Improvement of Cell Counting Program by using Image Processing Methods

Su Mon Aung

A Thesis Submitted in Fulfillment of the Requirements for the Degree of Master of Science in Biomedical Engineering
Prince of Songkla University
2017

Improvement of Cell Counting Program by using Image Processing Methods

Su Mon Aung

A Thesis Submitted in Fulfillment of the Requirements for the Degree of Master of Science in Biomedical Engineering
Prince of Songkla University
2017

Thesis Title	Improvement of cell counting program by using image processing methods
Author	Miss Su Mon Aung
Major Program	Biomedical Engineering

Major Advisor

.....
(Asst. Prof. Dr. Surapong Chatpun)

Examining Committee :

.....Chairperson
(Asst. Prof. Dr Theekapun Charoenpong)

.....Committee
(Assoc. Prof. Dr. Pornchai Phukpattaranont)

.....Committee
(Asst. Prof. Dr. Kanyanatt Kanokwiroon)

.....Committee
(Asst. Prof. Dr. Surapong Chatpun)

The Graduate School, Prince of Songkla University, has approved this thesis as fulfillment of the requirements for the Master of Science Degree in Biomedical Engineering.

.....
(Assoc. Prof. Dr. Damrongsak Faroongsarng)
Dean of Graduate School

This is to certify that the work here submitted is the result of the candidate's own investigations. Due acknowledgement has been made of any assistance received.

.....Signature
(Asst. Prof. Dr. Surapong Chatpun)
Major Advisor

.....Signature
(Miss Su Mon Aung)
Candidate

I hereby certify that this work has not been accepted in substance for any degree, and is not being currently submitted in candidature for any degree.

.....Signature
(Miss Su Mon Aung)
Candidate

Thesis Title Improvement of Cell Counting Program by using Image Processing Methods

Author Miss Su Mon Aung

Major Program Biomedical Engineering

Academic Year 2017

ABSTRACT

Cell counting is one of the most important, common tasks in experiments that deal with cell. Traditionally, manual cell counting uses hemocytometer which is the gold standard for cell counting for biomedical research. However, manual cell counting is time consumed; this may also lead to inaccurate counting results, especially if the counting operators have many samples to analyze over a short period of time. In particular, computer-assisted programs can reduce time consumption, as well as avoiding manual artifact about cell counting. Furthermore, it can diminish the variation of counting results among the counting operators which sometimes is related to personal judgment. For this study, input data are microscopic images that are stained with Trypan Blue dye. The main objectives of this study are to propose a computer-assisted cell counting approach based on image processing techniques and to improve the accuracy and reliability of cell counting, especially dead cells counting. The proposed computer-assisted cell counting algorithm works in four stages. First, image denoising is applied by using image guided filter, since original microscopic staining images have background noise and small debris. Secondly, a thresholding method is used to extract background and foreground objects, and then the third stage is to apply the morphological operations for image analysis. The final stage is to analyze and identify live and dead cells by using object analysis. There are two approaches in this study: the first cell counting algorithm by using morphological operations of digital image approaches and image

segmentation and the second algorithm by using the detection of live and dead cell based on adaptive K-means clustering. The first approach gave 83±6% and 63±10% for the accuracy of live cell counting and dead cell counting respectively when compared with the experts. The results show that the performance of the second approach using an adaptive K-means clustering reaches 89±4% of live cells from three experts, showing a good likelihood in clusters of live cells and 61±23% for dead cells accuracy, whereas ImageJ obtained 49±1% for total counting cells only compared with manual counting.

The correlation coefficients between the counting results from the first approach and the experts were 0.99 and 0.74 for the living cells and dead cells. The second counting algorithm also displayed the highest correlation (r = 0.99, r=0.99, r=0.99) with three manual counting for live cells while the ImageJ did not correlate well with manual counting (r=0.91, r=0.92, r=0.91). In addition, the second approach has a good correlation with manual counting for dead cells (r=0.93, r=0.80, r=0.85) which is higher than the first approach. Therefore, both proposed computer-assisted show that the reliability and accuracy of counting are increased, especially live cells when compared with the experts. Finally, Graphical user interface (GUI) is developed to make this a user-friendly application for Trypan Blue stained microscopic image cell counting.

ACKNOWLEDGEMENT

I would first like to thank my thesis advisor Asst. Prof. Dr. Surapong Chatpun (Faculty of Medicine, Prince of Songkla University) for his encouragement, supervision and guidelines.

I would like to thank the experts who was involved in the manual counting for this research project. I would like to give my pleasure to Asst. Prof. Kanyanatt Kanokwiroon (Faculty of Medicine, Prince of Songkla University) who gave the advice for my cell counting program. Without her passionate participation and input, the cell counting program could not be successfully conducted.

I would also like to acknowledge Assoc. Prof. Pornchai Phukpattaranont (Faculty of Engineering, Prince of Songkla University). I am gratefully indebted for his very valuable suggestions on image processing.

I would like to thanks Asst. Prof. Dr. Theekapun Charoenpong (Faculty of Engineering, Srinakharinwirot University) for his kind to be the chairperson.

I also would like to thank IBME staffs for taking care of me while I have stayed in Thailand for two years.

Finally, I must express my very profound gratitude to my parents and my siblings for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Su Mon Aung

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In cancer drug discovery researches, the effects of novel drug treatment on cancer cells have been studied in many aspects including morphological change, cellular structure and drug response(1-3). A simple investigation to evaluate the efficacy of cancer drug treatment is the identification of apoptotic cells and live cells(4). Apoptosis is a programmable cell death, which categorically observed from cell morphological changes. Counting apoptotic cells or dead cells per total cells after in vitro drug treatment is a common work for biologists and biomedical researchers to evaluate the drug efficiency. The counting of cells in biological research is mostly carried out by microscope-based counting and the Neubauer, Burker and Fuchs-Rosenthal chambers are widely-recognized methods of counting cells and cell densities in matter (31). Manual cell counting by the expert with a tally counter is a conventional method which is an individual dependence, tedious and laborious. To overcome the drawbacks of manual cell counting task, many automated visual analyses have been developed such as ImageJ, Fiji, CellProfiler™, CellC, Image-Pro Plus and MetaMorph® which are available software packages for cell analysis(5-11). However, some of these methods are based on open-source softwares and are designed for use with images of a wide variety of biological objects so sometimes do not meet the requirements of a specific purpose.

Computer-aided methods can solve this problem by employing image segmentation which whilst fundamental to cell counting applications, is a difficult problem from a programming perspective. These methods rely on cell images with a high contrast between cells and their background and the problem of analyzing images which do not naturally present such a high contrast is described as the dark field cell identification issue. Several

image filters and segmentation methods have been proposed for use in cell identification and counting in microscope-based images, such as the watershed transform-based method (34).

Additionally, computer-assisted cell counting using image processing and analysis has been continuously developed and improved to provide the requirements of high reliability, specificity and sensitivity due to different cell types and test protocols. For example, Sui et al. developed cell counting method for host cells in the bright field for insect cells by using nonlinear transformed sliding band filter. They obtained a low error rate and a high accuracy when compared their proposed method with manual counting (12). The cell counting method for white blood cells has been developed with Matlab to evaluate the hematic pathologies in terms of numbers, sizes and types of white blood cells(13). They applied image enhancement and image segmentation as pre-processing steps and neural network for classification. Piccinini et al. has used a fully automated mosaicing method to improve the reliability and reproducibility of live and dead cell counting(14). Moreover, Mouelhi and colleagues also described automatic image segmentation with active contour for stained nuclei in breast cancer tissue which could segment touching nuclei to get a total number of breast cancer nuclei(15).

Quality of light microscopy digital images taken from hemocytometry significantly influences the visual analysis including cell counting. Therefore, most light microscopy digital images need to do a pre-processing before performing cell segmentation. Noise filtering is one of the pre-processing steps in every digital image processing technique. Median filter is a popular noise filter which is used to remove salt and pepper noise in digital images(16-19). It has been reported that impulse noise removal in high-density noisy images is effectively removed by using an improved algorithm such as decision based adaptive neighborhood median filter(20). Mahmood and Mansor used morphological tool with Hough transform technique to detect and estimate the number of red blood cells in the blood sample image(21). Liew et al. applied image processing methods to develop a prototype for cell

detection algorithm using circular Hough transform detecting the number of cells in bee comb from digital images(22). Although there are many methods for cell detection and cell counting, computer-assisted cell counting methods are still increasingly developed to improve the accuracy and to match with the applications.

In general, image analysis requires the image to be pre-processed to enhance its quality and to make it suitable for the following steps of image analysis. Image pre-processing usually consists of noise removal, contrast enhancement, region separation and the conversion of color images to gray scale (36) or HSV images (37,38). Further, image segmentation is used to separate objects from the background and there are many segmentation methods used, such as histogram thresholding, Otsu thresholding, global thresholding, the Hough transform and watershed transform algorithms as well as by K-means clustering. The Circular Hough Transform (CHT) is a frequently used method for detecting circular objects in an image. However, it often suffers from degradation in performance, especially in terms of speed, because of the large amount of edges presented by a complex background or texture. Some applications use segmentation to classify abnormalities in cells. In the analysis of white blood cells, some techniques used are gradient vector flow , the snake algorithm and Zack thresholding which can be used for segmenting the nuclei of cells (39). Methods based on morphological operators, gray-level threshold based methods (33) contour or region-based methods (35) and artificial neural networks (ANN) have been proposed.

Image post-processing includes feature extraction and morphological operations to identify objects, which include dilation, erosion, granulometry and morphological filtering. Further, a closing operation is used to fill holes and gaps and an opening operation is used to smoothen an image (40). Feature extraction determines the features that contain quantitative information about objects of interest. Shape features frequently used in biological images are geometric parameters like cell area, cell perimeter and the ratio of the nucleus to the overall cell area, the boundary of the nucleus and the circularity index (39).

Circular Hough Transform (CHT) is a technique used in image analysis to detect objects in a circular form (41). Classifiers like nearest neighbor , k-nearest neighbor , Bayesian analysis, support vector machines , neural networks (36),local linear maps , fuzzy cellular neural networks are methods which can be included in feature extraction and are often used to classify blood cells.

Therefore, a combination of suitable pre-processing and post-processing techniques can improve the efficiency of cell detection and cell counting. The robust and reliable systems are required to enable biotechnologists to handle larger cell-image data sets. Moreover, providing a user-friendly interface and counting record report are important considerations in automatic cell counting applications.

The objective of this work was to develop algorithms that improve the counting efficiency of live cells and dead cells from Trypan Blue stained microscopic digital images and to create a graphical user interface (GUI) for this automatic cell counting algorithm.

Chapter 2

Materials and Methods

2.1 Sample preparation and image acquisition

Human breast cancer cell line (MDA-MB-231) purchased from American Type Culture Collection (ATCC, Canada) was used as an example in this study. The procedure of the sample preparation is overviewed in Figure 2.1.

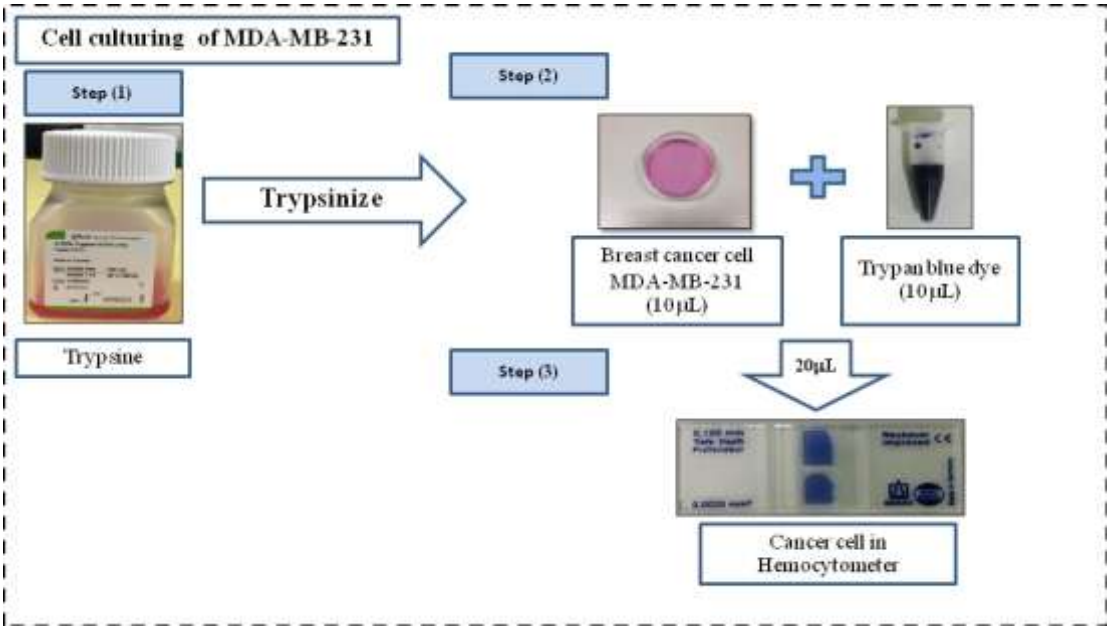


Figure 2.1 Trypan Blue stained MDA-MB-231 cells in hemocytometer

The sample preparation was performed according to the procedure stated in the work of Piccinini et al.(14) In brief, the cells were detached from the flask by shortly exposure to trypsin/EDTA, washed in the culture media by centrifugation and resuspended

in 10 mL of DMEM medium. After trypsinization, cells were stained with Trypan Blue solution and then the 20 µL of this mixture were distributed inside the counting chamber of hemocytometer.

Images of the stained cells were obtained by an inverted Olympus IX51 microscope equipped with a digital Olympus DP72 camera with 2/3-inch CCD. The images were acquired in the bright field by using Olympus UPLanFL N 10x0.13 as a standard objective lens. An example of digital images of Trypan blue stained cells was shown in Figure 2.2.

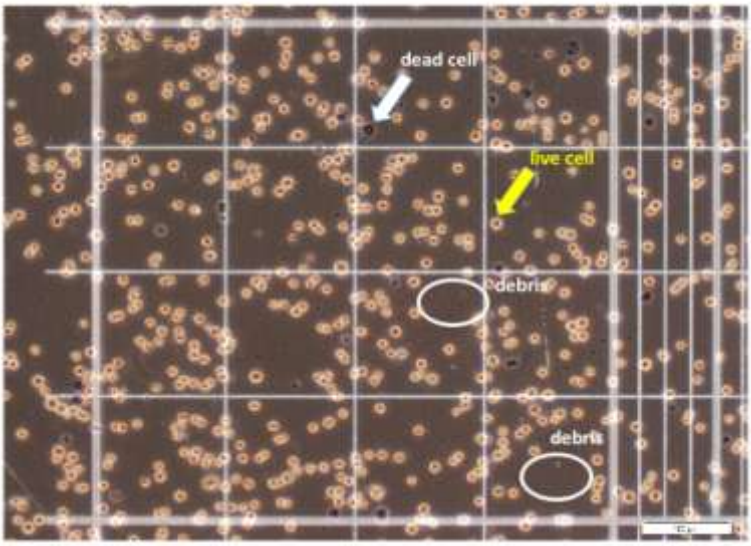


Figure 2.2 An example of original image of Trypan Blue stained

The image was cropped to a 16-square grid and then the image processing was performed to the image as shown in the flowchart of our cell counting approach presented in Figure 2.3.

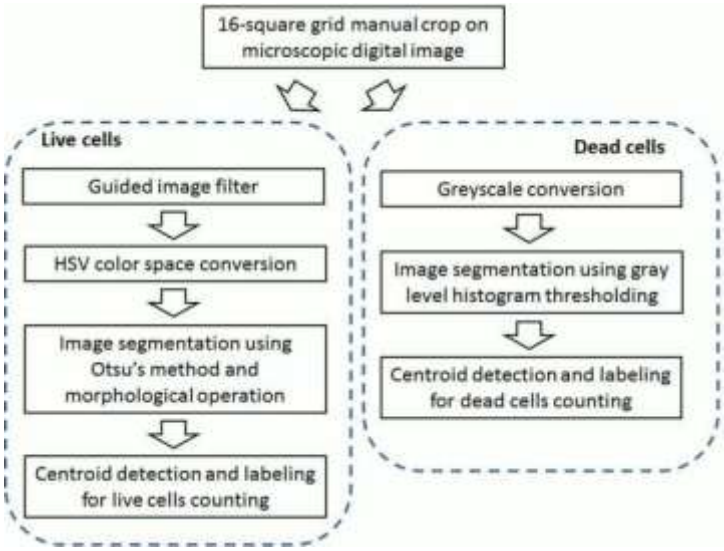


Figure 2.3 Overview of the image processing for live and dead cells counting

2.2 Image processing approaches for cell counting

2.2.1 Approach 1: Image segmentation and morphological operations

2.2.1.1 Image filtering

In our study, it is noticed that there are small debris (dark spot) in Trypan Blue stained microscopic images of hemocytometer which can cause the artifacts of cells as shown in Figure 2.2. In order to solve this problem, the guided image filter was applied and spatially distributed the filtering strengths with low-level features to the image to smooth and remove the small debris from background (23, 24). Furthermore, the guided image filter maintains the edge features of image, better content-specific visual effects and introduce less artifacts. The filtering process need constraints from the input p to determine the linear coefficients (a_k , b_k). The filtering output q is a linear transform of the guidance I in a window

ωk centered at the pixel k since the input p needs to subtract some unwanted components n such as noise/textures as follows:

$$q_i = a_k I_i + b_k, \forall_i \in \omega_k \quad \text{(i)}$$

$$q_i = p_i - n_i \quad \text{(ii)}$$

The function of guided image filtering is similar to other filtering operations, which reduces noise and maintains the edge without blurring. This noise filtering algorithm uses the image correlation to function the features of the filtering mask over the image and adaptively resizes the mask according to the noise levels of the mask. Furthermore, the guided image filter needs neighborhood size around each pixel as a scalar or two-element vector, [M, N] of positive integers. In this study, neighborhood radius size [M, N] = [30, 30] to remove small debris in the background.

2.2.1.2 Color space and grayscale conversion

There are several color spaces such as RGB (red, green, blue), HSV (hue, saturation, value) and HSL (hue, saturation, and luminance) for color images. The purpose of color model or color space is to simplify the specifications of colors in some standard. The most common color space is RGB in which color space can be obtained by the sum of three primary colors: red, green and blue. The original Trypan Blue stained image is represented by the RGB color space in the RGB model. In this study, a color space of the Trypan Blue stained images which was originally RGB color images was converted by applying HSV color space. The HSV color space is quite similar to the way in which humans perceive color.

The colors used in this space can be clearly defined by human perception, which is not always the case with RGB. These characteristics make the HSV color space more suitable for image segmentation and analysis than the RGB model. The original RGB images were also converted to grayscale for dead cells detection. The segmentation is based on histogram thresholding and morphological operations, and then counting is based on the labelling of binary image. The histogram of Trypan Blue stained gray scale images was studied, and it was found out the best thresholding value to extract dead cells. After cells separation, each image is preprocessed using morphological operators to obtain the area of dead cells using labelling.

2.2.1.3 Image segmentation

Before image segmentation, the brightness of a converted image was adjusted for a better contrast. It is necessary to separate a digital image into multiple regions or clusters i.e. objects and background. Based on the color space converted image, it was noticed that only live cells were appear clearly. This approach utilized Otsu's thresholding method to extract foreground (live cells) from the background and to turn the image to black and white(25). As the Trypan Blue stained microscopic images still contained counterfeit structures in the nuclei, these caused the difficulty for the extraction and segmentation. Therefore, the operations based on the morphological reconstruction were applied to overcome this drawback. Opening by reconstruction can remove the disconnected bright objects that are smaller than the structuring element (SE)(26). Similarly, dilatation by reconstruction can remove the disconnected dark objects smaller than the SE. These two

operators were applied in a sequence to our images. A disk shape SE with radius “n” was assigned in the algorithm. The amount of detail presented in the image depends on the size of the SE which relates to the size of nuclei and the resolution of the image. Furthermore, the effective threshold value from the histogram of a grayscale image was applied to obtain only dead cells in the black and white image.

2.2.2 Approach 2: Adaptive K-mean clustering

The general concept of the proposed algorithm is presented in Figure 2.4.

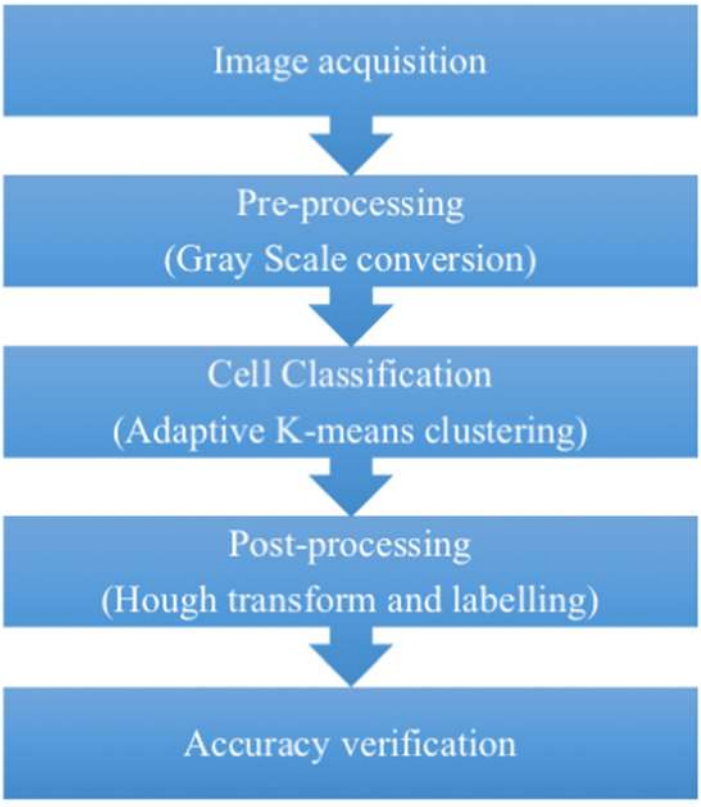


Figure 2.4 Illustration of concept for cell counting algorithm

2.2.2.1 Pre-processing

Pre-processing of an image was to convert a RGB image into a gray scale image. A weighted average of three different color components i.e., red, green and blue values are converted to equal gray scale values. The weighted gray scale average value is defined by gray in the following equation.

$$f_{gray} \text{ (Gray Image)} = 0.29R + 0.59G + 0.11B \qquad \text{(iii)}$$

The morphological concepts and methods have been proposed to constitute a powerful set of tools for separating features from an image. The basic operators of erosion, dilation, and reconstruction defined for both binary and gray scale image processing can be done with in sequence to represent an extended range of tasks. Morphological techniques can be applied for image segmentation. Furthermore, they play a significant part in algorithm for image information. Morphological opening, erosion occasionally was used in order to eliminate small objects, and the consequent dilation tends to reconstruct the pattern of the objects that remain. Nevertheless, the efficiency of this reconstruction depends on similarity between the shapes of cell and the radius of structuring element. The morphological opening by reconstruction of an image G using structuring element B, is defined as R_G (GΘB) in this study, the image extraction was performed from the stained image that involves living and dead cells. Both “opening” of morphological operations and reconstruct the “opening” pixel by erosion in common, we performed that step beginning, applying a ‘disk’ structuring element. Thus, the live and dead cells were reconstructed correctly. “Opening” suppresses bright detail smaller than the structuring element; in the meanwhile “Closing” suppresses dark

features smaller than the structuring element. Opening can be used to compensate for nonuniform background illumination. By subtracting this background from the original image, we could form an image of the live and dead cells with a reasonably smoothly background.

2.2.2.2 Clustering technique (Adaptive K-means Clustering)

In this step, the live and dead cells of stained image were extracted by using adaptive k-means clustering algorithm. Algorithm for adaptive K-means clustering composes of 5 steps. First, K elements are selected from the input data set. Second, the distance between a given element and a cluster is computed. Third, the distance between two data elements E1 = {E11, E12, E1n} and E2 = {E21, E22, E2n} is given by $\sqrt{(\boldsymbol{E_{11}} - \boldsymbol{E_{21}})^{\boldsymbol{2}} + (\boldsymbol{E_{12}} - \boldsymbol{E_{22}})^{\boldsymbol{2}} + \cdots + (\boldsymbol{E_{1n}} - \boldsymbol{E_{2n}})^{2}}$ Fourth, the distance of each cluster from every other cluster is determined and this value is stored in a 2D array as a triangular matrix. Lastly, a distance between any two clusters Cm_1 and Cm_2 is minimized as well as the identification of these two closest clusters. In this study, k=3 was assigned to group the clustering objects in images. There are 3 clusters can be separated as dead cells, total cells and live cells. Therefore, we choose the cluster 1 and cluster 3 to process for dead and live cell counting, respectively.

