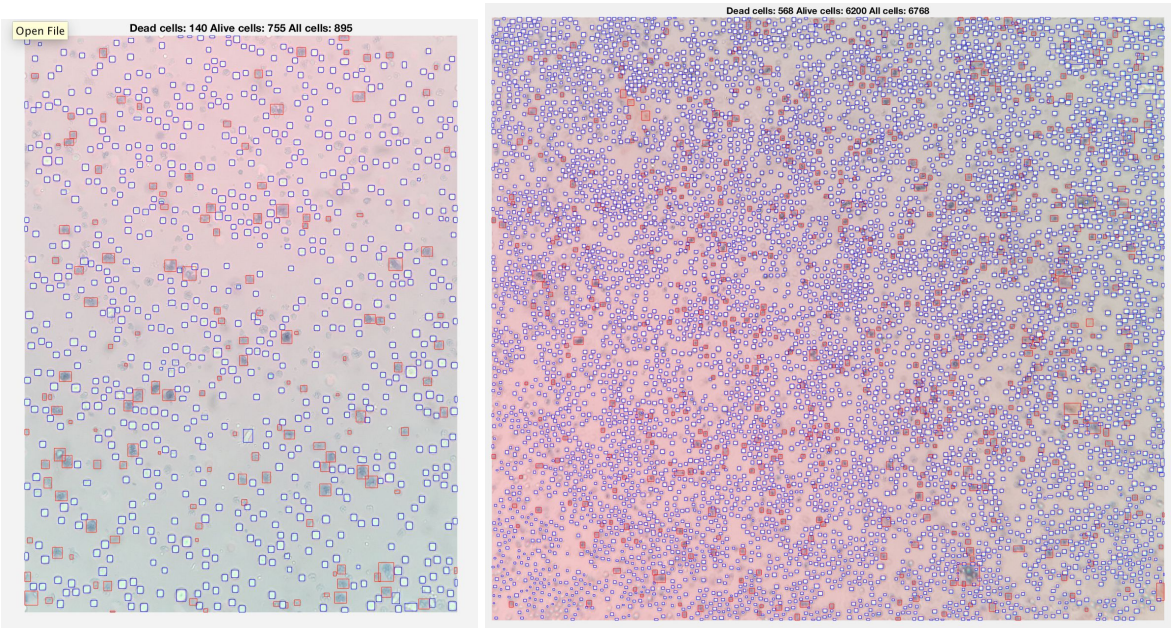
5) Using morphological operators: Morphological operators such as opening and closing are used for separation and identification of shapes and sizes. This allow to remove unnecessary noises and smudges.This also allow to identify overlapping cells to decrease miscounting. imfill() method is then used to fill in holes and gaps of the cells.

6) Object Identification using regionprops: Using regionprops() allowing to find the area of the alive and dead cells. This will help with identifying the cell's area and allowing for counting.



7) Determining final cell count: Using cat(num, blobs(:).Area) method to find area bigger to a set size. Using a for *loop and box and count all dead and alive cells and printed into the original cropped image. Figure 6 shows the final cell counts - "Dead cells: 568, Alive cells: 6200, All cells: 6768"*

Result:



This method was tested with the sample images provided by Dr Isaac Li. Overall the algorithm works great with timing. Avenging less than 10 seconds run time per cell photo. Comparing to manual count avenging 10 minute per cell photo. The algorithm was able to calculate almost all the alive cells (95%) in selected photos. We have issue, when alive cells and the background color is the same, the algorithm have trouble identifying the alive cells. When detecting dead cells, the algorithm have trouble to detecting lighter stain cells. The algorithm can detected 75% of all the dead cells. This is because some of the dead cells are very similar to the background color which the algorithm can't detect them.

Discussion:

It is very challenging to detect all of the dead and alive cells with a color gradient background. During the learning process, I have looked into converting images into grayscale and performing cell counting. When converting into black and white, we lost a lot of details and information about the photos. Gray Scale images tends work great if we are only detecting one type (dye or alive) of cells in the image.

The next step of the project is to automatic few processes including choosing the color light range and also allowing user to fix and edit cell counting from the result image. And also produce a histogram of all cell sizes.

Acknowledgment:
I would like to thank Dr. Isaac Li and Dr Abdallah Mohammed for the opportunity to work on this cell counting project and for helpful discussion and support during the directed study.

References:

D. Forsyth and J. Ponce, Computer vision, 2nd ed. Boston: Pearson, 2012.

R. Gonzalez and R. Woods, Digital image processing, 1st ed. Will be used alongside

Al-Khazraji et al. Biological Procedures Online 2011, 13:9, “An automated cell-counting algorithm for fluorescently stained cells in migration assays.” http//www.biologicalproduresonline.com/content/13/1/9 (19 October 2011)

https://www.thermofisher.com/ca/en/home/life-science/cell-analysis/cell-analysis-instruments/automated-cell-counters.html

Fuyong Xing, Lin Yang, "Robust Nucleus/Cell Detection and Segmentation in Digital Pathology and Microscopy Images: A Comprehensive Review", *Biomedical Engineering IEEE Reviews in*, vol. 9, pp. 234-263, 2016, ISSN 1937-3333.

Peter Markowsky, Svenja Reith, Tabea E. Zuber, Rainer König, Karl Rohr, Christoph Schnörr, "Segmentation of cell structures using Model-Based Set Covering with iterative reweighting", *Biomedical Imaging (ISBI 2017) 2017 IEEE 14th International Symposium on*, pp. 392-396, 2017, ISSN 1945-8452.