2020

A PCNN Framework for Blood Cell Image Segmentation

Carol D. Lenihan

A PCNN Framework for Blood Cell Image Segmentation

by

Carol D. Lenihan

A dissertation report for partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in
Computer Science

College of Computing and Engineering
Nova Southeastern University

2020

We hereby certify that this dissertation, submitted by Carol Lenihan conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

[illegible] — December 2, 2020

Michael J. Laszlo, Ph.D.
Chairperson of Dissertation Committee — **Date**

[illegible] — Dec 2, 2020

Francisco J. Mitropoulos, Ph.D.
Dissertation Committee Member — **Date**

Sumitra Mukherjee — December 2, 2020

Sumitra Mukherjee, Ph.D.
Dissertation Committee Member — **Date**

Approved:

[illegible] — December 2, 2020

Meline Kevorkian, Ed.D.
Dean, College of Computing and Engineering — **Date**

College of Computing and Engineering
Nova Southeastern University

2020

An Abstract of a Dissertation Submitted to Nova Southeastern University in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

A PCNN Framework for Blood Cell Image Segmentation

by
Carol D. Lenihan

This research presents novel methods for segmenting digital blood cell images under a Pulse Coupled Neural Network (PCNN) framework. A blood cell image contains different types of blood cells found in the peripheral blood stream such as red blood cells (RBCs), white blood cells (WBCs), and platelets. WBCs can be classified into five normal types – neutrophil, monocyte, lymphocyte, eosinophil, and basophil – as well as abnormal types such as lymphoblasts and others. The focus of this research is on identifying and counting RBCs, normal types of WBCs, and lymphoblasts. The total number of RBCs and WBCs, along with classification of WBCs, has important medical significance which includes providing a physician with valuable information for diagnosis of diseases such as leukemia.

The approach comprises two phases – segmentation and cell separation – followed by classification of WBC types including detection of lymphoblasts. The first phase presents two methods based on PCNN and region growing to segment followed by a separate method that combines Circular Hough Transform (CHT) with a separation algorithm to find and separate each RBC and WBC object into separate images. The first method uses a standard PCNN to segment. The second method uses a region growing PCNN with a maximum region size to segment.

The second phase presents a WBC classification method based on PCNN. It uses a PCNN to capture the texture features of an image as a sequence of entropy values known as a texture vector. First, the parameters of the texture vector PCNN are defined. This is then used to produce texture vectors for the training images. Each cell type is represented by several texture vectors across its instances. Then, given a test image to be classified, the texture vector PCNN is used to capture its texture vector, which is compared to the texture vectors for classification.

This two-phase approach yields metrics based on the RBC and WBC counts, WBC classification, and identification of lymphoblasts. Both the standard and region growing PCNNs were successful in segmenting RBC and WBC objects, with better accuracy when using the standard PCNN. The separate method introduced with this research provided accurate WBC counts but less accurate RBC counts. The WBC subimages created with the separate method facilitated cell counting and WBC classification. Using a standard PCNN as a WBC classifier, introduced with this research, proved to be a successful classifier and lymphoblast detector. While RBC accuracy was low, WBC accuracy for total counts, WBC classification, and lymphoblast detection were overall above 96%.

Acknowledgements

Listen to advice and accept instruction, that you may gain wisdom in the future.
(Proverbs 19:20)

I would like to thank my dissertation chair, Dr. Michael Laszlo, for agreeing to work with me on this project and providing tremendous feedback. Your valuable advice and guidance throughout instructed me in continuing and finishing this research. I would also like to thank my committee members, Dr. Francisco Mitropoulos and Dr. Sumitra Mukherjee, for taking the time to review. Also, thanks to all the faculty at NSU for their excellent teaching and dedication.

I would like to thank my parents, brother, sisters, friends, colleagues, and fellow students that provided me words of encouragement during this process. I am thankful to God that I was given the necessary tools to proceed down this path of achievement.

Trust in the Lord with all your heart, and do not lean on your own understanding. In all your ways acknowledge him, and he will make straight your paths.
(Proverbs 3:5-6)

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Background

A complete blood count (CBC) is a laboratory blood test that contains a count of leukocytes also called white blood cells (WBCs), erythrocytes also called red blood cells (RBCs), hemoglobin, hematocrit, and blood smear examination (Brown, 1980). A medical technologist performs a manual examination of a blood smear slide using a microscope to classify and count the percentage of each type of leukocyte. They also indicate the presence of any cells that are abnormal, premature, or contain parasites so those slides can be further analyzed by a hematologist or physician. The CBC can be used to diagnose some diseases, screen general health condition, and monitor patient during treatment (Brown, 1980).

The blood cells are created in the bone marrow and mature cells are circulated in the blood stream. Leukocytes help fight infections and erythrocytes carry oxygen to the body. However, under certain diseases and conditions, there is an increase (or decrease) in the number of blood cells, and sometimes immature or early cells can enter the blood stream. Immature cells seen in a blood smear can be indicative of disease.

A blood cell image typically has a higher number of RBCs than WBCs. The RBCs are usually smaller than WBCs, around 7-8 μm in diameter and the center may appear hollow, whereas WBCs are around 10-20 μm in diameter and contain a nucleus and depending on the type may have granules (Loddo, Putzu, Di Ruberto, and Fenu, 2016). The color absorbed during staining of the smear can be used to differentiate between an RBC and WBC, however, this can vary with the stain process. Leukocytes can be classified into neutrophil, lymphocyte, monocyte, eosinophil, basophil, and early (immature) cell types known as blasts. The types can be differentiated by a combination of color, size, number of lobes in nucleus, and presence of granules (Loddo et al., 2016). Erythrocyte size and shape can vary: microcell (smaller than normal), or macrocell (larger than normal), tear drop, sickle cell (shaped like a crescent and indicative of type of anemia), nucleated (premature RBC), malaria (parasite in RBC), among others (Brown, 1980). Thus, blood counts and classification can show indications for leukemia, anemia, and malaria to name a few diseases.

This section is organized as follows. First the problem is discussed and the dissertation goal is presented. The relevance and significance of the problem follows, along with the research questions of this dissertation. Barriers and issues are discussed next and this chapter concludes with a summary.

Problem Statement

Improper blood cell image segmentation and cell clumping lead to incorrect counts and classification that can result in misdiagnosis. To accurately count erythrocytes and leukocytes, the cells need to be separated from each other and the background (Loddo et al. 2016). Incorrect segmentation can result in counting one cell type as another (Acharya & Kumar, 2018). The staining process of blood smears can impact thresholding methods resulting in inaccurate results (Loddo et al. 2016). Choosing an incorrect threshold can cause improper segmentation (Acharya & Kumar, 2018). Quinones, Macawile, Ballado Jr., Dela Cruz, and Caya (2018) suggest that improvements in blood cell image segmentation are still needed to provide more accurate results. Thus, segmentation is a crucial step for blood cell image processing to provide accurate counts.

Besides segmentation, counting cells that are clumped together or visually overlapping can reduce accuracy if they are not correctly separated. Blood cell images can contain many clumps of multiple cells after segmentation (Loddo et al., 2016). Cells that are visually connected can be counted incorrectly (Acharya & Kumar, 2018). Improvements are needed for separation of overlapped and clumped cells as they can decrease segmentation accuracy impacting the cell counts (Savkare, Narote, & Narote, 2016). Counting the individual cells within a clump is important for obtaining correct counts.

Since a CBC normally includes a differential and detection of abnormalities, this should also be included during image processing of a blood smear for complete results. Determining types of WBCs is important as an increase in premature types of WBCs is associated with leukemia (Loddo et al., 2016). Abnormal growth of blood cells is indicative of leukemia, such as Lymphocytic which has an increase in lymphoblasts and Myelogenous (also known as Myeloid) which has an increase in myeloblasts, among others (Brown, 1980). Lymphocytic and Myeloid are two common types of leukemia containing different signatures, thus knowledge of the types of premature WBCs are crucial to diagnosis of which type the patient has (Khobragade, Mor, & Patil, 2015). Thus, detecting abnormal WBCs along with classifying the type of WBC should be included to aid in diagnosis.

A CBC provides important medical information to a physician to diagnose certain diseases that can be life-threatening to a patient. A hematology analyzer is an automated method to obtain these counts and is costly; a hemocytometer is a manual method for obtaining counts and thus is prone to error (Quinones et al., 2018). Some countries or regions do not have access to a laboratory for performing a CBC (Seth & Palodhi, 2017). To address these problems, image processing of digital blood images can potentially be used instead to identify, count, and classify leukocytes and erythrocytes and detect abnormalities. Therefore, the problem exists for improvements in segmentation and clump separation of blood cell image processing to provide accurate counts and classification.

Dissertation Goal

The goal of this research was to develop and assess image processing methods to segment and separate RBCs and WBCs from a blood smear image, classify WBCs, and count RBCs and WBCs. The development was split into three main areas: segmentation, separation, and classification. A framework was developed to experiment with different segmentation methods including threshold, watershed, and Pulse Coupled Neural Network (PCNN). A separate method was developed to find and separate the RBC and WBC objects from the segmented image using postprocessing, Circular Hough Transform (CHT), and separating the objects into sub images. The resulting WBC sub images were used for WBC classification using a PCNN classifier.

A PCNN is a single layer neural network where each pixel represents a neuron and neighbor neurons provide link information that pulses through the network (Kuntimad & Ranganath, 1999). PCNN was modeled after the cat's visual cortex (Eckhorn, Reitboeck, Arndt, and Dicke, 1990). It is an unsupervised method which can be used to segment objects from the background in a grey scale digital image. A PCNN was used to segment and count RBCs by Adagale and Pawar (2013) who combined it with templates and Ma, Liang, and Ma (2016) who combined it with image quality.

An issue with PCNN is tuning the parameters and knowing when to stop so that the best segmentation is achieved. Liu, Wang, Yan, and Huang (2016) used fuzzy entropy to determine the stopping criteria when segmenting WBCs using a simplified PCNN. Recently, Zhou and Shao (2018) proposed a multi object grey scale region growing PCNN image segmentation and Xu, Li, Lei, and Lv (2018) proposed a similar region growing PCNN image segmentation using color.

This dissertation examined two different PCNN stopping criteria. The first criterion uses a fixed number of iterations, whose value was determined by experiments that produce the best segmentation and subsequent separation of RBC and WBC objects. The second criterion uses a region growing PCNN with additional stopping criteria that specified a max region size. The PCNN segmentation experiments included using the intensity from the grey scale image for the feeding and linking part of a standard PCNN, and the spectral feeding for region growing PCNN as per Xu et al. (2018) for color values.

A separation method was developed that employs Circular Hough Transform (CHT) on the segmented image to find cells and then separates each RBC and WBC into images. Each object found is subsequently removed from the segmented image thus eliminating duplicated cells, unwanted edges, and separating cells in clumps. Preprocessing and postprocessing is also performed to facilitate separation.

For WBC classification, the textural information was captured with a PCNN that retrieved the entropy series and stored as the texture vector. A PCNN was used for image texture retrieval by Yang, Lyu, Liu, Zhou, Chen, Jiang, Li, Chen, Xu, and Wang (2017). The PCNN parameters were determined through experiments for the texture vector PCNN that produced the best classifier. The texture vectors and cell type were captured using the texture vector PCNN on the training dataset and stored. The PCNN classifier was used to capture the texture vector for a WBC from the testing dataset, and its texture vector compared to the stored texture vectors for WBC classification. WBCs were classified into neutrophil, lymphocyte, monocyte, and eosinophil, along with basophil and lymphoblasts depending on the dataset.

Relevance and Significance

There are several steps associated with counting blood cells using image processing: segmentation, classification, and counting. The segmentation stage separates the WBC and/or RBC from the background (Kolhatkar & Wankhade, 2016). Classification methods are used to separate types of WBCs as is done by a manual differential (Macawile, Quinones, Ballado Jr., Dela Cruz, and Caya, 2018). Counting may include separating cells that are clumped into single cells for more accurate counting (Loddo et al., 2016). This section discusses the relevance and significance as related to segmentation, classification, and counting blood cells.

Segmentation can be done by using a threshold to separate an object from the background. For thresholding into object and background, each pixel of a grey scale image is compared and if it is above a certain value it is specified as a 1 for object and otherwise 0 for background (Gonzalez and Woods, 2002). Otsu threshold is an algorithm for finding the optimal threshold that splits an object from the background. The Otsu threshold is used by Acharya and Kumar (2018) to segment RBCs from the background. Shankar, Deshpande, Chaitra, and Aditi (2016) use Zack threshold on converted color space to segment the WBCs from the background.

Another segmentation method is clustering, where the objects of similar values are grouped together. A common clustering method is k-means where the number of clusters is specified by the value of *k*. A blood cell image is typically stored by its red, green, and blue color values known as RGB color space. It can be converted to another color space or grey scale. CMYK represents an image using cyan, magenta, yellow, and black, and Lab uses luminance and chromaticity components a and b. Abdul Nasir, Mashor, and Rosline (2011) used clustering to segment the WBC in one step and the WBC nucleus in the next step. Savkare and Narote (2015) used k-means clustering, where *k* is equal to 2 to separate the cells from the background. Vogado, Veras, Andrade, Araujo, Silva, and Medeiros (2016) converted the image to CMYK and Lab, extract the M and b components, and perform k-means clustering to segment the WBCs. Most of the k-means clustering methods segment either the RBCs or WBCs. However, Jagadev and Virani (2017) used k-means clustering on Lab color space to separate into WBC nucleus, RBC and WBC cytoplasm, and background. While Zhang et al. (2014) used a combination of color transfer and k-means clustering to separate the background, RBC, and WBC nucleus.

A support vector machine (SVM) which is a supervised machine learning classification method can also be used for segmentation. Di Ruberto, Loddo, and Putzu (2016) used a Nearest Neighbor Search (NNS) and SVM model to segment the image into WBC, RBC, and plasma.

Pulse Coupled Neural Network (PCNN) is another method for image segmentation. A PCNN was used by Mao-jun, Zhao-bin, Hong-juan, and Yi-de, (2008), Adagale and Pawar (2013), and Ma et al. (2016) to segment the RBCs from the background. A simplified PCNN and fuzzy entropy was used by Liu et al. (2016) to segment WBCs. This dissertation used a PCNN to segment the image which subsequently was separated into WBC and RBC objects.

Once the image is segmented the cells need to be separated for accurate counting. From the literature, Watershed Transform, Circular Hough Transform (CHT), and templates are used as well as combinations. Watershed transform is used by Savkare et al. (2016) to separate clumped RBCs. Acharya and Kumar (2018) used watershed and Circular Hough Transform (CHT) with a specified radius to separate and count RBCs. Di Ruberto et al. (2016) counts clumped WBCs using CHT and a specified radius along with the grey level reference values. Dela Cruz Valiente Jr., Castor, Mendoza, Song, and Torres (2017) use an estimated count for RBC clumps based on the clump size. Ma et al. (2016) used CHT and average radius for template creation. Templates are also used by Adagale and Pawar (2013) to determine the count based on the clump size. The separation of cell objects for this research used CHT with a specified radius based on cell type along with object acceptance criteria.

Determining the type or classification of a WBC is crucial for accurate counts. Alreza and Karimian (2016) used and SVM model for classification after extracting WBC features such as color, texture, and number of lobes in the nucleus. An Artificial Neural Network (ANN) was used by Manik, Saini, and Vadera (2016) to classify WBCs into three categories (neutrophil, lymphocyte, and eosinophil). Jagadev and Virani (2017) used an SVM to determine if cells were leukemic after extracting statistical, geometrical, color, and textural features. Ghosh, Singh, and Sheet (2017) used a deep Convolutional Neural Network (CNN) with average pooling to determine if the image contained lymphoblasts. Macawile et al. (2018) used transfer learning and a CNN to classify WBC cells from blood image into neutrophils, lymphocytes, monocytes, eosinophils, and basophils. Liang, Hong, Xie, and Zheng (2018) used a combination of CNN and Recurrent Neural Network (RNN) to classify WBCs into lymphocyte, eosinophils, monocyte, and neutrophils. SVM and NN methods typically require a large set of labeled data for the training of the network.

Other classification methods compare different features. Khobragade et al. (2015) detected abnormal types of WBCs by comparing statistical, textural, geometrical, and color features between normal and blast cells. A PCNN can be used to extract statistical features for texture classification (Yang, Lyu, Liu, Zhou, Chen, Jiang, Li, Chen, Xu, Wang, 2017). Since a PCNN process is iterative until the stop criterion is reached, the resulting output contains a series of images. A feature vector can be calculated from the time and entropy series that is unique and invariant to large changes in scale and rotation

(Zhan, Zhang, Ma, 2009). The classification of WBCs for this project used a PCNN and calculated a feature vector from the entropy series which was compared to a known set of vectors for WBC types.

Research Questions

There were three main stages performed by this research: segmentation, separation, and classification. The questions associated with each stage in this section were answered by this research as described in Chapter 4. All questions are related specifically for blood cell images.

The segmentation step was used to segment the objects of interest from everything else. Segmentation for the purpose of this research was generation of a binary image that contained either WBCs, RBCs or both and was used by the separation stage to separate into WBC and RBC objects. The first experiment used the intensity grey scale.

RQ1: What are the significant PCNN parameters impacting PCNN segmentation?

The region growing PCNN added a color option and experiments were done on different color spaces. It also segmented the image into regions reducing the number of objects in the separation stage.

RQ2: What color channels and image processing methods improve the results of PCNN segmentation and separation?

The separation stage generated an image for each circular object found in the segmented image that was used for counting and WBC classification. Some postprocessing was required during this stage. This stage captured each object into a WBC and RBC list. This was a precursor for WBC classification and for counting.

RQ3: Does PCNN segmentation with postprocessing identify edges for CHT to find and differentiate between WBC and RBC objects?

The classification stage was used to classify the type of WBC. This stage used a PCNN to capture the texture vector of the WBC object and compared with the texture vector list stored per type of WBC. The PCNN and parameters used to capture the texture vector for the WBC test object was the same as the one used to generate the texture vectors. There were two sets of stored texture vectors, one for each dataset (ALL_DB and Kaggle).

RQ4: What are the significant PCNN parameters that yield the best texture vector results for each dataset, or which worked generally across datasets?

Barriers and Issues

Some of the PCNN code was available from Lindblad and Kinser (2013) which was used as a starting point for the standard model, however, additional code was developed for other PCNN models along with the *separate* method. A PCNN framework was developed to select different segmentation methods for verification of this research.

A texture vector representation for each cell type was created for WBC classification. The two datasets used were the ALL_DB from Labati, Piuri, and Scotti (2011) and the Kaggle blood-cell dataset from Mooney (2018). Difference between these datasets required generation of two texture vector representations, one for each dataset. The real counts of RBC and WBCs for each image tested was manually calculated for analysis of metrics.

The lymphoblasts from the ALL_IDB contained center values which were used to obtain the real counts. The other WBCs from the ALL_IDB were manually classified into lymphocytes, neutrophils, monocytes, basophils, and eosinophils. The Kaggle dataset contained WBC classifications for lymphocytes, eosinophils, monocytes, and neutrophils, but did not contain basophils or lymphoblasts; thus, those types were not included in the results for that dataset. Only a subset from each dataset was used for counting and classification in this research. The quality of images from the Kaggle dataset was different than those of the ALL_IDB and as such required different preprocessing and was not used for RBC counting.

Summary

The research for this project created a PCNN framework for adjusting parameters and testing segmentation methods, developed a separation method that used CHT, classified WBCs, and counted cells. The remainder of this paper is organized as follows. Chapter 2 contains the literature review, Chapter 3 contains details of the methodology used in this research, Chapter 4 contains the dissertation results, and Chapter 5 summarizes this report.