2.2.2.3 Post-processing

The segmented image for live cells from adaptive k-means (Cluster 3) was used as an input for Hough transform process. Hough transform approach is to distinct live

cells from dead cells and background. The Hough transform has been identified as particularly robust tool for the detection of parametric curves in images. It achieves a voting process that maps image edge points into manifolds in an approximately defined parameter range. The circular Hough transform, one of the Hough transform techniques, concentrates to locate circular patterns within an image. The circle Hough transforms is intended to obtain a curve characterized by a center point (0, 0).

$$(x-x_0)^2+(y-y_0)^2=r^2 \quad \text{(iv)}$$

Where, x_0 and y_0 are the coordinates of the center and r is the radius of circle. Corresponding to the sequence from the adaptive k-means segmented image (Cluster 1) which is dead cells area, the most suitable method used for counting in this study, was connected component labeling. Therefore, counting of dead cells was accomplished by identifying the number of connected components in segmented image of adaptive k-means clustering (Cluster 1) and the counted the connected objects in a segmented image.

2.2.3 Cell counting

To count live and dead cells, a label technique was applied to the binary images after thresholding to distinguish objects from the background and then measured the geometric properties using centroid and boundaries to select live cells from labeled regions.

2.3 Accuracy Verification

In this study, three experts counted thirty-six Trypan Blue stained images to evaluate the accuracy of program and ImageJ counting. We used ImageJ which is open source software to count the cells. CellContingMacro2 v1-01plugin in ImageJ was used to count the cells in Trypan Blue stained images. Furthermore, the correlation between two methods was determined.

2.4 Graphic User Interface (GUI)

GUI was developed to make the algorithms easier to use and more convenient for cell counting. The features of this GUI are composed of loaded images, parameter setting, live cell counting and dead cell counting. The percentage of cell death showed on GUI is computed from the ratio of dead cells per total cells.

Chapter 3

Results

3.1 Image segmentation and morphological operations

Thirty six Trypan blue stained microscopic images were obtained and used to test in this study. The debris in the image (Figure 3.1a) was removed after the image guided filtering as shown in Figure 3.1b. As we performed color space conversion, the images based on HSV channels are shown in Figure 3.2. It can be noticed that an image with the saturation channel (Figure 3.2b) gave a better contrast of live cells compared to other channels. Then after brightness adjustment, the live cells became clearly extracted from the background as shown in Figure 3.3.

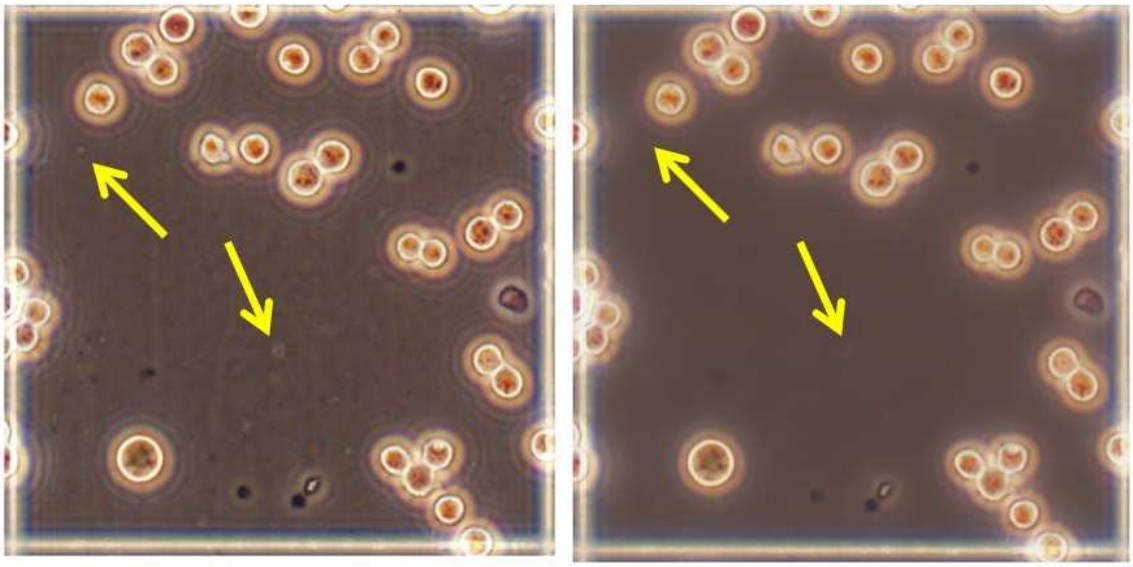


Figure 3.1 Examples of (a) original image and (b) image after guided image filter process

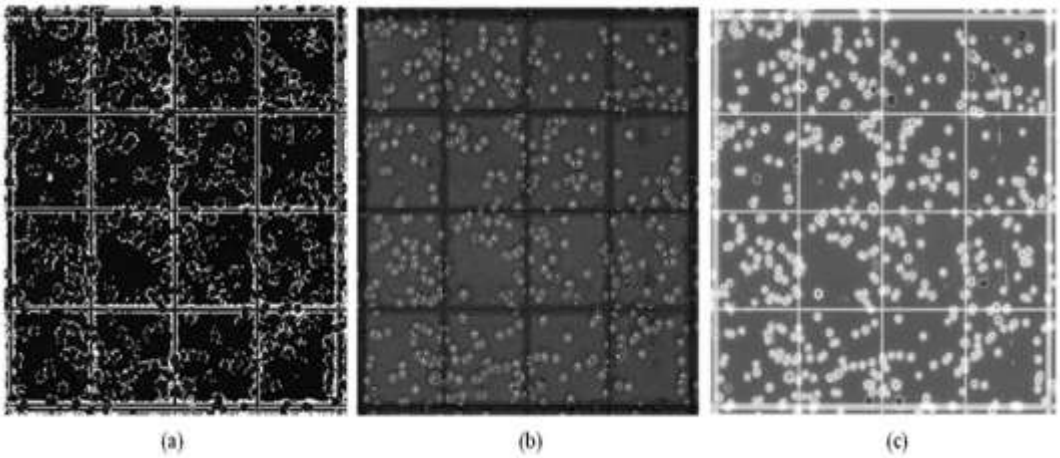


Figure 3. 2 Images after HSV color space conversion: (a) Hue channel image, (b) Saturation channel image and (c) Value channel image

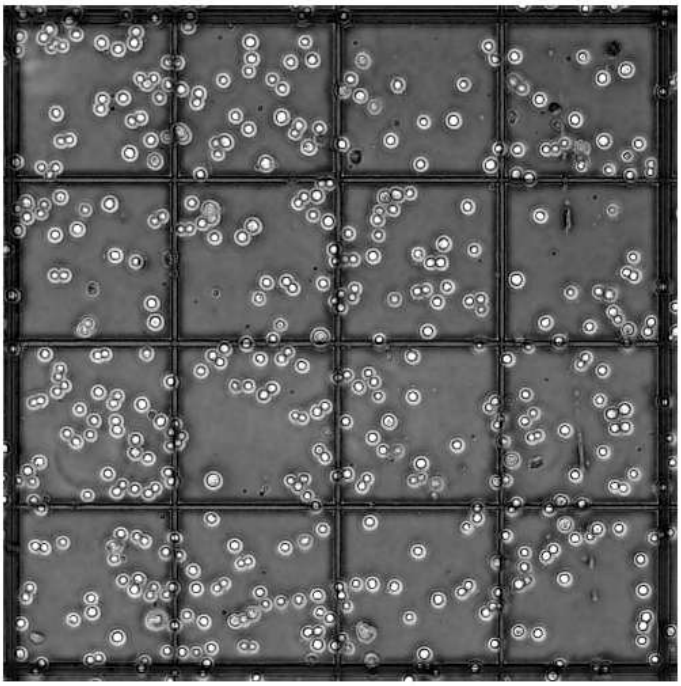


Figure 3.3 Saturation channel image after brightness adjustment showing better contrast of live cells from the background

Otsu's thresholding was performed to segment the live cells from the background and the nuclei of live cells are opened and dilated after thresholding. The result of segmentation step is demonstrated in Figure 3.4 for live cells as white spots. Our approach provided a very similar counting result of live cells to the experts' counting result as a high correlation coefficient (r=0.99) whereas a correlation coefficient in case of the dead cell counting is 0.74 (Figure 3.5). In comparison with the experts, the average accuracy percentages of cell counting from thirty six images were also determined. The accuracy of live cell counting and dead cell counting were 89±10% and 67±24%, respectively.

Figure 3.4 Image after applying Otsu's method showing a black and white image

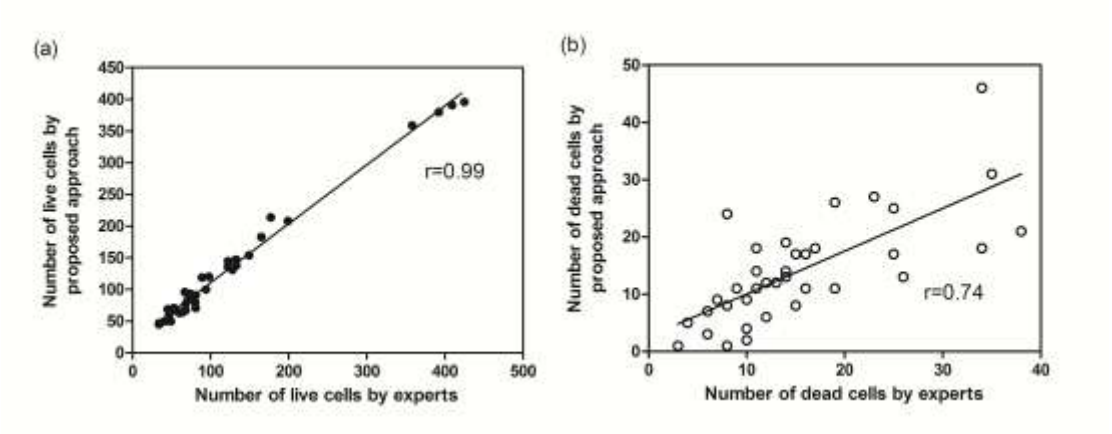


Figure 3.5 Correlation plots between the counting results by the first approach and experts for (a) live cells and (b) dead cells

After that the image was processed with counting algorithm to identify and count both live and dead cells. The final images showing the live and dead cells identification are respectively presented in Figure 3.6 and Figure 3.7.

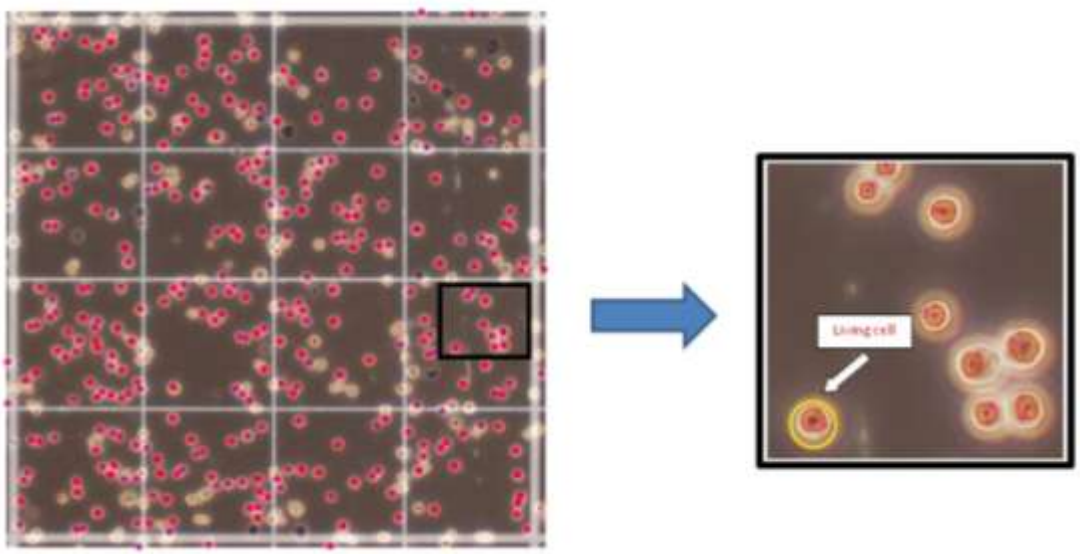


Figure 3.6 Example of image showing the live cells detection

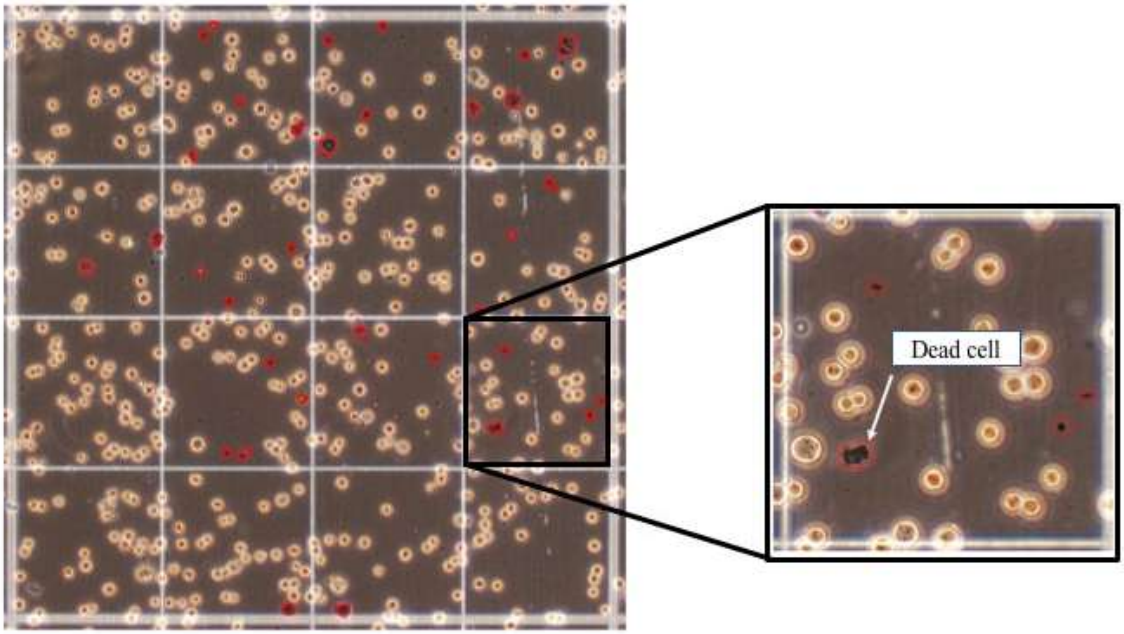


Figure 3.7 Example of image showing the dead cells detection

3.2 Adaptive K-mean clustering

Using the second approach, Figure 3.8 depicts the conversion of the original of one of the 36 Trypan Blue stained images used in this work to assess the cell counting program, to a gray-scale image.

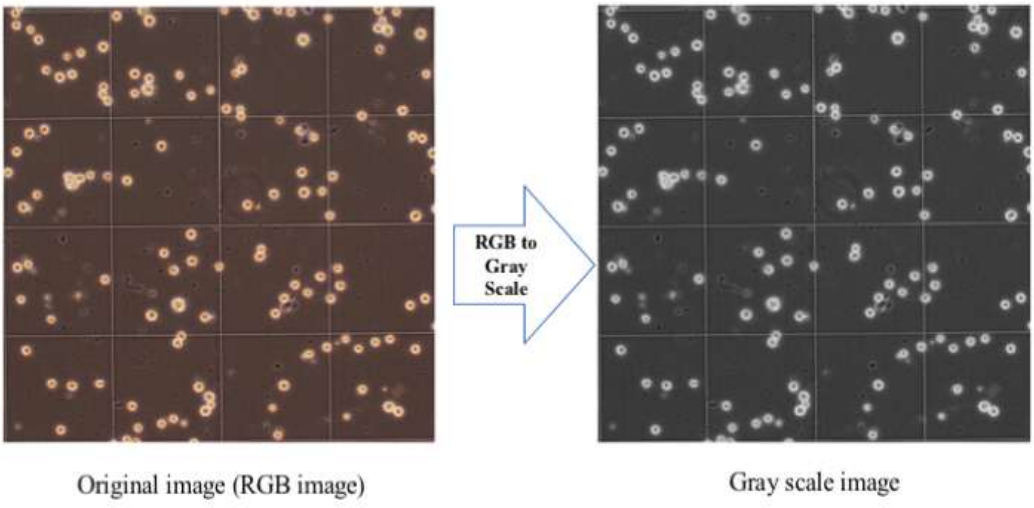


Figure 3.8 Example of original RGB to Gray converted image

As the opening and closing reconstruction of the morphological operations were applied at the beginning of a disk-structuring element, the live and dead cells were reconstructed perfectly as shown in Figure 3.9. Figure 3.10 illustrates the three clusters, dead cells, total cells and live cells from which cluster 1 and cluster 3 were used to count the dead and live cells, respectively, using the Hough transform technique which was applied after segmentation from the adaptive K-means clustering, to extract the live cells from the dead cells and the background as showed in Figure 3.11.

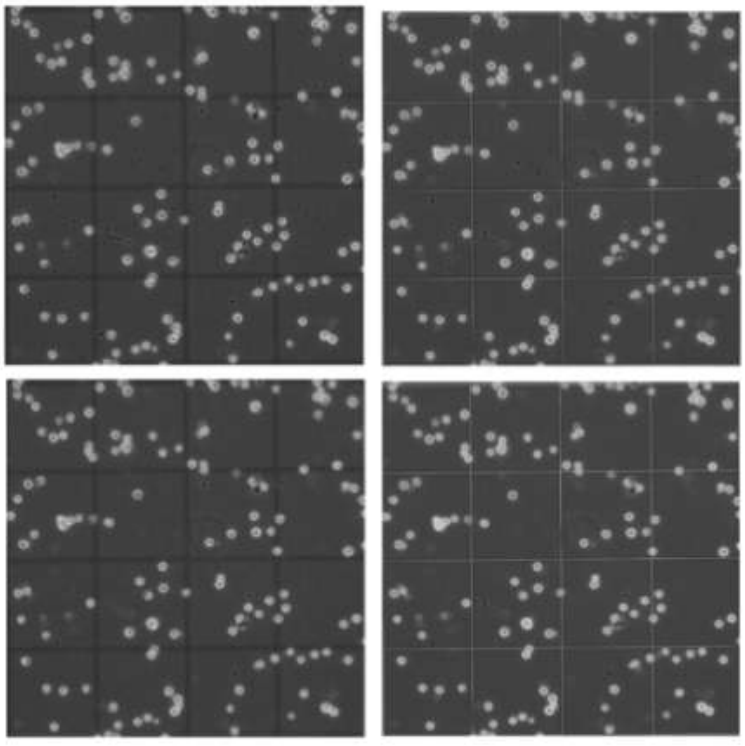


Figure 3.9 Example of morphological reconstruction of Trypan blue stained image

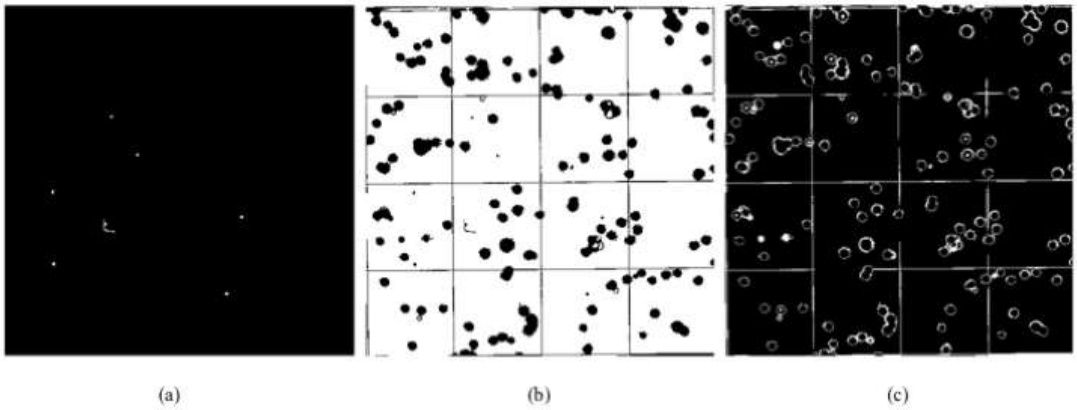


Figure 3.10 Adaptive K-means Clustering: (a) dead cells, (b) total cells and (c) live cells

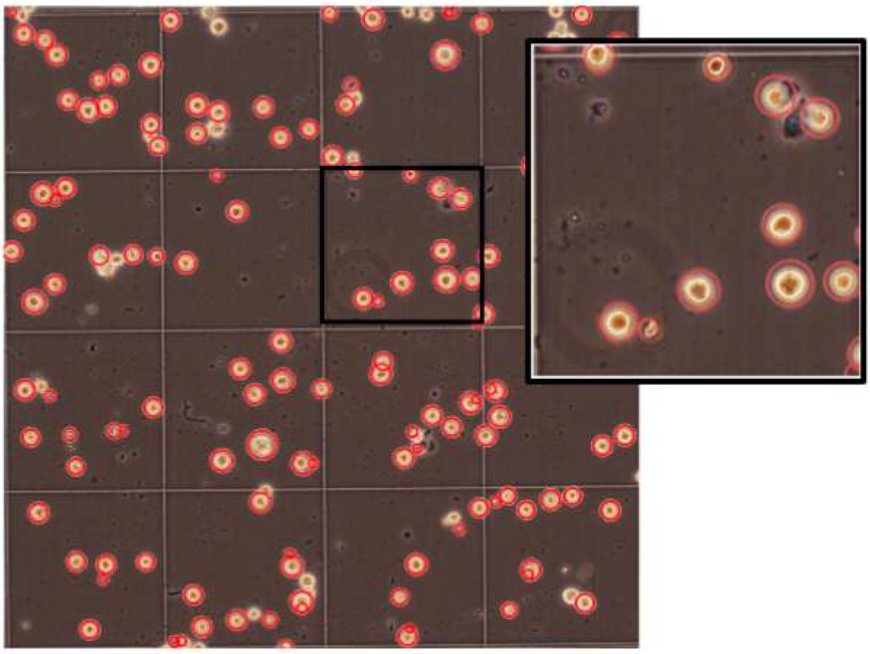


Figure 3.11 Example of Hough transform for living cell detection

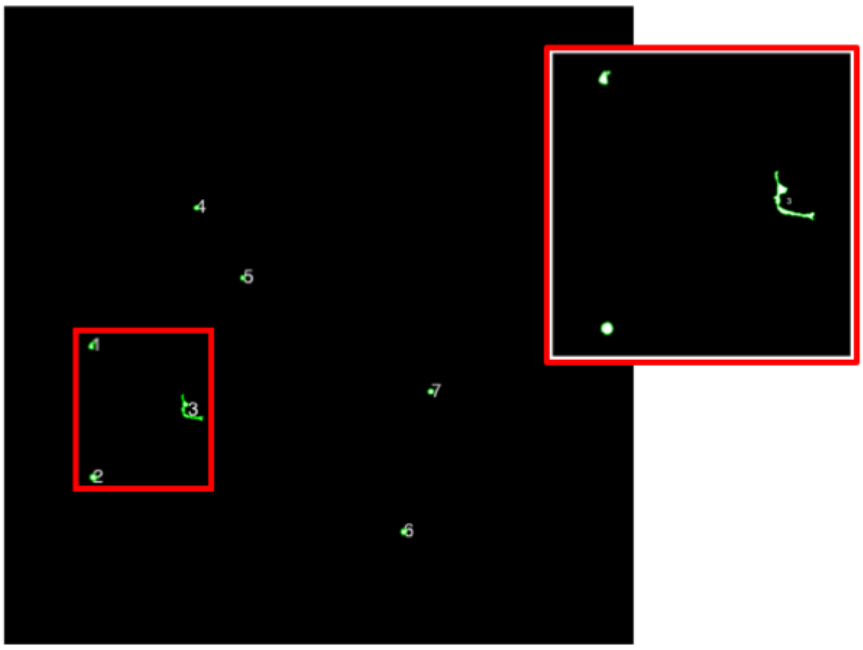


Figure 3.12 Example of labeling for dead cell detection

Since the counting of the dead cells is accomplished by finding a number of connected components in the segmented image of the adaptive K-means clustering (cluster 1) and identifies the connected objects in an image, the result obtained was that presented in Figure 3.12. ImageJ was used to measure the accuracy of the live- and dead-cell counts. However, ImageJ cannot classify live and dead cells automatically from the original image as shown in Figure 3.13.

