Segmentation of Confluent and Sub-Confluent Cell Cultures from Phase Contrast Images

By: Andrew (Jui-An) Kuo, Sohrab Roointan, Trong Shen

Abstract

Cell morphology is essential to biologists because it gives key information about cell physiology. Current methods available for extracting cell morphology are either not compatible with phase contrast images or with non-confluent images. We have developed a MATLAB program with DIP image toolbox which operates on confluent and non-confluent phase contrast images to output key cell morphological parameters including percent confluency, number of cells, cell boundary outlines, cell sizes, and cell orientation. The algorithm of our program involves extraction of cell-seeded area based on standard deviation, seed detection based on local minimum intensity, and watershed with seeds. We have evaluated the fidelity of our program by comparing with hand-drawn cell boundaries with less than 8% error in confluency, number of cells, and cell area. It also shows great compatibility with both confluent and non-confluent images. However, the key limitation is that standard deviation threshold and the gaussian filter standard deviation used which are essential to the accuracy for the program are determined empirically. One set of these optimized values are likely to be only valid for a specific optics on a specific cell type. Nonetheless, once the values are determined for all cell types and optics of interests, the program should be highly effective. We hope that our developed program can provide a simple but relatively accurate estimation of cell morphology to the biologists.

Introduction

Cell morphology is one of the key indicators of cell physiology. For instance, endothelial cells (ECs) elongate and align under flow shear stress, which drives them to an anti-atherosclerotic and anti-inflammatory phenotype [1]. Cell morphological studies often measure cell shape based on fluorescently labelled cytoskeletal elements or by manual tracing. Estimating cell shape from cytoskeletal elements requires additional immunofluorescence staining step and may not be accurate. On the other hand, manual cell segmentation is labour-intensive and not repeatable. Estimation of cell shape based on phase-contrast microscopy images has been used to study HBMEC morphology under shear [2]. The study reports that cell morphology obtained by phase contrast images showed comparable results to manual analysis of cell boundaries in immunofluorescence images. Therefore, phase contrast images present a direct and easy way to measure endothelial cell shape.

Cell segmentation approaches in the literature, although robust, are often needlessly complex for cell morphological studies like EC alignment, thereby increasing computational time and prevents biologist from using them. To this end, we have developed a cell segmentation algorithm that is computational simple, inexpensive and able to accurately segment cells in both confluent and sub-confluent cultures. Out segmentation algorithm can be broken down into four main steps: (1) determine confluency and cell denuded regions, (2) segment cells, (3) label and measure cells in cell populated regions, and finally (4) filter erroneous data. We evaluated the segmentation algorithm performance using two different cell lines; human umbilical vascular endothelial cells (HUVECs) and epithelial cells in different confluency level, orientation, and magnification. The segmentation algorithm was able to segment and measure cell shape to high accuracy. It can also account for variations in image quality and resolution by tuning a couple of parameters in the program.

Methods

We have developed a compilation of MATLAB scripts and functions with DIP Image toolbox to output cell morphology of images with various confluencies. Our cell morphology outputs include % confluency, number of cells, cell boundary, cell sizes, and cell orientation. The operation flow of our program (Figure 1) includes 7 functions and scripts. In this section, we will be explaining our program and its algorithm by referring to the operation flow.