Chapter 2

Review of the Literature

There are several steps associated with counting blood cells using image processing: preprocessing, segmentation, postprocessing, separation, counting, and classification. The preprocessing stage includes image enhancement and denoising and the segmentation stage segments the cells from the background (Kolhatkar & Wankhade, 2016). Counting may include separating cells that are clumped into single cells for more accurate counting (Loddo et al., 2016). Classification methods can be used to determine types of WBCs as is done by a manual differential (Macawile, Quinones, Ballado Jr., Dela Cruz, and Caya, 2018).

Since the goal of this research was related to segmentation, separation, counting, and classifying WBCs, this section contains a review of the literature in these areas. Some of the reviews in this section were concerned with just segmentation, counting, or classifying and not necessarily all of these. This chapter first contains a brief discussion on color spaces, followed by segmentation methods and includes preprocessing. Next is a section on separation and counting which also includes any postprocessing. The chapter is wrapped up with a section on classification methods followed by a chapter summary.

Color Spaces

Images can be represented in different color spaces. RGB is a common color space that is represented by the colors red, green, and blue. Grey scale is represented by shades of grey using a value between 0 and 1. A binary image is represented by black or white, represented with a value of 0 or 1, respectively. HSV is known as hue, saturation, and value, and HSI is hue, saturation, and intensity. The intensity component of HSI does not contain any color value as that is retained in the other two components (Zhang et al., 2014). The CMYK color space is represented by the colors, cyan, magenta, yellow, and black. It is considered a subtractive model and is used in color printing (Zhang, 2014). Lab color space is represented by luminance and chromaticity components a and b, which indicate brightness, and colors red to green, and blue to yellow, respectively (Savkare et al., 2015; Jagadev and Virani, 2017). The Haematoxylin Eosin Diaminobenzidine (HED) is a color space that contains haematoxylin (blue), eosin (magenta-red), and diaminobenzidine (brown) and is used in histology and cytology (Ruifrok & Johnston, 2001). The Luv color space, like Lab, consists of luminance containing the light or brightness along with the u, v parts containing the color for red to green, and blue to yellow, respectively. Blood cell images are typically stained using wright stain where RBCs are red, WBCs have a blue color for the nucleus, WBC cytoplasm is a lighter blue or red, eosinophils have an orange color to the granules, and basophils have a purple color to the granules. Using different color spaces can have an impact on the segmentation result.

Segmentation Techniques

Segmentation for blood cell images is the process that partitions the image into the objects of interest. In a binary segmented image, the objects of interest are represented with a pixel value of 1 and everything else is background with a pixel value of 0. For a blood cell image, it can represent either all cells or just RBCs or WBCs. An image can be segmented into multiple objects of interest, where each object contains a pixel value of non-zero with background pixel values of 0. In this case a blood cell image could represent WBCs with a value of 1 and RBCs with a value of 2.

Methods such as thresholding, clustering, region growing, edge based, PCNN, among others can be used to segment an image (Kolhatkar & Aankhade, 2016; Chouhan, Kaul, Singh, 2018). Soft computing methods can also be used for image segmentation such as Fuzzy Logic (FL), ANN, and Genetic Algorithm (GA) (Chouhan, Kaul, Singh, 2018). Since most methods used for blood cell segmentation contain some preprocessing, these are included in this section with the segmentation. This section includes those related to blood cell segmentation such as thresholding, clustering, edge, active contours, SVM, region growing, PCNN, and Neural Networks (NN).

Thresholding

Segmentation using a threshold can be based on a global or local threshold. A global threshold is one that is done for the entire image, and a local threshold can be used for a subset of the image. Kim, Kim, Song, Park (2000) used thresholding with fuzzy logic to select the threshold to segment WBCs and RBCs. Mohamed, Far, and Guaily (2012) enhanced the intensity of WBCs with linear contrast, histogram equalization, adding images, and Otsu thresholding to segment WBCs. Gautam and Bhadauria (2014) finds the optimal threshold with Otsu thresholding and uses it to segment the WBCs from the background using some morphological preprocessing and postprocessing to remove RBCs. Khobragade et al. (2015) perform WBC segmentation by using thresholding, first converting RGB to grey scale, then histogram equalization and linear contrasting, add and subtract the enhanced images, and Otsu thresholding to convert to binary. Le, Bui, Yu, and Bui (2015) convert to HED then grey scale manually choosing the best threshold to segment WBCs.

Gatc and Maspiyanti (2016) use a preprocessing median filter, double thresholding which includes filling holes, and morphology methods to segment both RBCs and WBCs. Shankar, Deshpande, Chaitra, and Aditi (2016) convert RGB to CMYK to obtain higher contrast of WBCs when converted to grey scale, then Zack thresholding is used on grey scale image to segment the WBCs from the background. Alreza and Karimian (2016) segmented WBCs by using a combination of RGB and CMYK converted to grey scale followed by the Zack thresholding algorithm, where the nucleus and cytoplasm are then obtained by subtracting the nucleus from the whole leukocyte. Manik et al. (2016) segments in two steps; first they convert the RGB to grey scale, use adaptive histogram equalization, Otsu threshold, and morphological operations to segment the cells; then they convert the RGB to HSV, obtain separate G and S segments from RGB, and HSV, respectively, subtract S from G, and apply morphological operations to segment the nucleus.

Dela Cruz, Valiente Jr., Castor, Mendoza, Song, and Torres (2017) convert RGB into HSV and use HSV thresholding to segment the blood cells into RBC, WBC, and platelets. Quinones et al. (2018) segment the WBC by first converting the image to HSV color space and extracting the S component and convert to grey scale, then binarization using a threshold. According to the authors using the S component eliminated the need for preprocessing using morphological methods. Acharya and Kumar (2018) segment RBCs using Otsu threshold where the mean intensity of the red channel is retained, after first converting from RGB to Lab color space and then converting to grey scale. They also preprocess using histogram equalization for images requiring contrast adjustment.

Clustering

Another segmentation method is clustering, where the objects of similar values are grouped together. K-medoids and k-means are two clustering methods. An issue with clustering is knowing the *k* value or number of clusters to generate as too small a value will combine unlike objects and too large can split objects. Sinha and Ramakrishnan (2003) used k-means clustering after converting to HSV color space to find the WBC nucleus, then crop around to capture the entire WBC image for further processing.

Rawat, Singh, Bhadauria, and Kumar (2014) compared different segmentation algorithms for WBCs where k-means clustering had the best results. Zhang et al. (2014) used a combination of color transfer and k-means clustering. The color transfer is done in RGB color space and is used to adjust the image color to match more closely that of a known good color image to correct for variations in staining. The RGB color is then converted to both HSI and CMYK to obtain characteristics that are more prominent in those spaces. K-means clustering is done on each color space to separate the background, RBC and WBC nucleus. RBC segmentation was done by subtracting the nucleus part from the combination of RBC and nucleus. Likewise, cytoplasm segmentation required image enhancement using bottom hat transformation and then subtracting the nucleus part from the entire WBC part. However, Zhang et al. (2014) did not separate clumped WBCs and determined accuracy based only on resulting segmentation.

Savkare and Narote (2015) used k-means clustering, where *k* is equal to 2 to separate the cells (both RBC and WBC) from the background. The image was first preprocessed to remove noise and enhance the image using both a median and Laplacian filter. Both the original and preprocessed images are converted into Lab color space and k-means clustering performed where the results are added together. A global threshold is used based on the Hue-Saturation in HSV color space on poorly stained images. Savkare et al. (2016) segment the cells from background by converting RGB to CMYK and then use k-mean clustering with *k*=2. First, they preprocess the image with background removal and contrast stretching to remove noise and enhance the image.

Vogado et al. (2016) converts to both CMYK and Lab to extract the M and b components, receptivity, subtracts the images and perform k-means clustering, followed by morphological postprocessing to segment the nucleus of WBCs. Jagadev and Virani (2017) also used k-means clustering but set *k*=3 to segment into WBC nucleus, RBCs and WBC cytoplasm, and background. Acharya and Kumar (2018) extracted WBCs using k-medoids algorithm.

Edge, Active Contour Methods

Edge based methods can be used to segment an image such as canny, sobel, prewit, and others. Active contours or snake method finds the contour of objects based on an energy minimizing curve that follows the contour or edges of objects. Ongun, Halici, Leblebicioglu, Atalay, Beksac, and Beksac (2001) use the active contour method also known as snakes to segment, by first finding the initial position for the snake based on threshold of WBC nucleus, then minimization where the center is considered the WBC location. Yang, Meer, and Foran (2005) modified a snake algorithm to work with a new color gradient after converting the RGB to Luv color space and tested on already created WBC images segmenting both cytoplasm and nucleus.

Puttamadegowda and Prasannakumar (2016) first preprocess the image by converting to grey scale, using a median filter and normalization, followed by a fuzzy clustering algorithm. The RGB image is also preprocessed with a Gaussian filter and a snake algorithm is used to get the WBC objects, the two images are fused into a segmented WBC image so they can be counted. Seth and Palodhi (2017) first preprocess the grey scale image using contrast adjustment, then segment using Gabor filter, and standard edge detection (sobel and prewit).

Support Vector Machine (SVM)

A support vector machine (SVM) is a supervised machine learning classification method. Di Ruberto, Loddo, and Putzu (2016) use a Nearest Neighbor Search (NNS) and SVM model to segment the image into WBC, RBC, and plasma. They trained the SVM using cross validation and labeling selected regions of interest (ROI) for each of the three classes. A nearest neighbor search (NNS) is used based on the RGB values, where duplicates, outliers, and intersections are all removed; thus, the results contain a clean segmentation. Loddo et al. (2016) used the same machine learning approach as Di Ruberto et al. (2016) to segment the image.

Region Growing

Region growing starts with a selected pixel and adds neighbor pixels to the region based on criteria on similarity of neighbor pixel to the selected pixel. The region continues to grow with neighboring pixels provided they meet the similarity measure. When there are no more neighbors, another pixel is chosen, and the next region is grown until all pixels in the image belong to a region. The selection of the start pixels has an impact on the segmentation outcome as the region grows from those pixels. Adams and

Bischof (1994) proposed a seeded region growing where the selected pixels are known as seeds and represent the starting points for each group or set, which are manually selected based on the images.

Abdul Nasir, Mashor, and Rosline (2011) proposed a method that used k-means clustering and region growing. They segment the WBCs in one step and then the WBC nucleus in the next step. In the first step, the min and max for each RGB color is obtained using a linear contrast technique and distributed over the histogram range, this stretched RGB image is converted to HSI color space, then k-means clustering is done using the H component and region growing is applied till cluster centers are stabilized resulting in the segmented WBCs. The second step takes the segmented WBCs and using the S component, k-means clustering, and region growing to segment the nucleus. Abdul et al. (2011) did not separate, count, or classify the WBCs.

Rashid, Mashor, and Hassan (2015) segment RBCs by first preprocessing to enhance the image with global contrast, then converting to HSI color space, followed by using a moving k-means clustering algorithm and median filter. Since they were looking to extract RBCs, they compared the segmented results from the H, S, and I components and removed the S component that contained the WBC nucleus. This was followed with a seeded region growing algorithm to remove platelets and any clumped RBCs and WBCs.

Pulse Coupled Neural Network (PCNN)

A PCNN is a different type of neural network that does not require training. It iteratively cycles through a set of equations using several parameters to generate a sequence of segmented images. Determining the parameter settings and iteration that contains the best segmentation is an area of active research. There are different variations of PCNN that use slightly modified equations from the standard.

A PCNN was used by Mao-jun, Zhao-bin, Hong-juan, and Yi-de, (2008) for both noise reduction, RBC segmentation, and adjusting parameters such that the autowave characteristics also removed artifacts. Li, Zhou, Chen, and Shi (2010) proposed a grey scale iterative PCNN where the threshold for determining the PCNN output is based on an iterative grey scale value. While not specific for blood cell images, they did show segmentation results for RBCs using their method.

Both Adagale and Pawar (2013) and Ma et al. (2016) used a PCNN to segment RBCs. A simplified PCNN was used by Liu et al. (2016) to segment WBCs from the blood image using fuzzy entropy for determining the best segmentation result, however they did not separate, classify, or count.

There are also some region growing PCNNs, while these were not used for blood cell image segmentation, they are included here for their similarity to what was done with this research. Stewart, Fermin, and Opper (2002) proposed a region growing PCNN as a replacement to seeded region growing, where the feeding input contains the grey scale value, the linking input is the sum over the eight nearest neighbors minus a positive constant, and the linking strength is updated each iteration. The pixel with the highest intensity is selected as the seed pixel and set as a fired neuron. The PCNN iterates and captures each region until the stopping criteria is met. Inside is a fast linking loop that iterates until no new neurons fire. Once the stopping criteria for a region is met, a new seed pixel with the highest intensity of the remaining pixels is selected for the next region which stops once all neurons have pulsed.

Zhou and Shao (2018) proposed a modified PCNN region growing algorithm that segments and separates regions of interest into multiple levels by subdividing the unfired region into a new level or class. The linking strength is calculated at each level based on fuzzy logic of the grey scale values for determining those belonging to that group. The stopping criteria is based on a calculated distance between the fired and unfired regions.

Xu et al. (2018) proposed a color region growing PCNN that adds a linking control unit so it can handle color pixel values, which they used Lab color space normalizing the L channel to a set range. Their PCNN and algorithm has similarities to Stewart et al. (2002) although they removed the positive constant from the linking network, randomly set the seed neuron, set the initial linking strength value, and added a new minimum size to the stopping criteria. The stopping criteria for Xu et al. (2018) includes: all neurons have fired, exceeded a maximum beta value, exceeded a mean difference, and exceeded a minimum region size, where the first three are the same as from Stewart et al. (2002). This research built on the work from Xu et al. (2018) and added another stop criterion for maximum size of a region to capture WBC and RBC cell objects.

Neural Networks (NN)

While NN are used for classification, they can also be used for image segmentation. An Artificial Neural Network (ANN) is based on the human brains neurological system (Chouhan et al., 2018). The input neurons are connected to the hidden layers which connect to the output layer with each connection containing a weight that is learned based on the training data. A convolutional Neural Network (CNN) is a NN that contains an input, convolutional, pooling, and output layers and can be used for image classification (Chouhan et al., 2018).

Ghosh, Singh, and Sheet (2017) uses a pretrained AlexNet model and tuned it with blood cell images from ALLDB, which are preprocessed to create more images by mirroring and rotation. The blood images are sent through the CNN's generated heatmap from an average pooling layer to generate a filtered image using a threshold that can segment out the WBCs. A second filter is created by first converting the RGB to HED space and using a threshold on the eosin channel. Yu, Chang, Yang, Zhang, Shen, Xia, Sha (2017) used a CNN with transfer learning using 5 different models pretrained from ImageNet. Their own blood cell images are preprocessed, sent through the five CNN models, and classification results are based on a vote. A contour aware CNN is used by Razzak and Naz (2017) to segment and separate cells, then using color descriptors for each cell image the cells are classified as RBC or WBC, whose cropped images are sent to an extreme learning machine for classification.

Macawile et al. (2018) used a CNN that was trained using models from AlexNet, ResNet101 and GoogleNet after first preprocessing the image to be the required size for the model, then segment and classify WBCs using transfer learning. A Recurrent Neural Network (RNN) is a NN that can be used for sequential data as they contain a memory of past data. Liang et al. (2018) used a combination of CNN and RNN to segment and classify WBCs. They use parameters from a CNN pretrained on ImageNet as input to their CNN, which has a convolution layer which uses two window sizes, a pooling layer, and output goes to the merge layer. The blood cell images are preprocessed with matrix transformation for rotation and to limit overfitting before being input to the RNN, which goes through its hidden layers and then to the merge layer. The merge layer combines the

CNN and RNN features, passes through a fully connected softmax layer to generate the output based on its probability distribution.

Separation and Counting Techniques

This section covers the methods used to separate the cells from the segmented image so they can be counted. The counts may be for total RBCs, WBCs or both and may include methods for separation of clump cells. Postprocessing methods are those methods used after segmentation and during the separation stage so the cells can be counted.

Common methods used to separate cells are Watershed Transform and Circular Hough Transform (CHT). Templates are also used to determine estimated counts based on clump sizes. Not all papers separated or counted the cells. This section includes a review related to CHT, templates and estimates, watershed, and distance transforms, connected component, and edge methods that were used for separation and/or counting of blood cells.

Circular Hough Transform (CHT)

CHT is a method for finding circular object in a digital image. It works by finding circles from the edge points and voting for those that intersect. Using a fixed radius reduces the number of circles and execution time. Shankar et al. (2016) use Hough transform to obtain a roundness ratio as part of their postprocessing. Di Ruberto et al. (2016) gets the reference size and shape from the training set, counts clumped WBCs using CHT with the specified radius and matching with the grey level values from the original image to exclude erroneous circles. Loddo et al. (2016) uses the same method as Di Ruberto et al. (2016) additionally counting RBCs.

Seth and Palodhi (2017) take the segmented edge image and use CHT specifying a radius for an RBC diameter of 7 to 8.5 um and double that for WBC to count single RBCs and WBCs. After segmentation and postprocessing to separate RBCs, Acharya, and Kumar (2018) counted RBCs using Circular Hough Transform (CHT) and a specified radius. According to their results CHT counts were more accurate than using a labeling algorithm.

Template and Estimates

Adagale and Pawar (2013) used template matching on the segmented image after passing it through a median filter to remove noise. The templates were bins of different sizes based on area where each bin is assigned a count value that is used to determine the count for those clumps. Ma et al. (2016) extract RBC edges using image quality and use CHT to get the average radius to create a template, which is used on the binary image to get a matching map that contains points for each RBC. Dela Cruz, et al. (2017) estimated the RBC count based on each clump size and the expected single RBC size.

Distance and Watershed

Watershed transform is a method that finds or follows edges based on grey scale values as if it were a geographical watershed basin. Distance transform determines a distance measure for each pixel to its nearest boundary. Savkare and Narote (2015) separated clumped RBCs using watershed transform after first detecting edges using Sobel edge detector. Le, Bui, Yu, and Bui (2015) first use a bilateral filter, then canny edge detector, followed by watershed to separate clumped WBCs.

Watershed transform is also used by Savkare et al. (2016) to separate clustered RBCs for counting. Shankar et al. (2016) separate clumped WBCs using watershed and distance transform, then further postprocess to remove unwanted objects, then the single and clumped WBCs are counted. Alreza and Karimian (2016) separated clumped WBCs for counting by using distance conversion and applying watershed on round leukocytes. Ghosh, Singh, and Sheet (2017) separate clumped WBC objects by using their centroid and distance-based algorithm to extract the potential WBC objects which are sent through their CNN to classify as normal or abnormal.

Acharya and Kumar (2018) separate RBCs from the binary image by first using watershed transform to remove overlapping or touching cells, using morphological open to filter out noise, then removing WBCs by deleting the largest objects until all WBCs are gone using a previously extracted WBC mask. Quinones et al. (2018) separate the WBCs for counting from the segmented image by cycling through all blobs, performing postprocessing, and based on area and eccentricity decide if it should be counted or if further splitting is required. Postprocessing included image cropping, distance and watershed transforms, and filtering.

Connected Component

Connected component labeling determines pixels are in the same region by checking its connectivity of the neighbor pixels and labels pixels belonging to the same region. Mohamed, Far, and Guaily (2012) used morphological operations on the binary image, determined objects based on neighbor connectivity, and removed objects smaller than a certain size retaining only WBC objects. WBCs. Dela Cruz, Valiente Jr., Castor, Gatc and Maspiyanti (2016) separate the WBCs and RBCs by determining blobs that are connected using a grass-fire algorithm that calculates a value based on intensity and size, then classifies as RBC or WBC based on the area. Mendoza, Song, and Torres (2017) use connected component labeling after morphological postprocessing on the segmented image which contains three types of cells, WBCs, RBCs, and platelets. This gives the count for each type in the segmented image; however, the RBC count is estimated based on the total area and approximate size of single RBC.