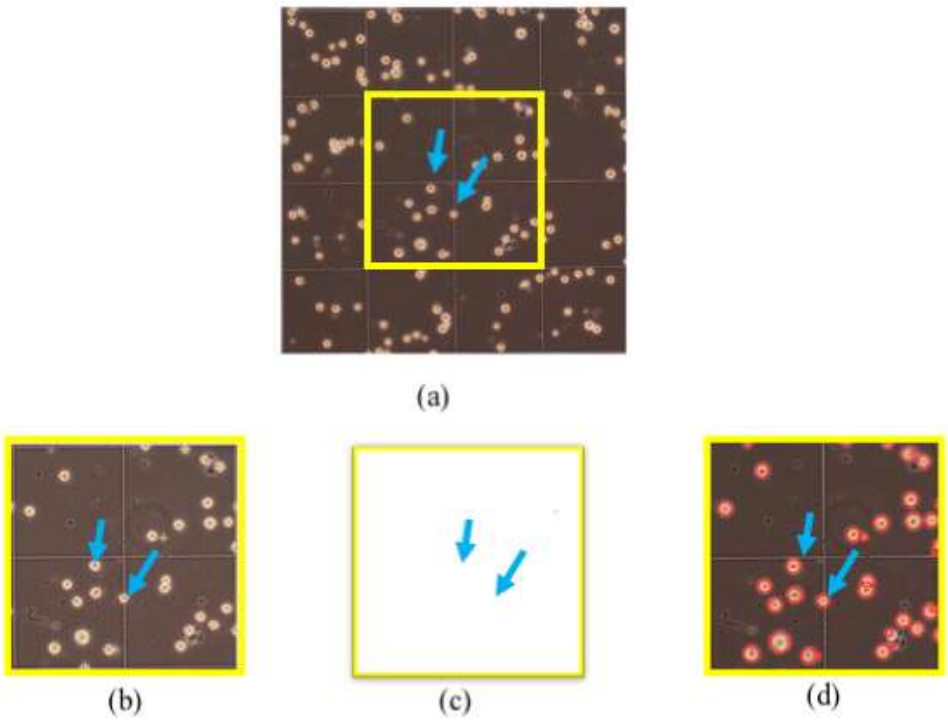


Figure 3.13 Example of comparison of Manual, ImageJ and Proposed

The findings of this study show that the results of counting the live cells consistently fit a regression line based on the counts conducted by the three experts as shown in Fig.3.14 and Fig.3.15 that is indicated for dead cell counting. Therefore, these results suggest that the results of the count of live cells can reasonably be accepted as accurate. Fig.3.16 shows the linear correlation of ImageJ and three experts for live cell counting.

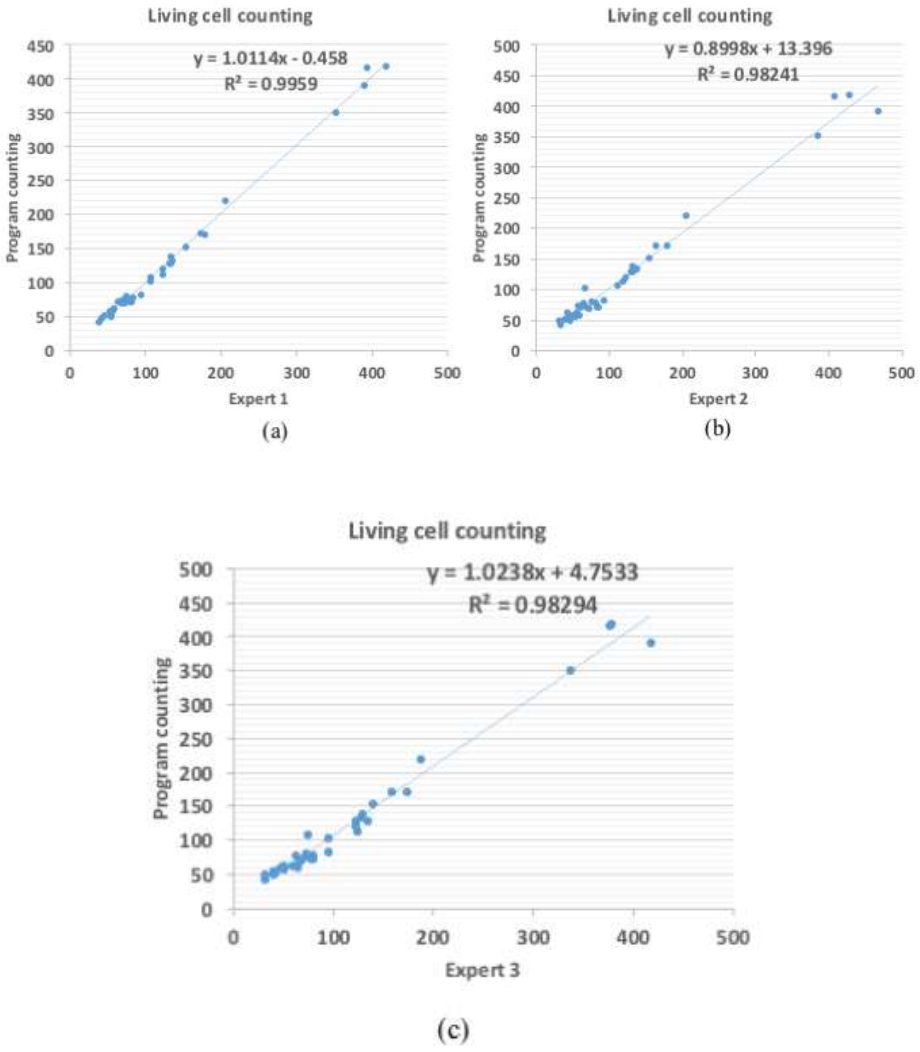


Figure 3.14 Accuracy verification: (a) (b) (c) linear correlation of program and three experts for live cell counting

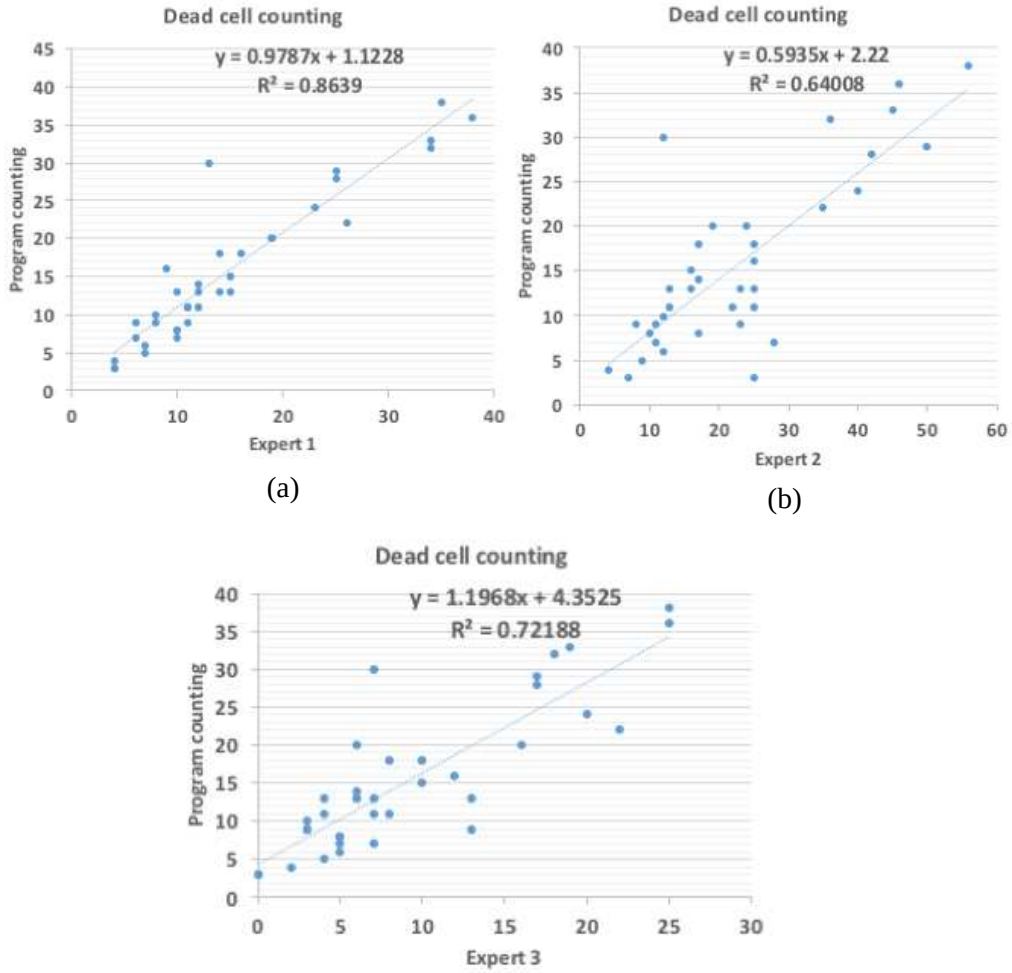


Figure 3.15 Accuracy verification: (a) (b) (c) linear correlation of program and three experts for dead cell counting

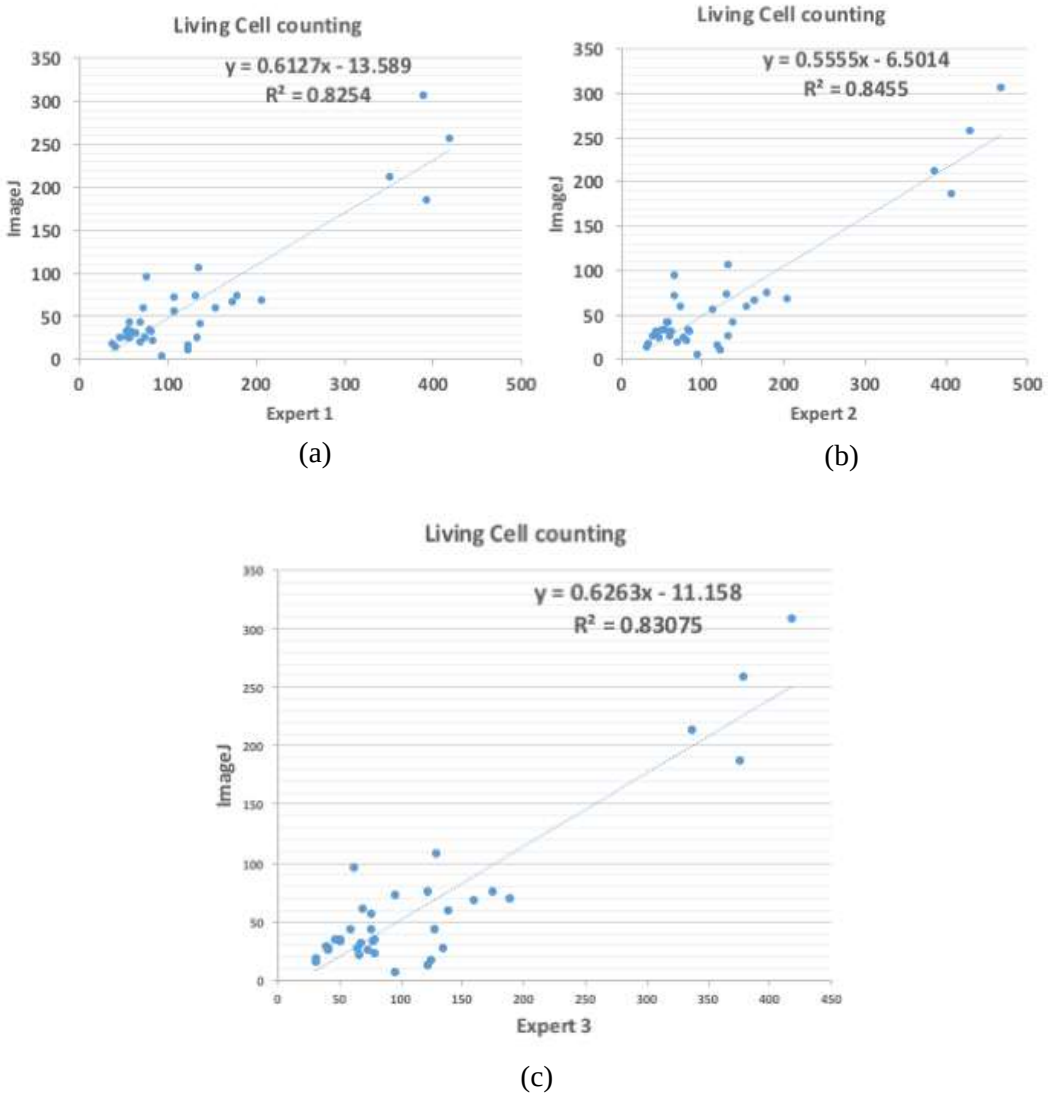


Figure 3.16 Accuracy verification: (a) (b) (c) linear correlation of ImageJ and three experts for live cell counting

Finally, a GUI was developed to provide a user-friendly screen display during cell counting which is presented in Figure 3.17. Figure 3.18 showed an example of loading input image into application. Figure 3.19 demonstrated the crop section of hemocytometer by manually and Figure 3.20 showed the result of counting in our cell counting program.

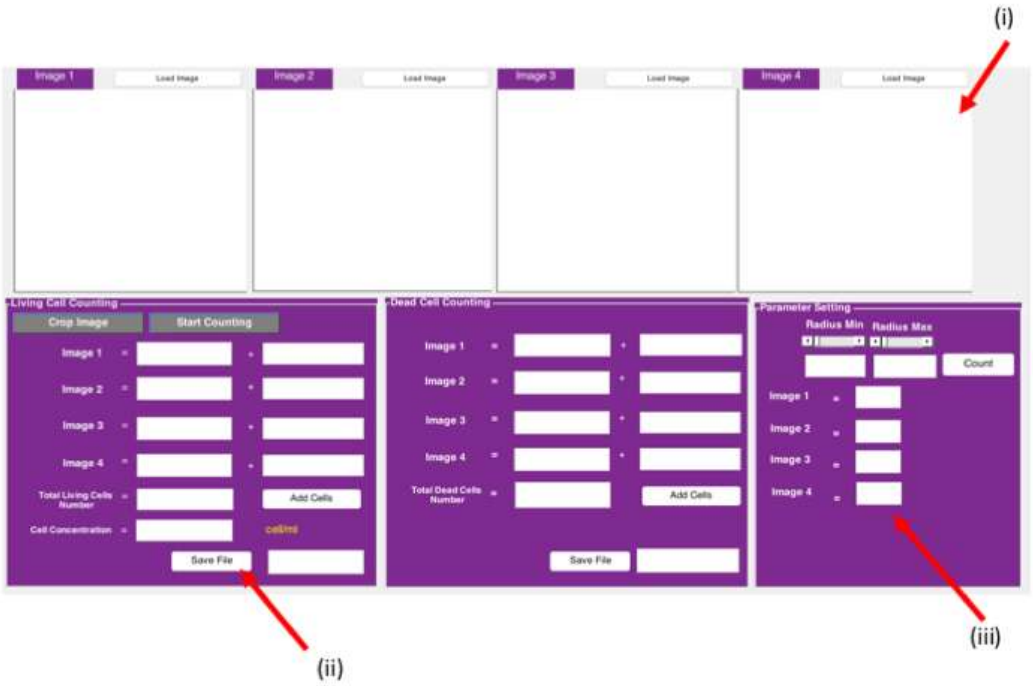


Figure 3.17 Graphical User Interface (GUI) of cell counting program: (i) Load image, (ii) Report document and (iii) Result box for total cell counting

Figure 3.18 Example of load images in GUI

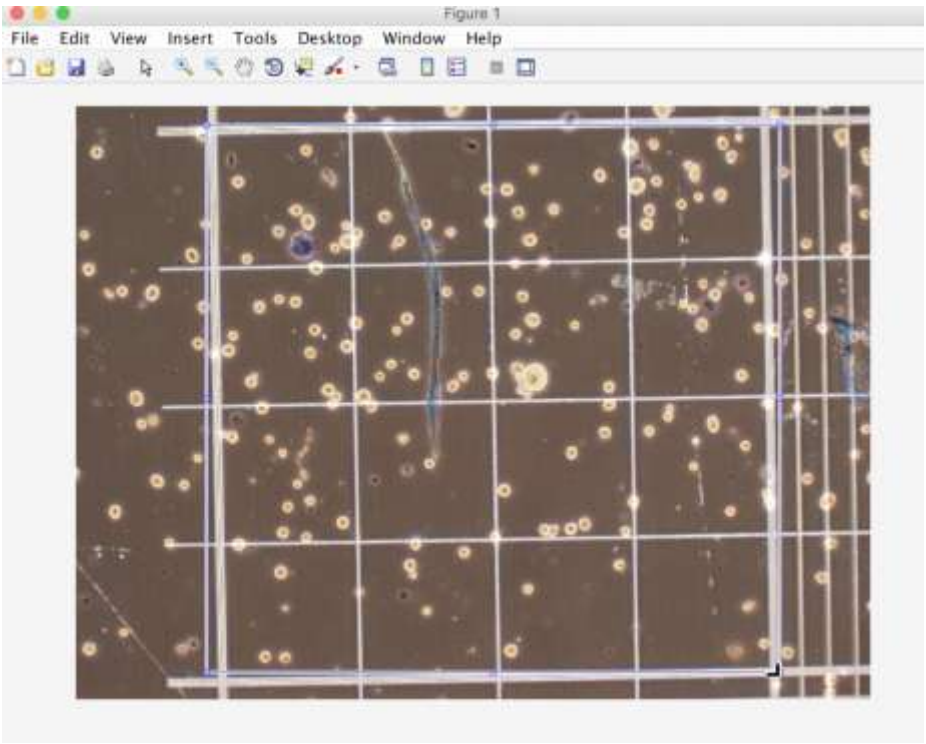


Figure 3.19 Example of cropped image section of original Trypan Blue stained image with grid lines in GUI

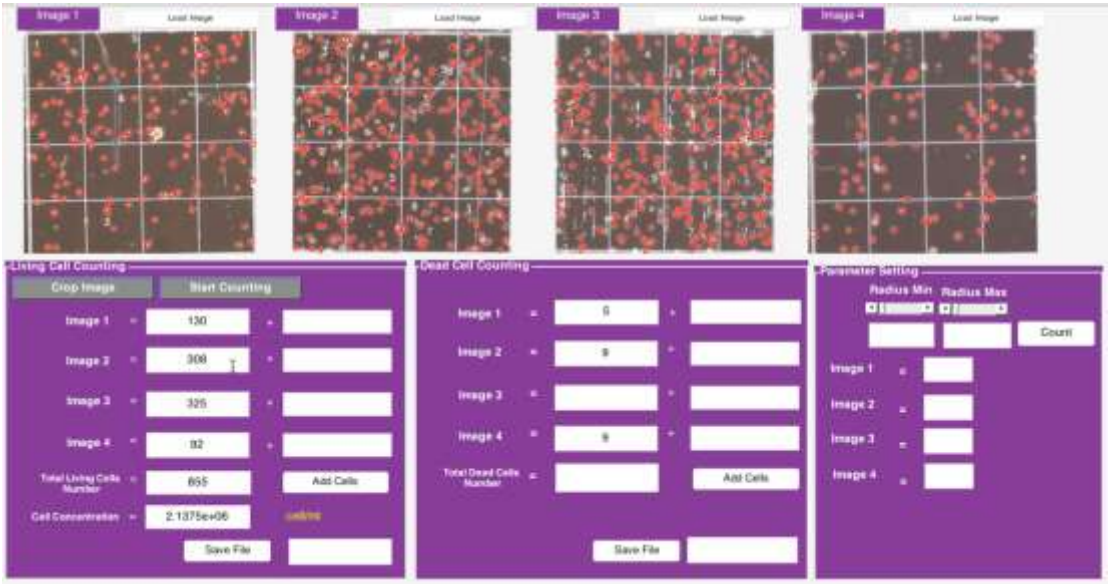


Figure 3.20 Example of cell counting results in from the program

Chapter 4

Discussion

This study presented the image analysis technique to count live and dead cells obtained from Trypan Blue stained microscopic images. As our results, the accuracy of live cells counting was very high (>85%) due to the good contrast between foreground and background after a saturated channel selection and the improvement from image noise reduction with a guided image filter. We experienced this high accuracy of live cells counting using similar sample images in our previous work using K-mean clustering technique(27). However, this new proposed method provided a higher accuracy. As the brightness of live cells is clearly different from the image background, this is the advantage for image thresholding for live cells detection. Piccinini and colleagues presented a high accuracy of live cells counting using an automated image mosaicing for A549 and KKP cell lines(14). The high accuracy of live cell counting in both their results and ours particularly relates to a clearly distinguishable shape of live cells and a good contrast to the background.

The number of dead cells from our counting approach was considerably less than the expert's counting as our high false negative. This might be caused by the ineffectiveness of dead cell segmentation. It should be noted that a segmentation result depends on the quality of preparation and the coloring of Trypan Blue stained microscopic images. Over staining and poor digitization could adversely affect the segmentation result (28, 29). These quandaries infrequently occur and can be remedied with a more rigorous

quality control during the sample preparation and image acquisition. Moreover, dead cells from apoptosis presented in fragments can lead to over counting (30). Therefore, counting on dead cells still has erroneous results much more than live cells counting. For example, the approach using automated image mosaicking had false positive results on dead cell counting compared to the experts much more than the live cells counting(14). This is quite a challenge to improve the accuracy of dead cells counting by improve either the image noise reduction or the image segmentation.

In the first approach, however, there are several limitations which effect on the counting accuracy. For example, we assumed that the cell diameters in all images are about the same size. In fact, the dead cells in our study had various diameters especially when cells became fragments from apoptosis. Therefore, some images had the number of dead cells counting by our approach greater than the results counted by the expert. Furthermore, the qualities of image such as the brightness of the image background and the image contrast can influence on the image processing and leads to lower accuracy of cell counting. As the Trypan Blue stained microscopic images had a dark background, the color of dead cells is quite similar to the background which requires several trials for the dead cells segmentation. Moreover, we applied the mean threshold value based on 36 images for dead cells segmentation. Therefore, it is necessary to recalculate the mean threshold value if the number of images or image background is changed. In addition, employing a large SE can oversimplify the image, while using too small SE does not always produce desirable results as many of the substructures within the large nuclei remain, affecting the segmentation

performance. Moreover, our cell counting program using the first approach is still needed to improve the accuracy for dead cells. By calculating the areas of all dead cells and assign the largest area of dead which is main dead cell and summation of other fragmented cells areas.

Furthermore, the second approach has been developed to improve the accuracy of cell counting from the first approach. In this approach, the cell segmentation and counting techniques for detecting live and dead cells of Trypan Blue stained microscopic images were investigated. The adaptive K-means clustering technique was applied to classify live and dead cells. It was found that adaptive K-means clustering required an accurate pre-processing process in order to obtain a highly accurate cell classification (42). If adequate pre-processing steps such as gray-scale conversion and morphological operations to remove grid lines are provided then an adequate classification is produced. ImageJ was used to count the Trypan Blue stained images of the cell line (MDA-MB-231) tested in this study (43). It was also found that the morphological operations structuring element was able to analyze the original image although if either larger or smaller pixels were used in the image the segmentation quality might be affected (44) . However, ImageJ lacks the ability to detect dead cells despite being a user-friendly image processing tools and the cell counter plug-in was not able to fully detect all live and dead cells nor could it distinguish dead cells from live cells and the background.

It was assumed that the cell diameters in all the images were approximately the same and most of the parameters used were standard for a Hough transform. Therefore, we set the minimum cell range (R_{min}) to 20 pixels, and the maximum (R_{max}) to 80 pixels in

order to ensure that the maximum radius covered a complete cell. However, these minimum and maximum radius pixel values can be adjusted according to cell types. The method used in this study currently takes about 20 seconds to count the cells in each image, which compares with manual counting which took approximately 2 minutes for one image. The cell counting program developed can therefore significantly reduce the time required for expert cell counting while providing high accuracy.

In spite of our counting program has accuracy, this second approach has some limitations. Adaptive K-means clustering algorithm operates by pre-processing steps such as gray scale and morphological operations, creating them affected to removing hemocytometer grid lines and invalid clustering.