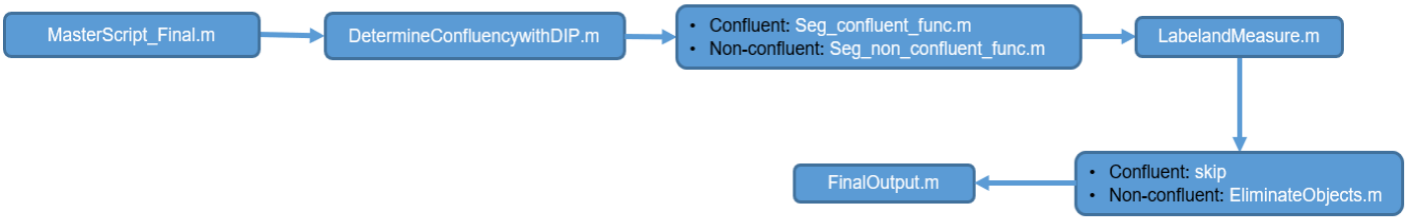


Figure 1: Operation flow of our program

The first script is *MasterScript_Final.m*. It initializes DIP image toolbox, reads the input image, convert to gray-scale if needed and calls all the other functions and scripts that follow. The first function that the master script calls is *DetermineConfluencywithDIP.m* which is based on published algorithm [3] and adapted to DIP image toolbox. It takes in an input gray-scale image and outputs % confluency and segments cell-denuded area against cell-seeded area. It operates by sweeping user-input small and large window sizes over the entire gray-scale image to calculate the standard deviations at each swept location and then applies a user-input threshold on the calculated image. If the standard deviation at a swept location is above the threshold, the window there is considered a cell-seeded area and if not, it is considered a denuded area. To ensure good capture of the image, the thresholded standard deviation image from small and large sweeping window were intersected to create a final segmented denuded image. The 100% minus the percentage of denuded area was then output as the variable *confluency* and the denuded area was output as *OutImage*.

After confluency is determined, the master script now calls a confluency-dependent segmentation function. If the confluency of the image is above 95%, the image will be considered confluent and *Seg_confluent_func.m* is called. If not, *Seg_non_confluent_func.m* will be called instead. The two functions are very similar in the algorithm in which both detect for cell seeds and segment cell boundaries using watershed. Both functions start with histogram equalization to enhance contrast, followed by median filter to remove speckles. Gaussian filter is then used to blur the gray-scale image to allow subsequent seed detection based on finding local minima in intensity values. The operation results in the dark spots of cell nuclei as the local minima and seeds. After detection of seeds, in the non-confluent function, seeds detected in the denuded area are removed, while the confluent function does not need this step. The seeds are then morphologically dilated and used for the subsequent watershedding with seeds on the gaussian-filtered gray-scale image. The final outputs for both functions are the segmented boundary *seg1* and the detected seeds *img_seed1*.

After segmentation of the boundaries, for both confluent and non-confluent input, the master script will call for the *LabelandMeasure.m* script. This script creates a label matrix for measuring desirable cell morphological outputs. First the script creates a binary image in which the segmented cell boundaries are 0s and pixels inside the cell boundaries are 1s. The label matrix is then obtained by labelling each binary object in the binary image. At this step, each binary object should be a cell. The measure function from DIP is then called to measure object size, orientation, and length of major and minor axes for every object. For non-confluent images, there is a slight over-estimation of the cell-seeded area, resulting in objects part of the denuded area being mislabeled as cells. To address this problem, a separate script *EliminateObjects.m* is called for non-confluent samples and eliminates them based on objects with too high (positive) or too low (negative) of a z score in length of major axis and pixel size. Finally, the master script calls *FinalOutput.m*, which outputs all the images obtained and cell morphological data including cell boundaries, cell length cell sizes, and cell orientation. The script also asks for micron to pixel ratio to convert the pixel sizes to microns.

Results

Our program is capable of generating all the previously mentioned cell morphological parameters for phase-contrast images, with variations in cell type and image quality, which will be later discussed. Figure 2

presents a sample of our algorithm's output for confluent and non-confluent cases, compared to the ground-truth (GT).

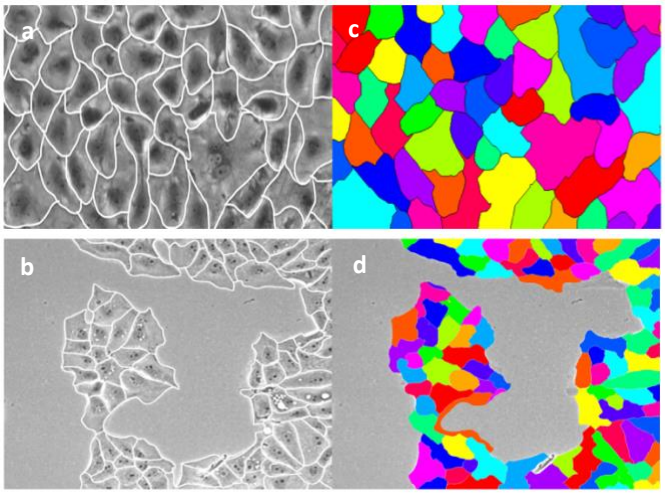


Table 1:

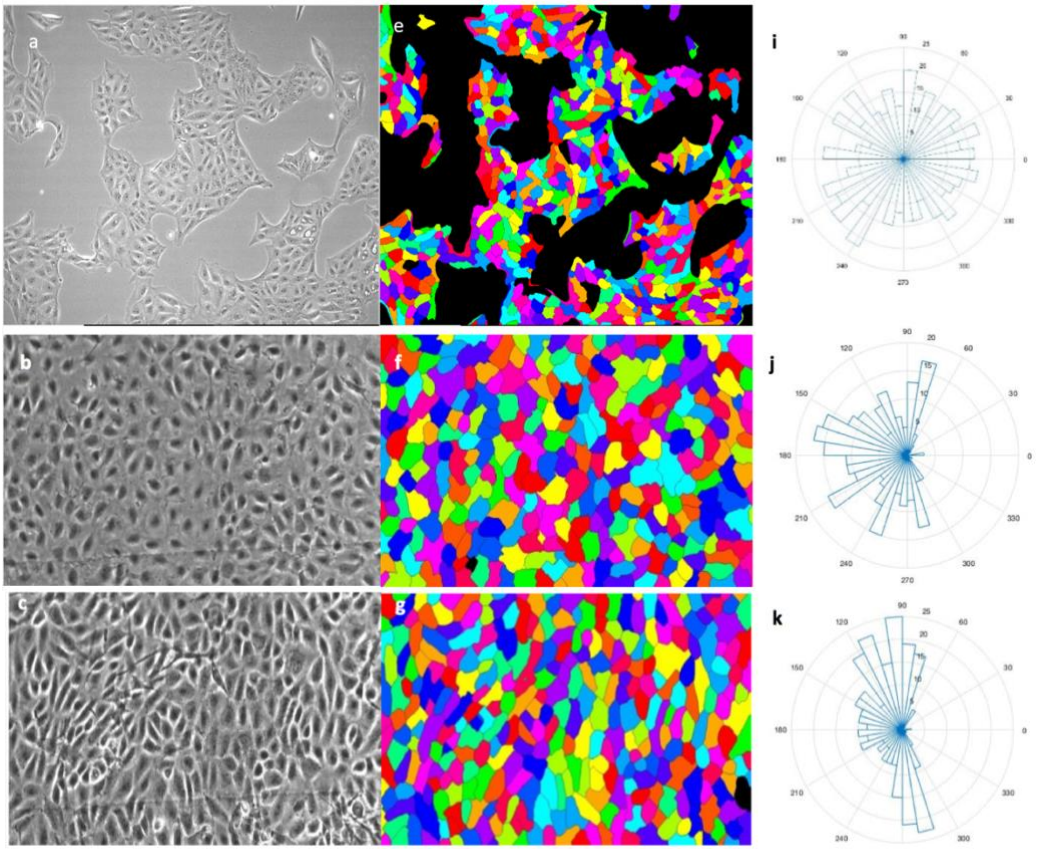
	Ground Truth	Segmented Result	% Error
Confluency	100%	100%	0
Num objects	68	64	5.88
Avg size (um^2)	1.38 +03	1.46e+3	5.69
Avg Orient. (deg)	20.6	26.78	30

Table 2:

	Ground Truth	Segmented Result	% Error
Confluency	56.94%	56.51%	0.76
Num objects	88	95	7.95
Avg size (um^2)	853.97	834.11	2.15
Avg Orient. (deg)	-16.99	-2.72	84

Figure 2: Comparison of the manually calculated ground-truth (a,b) and segmented results (c,d). The top row shows the results for the confluent case with results summarized in table 1. The bottom shows the segmented results for non-confluent case.

The images in figure 2 have been cropped from their full version (Figure 3, a and b), in order to facilitate the manual calculation of the GT. The fully segmented versions for these two cases along with the angle orientation histogram can be found in the first two rows of Figure 3, with the other measured morphological outputs in table 3. We also presents the segmentation and orientation results for two other images (c,d) with different polarities.