Edge Based

Khobragade et al. (2015) use a filter on the segmented binary image to remove noise and Sobel edge detection to capture the WBC nucleus. They extract features for leukemia detection, but do not count cells.

Classification Techniques

This section contains a review of the classification methods used for blood cell images. Determining the type of WBC or whether the WBC or RBC is normal or abnormal is also part of a CBC. Some papers were concerned with classifying the WBCs as being normal or cancer, whether RBCs were normal or abnormal, and others classified the WBCs by type such as lymphocyte, monocyte, basophil, eosinophil, and neutrophil. This research classified based on lymphoblast or normal and further classified normal types of WBCs into lymphocyte, monocyte, basophil, eosinophil, and neutrophil.

Kim, Kim, Song, Park (2000) extracted 76 features and used Principal Component Analysis (PCA) to reduce the number of features and a three-layer NN to classify RBCs and WBCs. Ongun, Halici, Leblebicioglu, Atalay, Beksac, and Beksac (2001) extracted 57 features from the WBCs and compared different classification methods including SVM which had the best results. Sinha and Ramakrishnan (2003) extracted features from the WBC nucleus and cytoplasm and compared different classification methods where the NN had the best results.

Gautam and Bhadauria (2014) extract features such as area, perimeter, circularity, and eccentricity from the WBC objects, where the min and max for each feature is calculated for each type during training and used later for classification into neutrophil, eosinophil, basophil, monocyte, and lymphocyte. Khobragade et al. (2015) detected abnormal types of WBCs (blasts) indicative of different types of leukemia by extracting statistical, textural, geometrical, and color features, where the statistical features had the most impact.

Alreza and Karimian (2016) extracted WBC features such as color, texture, and number of lobes in the nucleus with an SVM model for classification. An ANN was used by Manik et al. (2016) to classify WBCs into three categories (neutrophil, lymphocyte, and eosinophil) based on features extracted from both cell and nucleus segmentation. Vogado et al. (2016) classifies WBCs as normal or cancer, but paper did not specify what features were extracted or how the classification was performed.

Syahputra et al. (2017) classified RBCs as normal or abnormal type based on the shape using a Radial Bias Function Network (RBFN) where results showed two types of abnormal cells. Jagadev and Virani (2017) extracted statistical, geometrical, color, and textural features for WBCs and used an SVM to determine if cells were leukemic. Ghosh, Singh, and Sheet (2017) use their CNN to classify WBCs as normal or abnormal. Yu et al. (2017) classify WBCs into monocytes, lymphocytes, basophils, eosinophils, neutrophils, and atypical lymphocytes using a CNN and transfer learning. Razzak and

Naz (2017) classify RBC and WBC using an extreme learning machine using the ALL_IDB dataset for training and testing WBC classification.

Macawile et al. (2018) use transfer learning and a CNN to classify WBCs into neutrophils, lymphocytes, monocytes, eosinophils, and basophils. The models from AlexNet, ResNet101 and GoogleNet with transfer learning for WBC classification, although they did not mention how they performed the transfer learning. Acharya and Kumar (2018) used a form factor to determine abnormal RBCs and highlight those with a bounding box. Liang et al. (2018) classify WBCs into eosinophil, monocyte, lymphocyte, and neutrophil from the Kaggle dataset using a combination of CNN and RNN. Transfer learning is used to pass parameter weights from a pretrained model to the CNN, and the CNN and RNN are trained with the blood cell images. Their classification results had an accuracy of 90.79% for one of their CNN-RNN models. The training took approximately 14 hours with an average of 3.8 seconds for one blood cell test image.

From the literature, textural information is a feature useful for classifying WBCs. The time series and entropy from PCNN segmentation can be used to retrieve textural information. Zhan, Zhang, and Ma (2009) compared standard PCNN image segmentation for texture features (time series, entropy, average residual, and standard deviation) with the variants Spiking Cortical Module (SCM) and Intersecting Cortical Module (ICM) to determine impact of angle rotation and scale. Chacon and Mendoza (2011) used PCNN time series combined with Fuzzy C-Means (FCM) algorithm for image segmentation based on features. An SCM version of PCNN was used for image texture retrieval by Yang, Lyu, Liu, Zhou, Chen, Jiang, Li, Chen, Xu, and Wang (2017) using entropy series, time series, and average residual. This research attempted to use the PCNN textural information to classify WBCs.

Summary

There are several methods that can be used to segment blood cell images, however some pixel values are similar between RBCs and WBCs, for example WBC cytoplasm. Separating the RBCs and WBCs is also a challenge as they can be clumped together or close together that the edges overlap. There are several features that can be extracted from the image to classify WBCs and different methods have been used to achieve this. The most recent being the use of NN, however, these require large labeled datasets and long training periods. This research used PCNN to segment, a separate method employing CHT to find objects and then separate into object images, a PCNN classifier to retrieve textural information and classify WBCs and counted RBCs and WBCs.

Chapter 3

Methodology

Introduction

This project developed a PCNN framework to perform segmentation, separation, and classification of blood cells for subsequent counting and obtaining metrics. The principal objective was to identify the ideal PCNN parameters and variants used to provide the best segmentation of the image. After segmentation, the image was processed to find and separate the cells for counting and classification. A method called *separate*, was developed that employs post processing, CHT, and an algorithm to separate into object images. The algorithm in *separate* creates a list of individual WBC and RBC images for each cell object found that matches criteria specified in parameters; these objects were later used to count and classify. The details of the algorithm are described in this section. Each WBC object was classified using another PCNN to capture a texture vector that was compared to texture vectors of known types.

PCNN Overview

A PCNN is a neural network that does not require training. The PCNN works by receiving the image pixels, neighboring pixels, and state information and using an iterative series of equations produces a sequence of segmented images. The two inputs to the PCNN are the feeding and linking networks. For each PCNN iteration, the previous state from each input is combined using a convolution operator with a neighbor weight matrix and tuning parameters; the feeding network also receives the image pixel as input.

On each iteration the output is determined using a threshold which is adjusted so that it decays over time. The result is a sequence of segmented images, one for each iteration. The PCNN parameters are used to tune the segmentation behavior. Choosing the tuning parameters and stopping criteria that provides the best segmentation, separation, and classification of blood cells is one of the goals of this dissertation. Experiments performed showed the parameters that had an impact on segmentation and those that had an impact on classification, which are described in Chapter 4.

A PCNN diagram is shown in Figure 1, where there is one neuron for every pixel in an image, where I represents the image and x a pixel in the image. F_x represents the feeding network for pixel x, L_x represents the linking network for pixel x, and N_x represents the neighbors of pixel x. The values of V_L, V_F, V_E are normalizing constants and α_L, α_F, α_E represent decay factors. Each neuron receives input from the feeding network (F_x) and the linking network (L_x) where these are combined with a linking strength variable (β) to form an internal state (U_x) (Lindblad & Kinser, 2013). Both the feeding and linking networks receive input from pixel neighbors and the feeding network also receives the intensity of the pixel from the image. The impact of the neighbors of pixel x is determined by the weight matrices M and W for the feeding and linking networks, respectively. The output neuron Y_x is considered fired when U_x is above a threshold (E_x), so the output Y contains 1 for pixels considered fired and 0 for those not fired. The PCNN iteratively cycles through the following 5 equations where Y_y represents the output for the pixel and its neighbors from the previous iteration. Each iteration uses the previous iteration multiplied by a decay parameter and then combined with the neighbor weight matrix, normalizing constant, and previous output results. The result is a series of segmented images contained in *Y* for each iteration *n*. For the implementation, *F*, *L*, *U*, *Y*, and *E* are represented as *numpy* arrays for the dimension of the image since each of these is representative of each pixel in the image. The initial value for *E* can be 0 or can vary per the user implementation (Lindblad & Kinser, 2013).

1) $F_x[n] = e - {}^{\alpha F}\, F_x[n-1] + I_x + V_F \sum_{y \in N_x} M_{xy} Y_y[n-1]$

2) $L_x[n] = e - {}^{\alpha L}\, L_x[n-1] + V_L \sum_{y \in N_x} W_{xy} Y_y[n-1]$

3) $U_x[n] = F_x[n]\,(1 + \beta L_x[n])$

4) $Y_x[n] = \begin{cases} 1 & if\ U_x[n] > E_x[n-1] \\ & 0\ otherwise \end{cases}$

5) $E_x[n] = e - {}^{\alpha E}\, {}_{E_x}[n-1] + V_E Y_y[n]$

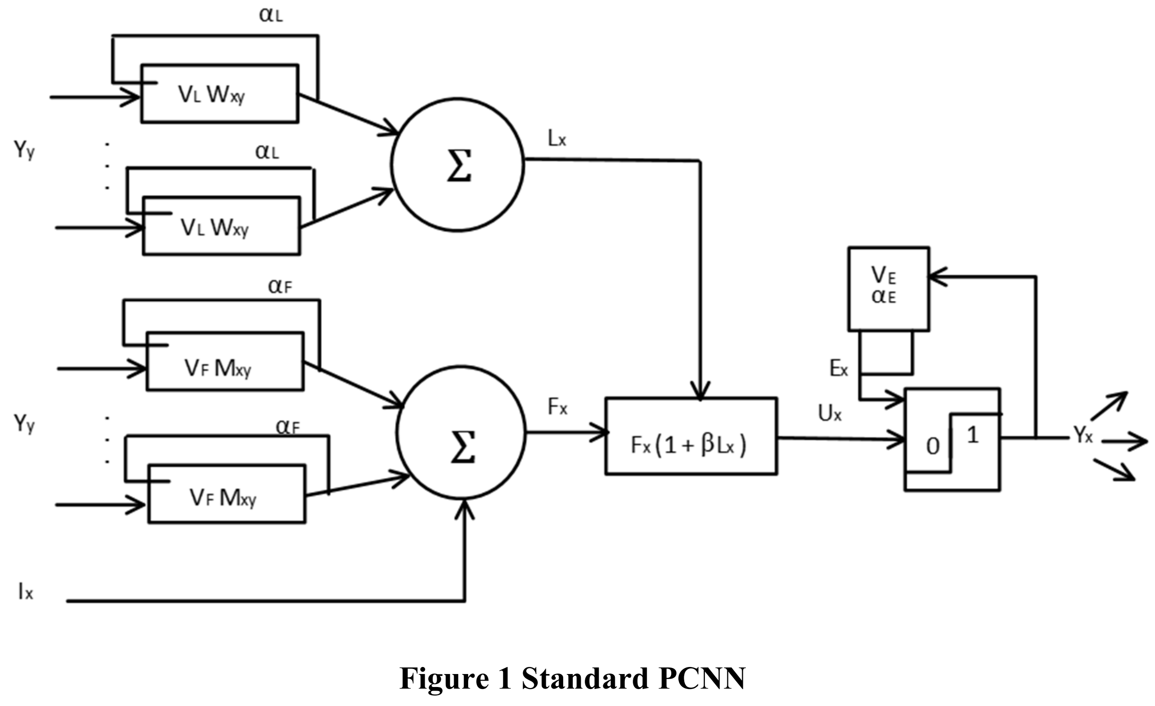


Figure 1 Standard PCNN

Key Parameters

The key parameters that were fine-tuned were the linking strength and normalizing constants (β, V_F, V_L, V_E); the decay parameters (α_F, α_L, α_E); the connected neighbor weight matrixes (*M* and *W)*; the PCNN type; and using color channels. Deng, Yan, and Ma (2019) show the impact of different PCNN parameters on firing times and other PCNN characteristics.

The Python *cspline2d()* Gaussian function was used for weight matrices *W* and *M* by Lindblad and Kinser (2013). A weak neighbor weight matrix = [[0, .01, 0,], [.01, .11, .01], [0, .01, 0]] was suggested by Deng, et al. (2019). Zhou and Shao (2018) proposed to set *M* as the center of a square matrix and *W* using Euclidean distance shown below where *x* is the center of the neighborhood, *y* is the spatial position related to the center, C_s and C_w are normalized constants, and σ_S is a scale factor. The values C_s, σ_S, and C_w were all set to 1 in the experiments.

$$M_{x,y} = C_S exp(-|| x - y||^2 / \sigma_S{}^2)$$

$$W_{x,y} = \begin{cases} C_W * 1/||x - y||_2, & x \neq 0 \\ 0, & x = 0 \end{cases}$$

The experiments included using the above for either *M* and/or *W*. The *M* and *W* values are used in this document by: Cspline represents the *cspline2d*() method, Weak represents weak neighbor coupling, Euclidean represents strong neighbor coupling, Exponential represents the formula above using *exp*, and Common represents a value = [[0.5, 1, 0.5,], [1, 0, 1], [0.5, 1, 0.5]].

Another key parameter explored was using the intensity value from a grey scale image or color space values. The standard PCNN model takes for a pixel value or a grey scale intensity value as input. However, Xu et al. (2018) extended a region growing PCNN variant for using color channels, combining the color channels as input to the PCNN model. The experiments used a standard PCNN with intensity value input, a PCNN region growing variant intensity value input, and another PCNN region growing variant with color inputs. The different color models tested, included RGB, HSV, HED, and Lab color space.

PCNN Variants

The PCNN variant types implemented included a standard Eckhorn and region growing models. Lindblad and Kinser (2013) contained PCNN Python code for the standard Eckhorn PCNN model. This was used as a base line for the PCNN standard implementation and modified for this research. The next variants were region growing PCNN, the first based on Stewart et al. (2002) that used grey scale intensities; the second was based on Xu et al. (2018) that was extended for using color channels.

Texture Features

As mentioned earlier PCNN is a sequence of equations where each iteration generates a segmented image. The Shannon entropy value can be calculated on each image, and the resulting sequence of entropy values, represented as a vector for the texture features of the image, also referred to as the entropy signature. The use of a texture vector for image classification was done in Yang et al. (2017). While a PCNN does not require training for segmentation, using PCNN for classification does require creating a vector list on a training set of known images. A list of texture vectors per dataset was created for the following WBC types: lymphoblast, lymphocyte, monocyte, neutrophil, eosinophil, and basophil depending on the dataset. The entropy values were obtained using the *skimage.filters.rank.entropy*() function. For completeness, the entropy calculation is shown below, where P_1 and P_0 are the probabilities of a pixel value being 1 or 0, respectively:

$$H(P) = -P_1 log_2(P_1) - P_0 log_2(P_0)$$

Framework Test Environment

The framework test environment contains segmentation, separation, counting, classification, and texture vector list creation for classification. Images were selected from the ALL_IDB and Kaggle datasets. Since training was not required for PCNN segmentation, a subset of images was chosen from both the training and testing sets. However, the training images were used to create the two sets of texture vector lists, one for each dataset. The classification was tested on a subset of images chosen from both training and testing sets.

The framework was used for segmentation, separation, counting, and classification. The classification step is dependent on the creation of the texture vector list which is described later in this section. Each image was segmented using the PCNN framework into a binary image representing WBCs, RBCs, or both. The cells were then separated into RBC and WBC cell objects using the separation method, *separate*, that employs post processing, CHT and an algorithm described in this section. The separated objects were saved as a separate image and stored in a Python list to facilitate counting; WBCs were classified after separation. The pseudo code is shown in Figure 2.

Framework Testing

```
1.  Foreach RGB image
2.     PCNN_framework_segmentation()
3.     Separate()
4.     Count()
5.     foreach WBC object
6.        classify()
7.     end
8.  end
```

Figure 2 Framework Test Pseudocode

Segmentation

The PCNN framework allowed experiments for different methods of segmentation including different PCNN variants and two conventional methods using the *PCNN_framework_segmentation()* method. The PCNN variants are described in detail later in this chapter. Preprocessing was done as part of the framework before the segmentation. This included converting the image to grey scale and inverting the grey scale image so that the WBCs would have the higher intensity. The Kaggle dataset also required the images to be cropped to remove extra white space that interfered with the PCNN segmentation. The PCNN framework segmentation is shown in Figure 3.

PCNN_framework_segmentation()

```
1. Input: RGB Image
2. Output: Segmented Binary Image(s)
3. Preprocess and segment image based on segmentation
   strategy
```

Figure 3 PCNN Framework Segmentation

Separation

A *separate* method was developed and included in the framework to find and separate cell objects in the segmented image using postprocessing, CHT, specified radius, and a separation algorithm. Postprocessing is done on the segmented image; for WBCs, small holes were removed followed by erosion; for RBCs, erosion was done, followed by removing small holes and then dilation. The method next finds circular objects by employing CHT with a specified radius searching first for WBCs and then RBCs. The last step was to perform the separation algorithm.

The CHT method was employed by using two methods from *skimage.transform* to create the Hough transform and retrieve the peaks using *hough_circle()* and *hough_circle_peaks(),* respectively. Since the region growing PCNN segments into different regions, the number of Hough transform and peaks should be smaller, so a parameter was included for the *separate* method called *find_num* to modify parameter settings for the *hough_circle_peaks()* method to adjust the number of peaks to retrieve from the Hough space. The value of *find_num* used for each segmentation method are described in Chapter 4. The *hough_circle()* creates the Hough transform and includes a *min*, *max,* and a *radii* value for the minimum and maximum radius, and the number of radii. The *radii* were set to a value of 3, and since WBCs are larger than RBCs, the *min* and *max* was based on the type, using 20 and 55 for RBCs and 45 and 175 for WBCs.

The separation algorithm cycles through the circle centers found from CHT. First it removes duplicate circle centers from the list to speed up processing and reduce finding duplicate cells. Next it generates a potential circle based on the center and radius found and compares it to the circle acceptance criteria. The circle acceptance is based on parameters for percentage, intensity, and radius offset. The percentage is the minimum value of how much of the circle is contained in the binary image (the *Y* output from PCNN), the intensity is the minimum value for the average intensity of the circle based on the original grey scale image, and the radius offset increases the radius based on these parameters. The acceptance criteria parameters for the *separate* method are: *wbc_offset*, *wbc_percent*, *wbc_intensity*, *rbc_intensity*, *rbc_percent*, and *rbc_offset.* The parameter values and defaults used are described in Chapter 4.

Once a circle object is found, it is matched to the above criteria and a circle is created using the center and determined radius. An image of size 257 by 257 is created containing a circle object in the center with the original grey scale image pixel values for the object and the radius of the circle. This image is added to the list of WBC or RBC images. The circle object is also removed from the working segmented image (the *Y* output from PCNN) by setting those pixel values to 0 and the method continues to find the next circle. The removal of the selected object from the working segmented image was to allow for overlapped cells to be found. WBC objects are used later for classification. The *separate* method is shown in Figure 4. This answered research question RQ3 discussed in Chapter 4.

separate()

```
1.  Input: Segmented binary image
2.  Output: List of WBC images, List of RBC images
3.  perform WBC postprocessing
4.  while find acceptable WBC object
5.    add object to WBC list
6.    remove object from segmented image
7.  end
8.  perform RBC postprocessing
9.  while find acceptable RBC object
10.   add object to RBC list
11.   remove object from segmented image
12. end
13. return WBC and RBC lists
```

Figure 4 Separate

Count Cells

Since the framework contained a list of WBC and RBC objects from the *separate* method, the count was obtained by using the Python *len()* function to get the number of objects in the list.

Classification

To classify a WBC, first the texture vector list had to be created. This was done by the framework using the *generate_texture_vector()* method which is described later in this section. The texture vector list contains a list of texture vectors (entropy series), for WBC types from the training dataset. The entropy series was used for this project as it produced the best results from Yang et al. (2017). A WBC object is sent through another PCNN, but this time it is used as a PCNN classifier as it retrieves the texture vector or entropy signature for that image.