Chapter 5

Conclusion

In this study, we presented a computer-assisted cell counting approach on Trypan Blue stained microscopic images of breast cancer cell line. The stained cell counting results indicated that the counting of Trypan Blue stained microscopic images using our approaches offers a remarkable accuracy, especially live cells counting. Our simple method can identify overlapping cells and count them quite correctly. The first approach focused on morphological operation of digital image processing and image segmentation while the second approach based on adaptive K-mean clustering algorithm which involved thresholding. Even there are several limitations from image itself and pre-processing and post-processing techniques but both automated counting approaches result in very time effectiveness to biologists and biomedical researchers. Furthermore, a graphic user interface (GUI) was created to provide a useful interface tool for biologists and biomedical researchers who are not familiar with image processing methods. However, the future work is still aiming to develop cell counting algorithm for higher accuracy for dead cells and other kinds of cell lines in cell culturing experiments.

References

1. Fang JY, Tan SJ, Wu YC, Yang Z, Hoang BX, Han B. From competency to dormancy: a 3D model to study cancer cells and drug responsiveness. J Transl Med. 2016 Feb 04;14:38.

2. Fesik SW. Promoting apoptosis as a strategy for cancer drug discovery. Nat Rev Cancer. 2005 Nov;5(11):876-85.

3. Kondo Y, Kanzawa T, Sawaya R, Kondo S. The role of autophagy in cancer development and response to therapy. Nat Rev Cancer. 2005 Sep;5(9):726-34.

4. Lowe SW, Lin AW. Apoptosis in cancer. Carcinogenesis. 2000;21(3):485-95.

5. Carpenter AE, Jones TR, Lamprecht MR, Clarke C, Kang IH, Friman O, et al. CellProfiler: image analysis software for identifying and quantifying cell phenotypes. Genome biology. 2006;7(10):R100.

6. Schindelin J, Arganda-Carreras I, Frise E, Kaynig V, Longair M, Pietzsch T, et al. Fiji: an open-source platform for biological-image analysis. Nat Methods. 2012 Jun 28;9(7):676-82.

7. Selinummi J, Seppala J, Yli-Harja O, Puhakka JA. Software for quantification of labeled bacteria from digital microscope images by automated image analysis. Biotechniques. 2005;39(6):859.

8. Grishagin IV. Automatic cell counting with ImageJ. Anal Biochem. 2015 Mar 15;473:63-5.

9. Vayrynen JP, Vornanen JO, Sajanti S, Bohm JP, Tuomisto A, Makinen MJ. An improved image analysis method for cell counting lends credibility to the prognostic significance of T cells in colorectal cancer. Virchows Arch. 2012 May;460(5):455-65.

10. Blatt RJ, Clark AN, Courtney J, Tully C, Tucker AL. Automated quantitative analysis of angiogenesis in the rat aorta model using Image-Pro Plus 4.1. Comput Methods Programs Biomed. 2004 Jul;75(1):75-9. 11. Narayan PJ, Gibbons HM, Mee EW, Faull RL, Dragunow M. High throughput quantification of cells with complex morphology in mixed cultures. J Neurosci Methods. 2007 Aug 30;164(2):339-49.

12. Sui D, Wang K, Park H, Chae J. Bright field microscopic cells counting method for BEVS using nonlinear convergence index sliding band filter. Biomedical engineering online. 2014;13(1):147.

13. Nazlibilek S, Karacor D, Ercan T, Sazli MH, Kalender O, Ege Y. Automatic segmentation, counting, size determination and classification of white blood cells. Measurement. 2014;55:58-65.

14. Piccinini F, Tesei A, Paganelli G, Zoli W, Bevilacqua A. Improving reliability of live/dead cell counting through automated image mosaicing. Comput Methods Programs Biomed. 2014;117(3):448-63.

15. Mouelhi A, Sayadi M, Fnaiech F, Mrad K, Romdhane KB. Automatic image segmentation of nuclear stained breast tissue sections using color active contour model and an improved watershed method. Biomedical Signal Processing and Control. 2013;8(5):421-36.

16. Zhu Y, Huang C. An improved median filtering algorithm for image noise reduction. Physics Procedia. 2012;25:609-16.

17. Soni T, Rathor N. Removal of High Density Impulse Noise using Efficient Median Filter for Digital Image. International Journal of Computer Applications. 2015;115(5).

18. Zeng H, Liu Y-z, Fan Y-m, Tang X. An improved algorithm for impulse noise by median filter. AASRI Procedia. 2012;1:68-73.

19. Sun C, Tang C, Zhu X, Li X, Wang L. An efficient method for salt-and-pepper noise removal based on shearlet transform and noise detection. AEU-International Journal of Electronics and Communications. 2015;69(12):1823-32.

20. Samantaray AK, Mallick P. Decision Based Adaptive Neighborhood Median Filter. Procedia Computer Science. 2015;48:222-7.

21. Mahmood NH, Mansor MA. Red blood cells estimation using Hough transform technique. Signal & Image Processing. 2012;3(2):53.

22. Liew LH, Lee BY, Chan M, editors. Cell detection for bee comb images using circular Hough transformation. Science and Social Research (CSSR), 2010 International Conference on; 2010: IEEE.

23. Hao S, Pan D, Guo Y, Hong R, Wang M. Image detail enhancement with spatially guided filters. Signal Processing. 2016;120:789-96.

24. He K, Sun J, Tang X. Guided Image Filtering. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2013;35(6):1397-409.

25. Otsu N. A threshold selection method from gray-level histograms. Automatica. 1975;11(285-296):237.

26. Veta M, van Diest PJ, Kornegoor R, Huisman A, Viergever MA, Pluim JP. Automatic nuclei segmentation in H&E stained breast cancer histopathology images. PLoS One. 2013;8(7):e70221.

27. Chobngam F, Kanokwiroon K, Chatpun S, Wichakool W, Limsiroratana S, Phukpattaranont P, editors. Preliminary results of death cell counting based on K-mean clustering. The 5th 2012 Biomedical Engineering International Conference; 2012 5-7 Dec. 2012.

28. Yagi Y, Gilbertson JR. Digital imaging in pathology: the case for standardization. J Telemed Telecare. 2005;11(3):109-16.

29. Osman MK, Ahmad F, Saad Z, Mashor MY, Jaafar H, editors. A genetic algorithmneural network approach for Mycobacterium tuberculosis detection in Ziehl-Neelsen stained tissue slide images. 2010 10th International Conference on Intelligent Systems Design and Applications; 2010 Nov. 29 2010-Dec. 1 2010.

30. Choi YH, Yoo YH. Taxol-induced growth arrest and apoptosis is associated with the upregulation of the Cdk inhibitor, p21WAF1/CIP1, in human breast cancer cells. Oncol Rep. 2012 Dec;28(6):2163-9.

31. Pollard JW, Walker JM. Basic cell culture protocols. Vol. 75. Springer Science & Business Media; 1997.

32. Lamprecht MR, Sabatini DM, Carpenter AE. CellProfilerTM: free, versatile software for automated biological image analysis. Biotechniques. 2007;42(1):71.

33. Preston Jr K. Image processing in medical microscopy. In: Proceedings of the Annual Symposium on Computer Application in Medical Care. American Medical Informatics Association; 1986. p. 213.

34. Malpica N, Ortiz de Solorzano C, Vaquero JJ, Santos A, Vallcorba I, Garcia-Sagredo JM, et al. Applying watershed algorithms to the segmentation of clustered nuclei. 1997;

35. Usaj M, Torkar D, Kanduser M, Miklavcic D. Cell counting tool parameters optimization approach for electroporation efficiency determination of attached cells in phase contrast images. J Microsc 2011 Mar;241(3):303–14.

36. Scotti F. Automatic morphological analysis for acute leukemia identification in peripheral blood microscope images. In: IEEE international conference on computational intelligence for measurement systems and applications. 2005.

37. Sahastrabuddhe AP. Counting of RBC and WBC using image processing: A Review. International Journal of Research in Engineering and Technology 2016; 5(5):356-60

38. Chourasiya S, Rani GU. Automatic red blood cell counting using watershed segmentation. Hemoglobin. 2014;14:17.

39. Sahastrabuddhe AP. Counting of Rbc and Wbc Using Image Processing: a Review. 2016;356–60.

40. Bhamare MMG, Patil DS. Automatic blood cell analysis by using digital image processing: a preliminary study. International Journal of Engineering Research and Technology. 2013;2(9):3137-41.

41. Maitra M, Kumar Gupta R, Mukherjee M. Detection and Counting of Red Blood Cells in Blood Cell Images using Hough Transform. Int J Comput Appl. 2012;53(16):13–7.

42. Patel BC, Sinha GR. An Adaptive K-means Clustering Algorithm for Breast Image Segmentation. Int J Comput Appl 2010;10(4):35–8.

43. Grishagin I V. Automatic cell counting with ImageJ. Anal Biochem. 2015 Mar 15;473:63–5.

44. Liew LH, Lee BY, Chan M. Cell detection for bee comb images using Circular Hough Transformation. 2010 International Conference on Science and Social Research (CSSR 2010). 2010. p. 191–5.

