

Differential Leukocyte Counting via Fluorescent Detection and Image Processing on a Centrifugal Microfluidic Platform

Max L. Balter, Alvin I. Chen, Christa Amara, Alexander Gorshkov, Brian Bixon, Vincent Martin, Alexander Fromholtz, Timothy J. Maguire, and Martin L. Yarmush

The image processing routines (i.e., *mergelImages.m*, *sort_nat.m*, *segmentLym_Mon.m*, and *segmentGran.m*) were written using MATLAB 2015b and require the installation of the Image Processing Toolbox to function. Acquired images (and example images) can be placed in either the same or a different folder. If located in a different folder, be sure to change the file paths in the code, otherwise, errors will occur.

Image Acquisition and Image Loading

In this study, green and red channel images were acquired length-wise along the capillary tube and stored in separate folders. These programs were written under the assumption that multiple images were acquired, however, the code also works for single images. For example, if one image fully captures the buffy coat region, then only one image is needed for processing. Furthermore, the green and red channel images can be placed in the same folder, but again, the file paths need to be updated in the code before running. All images stored in a specific folder can be processed during one run of the code. This is accomplished with a for loop, iterating through each image via the *imread* function.

Grayscale conversion

A monochrome camera was used for this study, in which captured images were represented as 8-bit grayscale images. If color images are acquired instead, the code can still run; the images just need to be converted to grayscale before processing. This can be added to the code itself, or a separate pre-processing routine can be created using the [*rgb2gray*](#) function.

Merging Green and Red Channel Images (*mergelImages.m*)

This code allows the user to input the trial number (*s*) and the number of image sets acquired (*v*). In our case, four image sets were acquired: one for each tube orientation, rotating the tube 90° for each set. This code also operates under the assumption that each image set, for both the green and red channel, is stored in a separate folder (*sort_nat.m* sorts images based on the natural order of a **cell** array of strings).

The [*montage*](#) function is used to separately stitch green and red channel images together, where the length of the montage corresponds to the number of images stored in the respective folder. Once individual montages are created for the green and red channel, the images are normalized to the maximum intensity value to correct for lighting differences across image sets. Finally, the images are fused together using the [*imfuse*](#) function to obtain a composite. Intensity values from the green image are colored green, and intensity values from the red image are colored red. At this point, the code saves images in a results folder ('2-Processed Images') before moving on to the following processing steps.

Image Processing, Segmentation, and Quantification of Buffy Coat

A number of processing steps are performed on the images before segmentation. Each image is processed and segmented sequentially based on the order specified by the filename strings.

- Green Channel Images (*segmentLym Mon.m*):
 1. **Crop Images:** By means of the *imcrop* function, the user is prompted to manually select the region of interest (i.e., the approximate location of the buffy coat). An interactive rectangle object appears on the image which can be dragged and/or moved until the desired area is selected. The user then must either double click the area or press enter to execute the image crop and proceed with the code.
 2. **Linear Contrast Stretching:** The function *imadjust* is used to adjust pixel intensity values for a more even distribution.
 3. **2-D Order-statistic Filter:** A 3x3 minimum *filter* is applied to the image using a for loop with three iterations. The filter size and number of loop iterations can be adjusted by the user as needed.
 4. **Segmentation:** A global threshold is applied to the image based on Otsu's method, using the function *graythresh* to automatically compute a threshold level. This level is selected to minimize intra-class variance between black and white pixels. At this point, the image is converted from grayscale to black and white, in which the white region indicates the lym/mon region of the buffy coat.
 5. **Morphological Operations:** Holes in the segmented region are filled using the function *imfill*, and small pixels are removed via the *imopen* function.
 6. **Identify Connected Objects:** Using the *bwconncomp* function, connected objects are located and analyzed for outlier segmented regions (e.g., abnormally large or small pixel areas away from the buffy coat region). The area of the segmented buffy coat region is quantified via the *regionprops* function with units in pixels.
 7. **Overlay Segmented Boundary:** A mask is created from the binary image and the region is then overlayed over the original grayscale image. Data from the overlay boundary, including area, perimeter and location obtained from the *bwboundaries* function, is saved for use in processing the red channel images.
- Red Channel Images (*segmentGran.m*):
 1. **Image Masking:** Segmented boundary data from the green channel image processing is loaded and converted to a region mask using the *poly2mask* function. This is used to black out the lym/mon region so this area won't interfere with segmentation of the granulocytes in the red channel images. Moreover, all regions above the buffy coat (i.e., platelets and **plasma**) are also masked out with black pixels.

2. **Median Filter:** The function *medfilt2* is applied to the image to reduce noise.
3. **2-D Order-statistic Filter:** Similar to the processing step for the green channel images, a statistic filter is also applied here. But in this case, a smaller 2x2 filter is used instead to better preserve pixel information.
4. **Segmentation:** Again, a global threshold is applied based on Otsu's method.
5. **Morphological Operations:** Holes in the segmented region are filled and small pixels are removed. In this case, a smaller disk object is used for cleaning operations to preserve segmented regions that drifted away from the buffy coat.
6. **Identify Connected Objects:** Connected objects are located and analyzed for outlier regions and the segmented buffy coat region is measured.
7. **Overlay Segmented Boundary:** A mask is finally created from the binary image and the region is overlaid over the original grayscale red channel image.

Buffy coat measurements are displayed at the end of *segmentLym_Mon.m* (greenBuffyArea = lymphocyte/monocyte area) and *segmentGran.m* (redBuffyArea = granulocyte area). These area measurements can then be collected for generating a standard curve, as demonstrated in this study, or used to reference a standard curve to output diagnostic leukocyte counts.

Image processing and segmentation steps for the acrylic chip data is identical to that previously described for the green channel images. However, in the case of the red channel images, pixel intensity values were summed after masking out the lym/mon buffy coat region.

Test Images

A series of green and red channel test images (stored in the '1-Test Images' folder) are included in the archive. Note: each set is stored in a separate folder ('Green' and 'Red'). The respective images are sequentially numbered based on the order they were acquired during experiments. For example, the first image file is: 'image1.png'. These images can be used to verify the included code and evaluate other segmentation routines.

Also included are results from *mergeImages.m* (stored in the '2-Processed Images' folder), *segmentLym_Mon.m*, and *segmentGran.m* (stored in the '3-Segmented Images' folder).

Contact

If you have any questions about the image processing algorithms discussed, please contact Max Balter at balterm53@gmail.com