The texture vector generated by the PCNN classifier was compared to the texture vector list to find the closest matching vector which determines the type. To compare the texture vectors, the Euclidean distance was used, as was done by Yang et al. (2017). The distance was calculated for a WBC object as follows, where j ranges over the number of iterations N, v_j is the j^{th} iteration for a vector v in the texture vector list, x_j is the j^{th} iteration of the WBC texture vector, and d is the distance value for the WBC vector and a vector v in the texture vector list. The distance was calculated for a WBC object and each vector v in the texture vector list.

$$d = \sqrt{\sum_{j=1}^{N} (x_j - v_j)^2}$$

The *argmin* was then used to find the *index* into the texture vector list that has the minimum distance value, where *i* ranges over the vectors stored in the list, v_i is the *i*th vector in the list, *x* is the query vector, and δ is the distance function. Once the *index* was found, the WBC type was retrieved using the *index* on the texture vector list.

$$index = argmin_i \delta(x, v_i)$$

The *classify()* method is shown in Figure 5. The value of *N* was determined by the experiments adjusting the PCNN parameters and answered research question RQ4 discussed in Chapter 4.

classify()

```
1. Input: Grey scale image of cell object
2. Output: WBC type
3. capture texture vector of cell object
4. foreach cell type in texture vector list
5.     calculate Euclidean distance
6. end
7. determine cell type based on min distance
8. return cell type
```

Figure 5 Classify

Generate Texture Vector

To classify WBC objects using the entropy texture vector, a list of known WBC types and texture vectors had to be created. This required a dataset of images containing the region of interest (ROI) for different types of WBCs (neutrophil, lymphocyte, monocyte, basophil, eosinophil, and lymphoblasts). The ROI images were available from the ALL2_IDB and Kaggle datasets, so texture vector representations were created from each dataset. However, the WBC still had to be segmented and separated from the rest of the image. The lymphoblasts were labeled in the ALL2_IDB, but the lymphocytes, neutrophils, eosinophils, monocytes, and basophils had to be labeled as part of this project. The Kaggle dataset contained labels for lymphocytes, monocytes, eosinophils, and neutrophils so only those types were tested for that dataset.

The ALL_IDB2 and Kaggle datasets contain training images with one or two labeled WBCs along with multiple RBCs. These images were used by the framework to segment out one WBC from the RBCs and once a WBC object was separated, the texture vector was obtained by segmenting the WBC image using the PCNN classifier and capture the texture vector for each WBC type. The texture vectors were stored along with their type. The pseudo code is shown in Figure 6.

Generate Texture Vectors

```
1. Input: RGB Images containing one WBC and some RBCs
2. Output: List of texture vectors per WBC type
3. foreach image
4.   segment_and_separate()
5.   capture_texture_vector()
6. end
7. generate texture vector list
```

Figure 6 Generate Texture Vectors Pseudocode

The *segment_and_separate()* method, shown in Figure 7, experimented using conventional segmentation methods (threshold, watershed) along with a standard PCNN method to segment the image. It called the *separate* method mentioned earlier from the framework to generate a WBC object image based on the original grey scale pixel values, which is the output of this method.

segment_and_separate()

```
1. Input: RGB Image with one WBC and RBCs
2. Output: Grey scale Image of just one WBC
```

Figure 7 Segment and Separate

The *capture_texture_vector()* method shown in Figure 8, used a standard PCNN as a PCNN classifier to capture the texture vector. The PCNN parameters were adjusted from experiments that determined the best classifier and value of *N*. The *texture_vector[s_1, s_2, s_3, …, s_N]* where s_i is the Shannon entropy for iteration *i* and *N* is the number of iterations.

```
capture_texture_vector()
1. Input: WBC object image, WBC type
2. Output: texture_vector[$s_1$, $s_2$, $s_3$, …, $s_N$]
```

Figure 8 Capture Texture Vector

The texture vector list contained several vectors for each type. The number saved depended on the dataset. Since the ALL_IDB2 dataset was relatively small (max 106 of one type), all texture vectors were included. However, the Kaggle dataset contained a larger training dataset (greater than 1000) so it saved the mean of every ten images, therefore keeping the texture vector list size at roughly 100 to 300 per WBC type. The idea was to keep the dataset size relatively small so classify would not have to compare a large texture vector list, but large enough to allow for differences in texture vectors per type.

Novel PCNN Methods

Depending on the PCNN model and parameters, knowing when to stop iterating is crucial for accurate segmentation. This project explored a few stopping methods. One method was to adjust parameters for PCNN segmentation and use the *separate* method to find all WBC and RBC objects. Another was to use region growing PCNN from Xu et al. (2018) and adding a new *Smax* parameter, then used the *separate* method to find the WBC or RBC object.

PCNN with CHT

The standard PCNN was used with different parameters to segment the image. The *separate* method was then used on the segmented image to find and separate the objects employing postprocessing, CHT, and the separation algorithm. The goal was to determine the parameters and iteration number that contained the best segmented image as input to the *separate* method such that it generated the most accurate WBC and RBC counts, as well as accurately capture the WBC objects. The ideal and significant parameters were determined from the experiments and shown in Chapter 4. This answered research question RQ1 as discussed in Chapter 4.

Region Growing PCNN with CHT

Based on the method from Xu et al. (2018), this research used a region growing PCNN. The new region growing PCNN added a *Smax* for each class (WBC, RBC) and saved the objects to a list as it grows to *Smax*, thus it tried to capture individual cells or small areas of cells. The seed selection picked the intensity from the WBC range first, then RBC range. The *separate* method was used to then find the cell or cells from each segmented image to capture the WBC and RBC cell objects. This answered research question RQ2 discussed in Chapter 4.

Conventional Segmentation and PCNN Prior Work

The PCNN segmentation methods were compared to a couple conventional segmentation methods along with a PCNN segmentation using the parameters from Ma et al. (2016). The conventional methods and PCNN parameters from Ma et al. (2016) were used to segment out cells from the background. All segmented binary images from these methods were used as input to the *separate* method to find and separate the cells. The output image list was used for counting total RBC or WBC for comparison results.

Threshold

One standard method to segment WBCs or RBCs from the background used a global threshold. A threshold was chosen from experiments that provided a segmentation of the cells from the background.

Watershed Region Growing

A watershed region growing method was available in scikit-image. This method works by first obtaining edges using the sobel method, then setting up markers for the basins based on thresholds. It was used to segment all cells from the background by setting different threshold values. The thresholds values were set from experiments.

Standard PCNN

A standard PCNN was used with the parameters from by Ma et al. (2016) to segment the cells from the background. This was used as a comparison of the separation and counting stages of this research.

Comparing Prior Work

The results of this research were compared to prior work in several ways which are described in this section. The first was to use the metrics described later in this chapter for determining accuracy of the total RBC and WBC counts. These metrics were calculated for prior work where applicable. The metrics from this research for WBC classification using the ALL_IDB and Kaggle datasets were compared based on the metrics in the literature. A metric based on the accuracy of lymphoblast detection from the ALL_IDB dataset was also compared.

Total RBC and WBC Counts

The results presented in the literature relied on several metrics. The metrics from Adagale and Pawar (2013) and Ma et al. (2016) were calculated based on a computed count percentage as described in Chapter 4. That metric was also used in this research for comparison with those papers. The accuracy, precision, recall, and F1 metrics were calculated for this research based on total WBC and RBC counts and compared to the metrics from Loddo et al. (2016) which also used the ALL_IDB dataset.

WBC Classification and Counts

A confusion matrix was generated that showed the predicted classification and the actual classification for each WBC type. The overall true positive, false negative, true negative, and false positive results were calculated, and the overall sensitivity, specificity, and accuracy metrics were generated. These metrics are described in the next section. The WBC classification metrics from this research using the ALL_IDB dataset was compared to the work done by Macawile et al. (2018) and included the average sensitivity, specificity, and accuracy per WBC type. The WBC classification metrics from this research for the Kaggle dataset were compared to the work presented in Liang et al. (2018) which included the overall accuracy metric.

Lymphoblast Detection

The metrics from this research for the detection of lymphoblasts using the ALL_IDB dataset was compared to the work presented in Ghosh et al. (2017), which included sensitivity, specificity, and accuracy regarding lymphoblast detection.

Metrics and Data

As was done by Macawile et al. (2018), Ghosh et al. (2017), and Loddo et al. (2016), the dataset from Labati, Piuri, and Scotti (2011) was used to determine metrics based on total RBC and WBC counts, WBC classification, and lymphoblast detection. The Kaggle dataset was used for WBC classification as was done by Liang et al. (2018). The ALL_IDB dataset was used for WBC classification and lymphoblast detection. The Kaggle dataset contained a labeled training set for eosinophils, monocytes, lymphocytes, and neutrophils. The ALL_IDB required labeling the eosinophils, monocytes, lymphocytes, basophils, and neutrophils. The classification required generation of the texture vector list from each dataset described earlier.

Analysis

Several metrics were calculated for comparison analysis. Accuracy, precision, recall, and F1-measure were calculated which are described in this section. Both Adagale and Pawar (2013) and Ma et al. (2016) calculated a computed count percentage described in Chapter 4. Loddo et al. (2016) calculated precision, recall, and F1-measure. To calculate the metrics the actual counts for each image were required. The ALL-IDB and Kaggle datasets contain counts for some of the WBC types, however, RBCs and some WBCs needed to be manually counted. For comparison results, some metrics were calculated from the data provided in the literature. Not all metrics below were calculated for all tests but calculated as needed for comparisons.

Accuracy

Accuracy is the percent of true values in proportion to the total possible values. Accuracy calculation is shown below where *TP* represents true positive, *TN* represents true negative, *FP* represents false positives, and *FN* represents false negatives.

$$accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} * 100$$

Precision

Precision is the number of true positives values in proportion to the total number of positive values detected. Precision calculation is shown below using the previously defined definitions.

$$precision = \frac{(TP)}{(TP + \ FP)}$$

Recall

Recall provides the sensitivity of the result, so it is also known as sensitivity and true positive rate. The recall calculation is shown below using the previously defined definitions.

$$recall = \frac{(TP)}{(TP + FN)}$$

Specificity

Specificity is the number of true negative values in proportion to the total number of negatives and false positive values detected. It is also known as the true negative rate. The specificity calculation is shown below using the previously defined definitions.

$$specificity = \frac{(TN)}{(TN + \ FP)}$$

F1-measure

The F1-measure represents the harmonic mean and uses precision and recall. The F1-measure is shown below using the previously defined definitions.

$$F1\text{-}measure = 2 * \frac{(precision * recall)}{(precision + recall)}$$

Actual Image Counts

The actual counts were retrieved from the dataset files or manually counted.

Resources

This section lists the type of tools that were necessary for this dissertation. For this research, Python (https://www.python.org/) was used since there are image processing and machine libraries available with scikit-image (https://scikit-image.org/) and scikit-learn (https://scikit-learn.org/stable/), which are all open source. Other libraries that were required were NumPy (http://www.numpy.org/) and SciPy (https://www.scipy.org/). The image processing library scikit-image was written in Python for a wide range of image processing functions (Van der Walt, Schonberger, Nunez-Iglesias, Boulogne, Warner, Yager, Gouillart, Yu, and the scikit-image contributors, 2014). The scikit-learn is a library of machine learning algorithms written in Python that includes methods for Support Vector Machine (SVM) and clustering (Varoquaux, Louppe, Pedregosa, Buitinck, Grisel, and Mueller, 2015). PCNN Python scripts were available from Lindblad and Kinser (2013) with the book purchase and were used for the basis of the standard PCNN and modified for use in this research.

These were installed on a PC running Windows. The ALL_IDB datasets from Labati et al. (2011) were requested and granted. The Kaggle dataset from Mooney, P. (2018) was also requested and received. Both datasets were used for this research.

Summary

The main purpose of this research was to identify PCNN parameters and variants for segmentation, separation into individual WBC and RBC images, and WBC classification using a PCNN classifier. The framework presented here facilitated the use of different segmentation methods. The framework was designed and implemented so that PCNN variant types, parameters, and stopping criteria were easily changed. The strategy design pattern was used so that different stopping criteria and other algorithmic changes could be plugged in and interchanged by supporting a common interface.

Chapter 4

Results

Introduction

This section discusses the results from the experiments. The ideal PCNN parameters and variants that provided the best segmentation of the image from the experiments are described in this section. The framework development allowed the preprocessing and segmentation methods used to be plugged in based on the segmentation method selected. The *separate* method used was the same for all segmentation methods, although adjustments in parameters were required which are explained in this chapter.

This chapter first discusses the different datasets and subsets used for the experiments. Next an overall discussion on preprocessing and separation, followed by the standard PCNN, Region Growing PCNN, and conventional segmentation and separation results. The discussion on the WBC classification results are described next, and then the comparison of results to prior work, with this chapter ending in a conclusion.

Datasets

Subsets of the ALL_IDB and Kaggle datasets were used based on the experiment type. The ALL_IDB and Kaggle datasets were requested and access provided from Labati, Piuri, and Scotti (2011) and Mooney (2018), respectively. A summary of those used for segmentation are shown in Table 1, those used for classification are shown in Table 2, and additional details including file sizes are in Table 3. The first set of experiments used the *ALL_IDB_subset* for determining PCNN segmentation and separation parameters. These experiments tested a wide range of different parameter combinations with those showing the biggest impact described in this chapter. There were approximately 16 different parameter settings and 7 different segmentation methods, although not every parameter was used for every method. However, due to the large number of combinations tested, each requiring a manual count for accuracy, 20 files from the ALL_IDB2 was selected that included all WBC types. The number 20 was chosen as it allowed for multiple experiments to test a wide range of parameters. The files were selected from the ALL_IDB2 dataset as the number of cells per slide was less, thus making the manual counts more accurate. Next the segmentation, separation, and cell counts were done with the *ALL_IDB_large*, which contained a subset of 6 randomly selected larger files from the ALL_IDB dataset. The number 6 was selected to test the parameter combinations on the different segmentation methods using the files that contained many cells, which also required manual counting and accuracy metrics. The *Kaggle_subset* files were chosen for similar reasons for segmenting, separation, and WBC counting.

For classification, the texture vector list was created for *ALL_IDB2* and *Kaggle_Train*. Experiments were unsuccessful to achieve PCNN parameters that worked across datasets so two texture vector lists were created, one for each dataset. Further details on the dataset creation are described later in this chapter. The files in *ALL_IDB2* were characterized into subdirectories by type and then the texture vector list was created. The *Kaggle_Train* files were already characterized by subdirectory which was used to create the texture vector list.

The *ALL_IDB_subset*, *ALL_IDB_large*, *ALL_IDB_large2, ALL_IDB_large3, Kaggle_Test_simple* and *Kaggle_Test* were each used to segment, separate, and classify WBCs with the corresponding dataset texture vector list. The *ALL_IDB_subset*, *ALL_IDB_large*, *ALL_IDB_large2, ALL_IDB_large3,* and *ALL_IDB_first33* were used for lymphoblast detection. The *ALL_IDB_large2 and ALL_IDB_large3* subsets were created to add more files to the testing data for WBC classification and lymphoblast detection. The *ALL_IDB_first33* were used for lymphoblast detection as these contained files with lymphoblasts; these files were used by Loddo et al. (2016) as they were all the same size and resolution. The Kaggle files provided some challenges as the staining was not as clear, and some files were created by rotation, resulting in white space that interfered with the PCNN segmentation, which was adjusted by cropping those files.

Dataset (total files)	**Segmentation Usage**		**Summary**	
	RBC Count	**WBC Count**	**Name**	**Number of Files**
ALL_IDB2 (260 files)	Yes	Yes	*ALL_IDB_subset*	20
ALL_IDB1 (108 files)	Yes	Yes	*ALL_IDB_large*	6
Kaggle (10,323 files)	No	Yes	*Kaggle_subset*	13

Table 1 Segmentation Dataset Summary

Dataset (total files)	Classification Usage		Summary	
	Training	Testing	Name	Number of Files
ALL_IDB2 (260 files)	Yes	No	*ALL_IDB2*	214
ALL_IDB2 (260 files)	No	Yes	*ALL_IDB_subset*	20
ALL_IDB1 (108 files)	No	Yes	*ALL_IDB_large*	6
ALL_IDB1 (108 files)	No	Yes	*ALL_IDB_large2*	10
ALL_IDB1 (108 files)	No	Yes	*ALL_IDB_large3*	11
ALL_IDB1 (108 files)	No	Yes	*ALL_IDB_first33*	33
Kaggle (9957 files)	Yes	No	*Kaggle_Train*	9957
Kaggle (71 files)	No	Yes	*Kaggle_Test_Simple*	71
Kaggle (2501 files)	No	Yes	*Kaggle_Test_subset*	80

Table 2 Classification Dataset Summary

Dataset (total files)	Subset Details		
	Name	File Names or Number	Size
ALL_IDB2 (260 files)	*ALL_IDB_subset*	1. Im001_1.tif	257 x 257
		2. Im002_1.tif	257 x 257
		3. Im021_1.tif	257 x 257
		4. Im024_1.tif	257 x 257
		5. Im084_1.tif	257 x 257
		6. Im091_1.tif	257 x 257
		7. Im123_1.tif	257 x 257
		8. Im135_0.tif	257 x 257
		9. Im153_0.tif	257 x 257
		10. Im154_0.tif	257 x 257
		11. Im156_0.tif	257 x 257
		12. Im166_0.tif	257 x 257
		13. Im192_0.tif	257 x 257
		14. Im201_0.tif	257 x 257
		15. Im203_0.tif	257 x 257
		16. Im212_0.tif	257 x 257
		17. Im246_0.tif	257 x 257
		18. Im251_0.tif	232 x 257
		19. Im253_0.tif	232 x 257
		20. Im260_0.tif	232 x 257
ALL_IDB1 (108 files)	*ALL_IDB_large*	1. Im001_1.jpg	1712 x 1368
		2. Im004_1.jpg	1712 x 1368
		3. Im016_1.jpg	1712 x 1368
		4. Im088_0.jpg	2592 x 1944
		5. Im091_0.jpg	2592 x 1944

		6. Im108_0.jpg	2592 x 1944
ALL_IDB1 (108 files)	*ALL_IDB_large2*	1. Im005_1.jpg	1712 x 1368
		2. Im006_1.jpg	1712 x 1368
		3. Im017_1.jpg	1712 x 1368
		4. Im020_1.jpg	1712 x 1368
		5. Im062_1.jpg	2592 x 1944
		6. Im063_1.jpg	2592 x 1944
		7. Im05_0.jpg	2592 x 1944
		8. Im079_0.jpg	2592 x 1944
		9. Im090_0.jpg	2592 x 1944
		10. Im093.0.jpg	2592 x 1944
ALL_IDB1 (108 files)	*ALL_IDB_large3*	1. Im068_0.jpg	2592 x 1944
		2. Im073_0.jpg	2592 x 1944
		3. Im074_0.jpg	2592 x 1944
		4. Im075_0.jpg	2592 x 1944
		5. Im076_0.jpg	2592 x 1944
		6. Im077_0.jpg	2592 x 1944
		7. Im078_0.jpg	2592 x 1944
		8. Im081_0.jpg	2592 x 1944
		9. Im082_0.jpg	2592 x 1944
		10. Im083_0.jpg	2592 x 1944
		11. Im084_0.jpg	2592 x 1944
ALL_IDB1 (108 files)	*ALL_IDB_first33*	12. Im001_1.jpg – Im033_1.jpg	1712 x 1368
Kaggle (10,323 files)	*Kaggle_subset*	1. _0_687.jpeg	320 x 240
		2. _0_884.jpeg	320 x 240
		3. _0_1022.jpeg	320 x 240
		4. _0_1338.jpeg	320 x 240
		5. _0_2399.jpeg	320 x 240
		6. _0_4170.jpeg	320 x 240
		7. _1_6343.jpeg	320 x 240
		8. BloodImage_0002.jpg	640 x 480
		9. BloodImage_0007.jpg	640 x 480
		10. BloodImage_0020.jpg	640 x 480
		11. BloodImage_0053.jpg	640 x 480
		12. BloodImage_0066.jpg	640 x 480
		13. BloodImage_0074.jpg	640 x 480
Kaggle (9957 files)	*Kaggle_Train*	Eosinophil - 2497 files	320 x 240
		Lymphocyte – 2483 files	320 x 240
		Monocyte – 2478 files	320 x 240
		Neutrophil – 2499 files	320 x 240
Kaggle (71 files)	*Kaggle_Test_Simple*	Eosinophil – 13 files	320 x 240
		Lymphocyte – 6 files	320 x 240
		Monocyte – 4 files	320 x 240
		Neutrophil – 48 files	320 x 240
Kaggle (2501 files)	*Kaggle_Test_subset*	Eosinophil - 20 files	320 x 240
		Lymphocyte – 20 files	320 x 240
		Monocyte – 20 files	320 x 240
		Neutrophil – 20 files	320 x 240

Table 3 Dataset Details

Preprocessing

As mentioned in the Chapter 3, preprocessing converted the color image to grey scale and then inverted the grey scale image so that WBCs would appear brighter. The Kaggle dataset also required some cropping of the images to remove the white space that was present due to creation of those images via rotation.