Appendix A

MATLAB CODE

```
function varargout = untitled2(varargin)
% UNTITLED2 MATLAB code for untitled2.fig
%      UNTITLED2, by itself, creates a new UNTITLED2 or raises the existing
%      singleton*.
%
%      H = UNTITLED2 returns the handle to a new UNTITLED2 or the handle to
%      the existing singleton*.
%
%      UNTITLED2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in UNTITLED2.M with the given input arguments.
%
%      UNTITLED2('Property','Value',...) creates a new UNTITLED2 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before untitled2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to untitled2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help untitled2

% Last Modified by GUIDE v2.5 09-May-2017 16:02:15

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @untitled2_OpeningFcn, ...
                   'gui_OutputFcn',  @untitled2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before untitled2 is made visible.
function untitled2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to untitled2 (see VARARGIN)

% Choose default command line output for untitled2
handles.output = hObject;
% handles.initial_parameter=varargin{1};
% handles.counter=0;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes untitled2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = untitled2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbuttonLoad1.
function pushbuttonLoad1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonLoad1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global I1

[filename pathname] = uigetfile('*.jpg;*.tif;*.png;*.gif','All Image Files');
complete = strcat(pathname,filename);

if pathname ~= 0
   I1 = imread(complete);
   handles.current_data1 = I1;
   axes(handles.axes1);
   image(I1);
   axis off
end
handles.output = hObject;
guidata(hObject, handles);
```

```
function cells1_Callback(hObject, eventdata, handles)
% hObject    handle to cells1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of cells1 as text
%        str2double(get(hObject,'String')) returns contents of cells1 as a double


% --- Executes during object creation, after setting all properties.
function cells1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cells1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbuttonAdd.
function pushbuttonAdd_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonAdd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

add1 = get(handles.add_cells1,'String');
if isempty(add1)
   add1 = 0;
else
   add1 = str2double(add1);
end
cells1 = str2double(get(handles.cells1,'String'))+ add1;
total = cells1;
set(handles.total_cells,'String',total)
%

add2 = get(handles.add_cells3,'String');
if isempty(add2)
   add2 = 0;
else
   add2 = str2double(add2);
end
cells3 = str2double(get(handles.cells3,'String'))+ add2;
total = cells3;
```

```
set(handles.total_cells,'String',total)
%

add3 = get(handles.add_cells5,'String');
if isempty(add3)
   add3 = 0;
else
   add3 = str2double(add3);
end
cells5 = str2double(get(handles.cells5,'String'))+ add3;
total = cells5;
set(handles.total_cells,'String',total)
%
add4 = get(handles.add_cells7,'String');
if isempty(add4)
   add4 = 0;
else
   add4 = str2double(add4);
end
cells1 = str2double(get(handles.cells1,'String'))+ add1;
cells3 = str2double(get(handles.cells3,'String'))+ add2;
cells5 = str2double(get(handles.cells5,'String'))+ add3;
cells7 = str2double(get(handles.cells7,'String'))+ add4;


total_living = cells1 + cells3 + cells5 + cells7;

set(handles.total_cells,'String',total_living)

concentration = (total_living*10000)/4;
set(handles.cell_concentration,'String',concentration)


function add_cells1_Callback(hObject, eventdata, handles)
% hObject    handle to add_cells1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of add_cells1 as text
%        str2double(get(hObject,'String')) returns contents of add_cells1 as a double


% --- Executes during object creation, after setting all properties.
function add_cells1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to add_cells1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function total_cells_Callback(hObject, eventdata, handles)
% hObject    handle to total_cells (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of total_cells as text
%        str2double(get(hObject,'String')) returns contents of total_cells as a double


% --- Executes during object creation, after setting all properties.
function total_cells_CreateFcn(hObject, eventdata, handles)
% hObject    handle to total_cells (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function cells2_Callback(hObject, eventdata, handles)
% hObject    handle to cells2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of cells2 as text
%        str2double(get(hObject,'String')) returns contents of cells2 as a double


% --- Executes during object creation, after setting all properties.
function cells2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cells2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
```

```
end



function add_cells2_Callback(hObject, eventdata, handles)
% hObject    handle to add_cells2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of add_cells2 as text
%        str2double(get(hObject,'String')) returns contents of add_cells2 as a double


% --- Executes during object creation, after setting all properties.
function add_cells2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to add_cells2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbuttonAddDead.
function pushbuttonAddDead_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonAddDead (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
add1 = get(handles.add_cells2,'String');
if isempty(add1)
   add1 = 0;
else
   add1 = str2double(add1);
end
cells2 = str2double(get(handles.cells2,'String'))+ add1;
total = cells2;
set(handles.total_dead_cells,'String',total)
%

add2 = get(handles.add_cells4,'String');
if isempty(add2)
   add2 = 0;
else
   add2 = str2double(add2);
end
cells4 = str2double(get(handles.cells4,'String'))+ add2;
total = cells4;
```

```
set(handles.total_dead_cells,'String',total)
%

add3 = get(handles.add_cells6,'String');
if isempty(add3)
   add3 = 0;
else
   add3 = str2double(add3);
end
cells6 = str2double(get(handles.cells6,'String'))+ add3;
total = cells6;
set(handles.total_dead_cells,'String',total)
%
add4 = get(handles.add_cells8,'String');
if isempty(add4)
   add4 = 0;
else
   add4 = str2double(add4);
end
cells2 = str2double(get(handles.cells2,'String'))+ add1;
cells4 = str2double(get(handles.cells4,'String'))+ add2;
cells6 = str2double(get(handles.cells6,'String'))+ add3;
cells8 = str2double(get(handles.cells8,'String'))+ add4;


total = cells2 + cells4 + cells6 + cells8;

set(handles.total_dead_cells,'String',total)



function total_dead_cells_Callback(hObject, eventdata, handles)
% hObject    handle to total_dead_cells (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of total_dead_cells as text
%        str2double(get(hObject,'String')) returns contents of total_dead_cells as a double


% --- Executes during object creation, after setting all properties.
function total_dead_cells_CreateFcn(hObject, eventdata, handles)
% hObject    handle to total_dead_cells (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
```

```
end


function SampleName_Callback(hObject, eventdata, handles)
% hObject    handle to SampleName (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of SampleName as text
%        str2double(get(hObject,'String')) returns contents of SampleName as a double


% --- Executes during object creation, after setting all properties.
function SampleName_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SampleName (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbuttonSave.
function pushbuttonSave_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonSave (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global cells1
global cells3
global cells5
global cells7
global total_living
global concentration

file = 'TrypanBlueDye.xlsx';

Name = get(handles.SampleName,'String');


N={Name, cells1, cells3, cells5, cells7, total_living, concentration};


%Read excel file return data on it and dimension
[coef, texte]=csvread(file,1); % old xlsread cannot use on mac use csvread for mac
```

```
%return the number of lines already used
[l, c]=size(texte);

%define the line we must right on
d=[char('A'),num2str(l+1)];

%right the new data in this line
csvwrite(file,N,1,d);

set(handles.Saved,'String','Data Saved !')



function cells4_Callback(hObject, eventdata, handles)
% hObject    handle to cells4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of cells4 as text
%        str2double(get(hObject,'String')) returns contents of cells4 as a double


% --- Executes during object creation, after setting all properties.
function cells4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cells4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function add_cells4_Callback(hObject, eventdata, handles)
% hObject    handle to add_cells4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of add_cells4 as text
%        str2double(get(hObject,'String')) returns contents of add_cells4 as a double


% --- Executes during object creation, after setting all properties.
function add_cells4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to add_cells4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function cells6_Callback(hObject, eventdata, handles)
% hObject    handle to cells6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of cells6 as text
%        str2double(get(hObject,'String')) returns contents of cells6 as a double


% --- Executes during object creation, after setting all properties.
function cells6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cells6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function add_cells6_Callback(hObject, eventdata, handles)
% hObject    handle to add_cells6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of add_cells6 as text
%        str2double(get(hObject,'String')) returns contents of add_cells6 as a double


% --- Executes during object creation, after setting all properties.
function add_cells6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to add_cells6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function cells8_Callback(hObject, eventdata, handles)
% hObject    handle to cells8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of cells8 as text
%        str2double(get(hObject,'String')) returns contents of cells8 as a double


% --- Executes during object creation, after setting all properties.
function cells8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cells8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function add_cells8_Callback(hObject, eventdata, handles)
% hObject    handle to add_cells8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of add_cells8 as text
%        str2double(get(hObject,'String')) returns contents of add_cells8 as a double


% --- Executes during object creation, after setting all properties.
function add_cells8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to add_cells8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end



function edit29_Callback(hObject, eventdata, handles)
% hObject    handle to edit29 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit29 as text
%        str2double(get(hObject,'String')) returns contents of edit29 as a double


% --- Executes during object creation, after setting all properties.
function edit29_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit29 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



function cells3_Callback(hObject, eventdata, handles)
% hObject    handle to cells3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of cells3 as text
%        str2double(get(hObject,'String')) returns contents of cells3 as a double


% --- Executes during object creation, after setting all properties.
function cells3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cells3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function add_cells3_Callback(hObject, eventdata, handles)
% hObject    handle to add_cells3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of add_cells3 as text
%        str2double(get(hObject,'String')) returns contents of add_cells3 as a double


% --- Executes during object creation, after setting all properties.
function add_cells3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to add_cells3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function cells5_Callback(hObject, eventdata, handles)
% hObject    handle to cells5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of cells5 as text
%        str2double(get(hObject,'String')) returns contents of cells5 as a double


% --- Executes during object creation, after setting all properties.
function cells5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cells5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function add_cells5_Callback(hObject, eventdata, handles)
% hObject    handle to add_cells5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of add_cells5 as text
%        str2double(get(hObject,'String')) returns contents of add_cells5 as a double


% --- Executes during object creation, after setting all properties.
function add_cells5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to add_cells5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function cells7_Callback(hObject, eventdata, handles)
% hObject    handle to cells7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of cells7 as text
%        str2double(get(hObject,'String')) returns contents of cells7 as a double


% --- Executes during object creation, after setting all properties.
function cells7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cells7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
```

```
end


function add_cells7_Callback(hObject, eventdata, handles)
% hObject    handle to add_cells7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of add_cells7 as text
%        str2double(get(hObject,'String')) returns contents of add_cells7 as a double


% --- Executes during object creation, after setting all properties.
function add_cells7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to add_cells7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function cell_concentration_Callback(hObject, eventdata, handles)
% hObject    handle to cell_concentration (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of cell_concentration as text
%        str2double(get(hObject,'String')) returns contents of cell_concentration as a double


% --- Executes during object creation, after setting all properties.
function cell_concentration_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cell_concentration (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in pushbuttonLoad2.
function pushbuttonLoad2_Callback(hObject, eventdata, handles)
global I2

[filename pathname] = uigetfile('*.jpg;*.tif;*.png;*.gif','All Image Files');
complete = strcat(pathname,filename);

if pathname ~= 0
   I2 = imread(complete);
   handles.current_data2 = I2;
   axes(handles.axes2);
   image(I2);
   axis off
end

handles.output = hObject;
guidata(hObject, handles);

% --- Executes on button press in pushbuttonLoad3.
function pushbuttonLoad3_Callback(hObject, eventdata, handles)
global I3

[filename pathname] = uigetfile('*.jpg;*.tif;*.png;*.gif','All Image Files');
complete = strcat(pathname,filename);

if pathname ~= 0
   I3 = imread(complete);
   handles.current_data3 = I3;
   axes(handles.axes3);
   image(I3);
   axis off
end

handles.output = hObject;
%Analysis(handles.I);
%Update  hndles  structure
guidata(hObject, handles);
% --- Executes on button press in pushbuttonLoad4.
function pushbuttonLoad4_Callback(hObject, eventdata, handles)
global I4

[filename pathname] = uigetfile('*.jpg;*.tif;*.png;*.gif','All Image Files');
complete = strcat(pathname,filename);

if pathname ~= 0
   I4 = imread(complete);
   handles.current_data4 = I4;
   axes(handles.axes4);
   image(I4);
   axis off
end
```

```
handles.output = hObject;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function pushbuttonLoad2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbuttonLoad2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes on slider movement.
function initial_value_Callback(hObject, eventdata, handles)
valor=get(hObject,'Value');
set(handles.edit33,'String',valor);


% --- Executes during object creation, after setting all properties.
function initial_value_CreateFcn(hObject, eventdata, handles)
% hObject    handle to initial_value (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on slider movement.
function total_value_Callback(hObject, eventdata, handles)
valor=get(hObject,'Value');
set(handles.edit34,'String',valor);



% --- Executes during object creation, after setting all properties.
function total_value_CreateFcn(hObject, eventdata, handles)
% hObject    handle to total_value (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end



function edit33_Callback(hObject, eventdata, handles)
% hObject    handle to edit33 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit33 as text
%        str2double(get(hObject,'String')) returns contents of edit33 as a double


% --- Executes during object creation, after setting all properties.
function edit33_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit33 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function edit34_Callback(hObject, eventdata, handles)
% hObject    handle to edit34 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit34 as text
%        str2double(get(hObject,'String')) returns contents of edit34 as a double


% --- Executes during object creation, after setting all properties.
function edit34_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit34 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in new_count.
function new_count_Callback(hObject, eventdata, handles)
% hObject    handle to new_count (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

rmin = get(handles.edit33,'String');
```

```
% if isempty(rmin)
%     rmin = 0;
% else
%     rmin = str2double(rmin);
% end
% new_rmin1 = str2double(get(handles.new_rmin1,'String'))+rmin;
new_rmin1=str2double(rmin);
set(handles.total_cell1,'String',new_rmin1)

rmax = get(handles.edit34,'String');
% if isempty(rmax)
%     rmax = 0;
% else
%     rmax = str2double(rmax);
% end
% new_rmax1 = str2double(get(handles.new_rmax1,'String'))+ rmax;
new_rmax1=str2double(rmax);
% set(handles.total_radius2,'String',new_rmax1)
global I1
 global I2
 global I3
 global I4



grayImage=rgb2gray(I1);
Iobrcbr=grayImage;
imData=reshape(Iobrcbr,[],1);
imData=double(imData);
[IDX, nm]=adaptcluster_kmeans(imData);
imIDX=reshape(IDX,size(Iobrcbr));

axes(handles.axes1);
imshow(I1);
axis off
% title('Living and Dead Cells','FontSize',fontSize);
[centersAll, radiiAll] = imfindcircles(imIDX==3,[new_rmin1
new_rmax1],'ObjectPolarity','dark');
viscircles(centersAll, radiiAll,'EdgeColor','b');
% viscircles(centersAll, radiiAll,'LineStyle','--');
length([centersAll, radiiAll]);
total_radius1 = length([centersAll, radiiAll]);

set(handles.total_cell1,'String',total_radius1)

grayImage=rgb2gray(I2);
Iobrcbr=grayImage;
imData=reshape(Iobrcbr,[],1);
imData=double(imData);
[IDX, nm]=adaptcluster_kmeans(imData);
imIDX=reshape(IDX,size(Iobrcbr));
```

```
axes(handles.axes2);
imshow(I2);
axis off
% title('Living and Dead Cells','FontSize',fontSize);
[centersAll, radiiAll] = imfindcircles(imIDX==3,[new_rmin1
new_rmax1],'ObjectPolarity','dark');
viscircles(centersAll, radiiAll,'EdgeColor','b');
% viscircles(centersAll, radiiAll,'LineStyle','--');
length([centersAll, radiiAll]);
total_radius1 = length([centersAll, radiiAll]);

set(handles.total_cell2,'String',total_radius1)

grayImage=rgb2gray(I3);
Iobrcbr=grayImage;
imData=reshape(Iobrcbr,[],1);
imData=double(imData);
[IDX, nm]=adaptcluster_kmeans(imData);
imIDX=reshape(IDX,size(Iobrcbr));

axes(handles.axes3);
imshow(I3);
axis off
% title('Living and Dead Cells','FontSize',fontSize);
[centersAll, radiiAll] = imfindcircles(imIDX==3,[new_rmin1
new_rmax1],'ObjectPolarity','dark');
viscircles(centersAll, radiiAll,'EdgeColor','b');
% viscircles(centersAll, radiiAll,'LineStyle','--');
length([centersAll, radiiAll]);
total_radius1 = length([centersAll, radiiAll]);

set(handles.total_cell3,'String',total_radius1)

grayImage=rgb2gray(I4);
Iobrcbr=grayImage;
imData=reshape(Iobrcbr,[],1);
imData=double(imData);
[IDX, nm]=adaptcluster_kmeans(imData);
imIDX=reshape(IDX,size(Iobrcbr));

axes(handles.axes4);
imshow(I4);
axis off
% title('Living and Dead Cells','FontSize',fontSize);
[centersAll, radiiAll] = imfindcircles(imIDX==3,[new_rmin1
new_rmax1],'ObjectPolarity','dark');
viscircles(centersAll, radiiAll,'EdgeColor','b');
% viscircles(centersAll, radiiAll,'LineStyle','--');
length([centersAll, radiiAll]);
total_radius1 = length([centersAll, radiiAll]);
```

```
set(handles.total_cell4,'String',total_radius1)


% % axis off

function edit35_Callback(hObject, eventdata, handles)
% hObject    handle to edit35 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit35 as text
%        str2double(get(hObject,'String')) returns contents of edit35 as a double


% --- Executes during object creation, after setting all properties.
function edit35_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit35 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbuttonLoad2.
function pushbutton12_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonLoad2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbuttonLoad3.
function pushbutton13_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonLoad3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbuttonLoad4.
function pushbutton14_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonLoad4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbuttonCrop.
```

```
function pushbuttonCrop_Callback(hObject, eventdata, handles)
handles.output = hObject;
guidata(hObject, handles);

 global I1
 global I2
 global I3
 global I4

num_crop = 4;
 crop_imge = cell(1,4);
 crop_imge{1} = I1;
 crop_imge{2} = I2;
 crop_imge{3} = I3;
 crop_imge{4} = I4;

 for n = 1:num_crop
    figure,imshow(crop_imge{n}),title('image')
    image_crop = imcrop(crop_imge{n});
    crop_image{n} =  image_crop;
    clear  image_crop
    close ;
 end
    I1 = crop_image{1};
    I2 = crop_image{2};
    I3 = crop_image{3};
    I4 = crop_image{4};

axes(handles.axes1);
image(I1);
axis off

axes(handles.axes2);
image(I2);
axis off

axes(handles.axes3);
image(I3);
axis off

axes(handles.axes4);
image(I4);
axis off

%  save  mydata1.mat crop1 crop2 crop3 crop4
  clear img imag_crop crop_image num_crop
% % The blanks filled with the number of Viable Cells counted


% --- Executes on button press in pushbuttonProcessed.
function pushbuttonProcessed_Callback(hObject, eventdata, handles)
```

```
handles.output = hObject;
guidata(hObject, handles);

% set(handles.Saved,'String','')

 global I1
 global I2
 global I3
 global I4

 global cells1
 global cells3
 global cells5
 global cells7
 global total_living
 global concentration


grayImage=rgb2gray(I1);
Iobrcbr=grayImage;
imData=reshape(Iobrcbr,[],1);
imData=double(imData);
[IDX, nm]=adaptcluster_kmeans(imData);
imIDX=reshape(IDX,size(Iobrcbr));
% figure,imshow(imIDX,[]),title('Index Image');
%
% figure,imshow(imIDX==1,[]);title('Cluster 1');
% figure,imshow(imIDX==2,[]);title('Cluster 2');
% figure,imshow(imIDX==3,[]);title('Cluster 3');

% figure
% imshow(grayImage,[]), title('Gray')
% handles.output=hObject;
% guidata(hObject, handles);
% axes(handles.axes2);
% axis off


axes(handles.axes1);
imshow(I1);
axis off
% title('Living and Dead Cells','FontSize',fontSize);
[centersAll, radiiAll] = imfindcircles(imIDX==3,[20 80],'ObjectPolarity','dark');
viscircles(centersAll, radiiAll,'EdgeColor','b');
viscircles(centersAll, radiiAll,'LineStyle','-');
length([centersAll, radiiAll]);
cells1 = length([centersAll, radiiAll]);

set(handles.cells1,'String',cells1)
%
BW2 = bwareaopen(imIDX==1, 400);
```

```
% figure,imshow(BW2,[]);title('Remove noise');

[L1,num1]=bwlabel(BW2);
STATS = regionprops(L1,'Centroid','Area');
auxA=zeros(num1,1);
radius=zeros(num1,1);
% figure;
% axes(handles.axes2);
% imshow(cropImage);title('result'); hold on;
% imshow(cropImage);
for k=1:num1
   aux=STATS(k).Centroid;
   text(aux(1),aux(2),num2str(k),'FontSize',14,'Color','w');
   auxA(k,1)=STATS(k).Area;
   radius(k,1)= sqrt(auxA(k,1)/pi);
end
% imcontour(BW2,'Color','g');
% figure,imshow(cropImage); title('original');
% [(1:num1)' radius]

handles.output=hObject;
guidata(hObject, handles);
cells2 = numel(STATS);

set(handles.cells2,'String',cells2)


grayImage=rgb2gray(I2);
Iobrcbr=grayImage;
imData=reshape(Iobrcbr,[],1);
imData=double(imData);
[IDX, nm]=adaptcluster_kmeans(imData);
imIDX=reshape(IDX,size(Iobrcbr));
% figure,imshow(imIDX,[]),title('Index Image');
%
% figure,imshow(imIDX==1,[]);title('Cluster 1');
% figure,imshow(imIDX==2,[]);title('Cluster 2');
% figure,imshow(imIDX==3,[]);title('Cluster 3');

% figure
% imshow(grayImage,[]), title('Gray')
% handles.output=hObject;
% guidata(hObject, handles);
% axes(handles.axes2);
% axis off
axes(handles.axes2);
imshow(I2);
axis off
% title('Living and Dead Cells','FontSize',fontSize);
[centersAll, radiiAll] = imfindcircles(imIDX==3,[20 80],'ObjectPolarity','dark');
```

```
viscircles(centersAll, radiiAll,'EdgeColor','b');
viscircles(centersAll, radiiAll,'LineStyle','-');
length([centersAll, radiiAll]);
cells3 = length([centersAll, radiiAll]);

set(handles.cells3,'String',cells3)

BW2 = bwareaopen(imIDX==1, 400);

% figure,imshow(BW2,[]);title('Remove noise');

[L1,num1]=bwlabel(BW2);
STATS = regionprops(L1,'Centroid','Area');
auxA=zeros(num1,1);
radius=zeros(num1,1);
% figure;
% axes(handles.axes2);
% imshow(cropImage);title('result'); hold on;
% imshow(cropImage);
for k=1:num1
   aux=STATS(k).Centroid;
   text(aux(1),aux(2),num2str(k),'FontSize',14,'Color','w');
   auxA(k,1)=STATS(k).Area;
   radius(k,1)= sqrt(auxA(k,1)/pi);
end
% imcontour(BW2,'Color','g');
% figure,imshow(cropImage); title('original');
% [(1:num1)' radius]

handles.output=hObject;
guidata(hObject, handles);
cells4 = numel(STATS);
set(handles.cells4,'String',cells4)
grayImage=rgb2gray(I3);
Iobrcbr=grayImage;
imData=reshape(Iobrcbr,[],1);
imData=double(imData);
[IDX, nm]=adaptcluster_kmeans(imData);
imIDX=reshape(IDX,size(Iobrcbr));
% figure,imshow(imIDX,[]),title('Index Image');
%
% figure,imshow(imIDX==1,[]);title('Cluster 1');
% figure,imshow(imIDX==2,[]);title('Cluster 2');
% figure,imshow(imIDX==3,[]);title('Cluster 3');

% figure
% imshow(grayImage,[]), title('Gray')
% handles.output=hObject;
% guidata(hObject, handles);
% axes(handles.axes2);
% axis off
```

```
axes(handles.axes3);
imshow(I3);
axis off
% title('Living and Dead Cells','FontSize',fontSize);
[centersAll, radiiAll] = imfindcircles(imIDX==3,[20 80],'ObjectPolarity','dark');
viscircles(centersAll, radiiAll,'EdgeColor','b');
viscircles(centersAll, radiiAll,'LineStyle','-');
length([centersAll, radiiAll]);
cells5 = length([centersAll, radiiAll]);

set(handles.cells5,'String',cells5)
% %
BW2 = bwareaopen(imIDX==1, 400);

% figure,imshow(BW2,[]);title('Remove noise');

[L1,num1]=bwlabel(BW2);
STATS = regionprops(L1,'Centroid','Area');
auxA=zeros(num1,1);
radius=zeros(num1,1);
% figure;
% axes(handles.axes2);
% imshow(cropImage);title('result'); hold on;
% imshow(cropImage);
for k=1:num1
   aux=STATS(k).Centroid;
   text(aux(1),aux(2),num2str(k),'FontSize',14,'Color','w');
   auxA(k,1)=STATS(k).Area;
   radius(k,1)= sqrt(auxA(k,1)/pi);
end
% imcontour(BW2,'Color','g');
% figure,imshow(cropImage); title('original');
% [(1:num1)' radius]
grayImage=rgb2gray(I4);
Iobrcbr=grayImage;
imData=reshape(Iobrcbr,[],1);
imData=double(imData);
[IDX, nm]=adaptcluster_kmeans(imData);
imIDX=reshape(IDX,size(Iobrcbr));
% % % figure,imshow(imIDX,[]),title('Index Image');
%
% figure,imshow(imIDX==1,[]);title('Cluster 1');
% figure,imshow(imIDX==2,[]);title('Cluster 2');
% figure,imshow(imIDX==3,[]);title('Cluster 3');

% figure
% imshow(grayImage,[]), title('Gray')
% handles.output=hObject;
% guidata(hObject, handles);
% axes(handles.axes2);
% axis off
```

```
axes(handles.axes4);
imshow(I4);
axis off
rmin=20;
rmax=80;
% title('Living and Dead Cells','FontSize',fontSize);
[centersAll, radiiAll] = imfindcircles(imIDX==3,[rmin rmax],'ObjectPolarity','dark');
viscircles(centersAll, radiiAll,'EdgeColor','b');
viscircles(centersAll, radiiAll,'LineStyle','-');
length([centersAll, radiiAll]);
cells7 = length([centersAll, radiiAll]);

set(handles.cells7,'String',cells7)
% %
% % BW2 = bwareaopen(imIDX==1, 300);

% figure,imshow(BW2,[]);title('Remove noise');

[L1,num1]=bwlabel(BW2);
STATS = regionprops(L1,'Centroid','Area');
auxA=zeros(num1,1);
radius=zeros(num1,1);
% figure;
axes(handles.axes2);
% imshow(cropImage);title('result'); hold on;
% imshow(cropImage);
for k=1:num1
   aux=STATS(k).Centroid;
   text(aux(1),aux(2),num2str(k),'FontSize',14,'Color','w');
   auxA(k,1)=STATS(k).Area;
   radius(k,1)= sqrt(auxA(k,1)/pi);
end
% imcontour(BW2,'Color','g');
% figure,imshow(cropImage); title('original');
% [(1:num1)' radius]

handles.output=hObject;
guidata(hObject, handles);
cells8 = numel(STATS);

set(handles.cells8,'String',cells8)
total_living = cells1 + cells3 + cells5 + cells7;
set(handles.total_cells,'String',total_living)

concentration = (total_living*10000)/4;
set(handles.cell_concentration,'String',concentration)
function [lb,center] = adaptcluster_kmeans(im)

% This code is written to implement kmeans clustering for segmenting any
% Gray or Color image. There is no requirement to mention the number of cluster for
% clustering.
```

```
% IM - is input image to be clustered.
% LB - is labeled image (Clustered Image).
% CENTER - is array of cluster centers.
% Execution of this code is very fast.
% It generates consistent output for same image.

% Written by Ankit Dixit.
% January-2014.

if size(im,3)>1
  [lb,center] = ColorClustering(im); % Check Image is Gray or not.
else
   [lb,center] = GrayClustering(im);
end


function [lb,center] = GrayClustering(gray)
gray = double(gray);
array = gray(:); % Copy value into an array.
% distth = 25;
i = 0;j=0; % Intialize iteration Counters.
% tic
while(true)
   seed = mean(array); % Initialize seed Point.
   i = i+1; %Increment Counter for each iteration.
   while(true)
      j = j+1; % Initialize Counter for each iteration.
      dist = (sqrt((array-seed).^2)); % Find distance between Seed and Gray Value.
      distth = (sqrt(sum((array-seed).^2)/numel(array)));% Find bandwidth for Cluster Center.
      %        distth = max(dist(:))/5;
      qualified = dist<distth;% Check values are in selected Bandwidth or not.
      newseed = mean(array(qualified));% Update mean.

      if isnan(newseed) % Check mean is not a NaN value.
         break;
      end

      if seed == newseed || j>10 % Condition for convergence and maximum iteration.
         j=0;
         array(qualified) = [];% Remove values which have assigned to a cluster.
         center(i) = newseed; % Store center of cluster.
         break;
      end
      seed = newseed;% Update seed.
   end

   if isempty(array) || i>10 % Check maximum number of clusters.
      i = 0; % Reset Counter.
      break;
   end
```

```
end
% toc

center = sort(center); % Sort Centers.
newcenter = diff(center);% Find out Difference between two consecutive Centers.
intercluster = (max(gray(:)/10));% Findout Minimum distance between two cluster Centers.
center(newcenter<=intercluster)=[];% Discard Cluster centers less than distance.

% Make a clustered image using these centers.

vector = repmat(gray(:),[1,numel(center)]); % Replicate vector for parallel operation.
centers = repmat(center,[numel(gray),1]);

distance = ((vector-centers).^2);% Find distance between center and pixel value.
[~,lb] = min(distance,[],2);% Choose cluster index of minimum distance.
lb = reshape(lb,size(gray));% Reshape the labelled index vector.


function [lb,center] = ColorClustering(im)

im = double(im);
red = im(:,:,1); green = im(:,:,2); blue = im(:,:,3);

array = [red(:),green(:),blue(:)];
% distth = 25;
i = 0;j=0;
tic
while(true)

   seed(1) = mean(array(:,1));
   seed(2) = mean(array(:,2));
   seed(3) = mean(array(:,3));

   i = i+1;
   while(true)
      j = j+1;

      seedvec = repmat(seed,[size(array,1),1]);

      dist = sum((sqrt((array-seedvec).^2)),2);

       distth = 0.25*max(dist);
      qualified = dist<distth;

      newred = array(:,1);
      newgreen = array(:,2);
      newblue = array(:,3);

      newseed(1) = mean(newred(qualified));
      newseed(2) = mean(newgreen(qualified));
      newseed(3) = mean(newblue(qualified));
```

```
        if isnan(newseed)
            break;
        end

        if (seed == newseed) | j>10
            j=0;
            array(qualified,:) = [];
            center(i,:) = newseed;
            %           center(2,i) = nnz(qualified);
            break;
        end
        seed = newseed;
    end

    if isempty(array) || i>10
        i = 0;
        break;
    end

end
toc
centers = sqrt(sum((center.^2),2));
[centers,idx]= sort(centers);


while(true)
newcenter = diff(centers);
intercluster =25; %(max(gray(:)/10));
a = (newcenter<=intercluster);
% center(a,:)=[];
% centers = sqrt(sum((center.^2),2));
centers(a,:) = [];
idx(a,:)=[];
% center(a,:)=0;
if nnz(a)==0
    break;
end

end
center1 = center;
center =center1(idx,:);
% [~,idxsort] = sort(centers) ;
vecred = repmat(red(:),[1,size(center,1)]);
vecgreen = repmat(green(:),[1,size(center,1)]);
vecblue = repmat(blue(:),[1,size(center,1)]);

distred = (vecred - repmat(center(:,1)',[numel(red),1])).^2;
distgreen = (vecgreen - repmat(center(:,2)',[numel(red),1])).^2;
distblue = (vecblue - repmat(center(:,3)',[numel(red),1])).^2;
```

```
distance = sqrt(distred+distgreen+distblue);
[~,label_vector] = min(distance,[],2);
lb = reshape(label_vector,size(red));



function new_rmin1_Callback(hObject, eventdata, handles)
% hObject    handle to new_rmin1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of new_rmin1 as text
%        str2double(get(hObject,'String')) returns contents of new_rmin1 as a double


% --- Executes during object creation, after setting all properties.
function new_rmin1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to new_rmin1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function new_rmax1_Callback(hObject, eventdata, handles)
% hObject    handle to new_rmax1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of new_rmax1 as text
%        str2double(get(hObject,'String')) returns contents of new_rmax1 as a double


% --- Executes during object creation, after setting all properties.
function new_rmax1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to new_rmax1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

```
function total_cell1_Callback(hObject, eventdata, handles)
% hObject    handle to total_cell1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of total_cell1 as text
%        str2double(get(hObject,'String')) returns contents of total_cell1 as a double


% --- Executes during object creation, after setting all properties.
function total_cell1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to total_cell1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function total_radius2_Callback(hObject, eventdata, handles)
% hObject    handle to total_radius2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of total_radius2 as text
%        str2double(get(hObject,'String')) returns contents of total_radius2 as a double


% --- Executes during object creation, after setting all properties.
function total_radius2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to total_radius2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

```
function new_rmin2_Callback(hObject, eventdata, handles)
% hObject    handle to new_rmin2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of new_rmin2 as text
%        str2double(get(hObject,'String')) returns contents of new_rmin2 as a double


% --- Executes during object creation, after setting all properties.
function new_rmin2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to new_rmin2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function new_rmax2_Callback(hObject, eventdata, handles)
% hObject    handle to new_rmax2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of new_rmax2 as text
%        str2double(get(hObject,'String')) returns contents of new_rmax2 as a double


% --- Executes during object creation, after setting all properties.
function new_rmax2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to new_rmax2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function total_cell2_Callback(hObject, eventdata, handles)
% hObject    handle to total_cell2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of total_cell2 as text
%        str2double(get(hObject,'String')) returns contents of total_cell2 as a double


% --- Executes during object creation, after setting all properties.
function total_cell2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to total_cell2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function total_radius4_Callback(hObject, eventdata, handles)
% hObject    handle to total_radius4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of total_radius4 as text
%        str2double(get(hObject,'String')) returns contents of total_radius4 as a double


% --- Executes during object creation, after setting all properties.
function total_radius4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to total_radius4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function new_rmin3_Callback(hObject, eventdata, handles)
% hObject    handle to new_rmin3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of new_rmin3 as text
%        str2double(get(hObject,'String')) returns contents of new_rmin3 as a double
```

```
% --- Executes during object creation, after setting all properties.
function new_rmin3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to new_rmin3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function new_rmax3_Callback(hObject, eventdata, handles)
% hObject    handle to new_rmax3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of new_rmax3 as text
%        str2double(get(hObject,'String')) returns contents of new_rmax3 as a double


% --- Executes during object creation, after setting all properties.
function new_rmax3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to new_rmax3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function total_cell3_Callback(hObject, eventdata, handles)
% hObject    handle to total_cell3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of total_cell3 as text
%        str2double(get(hObject,'String')) returns contents of total_cell3 as a double


% --- Executes during object creation, after setting all properties.
function total_cell3_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to total_cell3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function total_radius6_Callback(hObject, eventdata, handles)
% hObject    handle to total_radius6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of total_radius6 as text
%        str2double(get(hObject,'String')) returns contents of total_radius6 as a double


% --- Executes during object creation, after setting all properties.
function total_radius6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to total_radius6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function new_rmin4_Callback(hObject, eventdata, handles)
% hObject    handle to new_rmin4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of new_rmin4 as text
%        str2double(get(hObject,'String')) returns contents of new_rmin4 as a double


% --- Executes during object creation, after setting all properties.
function new_rmin4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to new_rmin4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function new_rmax4_Callback(hObject, eventdata, handles)
% hObject    handle to new_rmax4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of new_rmax4 as text
%        str2double(get(hObject,'String')) returns contents of new_rmax4 as a double


% --- Executes during object creation, after setting all properties.
function new_rmax4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to new_rmax4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function total_cell4_Callback(hObject, eventdata, handles)
% hObject    handle to total_cell4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of total_cell4 as text
%        str2double(get(hObject,'String')) returns contents of total_cell4 as a double


% --- Executes during object creation, after setting all properties.
function total_cell4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to total_cell4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function total_radius8_Callback(hObject, eventdata, handles)
% hObject    handle to total_radius8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of total_radius8 as text
%        str2double(get(hObject,'String')) returns contents of total_radius8 as a double


% --- Executes during object creation, after setting all properties.
function total_radius8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to total_radius8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
function slider4_Callback(hObject, eventdata, handles)
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function slider4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
```

Appendix B

Submitted Manuscript#1

Su Mon Aung, B.E, Kanyanatt Kanokwiroon PhD , Tonghathai Phairatana PhD and Surapong Chatpun, PhD. Computer-assisted cell counting for Trypan Blue stained cells images based on image segmentation and morphological operations. **(Submitted to Machine Vision and Applications)**

Machine Vision and Applications

Computer-assisted cell counting for Trypan Blue-stained cells images based on image segmentation and morphological operations

--Manuscript Draft--

Manuscript Number:	MVAP-D-17-00407
Full Title:	Computer-assisted cell counting for Trypan Blue-stained cells images based on image segmentation and morphological operations
Article Type:	Full Paper
Corresponding Author:	Surapong Chatpun, Ph.D. Prince of Songkla University Hatyai, Songkhla THAILAND
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Prince of Songkla University
Corresponding Author's Secondary Institution:	
First Author:	Su Mon Aung, B.Eng.
First Author Secondary Information:	
Order of Authors:	Su Mon Aung, B.Eng. Kanyanatt Kanokwiroon, Ph.D. Tonghathai Phairatana, Ph.D. Surapong Chatpun, Ph.D.
Order of Authors Secondary Information:	
Funding Information:	The Thailand's Education Hub for the Southern Region of ASEAN Countries (TEH-AC) scholarship given to Ms. Su Mon Aung from the Graduate School, Prince of Songkla University — Ms Su Mon Aung
Abstract:	Cell counting is a required procedure in biological and biomedical experiments and drug testing. Using a hemocytometer for cell counting has many advantages because it is simple and inexpensive. However, manual cell counting performed with a hemocytometer is time consuming and individual dependence. This study reports on the development of a computer-assisted program for Trypan Blue stained-cell counting using digital image analysis. Images of Trypan Blue-stained breast cancer cells of cell line MDA-MB-231 were obtained by microscope with a digital camera. Undesired noise and debris were removed by applying a guided image filter. Color space HSV (Hue, Saturation and Value) conversion and grayscale conversion were performed for distinguishing between live and dead cells. Image thresholding and morphological operators were applied for image segmentation. Live and dead cells were counted after image segmentation and the results were compared with manual counting by well-experienced counters. The computer-assisted cell counting from thirty-six Trypan Blue-stained microscopic images had a high correlation coefficient with the live cell results of the experts (r=0.99). The correlation coefficient of the number of dead cells comparing the computer-assisted count and the experts' count was 0.74. Furthermore, we found that our automated approach obtained false positives on the live cells count and false negatives on the dead cells count. Our approach offers high accuracy (>85%) on counting live cells from Trypan Blue-stained microscopic images compared with the experts' counting. This automated cell counting approach can assist biologists and biomedical researchers for time effectiveness in cell counting.
Suggested Reviewers:	Nasrul Humaimi Mahmood, Ph.D. Associate Professor, Universiti Teknologi Malaysia, Johor, MALAYSIA nasrulhumaimi@utm.my

	His research areas are related to three-dimensional (3D) object reconstruction from multiple views and biomedical image processing.
	Mita Nasipuri, Ph.D. Professor, Jadavpur University, India mitanasipuri@gmail.com He has just published about Blood smear analyzer for white blood cell counting: A hybrid microscopic image analyzing technique in 2016. His research are related to Bio-medical Signal Processing, Image Processing, Pattern Recognition and Soft Computing.