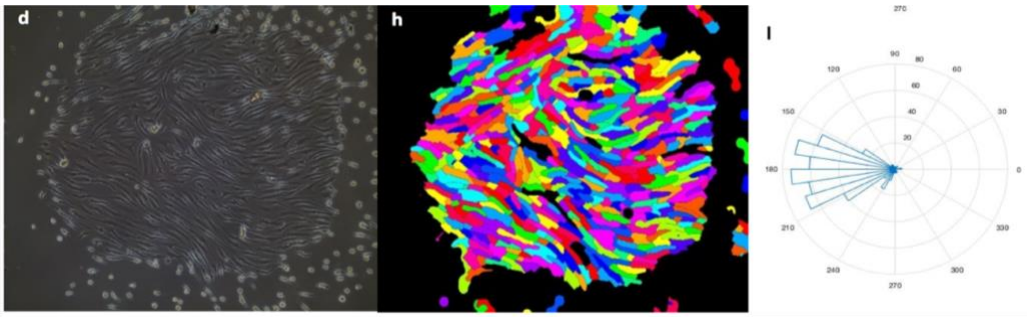


Figure 3: Segmented results and orientation angle distribution of Epithelial cell culture (a,e,i) and HUVEC culture (the rest) for (a,b) randomly oriented (c) aligned perpendicular and (d) aligned parallel cultures. Final segmented results are shown besides each phase contrast image and their respective orientation angle histograms are at the rightmost column.

	No. of Cells	Confluency (%)	Micron-Pixel Ratio (Input)	Mean Cell Size (μm²)
Confluent case	286	%100.0	0.625	1.4646e+03
Non-Confluent case	574	%62.1	0.43	834.1071

Table 3: Cell morphological output for nonconfluent case (figure 3a) and confluent case (figure 3b).

We tested our segmentation algorithm against images with different resolution, magnification, and contrast to understand which image quality measures can be used to adjust segmentation settings. We found that variations in image can be characterized by two variables: standard deviation of cell region pixels and cell density. They capture cell image quality and image resolution variations respectively. The optimal parameter values for each image condition were determined empirically and was used to form a guideline for parameter value selection. We found that segmentation results can be optimized in most cases by adjusting two parameters: threshold (in the confluency calculation function) and sigma (in the gaussian filter prior to seed detection). We plotted the optimized parameter values against their respective image quality measure (Figure 4).

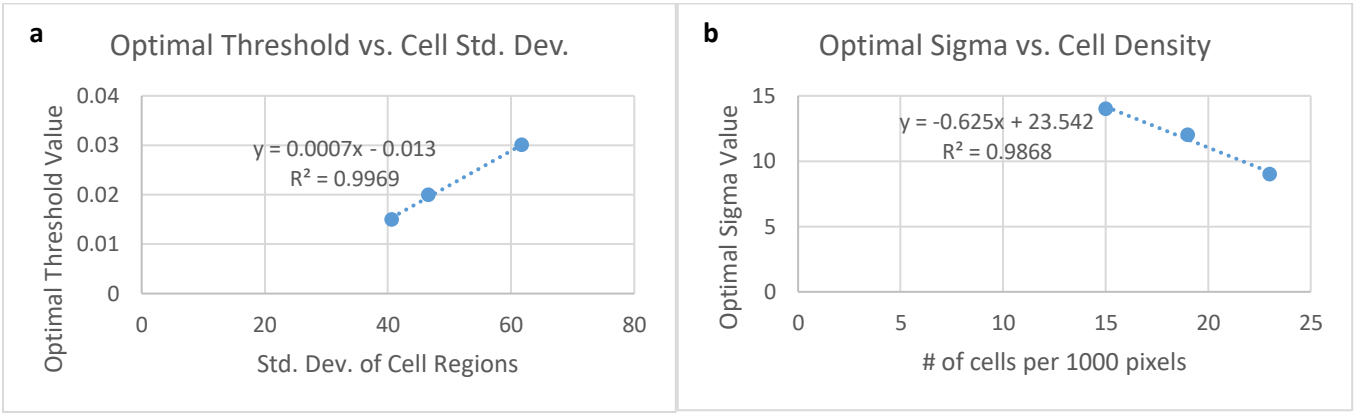


Figure 4: (a) Optimal threshold value plotted as a function of cell image quality and (b) optimal sigma value as a function of cell density. We took the standard deviation of cell regions *after applying histogram equalization* as a measure of threshold selection. Cell density was measured relative to the image resolution by counting the number of cells intersecting a 1000-pixel length line in a cell *confluent* region.