Separation

The *separate* method processes one or more segmented binary image. The standard PCNN contains one segmented image with all cells, however, the region growing PCNN methods contains a list of segmented images as there is a segmented image for each region. The method first searches for WBC cells and then RBCs by performing post processing mentioned earlier, then employs CHT to find the circles. The default values for *hough_circle()* was described in the Chapter 3. The *find_num* parameter was used here for setting the *max_peaks* value for the *hough_transform_peaks()* method for maximum peaks to retrieve. The default value for the *find_num* parameter is the Python *None* value, which indicates to use a *max_peaks*=500, which was used for standard PCNN and conventional methods, however, for region growing it was set lower as the goal was to capture like cells in a segmented region. Additional details of this parameter and other parameters settings for each segmentation method are described under those sections.

At this point the separation algorithm processes each circle center as described earlier. Once an object is found that passes the acceptance criteria, an image of that object is stored. The working image represents a binary image and 0 indicates background and 1 indicates object, which are displayed in Figure 10 as black and yellow, respectively. As objects are found, the pixels representing that object are set to 0 in the working image, thus removing that object. Figure 9 shows the original and segmented image and Figure 10 shows the working image after the WBC object was removed. The algorithm cycles to the next circle center which is processed using the acceptance criteria and the working image.

An example of a segmented image using file ***Im201_0*** from the *ALL_IDB_subset* and the standard PCNN is shown in Figure 9, along with the original image and an image containing the found objects marked with circles (red for RBC and blue for WBC). Figure 10 shows a few working images of ***Im201_0***, where image *1* on the top left contains the working image with the first object removed (the WBC) from the segmented image; images *2* and *3* contain subsequent working images, images 4 – 8 are not shown, and image *9* shows the last working image with 9 cell objects removed. Figure 11 shows two of the images created, the WBC object and an RBC that appears in the upper-left between two other cells in Figure 9 and Figure 10 image 1 and is shown removed from the upper-left of the working image in Figure 10 image *2*.

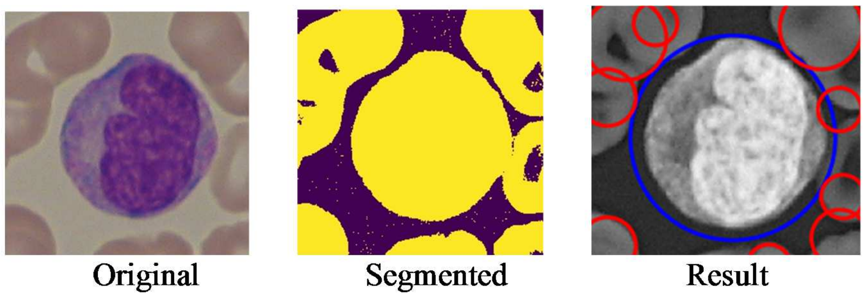


Figure 9 Original, Segmented, and Result of Im201_0

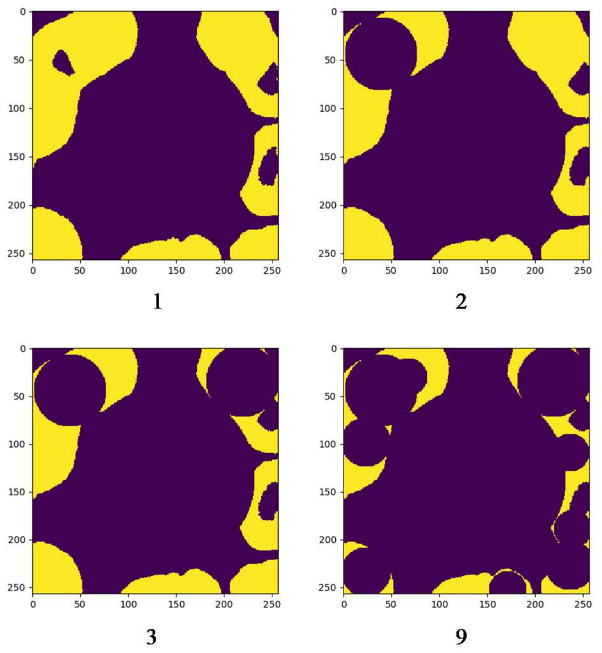


Figure 10 Working Images (Im201_0)

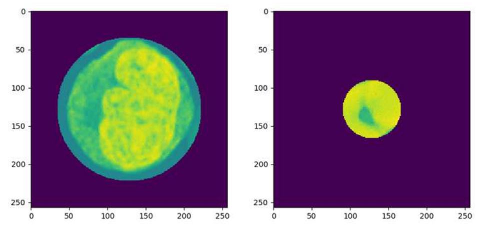


Figure 11 WBC and RBC (Im201_0)

PCNN with CHT

The first experiments were related to using standard PCNN and the CHT based separation method, called *separate*. The key PCNN parameters discovered from the experiments are described in this section along with the *separate* method parameters.

Key Parameters

As mentioned, the segmentation experiments were done on the *ALL_IDB_subset, ALL_IDB_large, and Kaggle_subset.* The intent was to determine the optimal parameters using *ALL_IDB_subset,* since those files were smaller, and then use those parameter settings for capturing the counts for the larger images. However, a couple of key parameters appeared to be impacted by the size of the image.

A few parameters shown in Table 4 were fixed from early experiments as they did not significantly impact segmentation results. These parameters were also used by Deng, Yan, and Ma (2019) include the normalizing constants V_F, V_L, V_E, and decay parameters α_L, α_E. The initial E value was set to .0001 as per the implementation from Lindblad and Kinser (2013).

V_F	V_L	V_E	α_L	α_E
.2	.2	.9740	1	.0771

Table 4 Standard PCNN Segmentation Fixed Parameters

The β parameter and the decay parameter, α_F were impacted by the images of larger size as shown in Table 5. The β parameter represents the linking strength and for smaller size files the value of .05 provided slightly better RBC accuracy, however, a value of .1 worked better on the larger images which may be related to a change in α_F decay parameter. For the smaller files, a decay parameter α_F value of 0.69 with the above β values provided the optimal segmentation, with 0.69 providing slightly better RBC accuracy. However, for the larger files, the α_F of 0.69 did not segment the entire image; These images segmented best with a α_F value of 0.72 and a β of 0.1.

File Size	$\boldsymbol{\beta}$	**Decay Value (α_F)**
257 x 257	0.05/0.1	0.69
1712 x 1368	0.1	0.69
2592 x 1944	0.1	0.72/0.73

Table 5 Standard PCNN Segmentation Parameters Impacted by Image Size

The standard PCNN generated a segmentation result for each iteration. A total of 20 iterations was done in the experiments and manually inspected to find the best iteration number. The iteration number was impacted by the M and W neighbor matrices as shown in Table 6. An *M* value of *Cspline* and *W* value of *Weak* provided the overall best segmentation using the other parameters previously mentioned with the segmented image obtained from iteration number 4. However, changing both the *M* and *W* values changed the iteration number for the segmented result image, such as, using *Exponential* for *M* and *Euclidean* for *W* as per Zhou and Shao (2018), which produced the best segmentation at iteration number 2. Using *Exponential* and *Euclidean* values for *M* and *W*, respectively also provided the best WBC accuracy on the *ALL_IDB_subset*, however, the RBC accuracy was reduced.

Weight Matrix (*M*)	**Weight Matrix (*W*)**	**Segmented Result Iteration**
Cspline	Weak	4
Exponential	Euclidean	2

Table 6 Standard PCNN Segmentation Weight Matrix Parameters

The parameter testing on *ALL_IDB_subset* results are shown in Figure 12, which helped to fine tune the parameters for the additional experiments. The parameters for best overall segmentation based on cell type can be seen in Table 7 and the metrics using the RBC based parameters is shown in Figure 13. In Figure 13, there are some overlap of metric values, especially those at 0 and 1. As mentioned earlier the *ALL_IDB_subset* contained a smaller number of cells for trying different PCNN parameters, as such there were limited numbers of WBCs per file in this dataset. Most of the files only contain 1 WBC, but there was one file with two WBCs and one file with no WBC. Figure 14 shows the data grouped by metric instead of each file so the values for each metric are displayed. While average WBC accuracy was above 97%, average RBC accuracy was just over 61%. The parameters for segmenting both RBC and WBC from *ALL_IDB_subset* were found to be like those from Deng, Yan, and Ma (2019) with a slight difference in the α_F value.

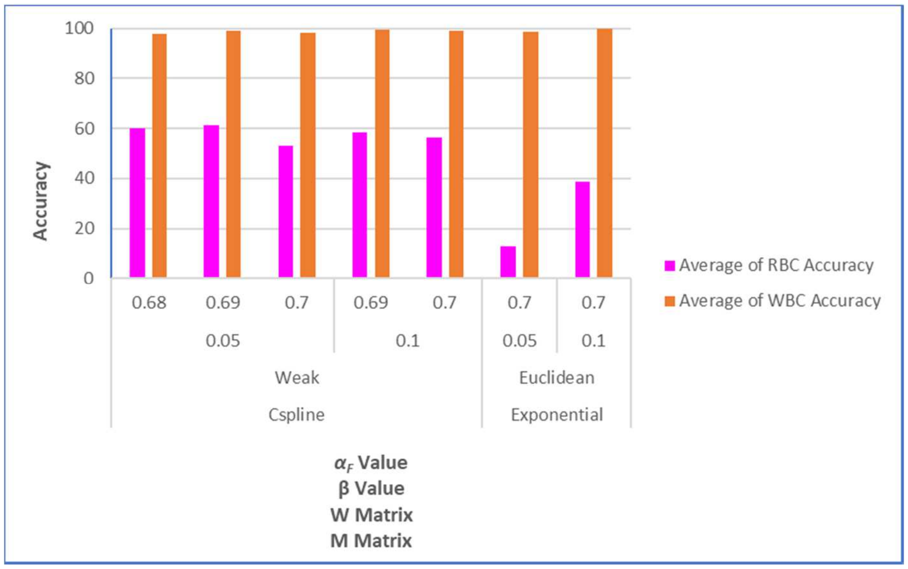


Figure 12 Standard PCNN Accuracy by Parameters (*ALL_IDB_subset*)

Type	**Key PCNN Parameters**			
	α_F	β	M	W
RBC	0.69	0.05	Cspline	Weak
WBC	0.7	0.1	Exponential	Euclidean

Table 7 Standard PCNN Segmentation Key Parameters by Cell Type (*ALL_IDB_subset*)

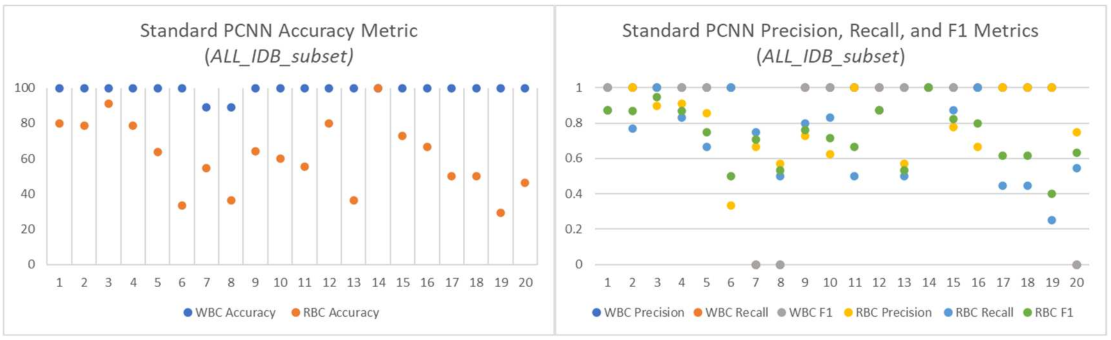


Figure 13 Standard PCNN Metrics (*ALL_IDB_subset*)

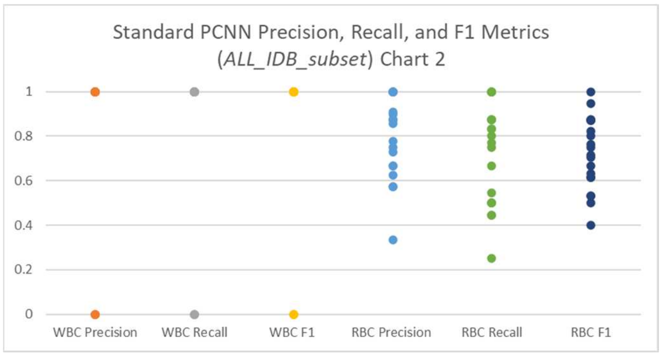


Figure 14 Standard PCNN Metrics (ALL_IDB_subset) Chart 2

The experiments using the *ALL_IDB_large* required some fine tuning of parameters, as mentioned earlier certain images with a higher resolution had an impact on segmentation. One of these experiments was to split the large images into four images and process each subimage. The idea was to provide an optimal α_F parameter for all files. A concern with this was the impact on splitting cells between subimages leaving them on the image border, which is discussed later. While the splitting did not necessarily provide an optimal α_F, it provided a slight increase in accuracy and had the advantage of breaking up some clumped WBCs. However, as was a concern it had a slight disadvantage of missing a WBC that got positioned on a border of a split image. The increase in accuracy suggests that the advantage out weighted the disadvantage; and the concern over cells being located on a border due to the split. The parameters that changed for the *ALL_IDB_large* are shown in Table 8.

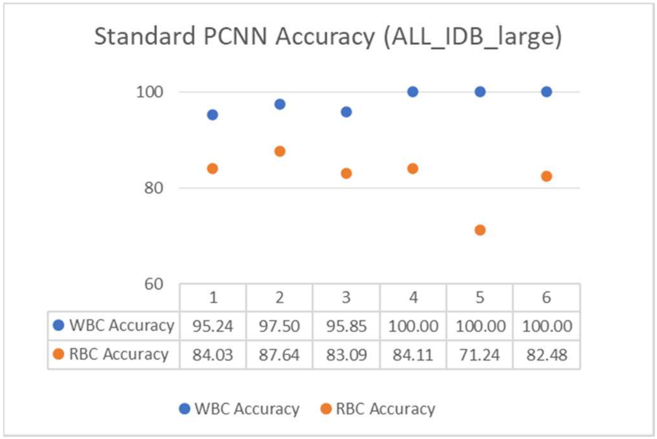
Size	**Key PCNN Parameters**	
	α_F	β
<= 1712 x 1368	0.69	0.1
>= 2592 x 1944	0.72	0.1

Table 8 Standard PCNN Segmentation Key Parameters by Image Size (*ALL_IDB_large*)

The experiments with splitting the large images resulted in a slight increase in average accuracy as shown in Table 9. The accuracy results on *ALL_IDB_large* subset using the split method and parameter values previously mentioned are shown in Figure 15. The WBC average accuracy was 98% and RBC average accuracy was 82%, where RBC accuracy was higher than with *ALL_IDB_subset*.

Split	Type	Average Metrics			
		Accuracy	*Precision*	*Recall*	*F1*
No	RBC	81.27	0.98	0.83	0.89
	WBC	97.96	0.78	0.88	0.81
Yes	RBC	82.10	0.93	0.88	0.90
	WBC	98.10	0.79	0.95	0.85

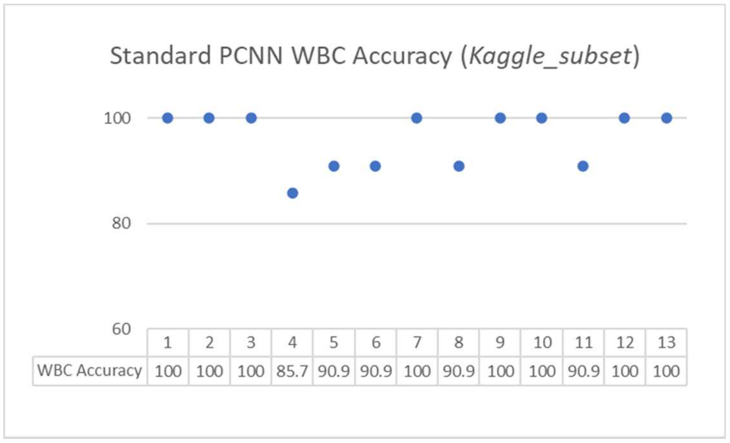
Table 9 Standard PCNN Metrics (*ALL_IDB_large*)



	1	2	3	4	5	6
● WBC Accuracy	95.24	97.50	95.85	100.00	100.00	100.00
● RBC Accuracy	84.03	87.64	83.09	84.11	71.24	82.48

Figure 15 Standard PCNN Accuracy Metric (*ALL_IDB_large*)

Figure 16 shows the results of the WBC counts on the *Kaggle_subset* using the parameter settings that provided the overall general segmentation from *ALL_IDB_subset*. The WBC accuracy was slightly lower on this data.



	1	2	3	4	5	6	7	8	9	10	11	12	13
WBC Accuracy	100	100	100	85.7	90.9	90.9	100	90.9	100	100	90.9	100	100

Figure 16 Standard PCNN WBC Accuracy Metric (*Kaggle_subset*)

This section provides the answer to RQ1 from Chapter 1, "*What are the significant PCNN parameters impacting PCNN segmentation?"*. The key parameters for the standard PCNN are α_F, β, W, and M, with certain images having an impact on the α_F parameter, which are shown in Table 10. An α_F =0.69 worked well for the smaller images and an α_F =0.72 worked better for the larger images. A β =0.05 or β =0.1 worked well based on the α_F value and had a slight impact on RBC accuracy on the smaller images. The weight matrices M=Cspline and W=Weak provided overall best segmentation for both RBC and WBC using iteration number 4, but M=Exponential, and W=Euclidean provided the best WBC segmentation using iteration number 2. Other parameters, while not providing an optimal segmentation for WBC and RBC counts, did have an impact on the texture vector generation used for classification, described later in that section. The average metric values for each subset based on the parameters used are shown in Table 11.