Computer-assisted cell counting for Trypan Blue-stained cells images based on image segmentation and morphological operations

Su Mon Aung[a], Kanyanatt Kanokwiroon[b], Tonghathai Phairatana[a] and Surapong Chatpun[a,*]

[a]Institute of Biomedical Engineering, [b]Department of Biomedical Sciences,

Faculty of Medicine, Prince of Songkla University, Hat Yai, Songkhla 90110, Thailand

Correspondence to:

Surapong Chatpun, Ph.D.

Institute of Biomedical Engineering

Faculty of Medicine, Prince of Songkla University

6th floor, 100 year Building

Hat Yai, Songkhla 90110 Thailand

Phone: +66-74451743; Fax: +66-74451744

E-mail: surapong.c@psu.ac.th

Abstract

Cell counting is a required procedure in biological and biomedical experiments and drug testing. Using a hemocytometer for cell counting has many advantages because it is simple and inexpensive. However, manual cell counting performed with a hemocytometer is time consuming and individual dependence. This study reports on the development of a computer-assisted program for Trypan Blue stained-cell counting using digital image analysis. Images of Trypan Blue-stained breast cancer cells of cell line MDA-MB-231 were obtained by microscope with a digital camera. Undesired noise and debris were removed by applying a guided image filter. Color space HSV (Hue, Saturation and Value) conversion and grayscale conversion were performed for distinguishing between live and dead cells. Image thresholding and morphological operators were applied for image segmentation. Live and dead cells were counted after image segmentation and the results were compared with manual counting by well-experienced counters. The computer-assisted cell counting from thirty-six Trypan Blue-stained microscopic images had a high correlation coefficient with the live cell results of the experts (r=0.99). The correlation coefficient of the number of dead cells comparing the computer-assisted count and the experts' count was 0.74. Furthermore, we found that our automated approach obtained false positives on the live cells count and false negatives on the dead cells count. Our approach offers high accuracy (>85%) on counting live cells from Trypan Blue-stained microscopic images compared with the experts' counting. This automated cell counting approach can assist biologists and biomedical researchers for time effectiveness in cell counting.

Keywords: Cell counting; Guided image filter; Color space; Image segmentation; Morphological operator

1. Introduction

In cancer drug discovery researches, many aspects of novel drug treatments on cancer cells have been studied, including morphological and cellular structure changes, and drug response.(Fang et al., 2016, Fesik, 2005, Kondo et al., 2005) One simple investigation to evaluate the efficacy of cancer drug treatments is the identification of apoptotic cells and live cells.(Lowe and Lin, 2000) Apoptosis is a programmable cell death, which can be categorically observed from cell morphological changes. Counting viable cells or live cells per total cells after in vitro drug treatment is a common task for biologists and biomedical researchers when they evaluate the efficacy. Manual cell counting by a biomedical expert with a tally counter is a conventional method but the counting result is an individual dependence, and it is tedious and time-consuming. To overcome the drawbacks of manual cell counting, many automated visual analysis tools have been developed such as ImageJ, Fiji, CellProfiler™, CellC, Image-Pro Plus and MetaMorph®, which are available as software packages for cell analysis.(Blatt et al., 2004, Carpenter et al., 2006, Grishagin, 2015, Narayan et al., 2007, Schindelin et al., 2012, Selinummi et al., 2005, Vayrynen et al., 2012)

Computer-assisted cell counting using image processing and analysis has been continuously developed and improved to provide the requirements of high reliability, specificity and sensitivity for different cell types and test protocols. For example, Sui et al. developed a cell counting method for host cells in a bright field for insect cells by using a nonlinear transformed sliding band filter. They obtained a low error rate and a high accuracy when comparing their method with manual counting. (Sui et al., 2014) A cell counting method for white blood cells was developed using Matlab to evaluate hematic pathologies in terms of numbers, sizes and types of white blood cells.(Nazlibilek et al., 2014) They applied image enhancement and image segmentation as pre-processing steps and used a neural network for classification. Piccinini et al. used a fully automated mosaicing method to

improve the reliability and reproducibility of live and dead cell counting.(Piccinini et al., 2014) Mouelhi and colleagues also described automatic image segmentation with active contour for stained nuclei in breast cancer tissue which could segment touching nuclei to get the total number of breast cancer nuclei.(Mouelhi et al., 2013)

The quality of light microscopy digital images taken from hemocytometry significantly influences visual analysis, including cell counting. Therefore, most light microscopy digital images need to undergo a pre-processing step before cell segmentation is performed. Noise filtering is one of the pre-processing steps in every digital image processing technique. A median filter is a popular noise filter which is used to remove salt and pepper noise in digital images.(Soni and Rathor, 2015, Sun et al., 2015, Zeng et al., 2012, Zhu and Huang, 2012) It has been reported that impulse noise in high-density noisy images can be effectively removed by using an improved algorithm such as a decision based adaptive neighborhood median filter.(Samantaray and Mallick, 2015) Mahmood and Mansor used a morphological tool with the Hough transform technique to detect and estimate the number of red blood cells in blood sample images.(Mahmood and Mansor, 2012) Liew et al. applied image processing methods to develop a prototype for a cell detection algorithm using circular Hough transform to detect the number of cells in bee combs from digital images.(Liew et al., 2010) Although there are many methods for cell detection and cell counting, computer-assisted cell counting methods are still being developed to improve the counting accuracy and to match with new applications. Furthermore, there is few automated counting software for Trypan Blue stained cells in biomedical research.

In this paper, we developed an automated counting approach for Trypan Blue-stained cells relied on simple digital image processing techniques including image noise reduction, image enhancement, image segmentation and image morphological operations. To evaluate the effectiveness of our approach, both live and dead cells were quantified using this

proposed method and compared with manual counting by three well-experienced counters and ImageJ, open source software.

2. Materials and Methods

2.1 Sample preparation and image acquisition

A human breast cancer cell line (MDA-MB-231) purchased from the American Type Culture Collection (ATCC, Canada) was used as an example in this study. An overview of the study procedure is shown in Fig.1. The sample preparation generally followed the procedure of Piccinini et al.(Piccinini et al., 2014) In brief, the cells were detached from the flask by short exposure to trypsin/EDTA, washed in the culture media by centrifugation and re-suspended in 5 mL of fresh medium. After trypsinization, the cells were stained with Trypan Blue solution and then 20 µL of this mixture was placed in a hemocytometer counting chamber.

Images of the stained cells were obtained by an inverted Olympus IX51 microscope equipped with a digital Olympus DP72 camera with 2/3-inch CCD. The images were acquired in the bright field by using Olympus UPLanFL N 10x0.13 as a standard objective lens. An example of digital Trypan Blue-stained cell images is shown in Fig. 2. The images were cropped to a 16-square grid and then image processing analysis was performed as shown in the cell counting approach flowchart as presented in Fig. 3.

2.2 Image filtering

In our study, there are small debris (dark spots) in Trypan Blue-stained microscopic images from the hemocytometer which can cause cell artifacts as shown in Fig.2. We applied a guided image filter that spatially distributed the filtering strengths with low-level features to the image to smooth and remove the small debris from background including maintaining the edge features of image and introducing less artifacts.(Hao et al., 2016, He et al., 2013)

The filtering process needs constraints from the input p to determine the linear coefficients (a_k , b_k). The filtering output q is a linear transform of the guidance I in a window ω_k centered at the pixel k since the input p needs to subtract some unwanted components n such as noise/textures as follows:

$$q_i = a_k I_i + b_k, \forall_i \in \omega_k \quad (1)$$

$$q_i = p_i - n_i \quad (2)$$

Furthermore, the neighborhood size around each pixel as a scalar or two-element vector, [M, N] was [30, 30] to remove small debris in the background.

2.3 Color space and grayscale conversion

In our study, we converted a color space of the Trypan Blue stained images which was originally RGB color images by applying HSV color space. The HSV color space is quite similar to the way in which humans perceive color and more suitable for image segmentation and analysis than the RGB model. The HSV converted images were then used to perform for live cells segmentation. Furthermore, the original RGB images were also converted to grayscale for dead cells detection.

2.4 Image segmentation

The segmentation step is very crucial because the accuracy of the subsequent processes mainly depends on the correct segmentation. Prior to image segmentation, the brightness of a HSV converted image is adjusted for getting the better contrast. Our approach utilized Otsu's thresholding method to extract foreground (live cells) from the background and to turn the image to black and white.(Otsu, 1975) As our Trypan Blue-stained microscopic images still contained counterfeit structures in the nuclei, however, which caused difficulty for extraction and segmentation. To eliminate this drawback, two operations

based on morphological reconstruction were used: opening by reconstruction and dilation by reconstruction. The first operation removes the disconnected bright objects that are smaller than the structuring element. (Veta et al., 2013) Similarly, the second one removes the disconnected dark objects smaller than the structuring element. A disk-shaped structuring element with radius "n" was assigned in the algorithm. The amount of detail presented in the image depends on the size of the structuring element with radius "n". Furthermore, we applied the effective threshold value from the histogram of a grayscale image to obtain only dead cells in the black and white image.

2.5 Cell counting

The process of counting the cells was achieved by applying the labeling technique to the binary images after thresholding to distinguish objects from the background and then we measured the geometric properties using a centroid and a boundary to select live cells from labeled regions. Furthermore, the counting results by our approach were compared to the manual counting by three experts as well as comparing to automatic cell counting with ImageJ using Trypan Blue exclusion plugin (open-source software). Then the correlation between the counting results of two methods was determined.

3. Results

The testing was carried out using thirty six Trypan Blue-stained microscopic images. The debris in the images was removed after image-guided filtering as shown in Fig. 4. As we performed color space conversion, the images based on HSV channels are shown in Fig. 5. It can be seen that the image with the saturation channel (Fig. 5b) gave a better contrast of live cells as compared to other channels. Then after brightness adjustment, the live cells became clearly extracted from the background as shown in Fig. 6. As we used Otsu's thresholding to segment the live cells from the background and the nuclei of live cells were opened and dilated after the thresholding, the result of segmentation step for live cell is demonstrated in

Fig. 7a. The image segmentation of dead cells after using the histogram thresholding is presented in Fig. 7b. After the image segmentation, the image was processed with the counting algorithm to identify and count both live and dead cells. The final images showing the live and dead cells identification are respectively presented in Fig. 8 and Fig. 9. The live cells are marked with yellow circles while the dead cells are marked as blue circles. As shown in Fig. 8, it was noticed that some live cells could not be detected and counted due to their nondistinctive nuclei. Fig. 10 shows the plot of the number of live and dead cells counted by our approach against the averaged results from manual counting. Our approach provided a very similar counting result of live cells to the experts' counting result with a high correlation coefficient (r=0.99) whereas a correlation coefficient in case of the dead cells counting is 0.74. In comparison with the experts, the average accuracy percentages of cell counting from 36 images were determined. We obtained 83±6% and 63±10% for the accuracy of live cell counting and dead cell counting respectively. We also show the time consumption from 15 images by each expert and our approach as illustrated in Fig.11. Our approach consumed shorter time than manual counting by experts and gave less variation on time consumption compared to manual counting. The average number of dead cells from our counting approach was considerably 11% less than the experts' counting, resulting in our higher false negative as illustrated in Fig.12. Fig.13 demonstrates the average number of live, dead and total cells from manual, ImageJ and proposed approach.

4. Discussion

In this study, we presented and tested the digital image analysis with counting algorithm based on simple image processing techniques for live and dead cells obtained from Trypan Blue-stained microscopic images. We showed that the guided image filtering provided smooth edge maintenance to our objects, leading to a better quality input image for our approach (He et al., 2013) . As our results, the accuracy of live cells counting, for each

image, was very high (>83%) due to the good contrast between foreground and background after preprocessing with the image noise reduction using a guided image filter and a saturated channel. (Choudhry, 2016) We experienced a similar high accuracy of live cells counting using similar sample images in our previous work using a K-mean clustering technique.(Chobngam et al., 2012) However, this new simple method provided a higher accuracy. With this approach, the brightness of live cells is clearly different from the image background, which is a distinct advantage for image thresholding for live cells detection. Piccinini and colleagues presented a high accuracy of live cells counting using an automated image mosaicing technique for A549 and KKP cell lines.(Piccinini et al., 2014) The high accuracy of live cell counting in both their results and ours particularly relates to the clearly distinguishable shape of live cells and a good background contrast. However, we have addressed that some images we could not completely detect all live cells because there was neither structure in nucleus nor smaller structure than our value of structuring element's radius in nucleus. This caused some images got an accuracy less than 90%. Furthermore, our approach did not require manual thresholding adjustment therefore it reduced time consuming.

Our approach provided less number of dead cells compared to manual counting by the experts. This might have been caused by the ineffectiveness of dead cell segmentation. It should be noted that segmentation results depend on the quality of preparation and the coloring of Trypan Blue-stained microscopic images. Overstraining and poor digitization can adversely affect segmentation results.(Osman et al., 2010, Yagi and Gilbertson, 2005) These quandaries infrequently occur and can be remedied with a more rigorous quality control during the sample preparation and image acquisition. Moreover, dead cells from apoptosis presented in fragments can lead to over-counting.(Choi and Yoo, 2012) Therefore, the counting of dead cells still has erroneous results much more than live cells counting. For

example, the approach using automated image mosaicking had false positive results on dead cell counting compared to the experts much more than the live cells counting.(Piccinini et al., 2014) It remains a challenge to improve the accuracy of dead cells counting by improving either image noise reduction or image segmentation as well as counting logic for fragmented dead cell.

Comparing the results of cell counting between our approach and ImageJ with automatic cell counting plugin, our approach could give the number of both live cells and dead cells whereas ImageJ could detect only live cells. This indicates that the ImageJ plugin for automatic cell counting does not serve our Trypan Blue stained images and requirements. It seems ImageJ could automatically count cells with good contrast between cells and background therefore dead cells in our images are quite difficult to be detected. It might be necessary to do an image preprocessing for our images before analyzing with ImageJ. On the other hand, our approach already included the image preprocessing such as noise filtering and contrast enhancement to improve the image quality in the algorithm.

In our approach, there were several limitations which affected the counting accuracy. For example, we assumed that the cell diameters in all images were about the same size, while in fact the dead cells in our study had various diameters, especially when the cells counted by our approach greater than the results counted by the experts. Furthermore, the qualities of the image, such as the brightness of the image background and the image became fragmented from apoptosis. Therefore, some images had the number of dead cells contrast, can influence the image segmentation and lead to lower accuracy of cell counting. As our Trypan Blue-stained microscopic images have a dark background, the color of the dead cells is quite similar to the background, which requires several trials for dead cells segmentation. Moreover, we applied a mean threshold value based on 36 images for dead cells segmentation, thus it is necessary to recalculate the mean threshold value if the number of

images or image background is changed. In addition, employing a large structuring element can oversimplify the image, while using too small a structuring element does not always produce desirable results as many of the substructures within the large nuclei remain, affecting the segmentation performance. Therefore, optimum parameters should be determined such as threshold value and radius of structuring element.

5. Conclusion

In this study, we present a computer-assisted cell counting approach on Trypan Blue-stained microscopic images of a breast cancer cell line. The stained cell counting results of the Trypan Blue-stained microscopic images using our approach offers a remarkable accuracy, especially in counting the number of live cells. Our simple method could identify overlapping cells and count them correctly. This automated counting approach is very time effective for biologists and biomedical researchers. The future development for this work will include the accuracy of dead cells counting by taking into account for the image noise reduction or the image segmentation.

6. Acknowledgments

The authors would like to acknowledge the Thailand's Education Hub for the Southern Region of ASEAN Countries (TEH-AC) scholarship given to Ms. Su Mon Aung from the Graduate School, Prince of Songkla University. We would like to thank Associate Professor Pornchai Phukpattaranont from the Faculty of Engineering, Prince of Songkla University for advice on image processing. We appreciated our technical discussions with Ms. Adeline Verel and Ms. Pamina Bernou, internship students from Telecom Physique Strasbourg, France. We also thank the International Affairs Office of the Faculty of Medicine, Prince of Songkla University for editing the English of the manuscript.

References

[1] Blatt RJ, Clark AN, Courtney J, Tully C, Tucker AL. Automated quantitative analysis of angiogenesis in the rat aorta model using Image-Pro Plus 4.1. Comput Methods Programs Biomed. 2004;75:75-79.

[2] Carpenter AE, Jones TR, Lamprecht MR, et al. CellProfiler: image analysis software for identifying and quantifying cell phenotypes. Genome biology. 2006;7:R100.

[3] Chobngam F, Kanokwiroon K, Chatpun S, Wichakool W, Limsiroratana S, Phukpattaranont P, editors. Preliminary results of death cell counting based on K-mean clustering. The 5th 2012 Biomedical Engineering International Conference; 2012 5-7 Dec. 2012.

[4] Choi YH, Yoo YH. Taxol-induced growth arrest and apoptosis is associated with the upregulation of the Cdk inhibitor, p21WAF1/CIP1, in human breast cancer cells. Oncol Rep. 2012;28:2163-2169.

[5] Choudhry P. High-Throughput Method for Automated Colony and Cell Counting by Digital Image Analysis Based on Edge Detection. PLOS ONE. 2016;11:e0148469.

[6] Fang JY, Tan SJ, Wu YC, Yang Z, Hoang BX, Han B. From competency to dormancy: a 3D model to study cancer cells and drug responsiveness. J Transl Med. 2016;14:38.

[7] Fesik SW. Promoting apoptosis as a strategy for cancer drug discovery. Nat Rev Cancer. 2005;5:876-885.

[8] Grishagin IV. Automatic cell counting with ImageJ. Anal Biochem. 2015;473:63-65.

[9] Hao S, Pan D, Guo Y, Hong R, Wang M. Image detail enhancement with spatially guided filters. Signal Processing. 2016;120:789-796.

[10] He K, Sun J, Tang X. Guided Image Filtering. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2013;35:1397-1409.

[11] Kondo Y, Kanzawa T, Sawaya R, Kondo S. The role of autophagy in cancer development and response to therapy. Nat Rev Cancer. 2005;5:726-734.

[12] Liew LH, Lee BY, Chan M, editors. Cell detection for bee comb images using circular Hough transformation. Science and Social Research (CSSR), 2010 International Conference on; 2010: IEEE.

[13] Lowe SW, Lin AW. Apoptosis in cancer. Carcinogenesis. 2000;21:485-495.

[14] Mahmood NH, Mansor MA. Red blood cells estimation using Hough transform technique. Signal & Image Processing. 2012;3:53.

[15] Mouelhi A, Sayadi M, Fnaiech F, Mrad K, Romdhane KB. Automatic image segmentation of nuclear stained breast tissue sections using color active contour model and an improved watershed method. Biomedical Signal Processing and Control. 2013;8:421-436.

[16] Narayan PJ, Gibbons HM, Mee EW, Faull RL, Dragunow M. High throughput quantification of cells with complex morphology in mixed cultures. J Neurosci Methods. 2007;164:339-349.

[17] Nazlibilek S, Karacor D, Ercan T, Sazli MH, Kalender O, Ege Y. Automatic segmentation, counting, size determination and classification of white blood cells. Measurement. 2014;55:58-65.

[18] Osman MK, Ahmad F, Saad Z, Mashor MY, Jaafar H, editors. A genetic algorithm-neural network approach for Mycobacterium tuberculosis detection in Ziehl-Neelsen stained tissue slide images. 2010 10th International Conference on Intelligent Systems Design and Applications; 2010 Nov. 29 2010-Dec. 1 2010.

[19] Otsu N. A threshold selection method from gray-level histograms. Automatica. 1975;11:23-27.

[20] Piccinini F, Tesei A, Paganelli G, Zoli W, Bevilacqua A. Improving reliability of live/dead cell counting through automated image mosaicing. Computer methods and programs in biomedicine. 2014;117:448-463.
[21] Samantaray AK, Mallick P. Decision Based Adaptive Neighborhood Median Filter. Procedia Computer Science. 2015;48:222-227.
[22] Schindelin J, Arganda-Carreras I, Frise E, et al. Fiji: an open-source platform for biological-image analysis. Nat Methods. 2012;9:676-682.
[23] Selinummi J, Seppala J, Yli-Harja O, Puhakka JA. Software for quantification of labeled bacteria from digital microscope images by automated image analysis. Biotechniques. 2005;39:859.
[24] Soni T, Rathor N. Removal of High Density Impulse Noise using Efficient Median Filter for Digital Image. International Journal of Computer Applications. 2015;115.
[25] Sui D, Wang K, Park H, Chae J. Bright field microscopic cells counting method for BEVS using nonlinear convergence index sliding band filter. Biomedical engineering online. 2014;13:147.
[26] Sun C, Tang C, Zhu X, Li X, Wang L. An efficient method for salt-and-pepper noise removal based on shearlet transform and noise detection. AEU-International Journal of Electronics and Communications. 2015;69:1823-1832.
[27] Vayrynen JP, Vornanen JO, Sajanti S, Bohm JP, Tuomisto A, Makinen MJ. An improved image analysis method for cell counting lends credibility to the prognostic significance of T cells in colorectal cancer. Virchows Arch. 2012;460:455-465.
[28] Veta M, van Diest PJ, Kornegoor R, Huisman A, Viergever MA, Pluim JP. Automatic nuclei segmentation in H&E stained breast cancer histopathology images. PLoS One. 2013;8:e70221.
[29] Yagi Y, Gilbertson JR. Digital imaging in pathology: the case for standardization. J Telemed Telecare. 2005;11:109-116.
[30] Zeng H, Liu Y-z, Fan Y-m, Tang X. An improved algorithm for impulse noise by median filter. AASRI Procedia. 2012;1:68-73.
[31] Zhu Y, Huang C. An improved median filtering algorithm for image noise reduction. Physics Procedia. 2012;25:609-616.