Discussion

As presented in figure 2, with optimal selection of the Guassian filter's sigma value and the confluency function's threshold value, our program can determine the confluency, number and size of cells, and cell boundaries in a phase-contrast image with minimal error compared to GT. Despite this, a large error exists for the case of cell orientation calculation. This is because orientation is calculated as the angle between the major axis (max feret diameter) and the global x axis of the image. Consequently, if the boundary differs even by a small amount, it will change the position of the major axis and thus alters the angle dramatically, resulting in the large reported error.

We have further characterized the function for selecting the optimal threshold and sigma values for given cell contrast and cell density. The plot in Figure 4 indicates a linear relationship exists between the parameters (threshold and sigma) and the corresponding image measures for cell contrast and density. Figure 4a shows that optimal threshold value is proportional to the standard deviation value of cell seeded regions. This follows the fact that *DetermineConfluencyDIP* function determines cell seeded regions based on the regions where the standard deviation is above a threshold. It follows that as the standard deviation of the cell seeded regions decrease, the threshold must decrease to maintain accurate confluency measures. Figure 4b shows an inverse relationship between optimal sigma value and the cell density. It should be noted that cell density is taken relative to the image resolution to account for resolution changes. For denser cell cultures, this decrease in sigma value is necessary to result in less blurring effect and allow for the subsequent local minima step to detect and localize cell seeds with more accuracy. However, since the segmentation performance is evaluated empirically, we cannot quantify the error in segmented results as the parameter values deviate from this relationship. One conclusion we can make from this relationship is the limit of our segmentation algorithm. By taking the threshold value to zero, we get the minimum standard deviation of cell regions required after histogram equalization to equal 18.6. Similarly, by taking the sigma value to zero, we get the maximum cell count per 1000 pixels in cell confluent regions to equal 37.7. However, the segmentation results will most likely break down before reaching these limits.

Despite having obtained an optimal sigma value for gaussian filtering, the local minima for seed detection can be erroneous. The use of local minima for detecting seeds is based on the assumption that holes (dark spots) are nuclei. However, a closer examination of figure 3 (a) shows that holes are most likely nucleoli instead of the nuclei depending on the optics during image acquisition. Therefore, the locations of the seeds are not very accurate which can affect watershed results. In addition, many cells undergoing cell division can have two nuclei. This leads to extra seed detection and potentially over-segmentation with watershed. Another key limitation is local minima's high sensitivity. It detects any local intensity minima that could be cell debris, bacteria, or small aggregates of proteins instead of the nucleoli. A practical improvement method is to stain the nuclei and use the stained nuclei as seeds to increase accuracy of seed detection and the subsequent watershed.

Conclusion

We have successfully developed an image analysis program in MATLAB with DIP image toolbox which measures cell morphological parameters on confluent and non-confluent images. Our program has proven to have high fidelity compared to ground truth from hand-drawn cell boundaries. However, the key limitation of our program exists in the selection of standard deviation threshold and sigma value to optimize segmentation results for different cell image quality and cell density/resolution. Despite this limitation, upon optimizing through the relationship described, the program should operate at high efficiency and fidelity. We hope this program can provide a fast, simple, and convenient estimate of cell morphology for biologists.

References

[1] Y. S. Chatzizisis, A. U. Coskun, M. Jonas, E. R. Edelman, C. L. .Feldman and P. H. Stone., "Role of Endothelial Shear Stress in the Natural History of Coronary Atherosclerosis and Vascular Remodeling: Molecular, Cellular, and Vascular Behavior," *Journal of the American College of Cardiology*, vol. 49, no. 25, pp. 2379-2393, 2007.

[2] A. Reinitz, J. DeStefano, M. Ye, A. D. Wong and P. C. Searson, "Human brain microvascular endothelial cells resist elongation due to shear stress," *Microvascular Research*, vol. 99, pp. 8-18, 2015.

[3] O. S.-Y. a. A. G. G. Topman, "A method for quick, low-cost automated confluency measurements.," *Microsc. Microanal.*, vol. 17, pp. 915-922, 2011.

Appendix: Commented Source Code

Functions and scripts are listed in the same order as shown in the operation flow (figure 1).