Name	Type	Key PCNN Parameters			
		α_F	β	*M*	*W*
ALL_IDB_subset	RBC	0.69	0.05	Cspline	Weak
	WBC	0.7	0.1	Exponential	Euclidean
ALL_IDB_large	RBC	0.69 and 0.72	0.1	Cspline	Weak
	WBC	0.72	0.1	Exponential	Euclidean
Kaggle_subset	WBC	0.69	0.05	Cspline	Weak

Table 10 Standard PCNN Segmentation Key Parameters by Cell Type

Name	Type	Average Metrics			
		Accuracy	*Precision*	*Recall*	*F1*
ALL_IDB_subset	RBC	61.35	0.81	0.72	0.72
	WBC	100	0.95	0.95	0.95
ALL_IDB_large	RBC	82.10	0.93	0.88	0.90
	WBC	98.10	0.79	0.95	0.85
Kaggle_subset	WBC	96.10	0.71	0.85	0.75

Table 11 Standard PCNN Segmentation Average Metric Results by Cell Type

Separate Cell Objects

The *separate* method was used with acceptance parameters *wbc_offset*=17, *wbc_percent*=0.75, *wbc_intensity*=0.6, *rbc_intensity*=0.4, *rbc_percent*=0.6, and *rbc_offset*=7 on the ALL_IDB dataset. The Kaggle dataset required a slight modification due to the staining differences and was set to *wbc_offset*=11, *wbc_percent*=0.17, *wbc_intensity*=0.26, and RBC values were not applicable. From the experiments, the standard PCNN segmentation did provide edges for CHT using *hough_circle*() and *hough_circle_peaks()* to find the circular objects and for the separation algorithm to separate the cells; thus, answering RQ3 from Chapter 1, "*Does PCNN segmentation with postprocessing identify edges for CHT to find and differentiate between WBC and RBC objects?"*. Removing found object circles from the working image did improve discovery of clumped cells. Distinguishing WBCs from RBCs was improved by using the matching acceptance parameters, but depending on the staining of the slides, the results varied. Although the separate function did find some overlapped cells, it did not do as well when there were larger clumps, as there were no edges in the large clump to separate. However, if the split segmentation method was used on clumped WBCs in the middle of the image, then some of the WBCs were split into the sub images and the separation method could find them. Figure 17 below shows WBCs from *Im001_1* on the left and the improvement using the split method on the right for some clumped WBCs.

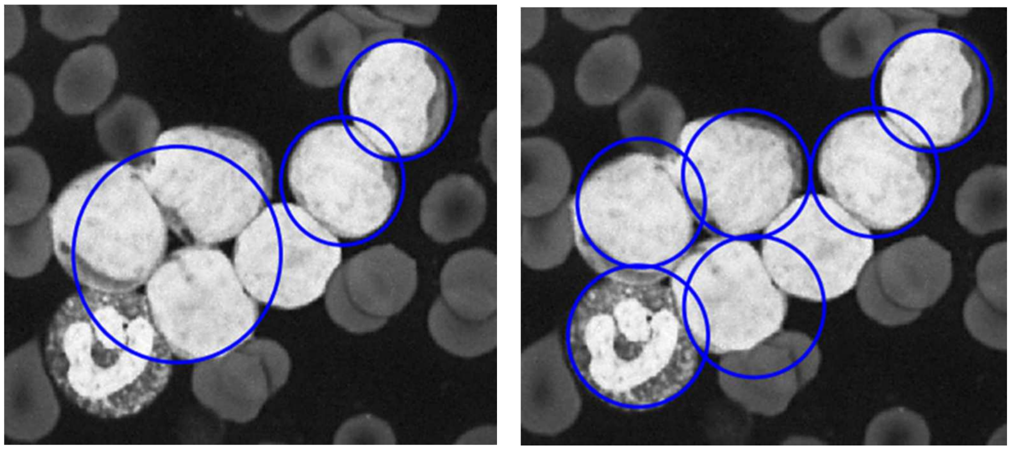


Figure 17 Clumped WBCs Improved with Split Segmentation

The separation method first identifies and removes WBC objects from the image, after which it identifies RBC objects. This approach impacted RBC detection since a missed WBC object could be detected as a false positive RBC object. This did not account for all RBC false positives, but it did account for some. For example, Figure 18 shows the results for Im123_1 and Im135_0 from *ALL_IDB_subset* using the parameters in Table 10 for RBCs to capture WBCs and RBCs. As shown in Figure 18, the WBCs were not found and so there were 3 false positives in each due to the missed WBC, with a false positive in Im135_0 that was not related to the missed WBC.

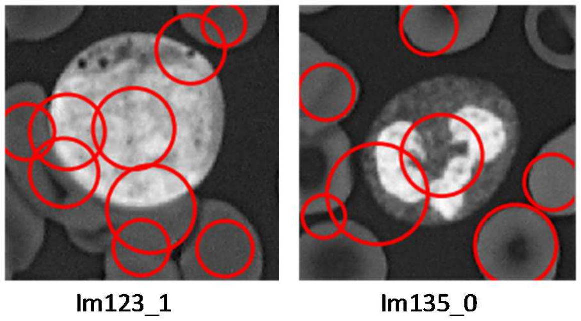


Figure 18 Missed WBCs

Region Growing PCNN

The Region growing PCNN segmentation method starts with a seed pixel and grows outward based on the PCNN until the stopping criteria is met for that region, then continues to the next region with the next start pixel until all pixels are assigned to a region. The experiments for Region Growing PCNN were split into two main areas, intensity and color which are described in this section.

Key Parameters

The β value is updated as part of the algorithm, so this parameter did not have the impact as with the standard PCNN. A starting value of $\beta=0.1$ was used. The M matrix value of Cspline was used as other values either did not work well or had no significant impact, although not all values were tried. The W matrix parameter did have an impact on the segmentation results which was different for intensity or color region growing and are described in those sections. The images that were impacted by the α_F value for standard PCNN were not impacted in region growing, so a value of 0.69 was used on all datasets. Table 12 shows the key parameters for region growing PCNN.

Key Parameters	
α_F	M
0.69	Cspline

Table 12 Region Growing PCNN Segmentation Key Parameters

Separation

For region growing, the *find_num* parameter of the *separate* method was set to a smaller number (compared to the standard PCNN) of circle peaks to retrieve with CHT, since a region should not contain the same number of circles as the entire image using standard PCNN. A value of 100 was determined by experiments and checking the segmented images of regions to determine the maximum number of circles in each. However, this value was different for the cell option described in that section.

Intensity

The parameters for region growing PCNN started with those previously described that were fixed and those from the standard PCNN. Experiments using *ALL_IDB_subset* were performed for determining parameters specific to PCNN region growing using the intensity of the grey scale image pixels (preprocessing converted the color image to grey scale) with the average accuracy results shown in Figure 19. As can be seen from Figure 19, the *W* value had an impact on segmentation results. A *W* value of Euclidean increased WBC accuracy but decreased RBC accuracy. A *W* value of Neighbors provided the overall best WBC and RBC accuracy, however, while average WBC accuracy was above 90%, average RBC accuracy was under 30%. The RBC accuracy was increased slightly using a smaller value for *separate* parameter *rbc_percent*. The intensity region growing specific key parameters by type on *ALL_IDB_subset* is shown in Table 13 and using the parameters for the overall best segmentation the metrics are shown in Figure

20. In Figure 20, there are some overlap of metric values due to the dataset as previously mentioned, which are shown grouped by metric in Figure 21.

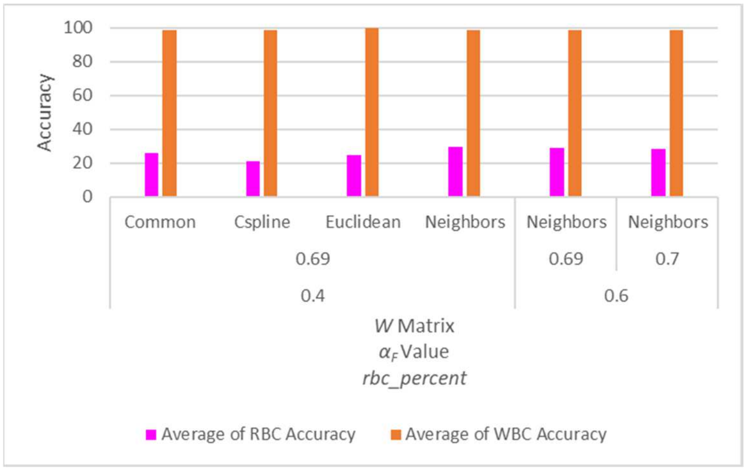


Figure 19 Region Growing Intensity PCNN Accuracy by Parameters (*ALL_IDB_subset*)

Type	**Key PCNN Parameters**	***Separate* Parameters**
	W	*rbc_percent*
RBC	Neighbor	0.4
WBC	Euclidean	N/A

Table 13 Region Growing Intensity PCNN Segmentation Key Parameters by Cell Type (*ALL_IDB_subset*)

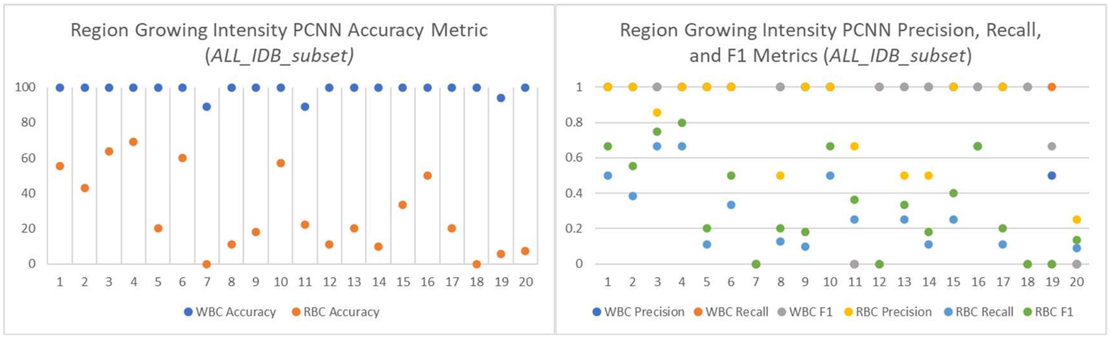


Figure 20 Region Growing Intensity PCNN (*ALL_IDB_subset*) Metrics

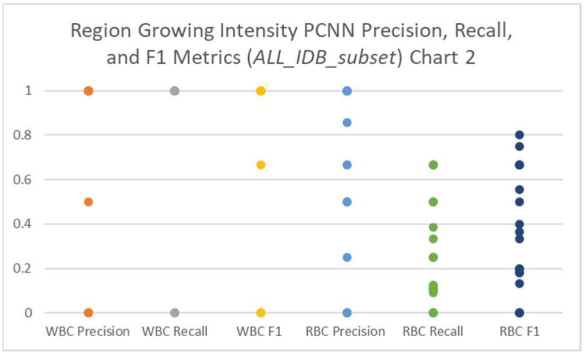
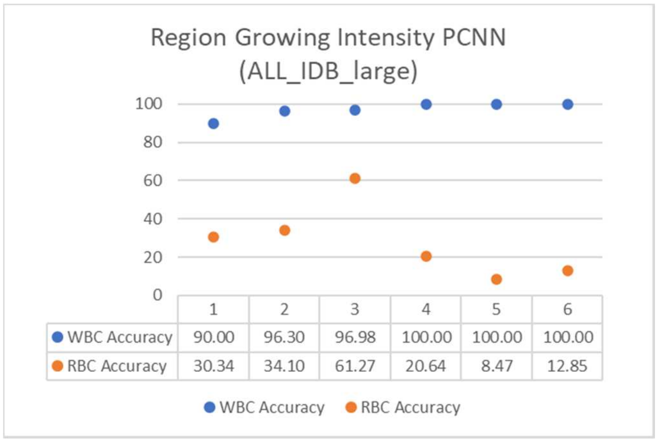


Figure 21 Region Growing Intensity PCNN (ALL_IDB_subset) Metrics Chart 2

The split option previously mentioned was also used for region growing intensity on ALL_IDB_large and the accuracy metrics are shown in Figure 22. While the WBC accuracy was above 90% for the worse case, RBC accurcy was below 60% for the best case.



	1	2	3	4	5	6
● WBC Accuracy	90.00	96.30	96.98	100.00	100.00	100.00
● RBC Accuracy	30.34	34.10	61.27	20.64	8.47	12.85

Figure 22 Region Growing Intensity PCNN WBC Accuracy (*ALL_IDB_large*)

The results for WBC segmentation for *Kaggle_subset* are shown in Figure 23 using the previously mentioned parameters that provided the best overall segmentation. However, the WBC accuracy is not as high on the Kaggle dataset and drops below 60% on some files.

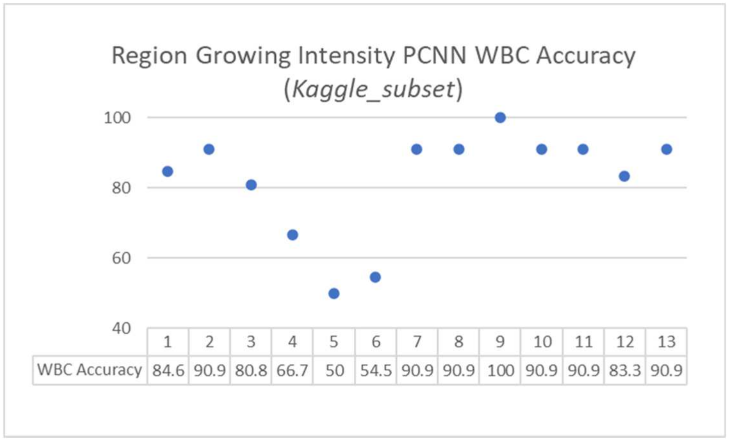


Figure 23 Region Growing Intensity PCNN WBC Accuracy (*Kaggle_subset*)

The key parameters and average metrics for region growing intensity PCNN are shown in Table 14 and Table 15.

Name	**Type**	**Key Parameters**		
		α_F	*M*	*W*
ALL_IDB_subset	RBC	0.69	Cspline	Neighbors
	WBC	0.69	Cspline	Euclidean
ALL_IDB_large	RBC	0.69	Cspline	Neighbors
	WBC	0.69	Cspline	Neighbors
Kaggle_subset	WBC	0.69	Cspline	Euclidean

Table 14 Region Growing Intensity Key Parameters

Name	**Type**	**Metrics**			
		Accuracy	*Precision*	*Recall*	*F1*
ALL_IDB_subset	RBC	29.52	0.63	0.26	0.34
	WBC	99.71	0.93	0.93	0.93
ALL_IDB_large	RBC	27.94	0.64	0.32	0.39
	WBC	97.21	0.82	0.98	0.89
Kaggle_subset	WBC	81.95	0.23	0.69	0.31

Table 15 Region Growing Intensity Average Metrics

Color

The experiments on *ALL_IDB_subset* for Color region growing PCNN showed that the *W* value had an impact on the segmentation results as can be seen in Figure 24. A *W* value of Cspline increased WBC accuracy but decreased RBC accuracy. A *W* value of Common provided the overall best WBC and RBC accuracy, however, while WBC accuracy was above 90%, RBC accuracy was under 40%. For the color space parameter, HSV had slightly higher accuracy for all *W* values except Cspline which had no difference between color spaces, which are also shown in Figure 24. This provides the answer to RQ2 from Chapter 1, "*What color channels and image processing methods improve the results of PCNN segmentation and separation*?*"*. The Color Region Growing worked best using a *separate* parameter of *rbc_percent=0.6*. From the experiments on *ALL_IDB_subset*, the color space did provide a 5% increase in RBC accuracy from the region growing with intensity, whether this is truly related to color or the algorithm differences is undetermined. The Color region growing specific key parameters by type on *ALL_IDB_subset* is shown in Table 16 and using the parameters for the overall best segmentation the metrics are shown in Figure 25. In Figure 25, there are some overlap of metric values due to the dataset as previously mentioned, which are shown grouped by metric in Figure 26.

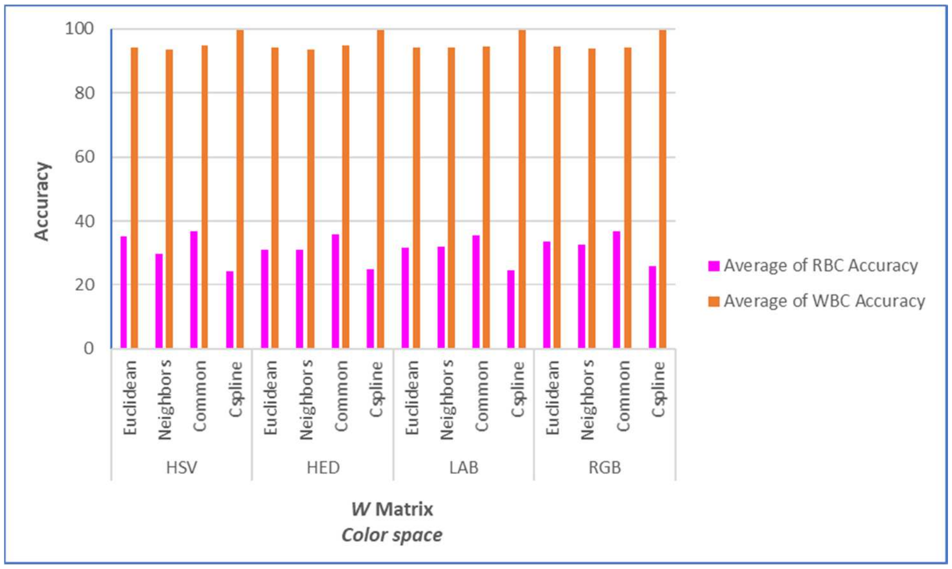


Figure 24 Color Region Growing PCNN Accuracy by Parameters (*ALL_IDB_subset*)

<table>
<tr><th rowspan="2">Type</th><th colspan="2">Key PCNN Parameters</th><th><i>Separate</i> Parameters</th></tr>
<tr><th><i>W</i></th><th><i>Color space</i></th><th><i>rbc_percent</i></th></tr>
<tr><td>RBC</td><td>Common</td><td>HSV</td><td>0.6</td></tr>
<tr><td>WBC</td><td>Cspline</td><td>N/A</td><td>N/A</td></tr>
</table>

Table 16 Color Region Growing PCNN Segmentation Key Parameters by Cell Type (*ALL_IDB_subset*)

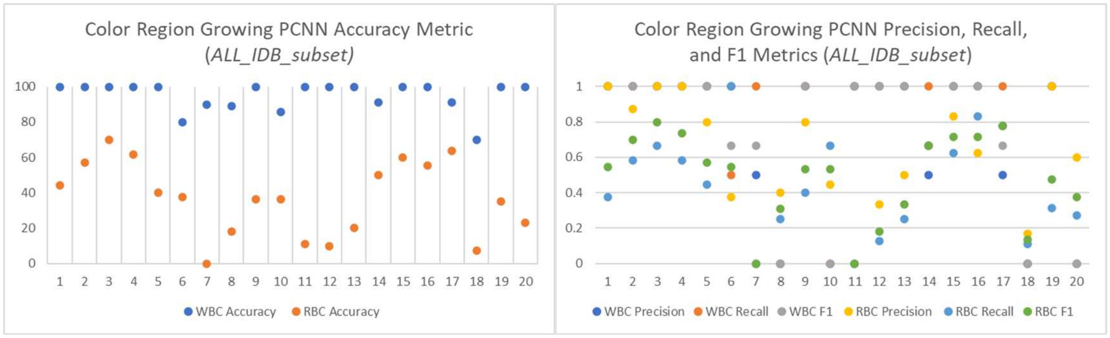


Figure 25 Color Region Growing PCNN Metrics (*ALL_IDB_subset*)

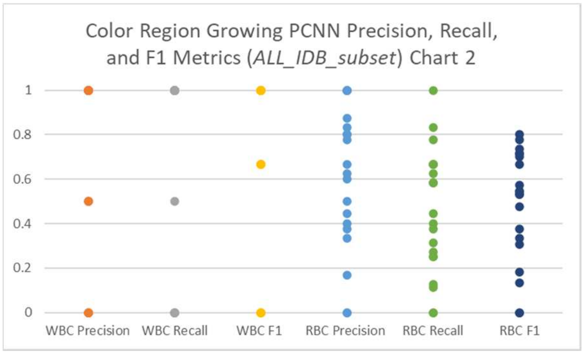
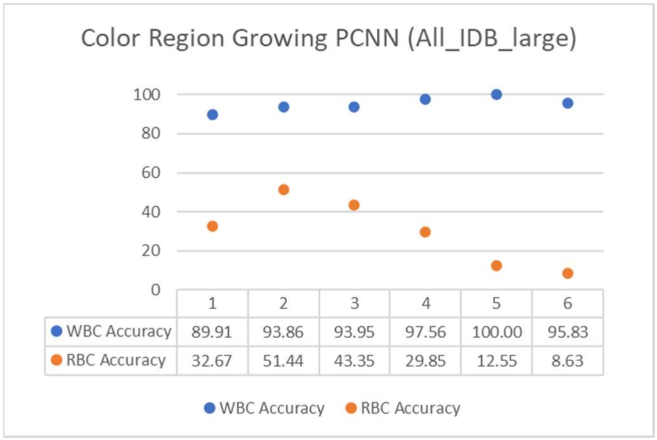


Figure 26 Color Region Growing PCNN Metrics (ALL_IDB_subset) Chart 2

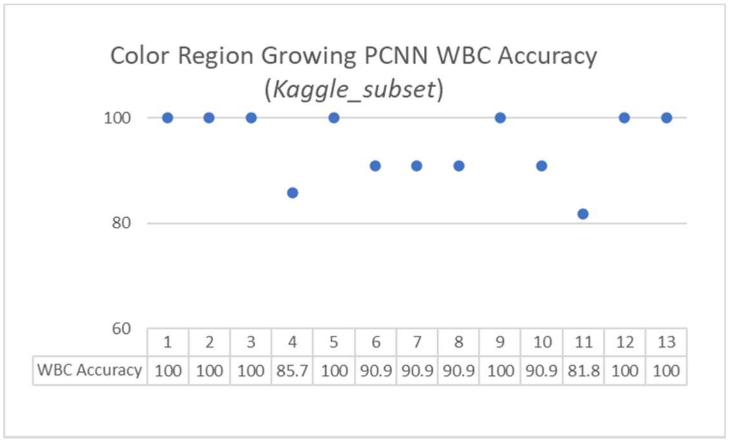
The split option previously mentioned was also used for color region growing on *ALL_IDB_large* and the accuracy metrics are shown in Figure 27. While the WBC accuracy was almost 90% for the worse case, RBC accuracy was just over 50% for the best case.