Figure legends

Figure 1. Trypan Blue-stained MDA-MB-231 cells in the hemocytometer

Figure 2. An example of an original Trypan Blue–stained image

Figure 3. Overview of the image processing for live and dead cells counting

Figure 4. Examples of (a) original image and (b) image after guided image filter process

Figure 5. Images after HSV color space conversion: (a) Hue channel image, (b) Saturation channel image and (c) Value channel image

Figure 6. Saturation channel image after brightness adjustment showing better contrast of live cells from the background

Figure 7. Image after applying Otsu's method showing a black and white image with live cells

Figure 8. Example of image showing live cells detection

Figure 9. Example of image showing dead cells detection

Figure 10. Correlation plots between the counting results by the new approach and experts for (a) live cells and (b) dead cells

Figure 11. Time consumption for cell counting measured by three experts and our proposed approach

Figure 12. A plot of average number of dead cells by manual counting and our proposed approach

Figure 13. A plot of the average number of live, dead and total cells counted by manual counting, ImageJ and our approach

Figure1

Click here to download Figure Figure 1 Cell culturing_600.tif

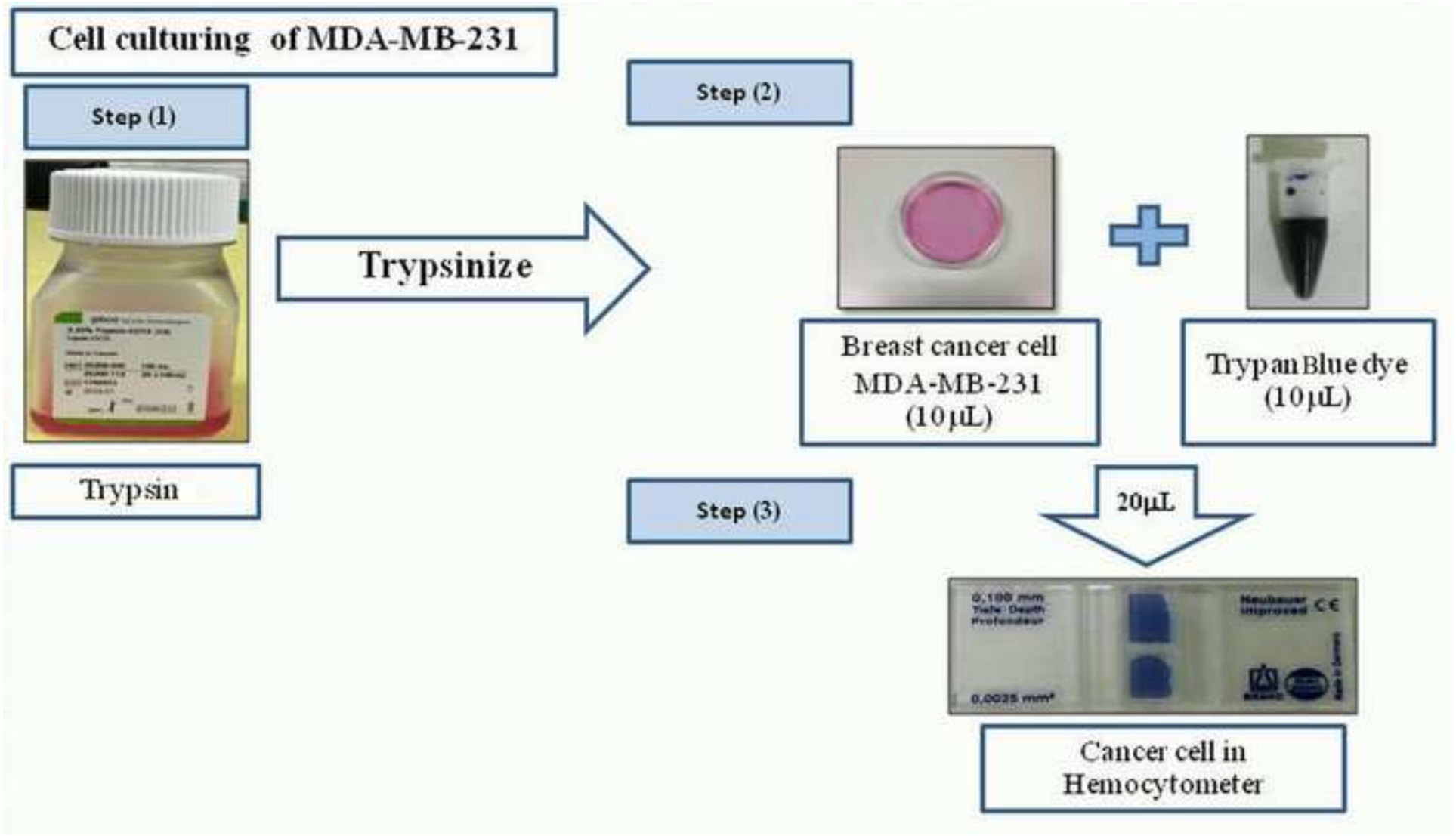


Figure2

Click here to download Figure Figure 2new_600.tif

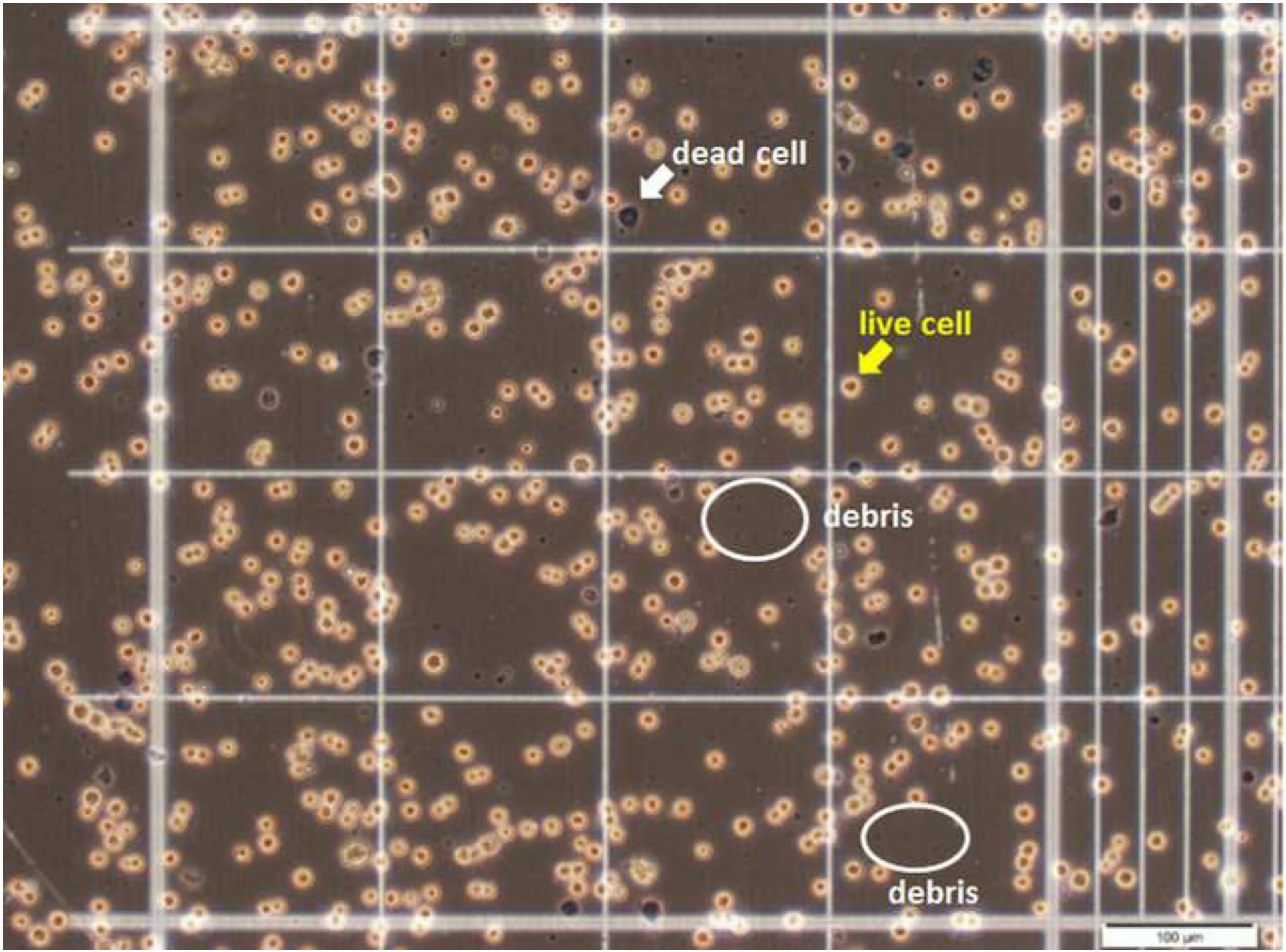


Figure3

Click here to download Figure Figure 3 Diagram for image processing_600.tif

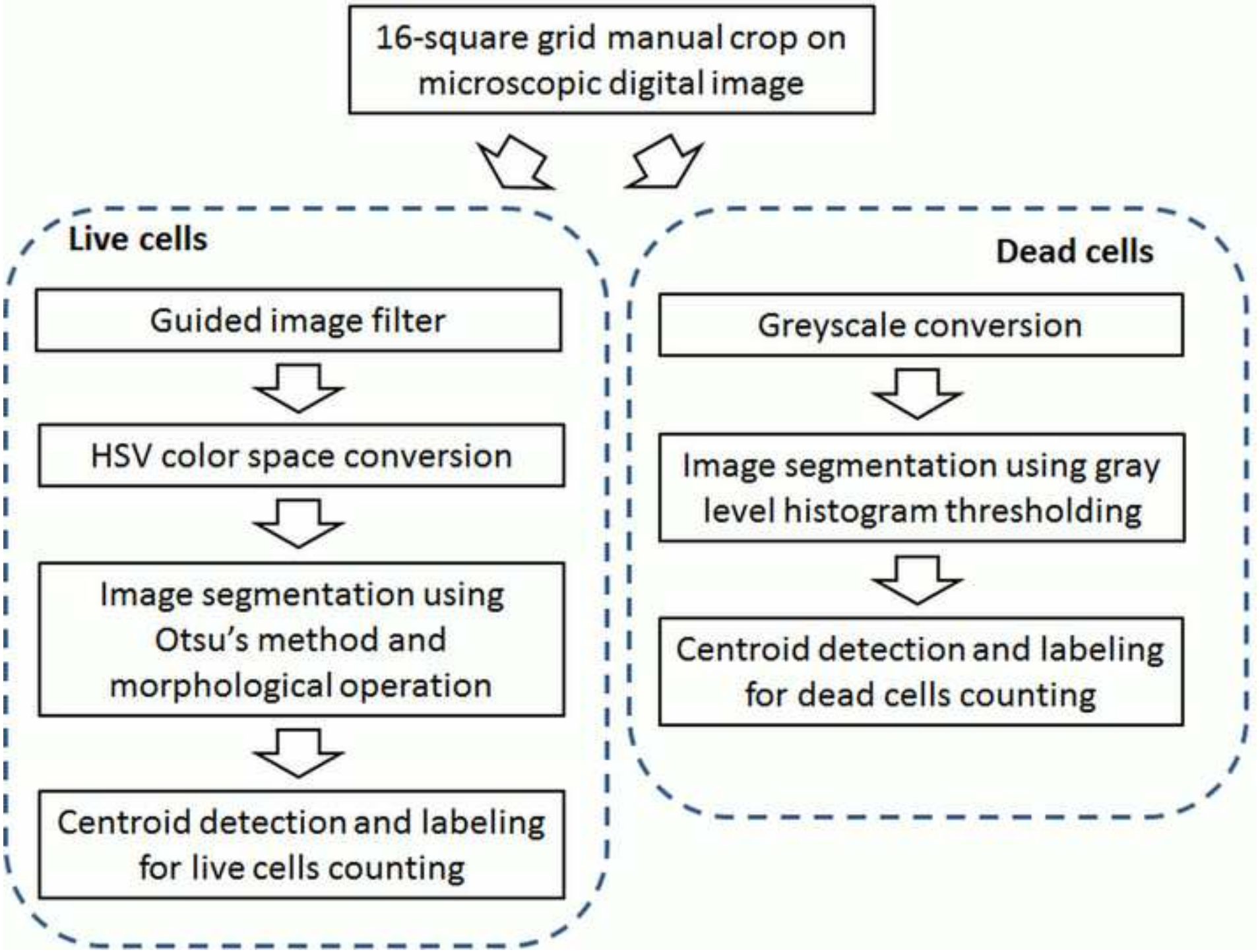


Figure4

Click here to download Figure Figure 4_600 newx.tif

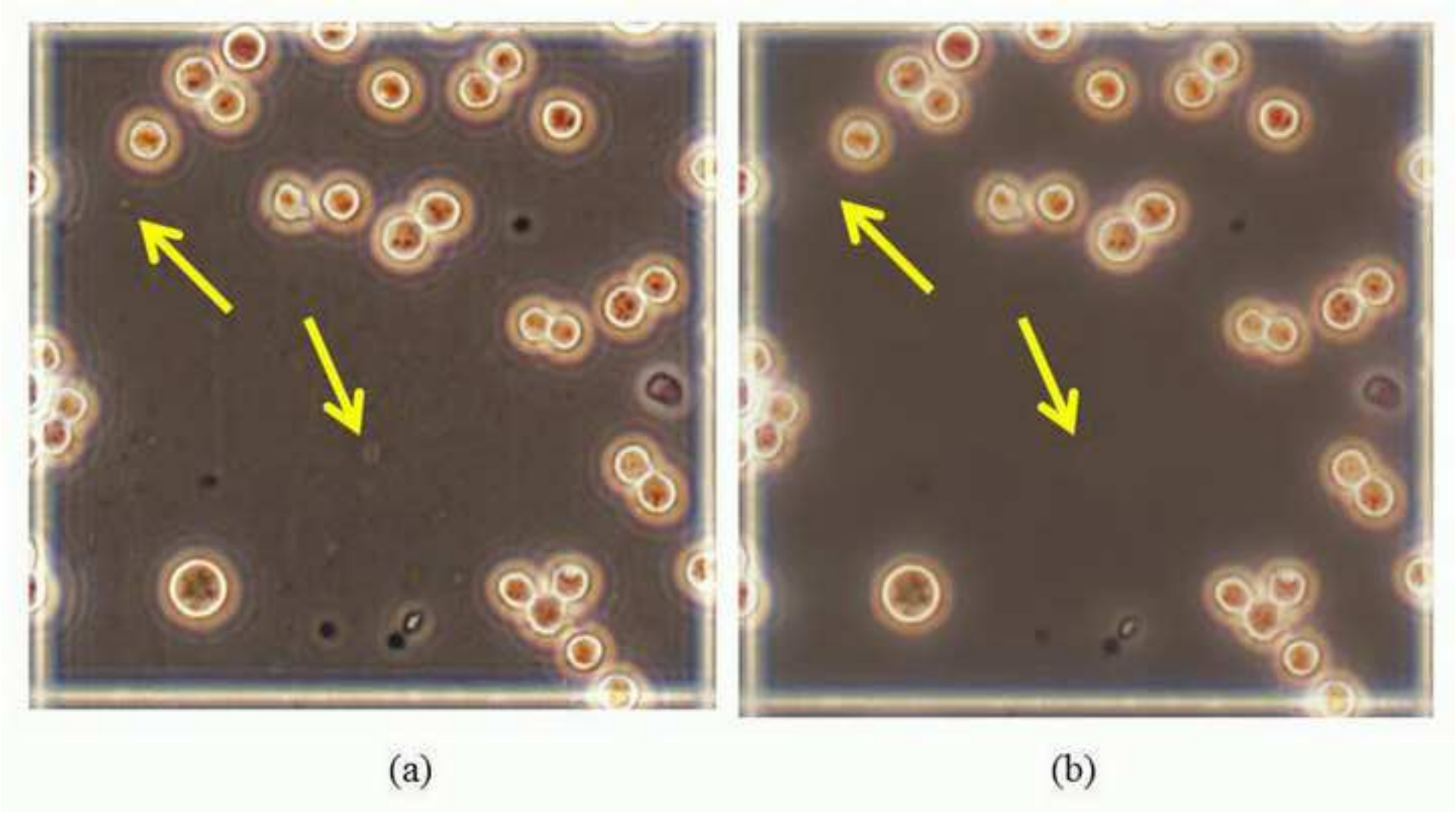


Figure5

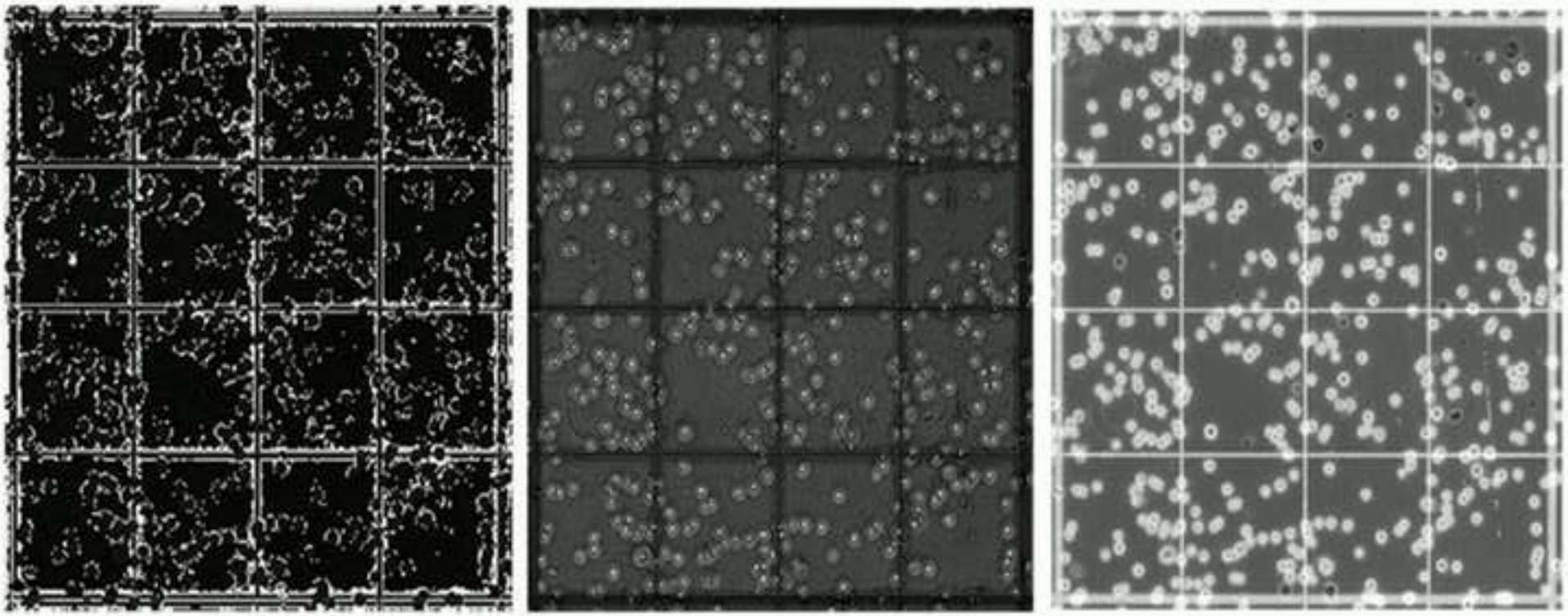


Figure6

Click here to download Figure Figure 6_600.tif

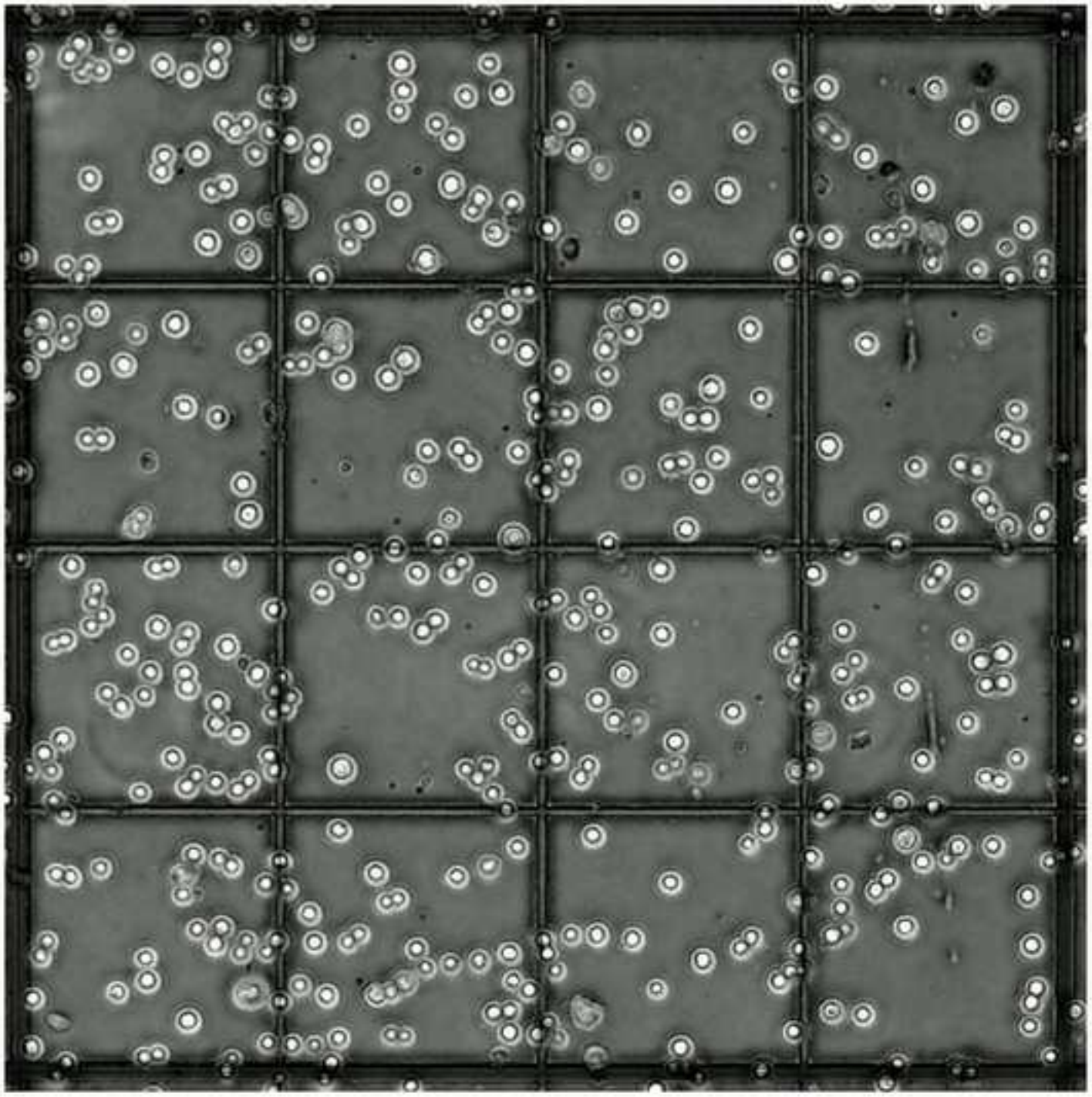


Figure7

Click here to download Figure Figure 7_new600x.tif

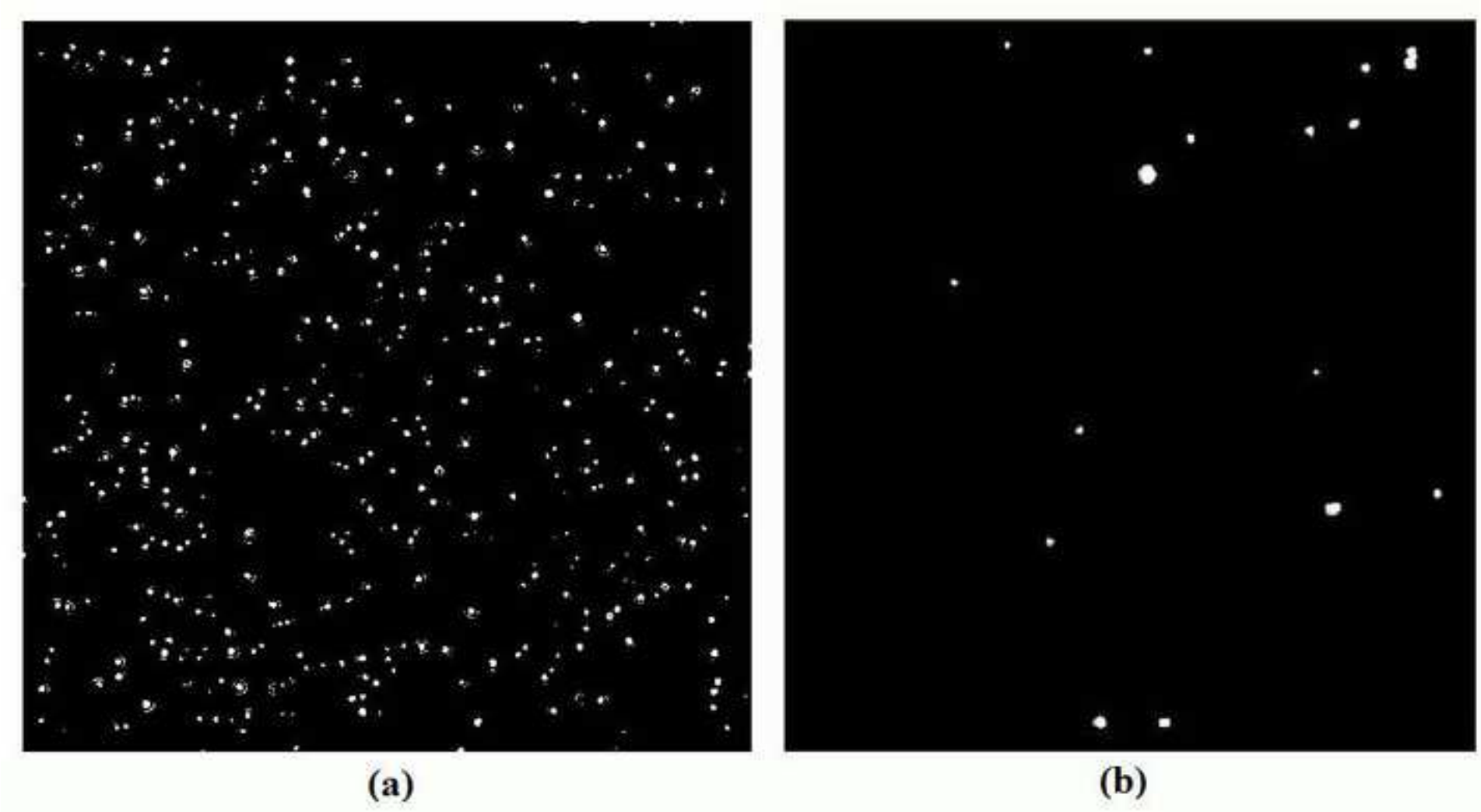


Figure8

Click here to download Figure Figure 8Live cell detect_600.tif

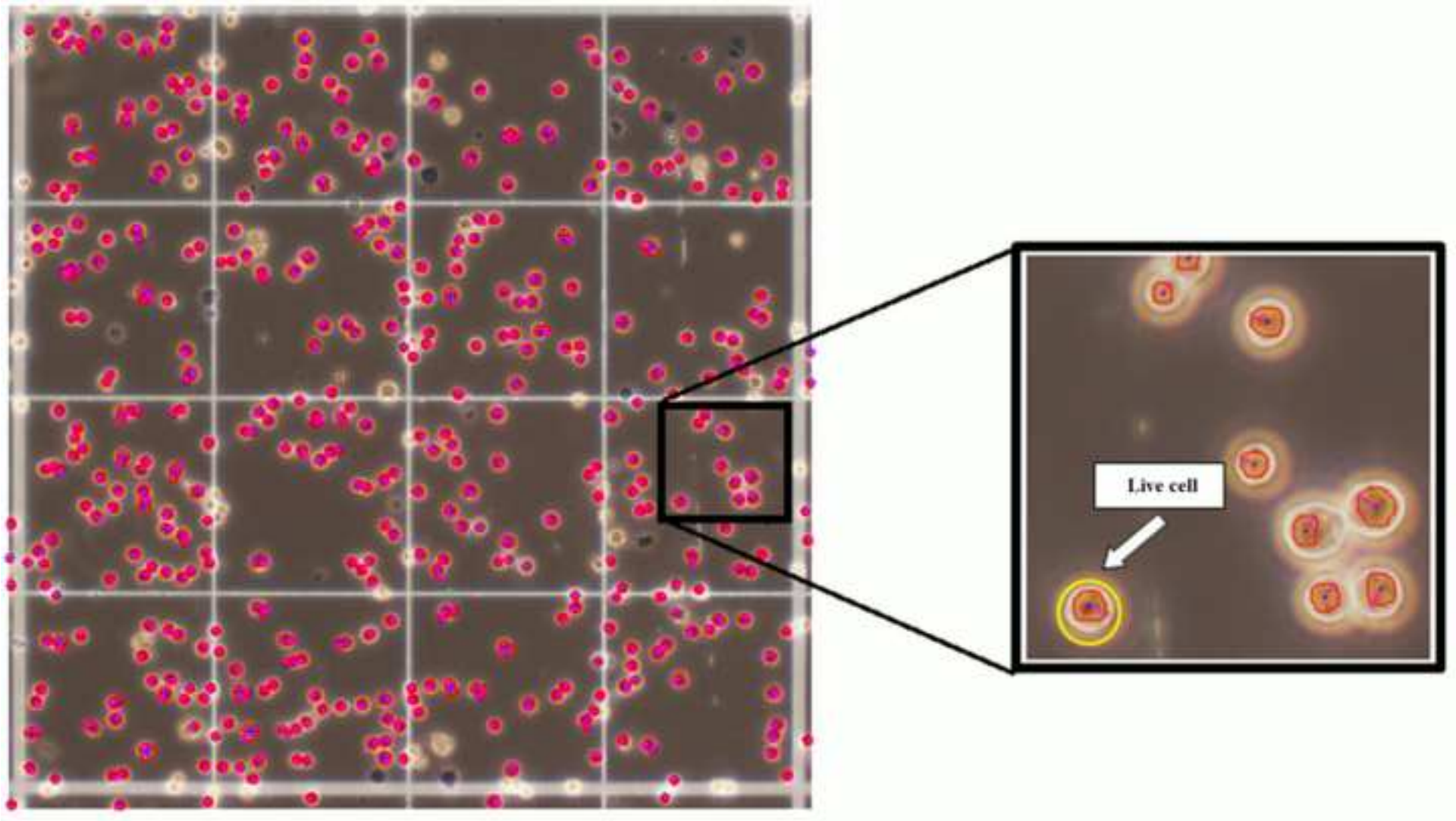


Figure9

Click here to download Figure Figure 9 Dead cell detection_600.tif

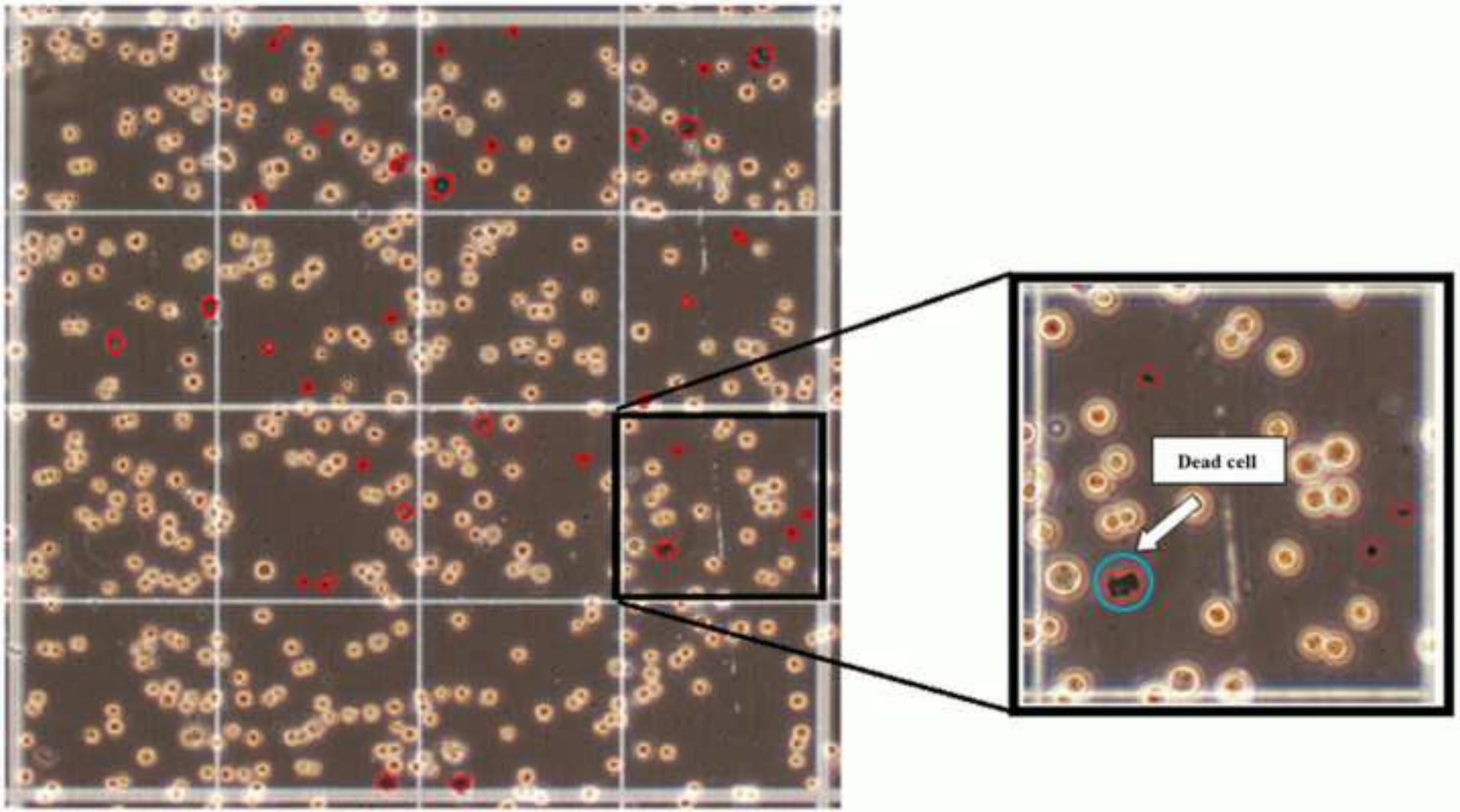


Figure10

Click here to download Figure Figure 10 new_600.tif

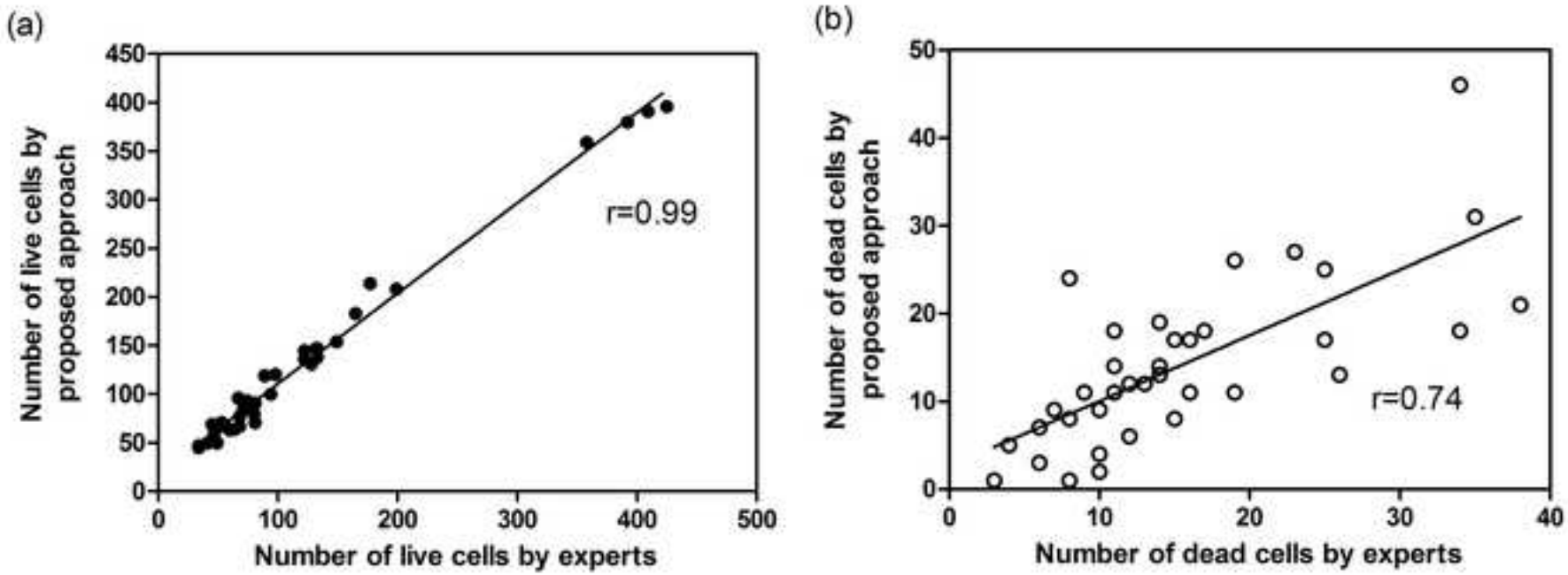