Masterscript_Final.m

```
%This Masterscript asks for image file input and output final
%segmented cell boundary image as well as numbers of cells, orientation,
%and cell area.

%Clear data and close all window
close all
clear all

%% Intialize with DIP
addpath('C:\Program Files\DIPimage 2.9\common\dipimage')
dip_initialise

%% Read Image
%Takes user input for the name or the path of the file
ImageName=input('Put the name of your image here');
%Read the input image
inImage=readim(ImageName);

%% Convert the image to gray value if it's RGB
%Obtain size of the image.
S=size(im2mat(inImage));
%Number of entries in the size matrix.
S_n=numel(S);

if S_n==2% If it doesn't have three channels for RGB
Image_Gray=stretch(inImage,0,100,0,255);
elseif S(3)==3 %RGB
%Convert to RGB based on the rgb2gary function in Image Processing Toolbox
Image_Gray=0.299 * inImage{1} + 0.587 * inImage{2} + 0.114 * inImage{3};
%Enhance contrast of the gray-scale image
Image_Gray=stretch(Image_Gray,0,100,0,255);
else %Not a gray-scale or RGB
    disp('Error. image needs to be gray-scale or RGB')
end

%% Run Confluency Determination Code
[ confluency, outImage ] = DetermineConfluencywithDIP( Image_Gray, ...
33, 3, 0.03);
%Inputs: original image, big window pixel size,
%small window pixel size, and standard deviation threshold
%Output Confluency and Final Denuded image


%% Confluency-Specfic Algorithm
%For confluent images with higher than 95 confluency
if confluency>=95
%Segment confluent
[seg1]=Seg_confluent_func(Image_Gray);
```

```
% Label and Measure the matrix
%and Measure objects
LabelandMeasure
% Output final data
FinalOutput

%For non-confluent image
else
%Segment non-confluent
[ seg1] = Seg_non_confluent_func(outImage,Image_Gray);
%Label and Measure
LabelandMeasure
%Eliminate extra objects obtained
EliminateObjects
%Output final data
FinalOutput
end
```

DetermineConfluencywithDIP.m

```
function [ confluency, outImage ] = ...
DetermineConfluencywithDIP( inImage, ...
bigWindow, smallWindow, Threshold )
%Inputs in order, image wish to be segmented in grayscale,
%the big window size in pixels,
%the small window size in pixels
%the threshold standard deviation value in which
%above is considered cell-covered area.

%% Rescale the intensity values to between 0 and 1.
image_stretched=stretch(inImage,0,100,0,1);

%% Applying Small Window Standard Deviation Kernel
%Need to convert back to MATLAB to use stdfilt from image processing
%toolbox
smallStd = stdfilt(im2mat(image_stretched), ...
ones(smallWindow,smallWindow));
%Threshold the obtained std results with the input threshold.
smallThresh_dip = ~threshold(mat2im(smallStd), 'fixed',Threshold);

%% Applying Big Window Standard Deviation Kernel
%Need to convert back to MATLAB to use stdfilt from image processing
%toolbox
bigStd = stdfilt(im2mat(image_stretched), ...
ones(bigWindow,bigWindow));
%Threshold the obtained std results with the input threshold.
bigThresh_dip = ~threshold(mat2im(bigStd), 'fixed',Threshold);
%Dilation of the big window to make the foreground closer in including cell
%boundary
%Big Size divided by 2 was recommended from the paper.
bigDilation = dilation(bigThresh_dip,bigWindow/2,'rectangular');

%% Intersection and Post-Processing
%Intersecting the obtained results from the big window and small window.
intersectImage = mat2im(im2mat(bigDilation) .* im2mat(smallThresh_dip));
%Perform morphological close and opening to better obtain the denuded area
closeImage = closing(intersectImage, ...
smallWindow, 'rectangular');
outImage = opening(closeImage, ...
smallWindow, 'rectangular');
%Morpholigcally open and close to remove unwanted specs and holes
closing_img=closing(outImage,50,'elliptic');
outImage=opening(closing_img,50,'elliptic');
%Ensure it is binary
outImage=logical(outImage)';

%% Calculate confluency
confluency = 100*(1-mean(outImage(:)));
end
```