	1	2	3	4	5	6
● WBC Accuracy	89.91	93.86	93.95	97.56	100.00	95.83
● RBC Accuracy	32.67	51.44	43.35	29.85	12.55	8.63

Figure 27 Color Region Growing PCNN Accuracy Metric (*ALL_IDB_large*)

The results for WBC segmentation for *Kaggle_subset* are shown in Figure 28 using the previously mentioned parameters that provided the best overall segmentation. The WBC accuracy on *Kaggle_subset* is higher for region growing using color compared to intensity.



	1	2	3	4	5	6	7	8	9	10	11	12	13
WBC Accuracy	100	100	100	85.7	100	90.9	90.9	90.9	100	90.9	81.8	100	100

Figure 28 Color Region Growing PCNN WBC Accuracy (*Kaggle_subset*)

The average metrics per dataset for color region growing PCNN are shown in Table 17.

Name	**Type**	**Metrics**			
		Accuracy	*Precision*	*Recall*	*F1*
ALL_IDB_subset	RBC	36.87	0.61	0.45	0.48
	WBC	99.71	0.93	0.95	0.93
ALL_IDB_large	RBC	29.75	0.53	0.43	0.42
	WBC	95.19	0.70	0.65	0.63
Kaggle_subset	WBC	94.71	0.60	0.69	0.62

Table 17 Color Region Growing PCNN Average Metrics

Cell

The last of the PCNN methods is the Cell variant of Color region growing, where additional parameters were added to the color region growing to limit the size of the region. This reduced the size of each region and increased the number of segmented images processed by *separate*. The purpose was to try and capture each cell object into a region. The parameters were set for choosing regions based on the size and intensity of WBCs and RBCs. The *find_num* parameter for *separate* was set to 1, although other values did not improve the results. This method increased the RBC accuracy from the Color region growing on *ALL_IDB_subset*, with a slight increase in WBC accuracy. The accuracy results based on experimented parameters are shown in Figure 29. The best parameters were the same as for color region growing. The metrics are shown in Figure 30 for *ALL_IDB_subset* using these parameters. In Figure 30, there are some overlap of metric values as previously mentioned related to the dataset, Figure 31 shows the overall per metric values.

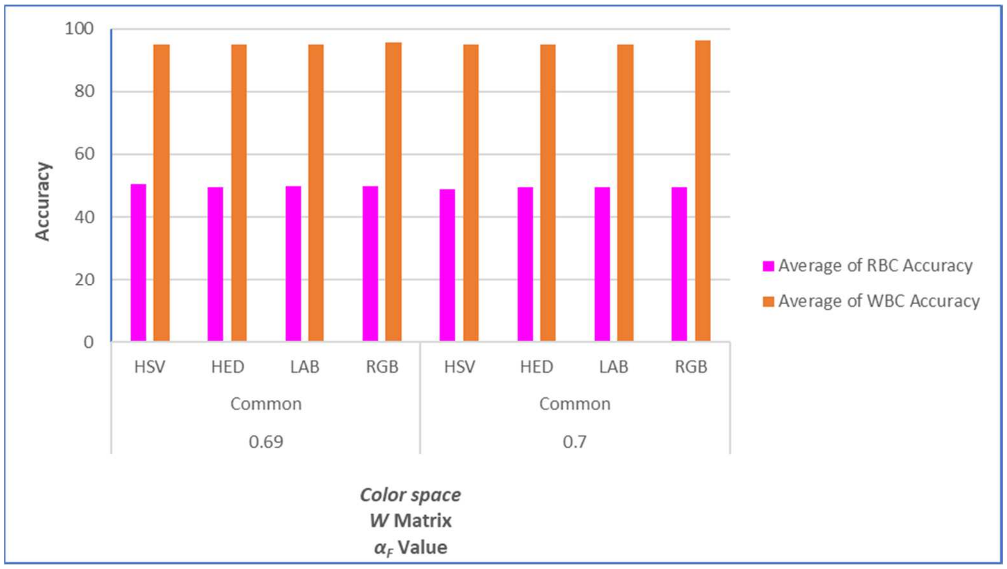


Figure 29 Color Region Growing Cell PCNN Accuracy by Parameters (*ALL_IDB_subset*)

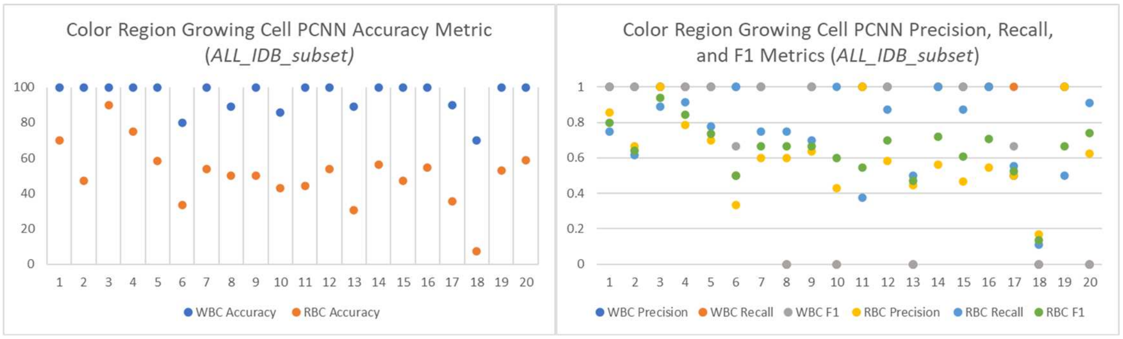


Figure 30 Color Region Growing Cell PCNN Metrics (*ALL_IDB_subset*)

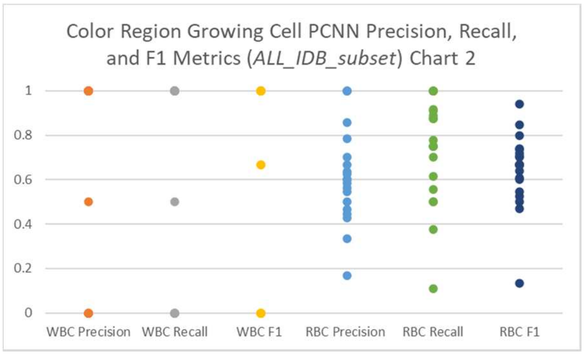
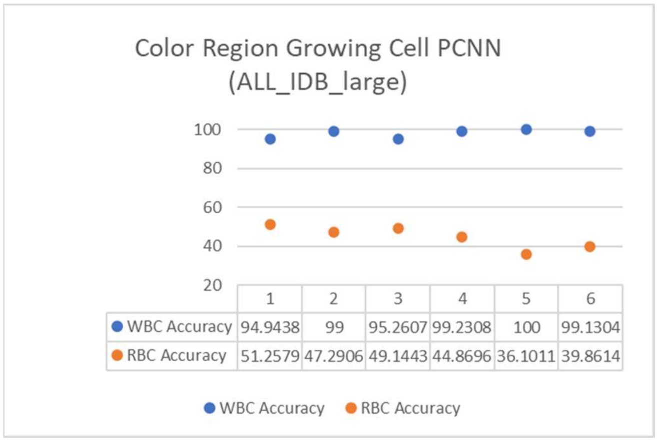


Figure 31 Color Region Growing Cell PCNN Metrics (ALL_IDB_subset) Chart 2

However, this method like the other region growing PCNN did not work well on RBC counts for the larger files, the metrics for the color region growing cell variant on

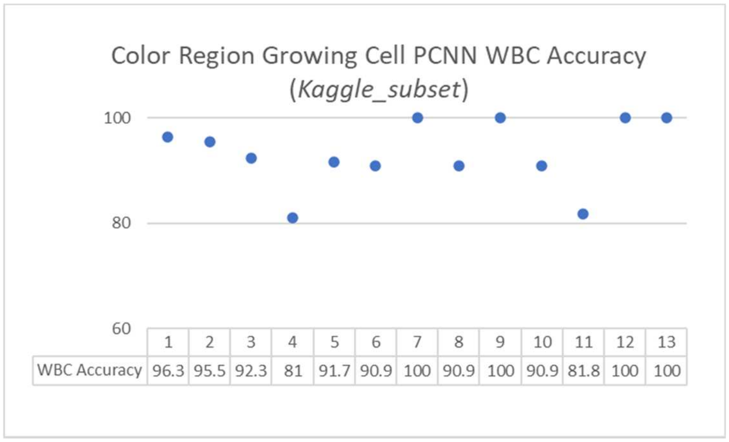
ALL_IDB_large using the parameters that proved the best overall segmentation are shown in Figure 32.



	1	2	3	4	5	6
● WBC Accuracy	94.9438	99	95.2607	99.2308	100	99.1304
● RBC Accuracy	51.2579	47.2906	49.1443	44.8696	36.1011	39.8614

Figure 32 Color Region Growing Cell PCNN WBC Accuracy (*ALL_IDB_large*)

The cell variant for color region growing on *Kaggle_subset* is shown in Figure 33, which showed a slight decrease in the WBC accuracy from the non-variant color region growing.



	1	2	3	4	5	6	7	8	9	10	11	12	13
WBC Accuracy	96.3	95.5	92.3	81	91.7	90.9	100	90.9	100	90.9	81.8	100	100

Figure 33 Color Region Growing Cell PCNN WBC Accuracy (*Kaggle_subset*)

The average metrics for color region growing cell PCNN are shown in Table 18.

Name	**Type**	**Metrics**			
		Accuracy	*Precision*	*Recall*	*F1*
ALL_IDB_subset	RBC	50.6	0.63	0.74	0.64
	WBC	96.39	0.80	0.78	0.78
ALL_IDB_large	RBC	44.75	0.59	0.67	0.61
	WBC	97.93	0.86	0.67	0.72
Kaggle_subset	WBC	93.17	0.50	0.77	0.58

Table 18 Color Region Growing Cell PCNN Average Metrics

Conventional Segmentation and PCNN Prior Work

For comparison results the *ALL_IDB_subset* was used on a couple conventional methods and the results are described in this section. These methods were not run for the other datasets.

Threshold

The threshold method uses a threshold value and every pixel less than or equal to that value is either in or out of the result. A threshold value of 0.5 provided the best segmentation for overall RBC and WBC segmentation and separation using *separate* method. The *rbc_percent* parameter for *separate* improved RBC counts using a smaller value of 0.4. The chart in Table 19 shows the metric values for the *ALL_IDB_subset.*

Name	**Type**	**Metrics**			
		Accuracy	*Precision*	*Recall*	*F1*
ALL_IDB_subset	RBC	32.78	0.50	0.40	0.42
	WBC	95.84	0.77	0.90	0.81

Table 19 Threshold Metrics (ALL_IDB_subset)

Watershed Region Growing

A watershed region growing method was available in scikit-image with an example for segmenting coins that was the basis for this implementation. An elevation map was created using a sobel filter on the image. Markers are determined by the threshold values. There are three 'basins' separated by threshold values of 0.4 and 0.51. These threshold values were set from experiments. The scikit-image *watershed* method was called with the elevation map and makers to segment. The three basins were chosen to segment and capture both types of cells. As with the threshold method, Watershed also had better accuracy with an *rbc_percent* parameter for *separate* set to 0.4, the metrics for *ALL_IDB_subset* are shown in Table 20.

Name	**Type**	**Metrics**			
		Accuracy	*Precision*	*Recall*	*F1*
ALL_IDB_subset	RBC	30.88	0.36	0.36	0.38
	WBC	96.67	0.83	0.90	0.85

Table 20 Watershed Metrics (*ALL_IDB_subset*)

PCNN from Literature

A standard PCNN with the parameters from by Ma et al. (2016) was used to segment the *ALL_IDB_subset* for comparison results. The PCNN parameters are shown in Table 21, and *separate* used an *rbc_percent* value of 0.4. The metrics are shown in Table 22, however as discussed later in the comparison section, their results used a computed count percentage. Their method also contained differences in preprocessing, post processing, and separation and counting.

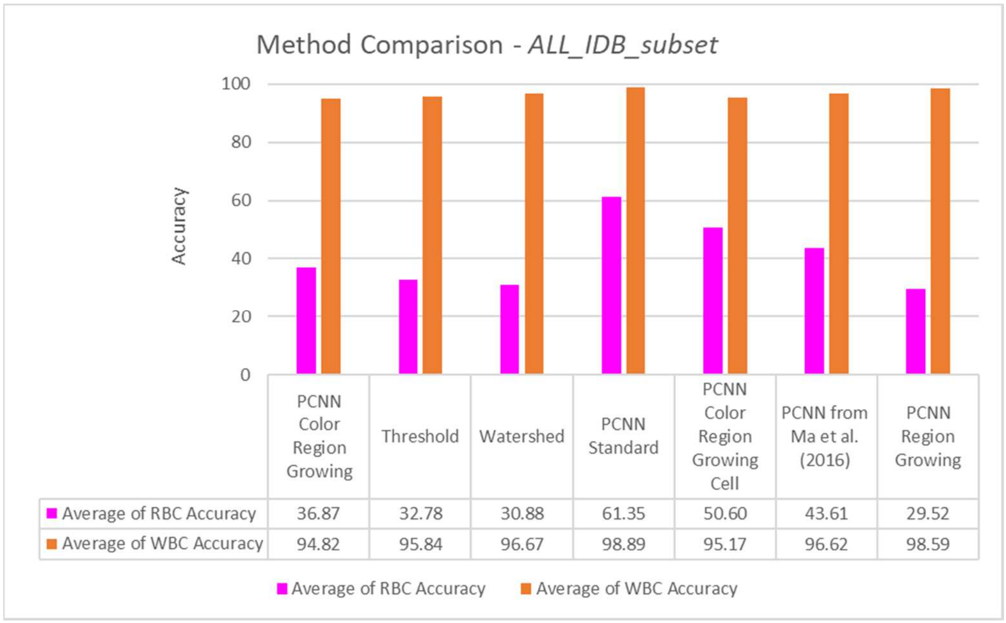
Name	**Parameters**								
	α_F	α_L	α_E	β	V_F	V_L	V_E	*M*	*W*
ALL_IDB_subset	0.8	1	1.35	0.1	0.2	0.2	2000	Cspline	Cspline

Table 21 PCNN Parameters from Literature

Name	**Type**	**Metrics**			
		Accuracy	*Precision*	*Recall*	*F1*
ALL_IDB_subset	RBC	43.62	0.53	0.65	0.56
	WBC	96.62	0.78	0.78	0.77

Table 22 PCNN with Parameters from Literature Metrics

Figure 34 shows the average accuracy metric comparison with the different segmentation methods and *separate* on the *ALL_IDB_subset* using the best overall segmentation results for each method.



	PCNN Color Region Growing	Threshold	Watershed	PCNN Standard	PCNN Color Region Growing Cell	PCNN from Ma et al. (2016)	PCNN Region Growing
🟪 Average of RBC Accuracy	36.87	32.78	30.88	61.35	50.60	43.61	29.52
🟧 Average of WBC Accuracy	94.82	95.84	96.67	98.89	95.17	96.62	98.59

Figure 34 Method Comparison Accuracy Methods

Classification

The classification was done for WBCs on both the ALL_IDB and Kaggle datasets described in Table 2. Classification required creation of the texture vector list using the labeled training data for each dataset, *ALL_IDB2* and *Kaggle_Train*. The *generate_texture_vector()* was described in Chapter 3. Several segmentation methods were used for segmentation with the separate method. The goal was to capture close to the entire WBC with limited background or RBCs contained in the WBC subimage created. The WBC segmentation method that achieved this goal best was the standard PCNN with the parameters shown in Table 23. The significant difference between datasets was that they required different α_E PCNN parameter values. The WBC objects are found as before using the *separate* method to capture the list of WBC images. The dataset specific parameters for *separate* are shown in Table 24.

Once the WBC objects were found, another standard PCNN was used with a different set of PCNN parameters also in Table 23 to capture the texture vectors for classification. The significant difference between datasets was that they required different α_E and V_E PCNN parameter values. These parameters were found through experiments where the output images produced were manually examined. The goal was to capture the texture of the image in a series and not the entire object in one segmentation. This was done by experimenting with different types of WBCs to capture the nucleus, granules, etc., in different indexes of the segmentation array. The Shannon entropy was calculated on each segmentation in the array and represented one element in the texture vector for an object, thus it captured differences between the types. This section provides the answer to RQ4 from Chapter 1, "*What are the significant PCNN parameters that yield the best texture vector results for each dataset, or which worked generally across datasets?*".

Usage	**Dataset**	**Parameters**									
		α_F	α_L	α_E	β	V_F	V_L	V_E	M	W	Initial E
Segmentation	*ALL_IDB*	0.72	1	0.0771	0.1	0.2	0.2	0.9740	Exponential	Euclidean	0.0001
	Kaggle			0.3							
Classification	*ALL_IDB*	1.2	0.6	0.08	0.01	0.1	0.1	1.1	Cspline	Common	1
	Kaggle			0.4				0.8			

Table 23 Texture Vector PCNN Parameters

Dataset	**Parameters**		
	wbc_offset	*wbc_percent*	*wbc_intensity*
ALL_IDB	15	0.6	0.5
Kaggle	11	0.17	0.26

Table 24 Texture Vector *Separate* Parameters

Once the parameters for generating the texture vector were found, the next experiments created and stored the list of texture vectors for each WBC type. The original expectation was to capture and save the average texture vector per type, but there was too much variance in the vectors, so for the *ALL_IDB2* training data it was decided to save the texture vectors and cell type for each image as was done by Yang et al (2017). However, for the *Kaggle_Train* there was a larger set, so the mean along with the mean plus and minus the variance were saved on every 10 images. The value of 10 was chosen by experiments that reduced the number of vectors stored and still captured the difference between types.