Figure11

Click here to download Figure Figure 11_600.tif

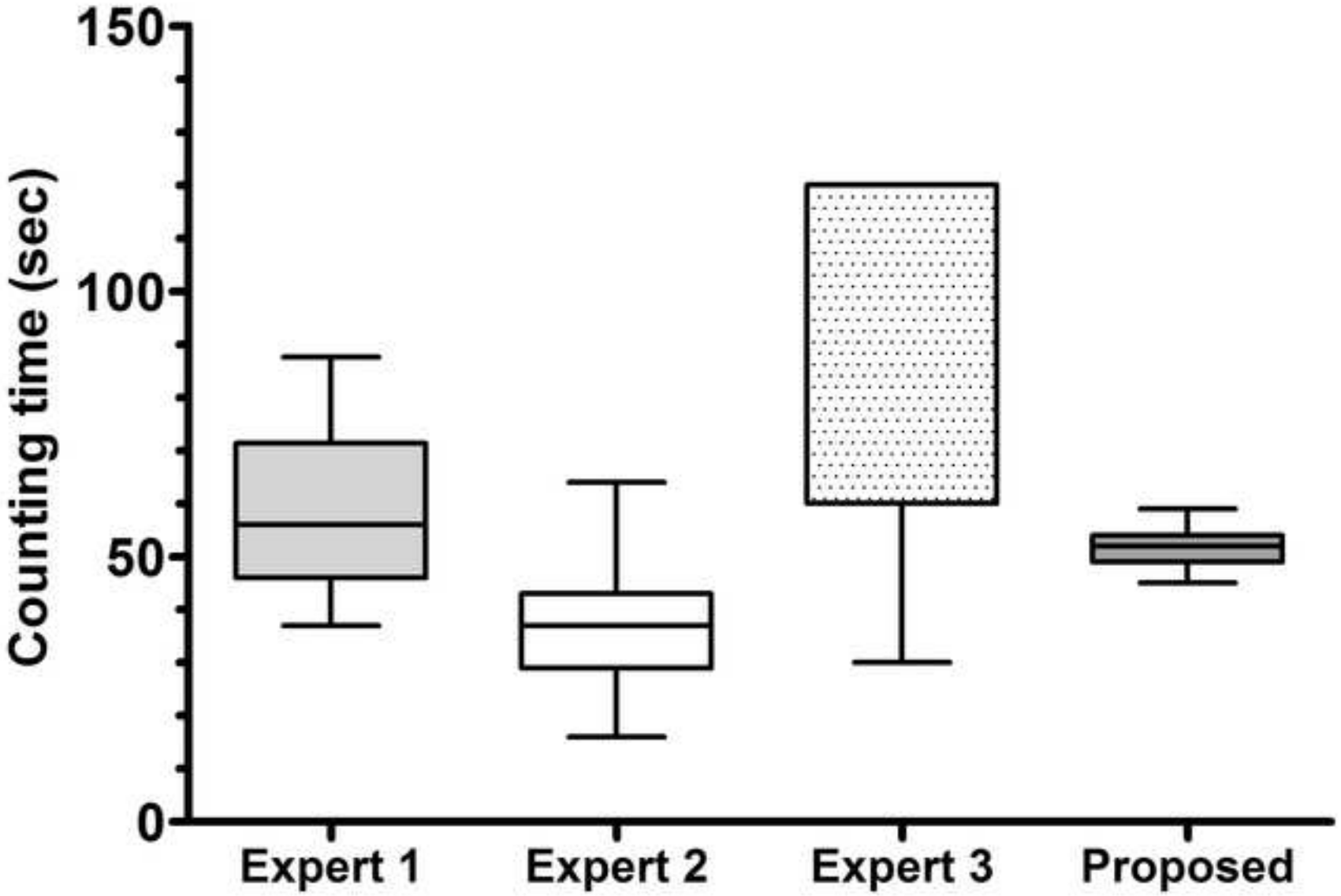


Figure12

Click here to download Figure Figure 12_600.tif

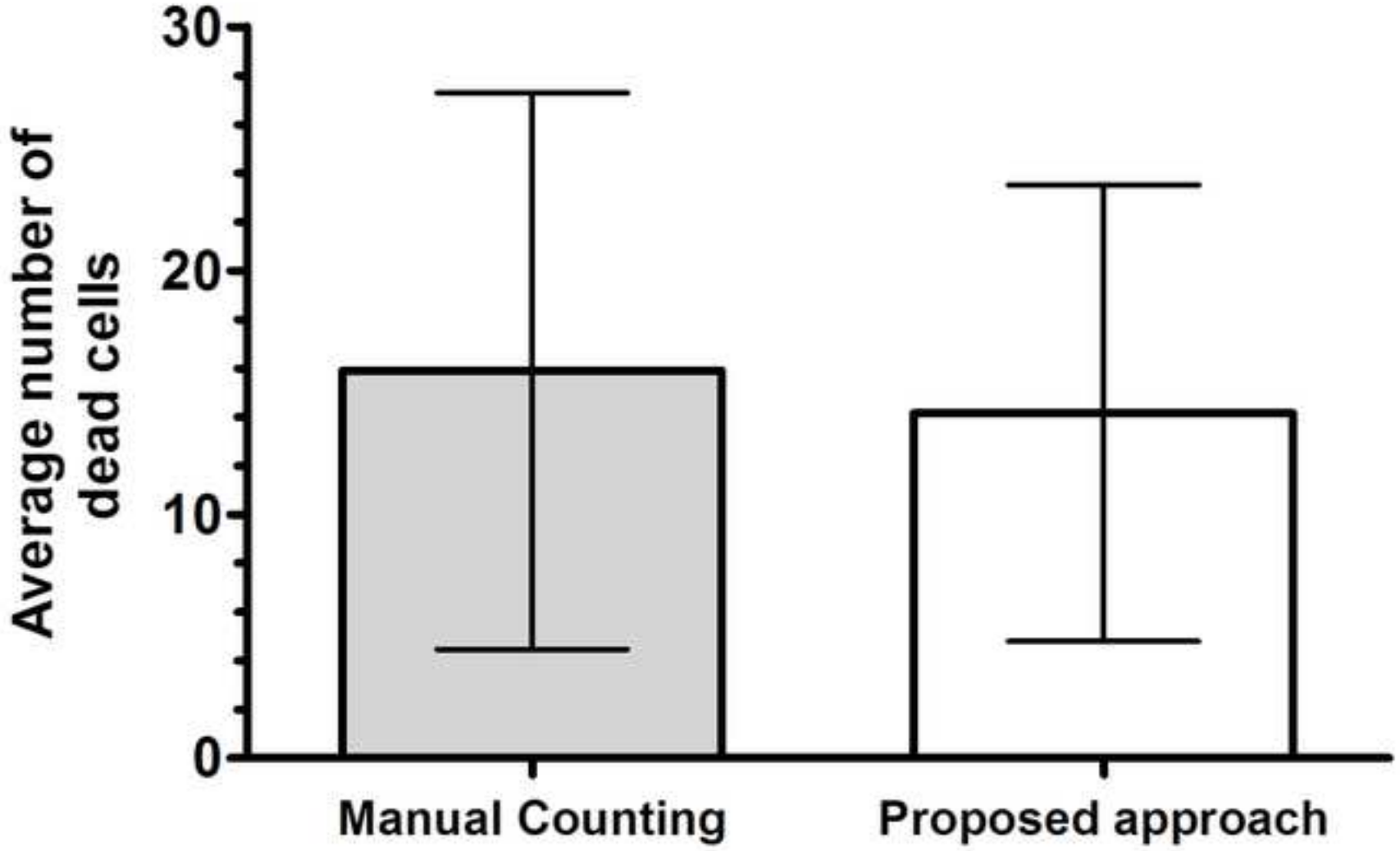
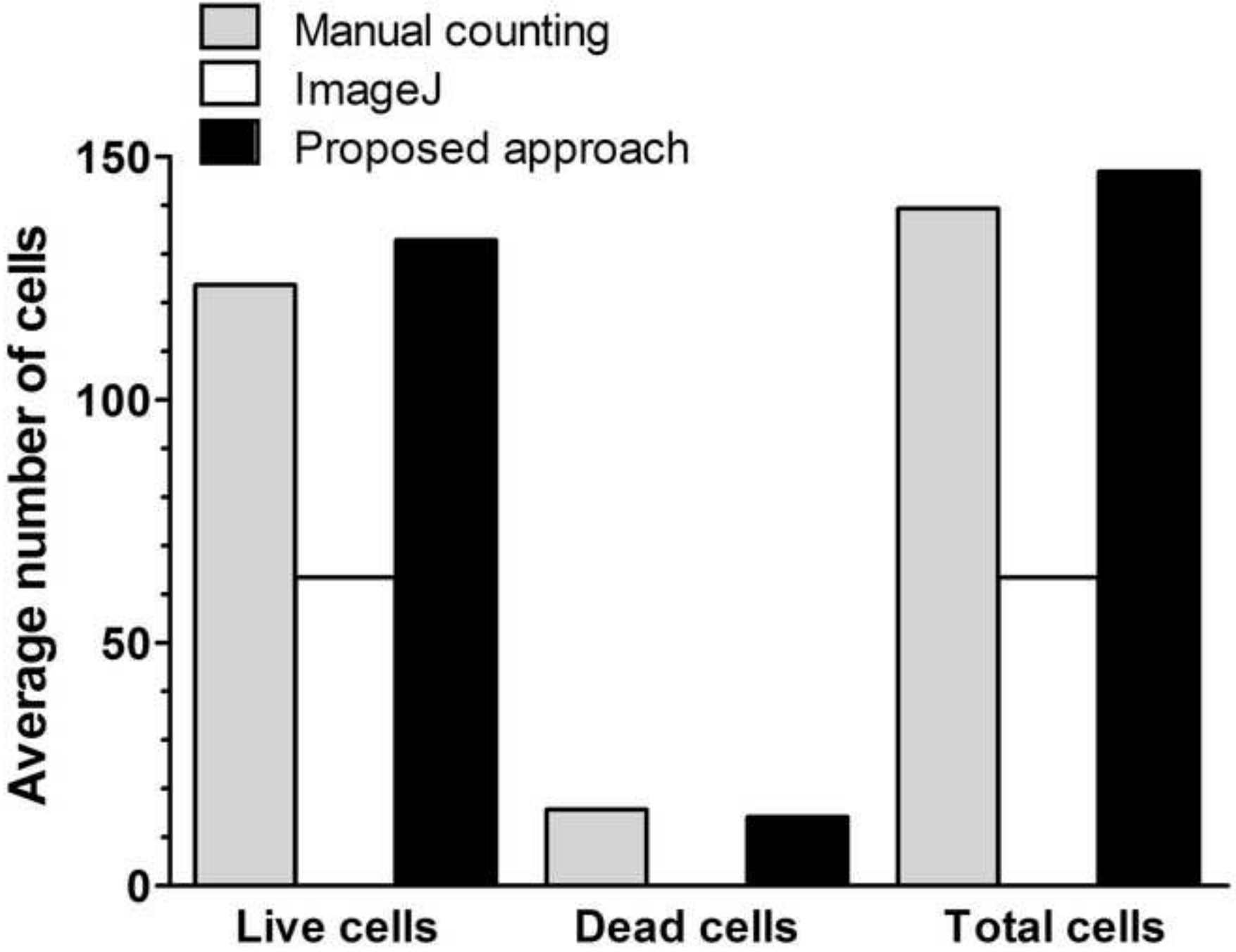


Figure13

Click here to download Figure Figure 13_600.tif



Appendix C

Counting results from the first approach

Table C.1 Results of live and dead cells counting from three experts and first proposed application

Image	Expert1 (Live)	Expert2 (Live)	Expert3 (Live)	Proposed (Live)	Expert1 (Dead)	Expert2 (Dead)	Expert3 (Dead)	Proposed (Dead)
1	389	470	417	396	12	25	13	18
2	393	407	376	380	11	23	13	11
3	351	385	337	359	6	28	7	13
4	419	429	378	391	9	25	12	8
5	106	66	95	119	4	25	0	2
6	74	76	73	93	8	12	3	1
7	106	112	75	120	10	11	5	11
8	72	72	68	84	7	9	4	9
9	69	69	66	67	4	4	2	1
10	83	81	79	91	8	8	3	3
11	56	59	64	63	4	83	0	5
12	79	83	79	80	7	12	5	8
13	133	132	134	138	14	17	8	12
14	123	119	124	136	15	16	10	14
15	93	93	95	100	10	13	7	9
16	123	122	122	145	13	12	7	11
17	81	85	77	71	10	10	5	24
18	134	132	129	147	16	25	10	17
19	131	130	122	131	19	24	16	26
20	173	164	159	183	14	23	6	19
21	55	46	40	60	15	16	4	17
22	45	38	41	50	12	17	6	12
23	76	65	62	77	19	19	6	11
24	41	31	31	45	11	13	4	14
25	57	55	59	66	23	40	20	27
26	58	43	50	68	38	46	25	21
27	69	57	75	96	35	56	25	31
28	64	61	67	64	25	50	17	25
29	52	50	46	50	26	35	22	13
30	54	42	39	69	34	45	19	18
31	54	54	50	71	34	36	18	46
32	38	33	31	47	25	42	17	17
33	136	137	127	147	6	11	3	7
34	206	204	188	208	12	25	8	6
35	154	154	139	154	10	17	5	4
36	179	179	174	214	11	22	7	18

VITAE

Name Miss Su Mon Aung

Student ID 5810320026

Educational Attainment

Degree	Name of Institution	Year of Graduation
Bachelor of Technology	Technological University (Mandalay)	2012
Bachelor of Engineering	Technological University (Mandalay)	2013

Scholarship Awards during Enrolment

- Thailand's Education Hub for ASEAN countries (TEH-AC) award for Master Degree Program (2015-2017)

List of Publication

1. **Su Mon Aung** B.E, Kanyanatt Kanokwiroon PhD, Tonghathai Phairatana PhD and Surapong Chatpun, PhD. Computer-assisted cell counting for Trypan Blue stained cell images based on image segmentation and morphological operations. **(Submitted to Machine Vision and Applications)**