Seg_confluent_func.m

```
function [seg1,img_seed1]=Seg_confluent_func(Image_Gray)
%Input the original greyscale image and output segmentation and detected
%seeds

%% Histogram equlization
Image_Gray=hist_equalize(Image_Gray);

%% Seed Detection
% Remove speckles with median filter
img_med = medif(Image_Gray, 6, 'elliptic');
%Gaussian Filter for blurring
img_gauss = gaussf(img_med,12,'iir');
%Detect seeds using local minimums
img_seed1 = minima(img_gauss,2);
%Dilate the seeds
img_seed1 = dilation(img_seed1,5,'elliptic');

%% Watershed with Seeds

%Preprocess the image with Gaussian filter
img_filt = gaussf(img_med,5,'iir');
% Apply watershed w/ seeds
seg1 = waterseed(img_seed1,img_filt,1);

end
```

Seg_non_confluent_func.m

```
function [ seg1,img_seed1] = Seg_non_confluent_func(denuded,Image_Gray)
%Input denuded image and original image
%Output segementation and detected seeds

%% Histogram equlization
Image_Gray=hist_equalize(Image_Gray);

%% Seed Detection
% Remove speckles with median filter
img_med = medif(Image_Gray, 6, 'elliptic');
%Gaussian Filter for blurring
img_gauss = gaussf(img_med,12,'iir');
%Find seeds using local minimums
img_seed1 = minima(img_gauss,1,true);
%Detect the seeds
img_seed1 = dilation(img_seed1,5,'elliptic');
%Eliminate seeds found in the denuded area.
img_seed1(logical(denuded)')=0;

%% Watershed with Seeds
%Segmenting the cell-cell border of non-conlfuent image after having
%segemneted the denuded area.

%Preprocess the image with Guaissian filter
img_filt = gaussf(Image_Gray,5,'iir');
% Apply watershed w/ seeds
seg1 = waterseed(img_seed1,img_filt,1);
%Ensure the data type is a binary image
seg1=logical(seg1);
end
```

LabelandMeasure.m

```
%This script creates a binary image and then label and measured

%% Create Binary Image for labelling
%Making a Binary Image where inside the cells are 1s and cell-bounadries are 0s
%Confluet image
if confluency>95
binary=logical(ones(size(Image_Gray)))'; %same size as input image
%non-confluent image
else
binary=logical(~outImage)'; %invert the denuded image to obtain cell area
end
```

```
%Enforce the watershed lines to be zero for labelling.
binary(seg1)=0;

% Label the Objects and measure the critical cell output
L=label(binary,1);

%Measure with the measure function
msr = measure(L, Image_Gray, ({'size', 'perimeter','P2A','Feret'}),[], 1);
```

EliminateObjects.m

```
%This script eliminates objects based on zscore of size and diameter to reduce errors

%% Caculate the z score
%Only using the z-score of the major axis
z_FeretMax=zscore(msr.Feret(1,:));
%z-score of the pixel area sizes
px_size=zscore(msr.size(:));

%% Elimate the z_scores that are out of scope
% zscore Smaller than -2 and larger than 2
[i2]=find(z_FeretMax<=-2|z_FeretMax>=2);
[i3]=find(px_size<=-2|px_size>=2);

%Eliminate entries in the original label matrix
%Eliminate based on with undesirable length of major axis
for j2=1:length(i2)
 L(find(L==i2(j2)))=0;

end
%Elimnate the entries with undesirable pixel size
for j3=1:length(i3)
    L(find(L==i3(j3)))=0;

end

%Display new label matrix
L_show=dipshow(L,'labels');
%% Measure again after eliminate of undesirbale extra objects.
msr = measure(L, Image_Gray, ({'size', 'perimeter','P2A','Feret'}),[], 1);
```

FinalOutput.m

```
%This script outputs final data
close all
%Output confluency
confluency
%Display denuded area
dipshow(outImage')
%Display input image
dipshow(inImage)
%Display segemented cell boundary
dipshow(seg1)
%Count number of cells
num_objects=length(msr)
%Ask for micron/pixel ratio
mic_pix=input('Micro/pixel ratio');
%Output mean average cell size in microns
avg_size=mean(msr.size(:))*mic_pix^2
%Output histogram of cell orientation
histogram(msr.Feret(4,:))
%Display the labelled matrix
L_show=dipshow(L,'labels')
```