The last step was to perform the segmentation and classification on the test data and capture the results. The test data images were segmented and separated to retrieve the WBC objects. Figure 35 shows the segmented image and resulting WBC objects found for Im004_1, where all WBCs are found along with some false positives. The false positives here are most likely WBCs that were either old or smeared and broken during making the slide. Three of the lymphoblasts from this segmentation are shown in Figure 36.

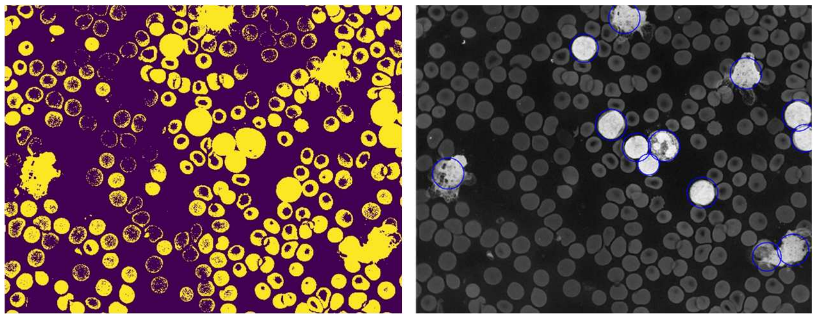


Figure 35 WBC Segmentation and Result (Im004_1)

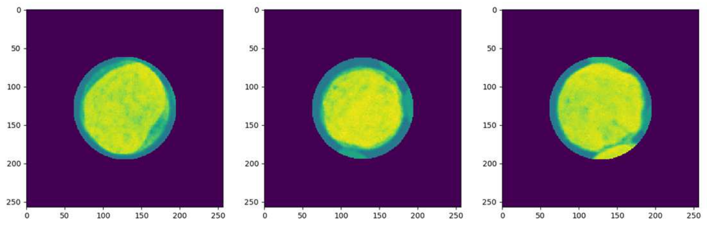


Figure 36 Lymphoblasts (Im004_1)

Once the WBC object was found, the texture vector was captured using the same parameters as was used for generating the texture vector list. The texture vector for the WBC object was then compared against the training texture vector list to find the closest similar vector as previously described in *classify()* in the Chapter 3. Figure 37 shows the one lymphocyte and two neutrophils from Im108_0 which were correctly classified, however, Figure 38 shows a neutrophil and two lymphocytes that were correctly classified and one monocyte that was incorrectly classified. The incorrectly classified WBC was not entirely captured from *separate*, as is evident from the image.

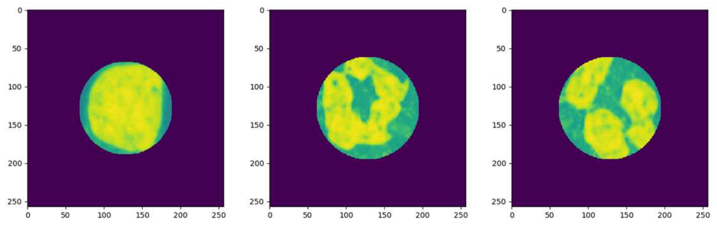


Figure 37 Lymphocyte, Neutrophils (Im108_0)

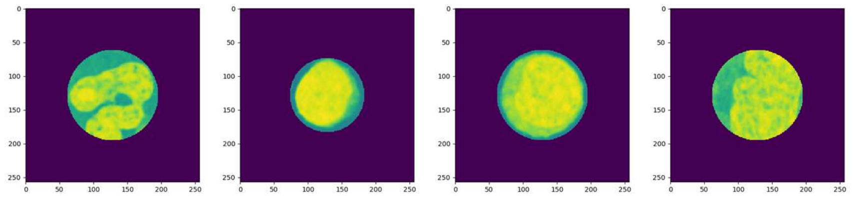


Figure 38 Neutrophil, Lymphocyte, Lymphocyte, and Misclassified Monocyte (Im088_0)

The confusion matrices and metrics are shown in this section for the different testing, along with the metrics for the combined results. Any false positive WBCs were not counted in the classification results as those results were already included with the metrics calculated for total WBC counts under segmentation.

ALL_IDB Dataset

As was described earlier in Table 2, the classification was done for WBCs on the testing data in *ALL_IDB_subset*, *ALL_IDB_large*, *ALL_IDB_large2*, and *ALL_IDB_large3*. The *ALL_IDB2* contained the training data and was used to label the WBCs and capture the texture vector list. The test images were segmented using the previously mentioned parameters with a slight adjustment in the *separate* parameters as shown in Table 25 and classified using the parameters mentioned previously to capture the texture vector.

Adjusted *Separate* Parameters	
wbc_offset	*wbc_intensity*
17	0.6

Table 25 Adjusted Classification Parameters (*ALL_IDB*)

The confusion matrix for the test data from *ALL_IDB_large* and *ALL_IDB_large2* are shown in Table 26.

		Predicted					
		Neutrophil	Lymphocyte	Lymphoblast	Monocyte	Eosinophil	Basophil
Actual	Neutrophil	11	2	3	0	0	0
	Lymphocyte	0	13	0	0	0	0
	Lymphoblast	1	5	93	3	0	0
	Monocyte	0	1	2	3	0	0
	Eosinophil	0	0	1	0	0	0
	Basophil	0	0	0	0	0	0

Table 26 Confusion Matrix (*ALL_IDB_large* and *ALL_IDB_large2*)

The *ALL_IDB_subset* which contain images from the training set was used as test data to determine the impact of slight changes to the *separate* parameters. The confusion matrix is shown in Table 27, which shows the *separate* parameters had a slight impact as the captured WBC was slightly different resulting in two WBCs being incorrectly classified.

		Predicted					
		Neutrophil	Lymphocyte	Lymphoblast	Monocyte	Eosinophil	Basophil
Actual	Neutrophil	2	1	0	0	0	0
	Lymphocyte	0	5	0	0	0	0
	Lymphoblast	0	0	8	0	0	0
	Monocyte	0	1	0	2	0	0
	Eosinophil	0	0	0	0	1	0
	Basophil	0	0	0	0	0	1

Table 27 Confusion Matrix (*ALL_IDB_subset*)

The confusion matrix for all test data which includes *ALL_IDB_large*, *ALL_IDB_large2*, and *ALL_IDB_large3* from ALL_IDB is shown in Table 28.

		Predicted					
		Neutrophil	Lymphocyte	Lymphoblast	Monocyte	Eosinophil	Basophil
Actual	Neutrophil	12	2	3	0	0	0
	Lymphocyte	0	26	0	0	0	0
	Lymphoblast	1	5	93	3	0	0
	Monocyte	0	1	2	3	0	0
	Eosinophil	0	0	1	0	0	0
	Basophil	0	0	0	0	0	1

Table 28 Confusion Matrix (*ALL_IDB Test Data)*

The metric table was calculated from the confusion matrix for the test data in *ALL_IDB_large*, *ALL_IDB_large2*, and *ALL_IDB_large3* is shown in Table 29. From the results the classification for lymphocytes was very good at 100%. Lymphoblast classification was 88.2 % accuracy with sensitivity of 91.2 %. The worse accuracy for neutrophils and monocytes was above 93%, but the best sensitivity was below 70% which may be related to staining variances resulting in segmentation and separation not capturing the entire cell. The limited number of basophils and eosinophils make those results not accurate.

Classification	**Sensitivity (%)**	**Specificity (%)**	**Accuracy (%)**
Neutrophil	70.59	96.06	93.06
Lymphocyte	100.00	100.00	100.00
Lymphoblast	91.18	82.00	88.16
Monocyte	50.00	97.76	95.71
Eosinophil	0.00	99.26	98.53
Basophil	0.00	100.00	100.00

Table 29 WBC Classification Metrics (*ALL_IDB Test Data*)

Kaggle Dataset

As was described earlier in Table 2, the classification was done for WBCs on the testing data in *Kaggle_Test_simple* and *Kaggle_Test.* The *Kaggle_Train* contained the training data and was used to capture the texture vector list. The test images were segmented using the previously mentioned parameters and classified using the parameters mentioned previously to capture the texture vector. The confusion matrix for the test data from *Kaggle_Test_simple* and *Kaggle_Test* are shown in Table 30 and Table 31, respectively.

Predicted					
		Neutrophil	Lymphocyte	Monocyte	Eosinophil
Actual	Neutrophil	25	4	3	6
	Lymphocyte	2	5	0	0
	Monocyte	1	1	2	0
	Eosinophil	0	5	4	4

Table 30 Confusion Matrix (*Kaggle_Test_simple*)

Predicted					
		Neutrophil	Lymphocyte	Monocyte	Eosinophil
Actual	Neutrophil	14	2	1	3
	Lymphocyte	7	12	0	1
	Monocyte	3	0	17	0
	Eosinophil	6	4	5	5

Table 31 Confusion Matrix (*Kaggle_Test)*

The metric table from the combined confusion matrix for *Kaggle_Test_simple* and *Kaggle_Test* is shown in Table 32. The classification was decent for lymphocytes and monocytes at 80% and just under 90%, respectively. The neutrophil and eosinophils were not as accurate at around 60%, which may be due to the staining and not capturing a consistent whole image. The Kaggle dataset did not contain any basophils or lymphoblasts.

Classification	**Sensitivity (%)**	**Specificity (%)**	**Accuracy (%)**
Neutrophil	57.35	60.27	58.87
Lymphocyte	61.54	87.01	80.58
Monocyte	79.17	92.75	89.25
Eosinophil	27.27	75.51	63.36

Table 32 Classification Metrics (*Kaggle*)

Comparing Prior Work

The results of this research were compared to prior work in this section. First the overall metrics calculated from the cell counts after segmentation and separation are compared. Next, the classification metrics for the two datasets are compared, and then the metrics related to lymphoblast detection are compared.

Total RBC and WBC Counts

Adagale and Pawar (2013) and Ma et al. (2016) both calculated their metrics based on a computed count percentage as follows:

$$Computed\ Count\ Percentage = \frac{Measured\ Blood\ Cell\ Count}{Actual\ Blood\ Cell\ Count}\,(100)$$

The computed count percentage from Adagale and Pawar (2013) was 90.1%; for Ma et al. (2016) their computed count percentage was 93.18%. The total cells from the data shown in both their papers was less than 100 RBCs per image. From the experiments, the computed count percentage was comparable at 92.86% on *ALL_IDB_large* and lower at 72.3% on *ALL_IDB_subset*. However, the RBC accuracy on the *ALL_IDB_large* using the standard PCNN and *separate* was 82% using the accuracy metric described in Chapter 3.

From Loddo, their overall accuracy was 99.2% on WBCs and 98% on RBCs, whereas the standard PCNN accuracy was slightly lower at 98% on WBCs and 82% on RBCs. Their method obtained a higher accuracy and other metrics then the PCNN methods in this paper using *ALL_IDB_large* as shown in Table 33. However, these results were for both RBC and WBC counting. Using the specific WBC parameters for WBC counts provided more accurate results as the WBC accuracy was 100% for the standard PCNN on *ALL_IDB_subset*.

Method	**Type**	**Average Metrics**			
		Accuracy	*Precision*	*Recall*	*F1*
Standard PCNN and *separate* (*ALL_IDB_large*)	RBC	82%	93%	88%	90%
	WBC	98.%	79%	95%	85%
Loddo et al. (2016)	RBC	98%	89%	98%	93%
	WBC	99.2%	100%	99.2%	99.6%

Table 33 Metric Comparison with Literature (*ALL_IDB_large*)

WBC Classification and Counts

Macawile et al. (2018) classified WBCs from the ALL_IDB dataset using three different models, with their AlexNet model providing the best results. The overall average sensitivity, specificity, and accuracy for both methods are shown in Table 34, these do not include the lymphoblasts since they were not in their results. While their results show a greater sensitivity, the accuracy and specificity were slightly better with the PCNN classifier. The sensitivity result was impacted by the number of eosinophils and basophils which were limited to two in Macawile et al. (2018) and one or less in the test sets used in this paper.

Classification	**Sensitivity (%)**	**Specificity (%)**	**Accuracy (%)**
PCNN	44.12	98.62	97.46
Macawile et al. (2018)	89.18	97.85	96.63

Table 34 Classification Metric Comparison (*ALL_IDB dataset*)

Liang et al. (2018) classified WBCs from the Kaggle dataset using different models, with their Xception-LSTM model providing the best results. Their accuracy results were better those using the PCNN classification, which may be related to the staining of the slides in the dataset. The authors did not provide sensitivity or specificity in their results. The overall average accuracy comparison between the two methods is shown in Table 34.

Classification	**Accuracy (%)**
PCNN	73.02
Liang et al. (2018)	90.79

Table 35 Classification Metric Comparison (*Kaggle dataset*)

Lymphoblast Detection

For lymphoblast detection, Ghosh et al. (2017) determined the metrics on a per slide basis indicating the presence of lymphoblasts and not a per cell classification. If a slide contains a lymphoblast and it is detected then it is counted as a true positive, if the slide contains a lymphoblast and it is not detected it is counted as a false negative, and similar for false positives and true negatives. The previous results displayed under the classification section was for the overall classification by type. The per slide metric used by Ghosh et al. (2017) was calculated for the results from *ALL_IDB_large*, *ALL_IDB_large2*, *ALL_IDB_large3*, and *ALL_IDB_first33* with any duplicates between subsets removed. Table 36 shows the comparison of Ghosh et al. (2017) and the PCNN classifier for lymphoblast detection using the per slide metric. The PCNN classifier metrics for accuracy and specificity are less than 1% lower with sensitivity being 3% lower; however, the lymphoblast cell level accuracy was 88.2% whereas Ghosh et al. (2017) stated they were not able to achieve good cell level accuracy. Ghosh et al. (2017) also did not support clumped cells which was supported by the PCNN segmentation and separation method in this research.

Classification	Sensitivity (%)	Specificity (%)	Accuracy (%)
PCNN	97.2	94.12	96.23
Ghosh et al. (2017)	100	94.9153	97.22

Table 36 Lymphoblast Detection Comparison

Summary

While the metrics from these experiments using PCNN produced lower RBC accuracy than those in the literature, using the standard PCNN, *separate*, and the PCNN classifier provided comparable or better results for WBC counts, classification, and lymphoblast detection on the ALL_IDB dataset. The staining on the Kaggle dataset had an impact on the WBC results as the accuracy values were lower. Some of the images that were in *ALL_IDB_large*, did require different parameter tuning which was most likely related to the larger size due to the resolution of those images.

Chapter 5

Conclusion

Introduction

This section discusses the overall experiments along with issues and concerns. The manual counting of RBCs was time consuming and had a potential for error when the number of cells was large and there were overlapped cells. The staining of the slides also had an impact on the results and some unexpected complications like the white space in the Kaggle dataset due to rotation that had to be removed during preprocessing as previously mentioned. However, most of the preprocessing and post processing was consistent between the datasets.

Conclusions

This section discusses the parameters and findings from the experiments. It provides comparative summary for segmentation, separation, and classification. The section concludes with an overall assessment of the experiments.

Parameters

There were several parameters associated with PCNN segmentation, where a few had a significant impact on blood cell segmentation. The α_F and β parameters were impacted by the file size (image resolution) for overall RBC and WBC standard PCNN segmentation but did not have the same impact with region growing PCNN. These two parameters along with M and W weight matrices was impacted by the cell type. For WBC only segmentation for use with classification or lymphoblast detection, these parameters improved WBC accuracy. For region growing PCNN, like standard PCNN the *W* matrix improved WBC segmentation, however, there was only one *M* matrix that was successful at segmentation. The *W* matrix value also impacted WBC accuracy with region growing PCNN when combined with the color option. The different color spaces used with region growing PCNN segmentation did not have a significant impact.

The PCNN parameters were dependent on usage, as using a standard PCNN as a classifier verses segmentation required a different set of parameters. A couple of PCNN parameters were dependent on the dataset, such as the α_E for both segmentation and classification and the V_E for classification. The separation parameters were also dependent on the dataset as the acceptance criteria was related to the quality of the staining and image.

Segmentation

The standard PCNN provided better segmentation and separation than the region growing PCNN variants. While the WBC results for the standard PCNN were comparable to the literature with an average accuracy above 98%, RBC accuracy results were lower with the best case just over 82%. An interesting finding was that the PCNN parameters had an impact on segmentation for certain images that were larger due to a higher resolution. Whether this was really related to the size and resolution or something else about the image was not clear.

The region growing PCNN variants did not produce comparable results to the standard PCNN. However, an interesting finding was related to the color option for region growing; there was improvement using color over grey scale intensity, but there was little impact between the different color spaces. This may be due to the overlap in WBC and

RBC colors as they both have variations of purple. Whether using intensity or color, the PCNN segmentation variants tended to pick up more of the nucleus for WBC objects and not as much of the cytoplasm.

Separation

The PCNN standard segmentation did provide the edges and the *separate* method did find and separate the cell objects. The *separate* method that created a cell object image and subsequently removed the image from the working image improved finding clumped cells. The split variant for segmenting a larger file into four subimages also improved finding clump cells. However, for some cells with more cytoplasm (such as a large monocyte or neutrophil), the entire cell was not always included in the generated WBC subimage due to variations in the segmentation. Overall, the standard PCNN worked better than the region growing versions. This may be due to region growing adding more pixels to the region resulting in a clumpier segmented image as input to the *separate* method which was subsequently unable to find the edges of the actual cell.

Classification

The WBC classification results on the ALL-IDB dataset were comparable to Macawile et al. (2018) at 97% accuracy; however, the sensitivity was slightly lower due to the limited number of certain cell types. The classification on the Kaggle dataset obtained an accuracy about 15% lower than the results from Liang et al. (2018), but as previously mentioned the PCNN seemed to be more sensitive to the variations in staining on that dataset. The lymphoblast detection using PCNN was comparable with Ghosh et al. (2017) at 96% for the per slide results and per cell lymphoblast classification was also achieved at 88.2%.

Assessment

The assessment from the experiments was that the standard PCNN provided comparable or better accuracy for WBC segmentation, counting, classification, and lymphoblast detection. For WBC segmentation and counting the standard PCNN produced comparable accuracy, but other metrics were lower. The PCNN classifier produced slightly better accuracy and specificity but had lower sensitivity. The PCNN classifier did not perform as well on the Kaggle dataset. For lymphoblast detection, the PCNN classifier was comparable on the per slide results but with lower sensitivity. The PCNN classifier also obtained per cell results, however, the comparable method in the literature was not able to produce per cell results with their method. The standard PCNN however produced lower RBC results compared to the literature.

The region growing PCNN had slightly lower accuracy for WBC and significantly lower RBC accuracy compared to the standard PCNN. Using the color option for region growing PCNN improved the overall results compared with using the intensity value. A region growing PCNN color cell option tried to improve the region growing by specifying a min and max size for a region, but this did not significantly improve the results.

Quality of results depended on characteristics of the dataset. The image resolution and magnification were different on some files in the ALL_IDB and caused variations in the segmentation parameters for the standard PCNN. The main parameters impacted were the α_F and β, with α_F being more sensitive to the size of the image and requiring a slightly larger value. The α_F parameter is a decay parameter and β is the linking strength which are combined and compared against the threshold value to determine the output for the iteration. A higher α_F parameter would decay faster allowing more pixels in the segmented output.

The staining of the images in the Kaggle dataset were slightly different from the ALL_IDB making the images not as clear with less distinct colors. This had an impact when segmenting and classifying WBCs. The α_E impacted both segmentation and classification and the V_E impacted classification only. The V_E is the normalizing constant and α_E is the decay parameter and these are associated with the threshold for determining the output of the PCNN. The lower image quality resulted in lower feeding and linking network values so using the same parameters generated a threshold value that resulted in little or no output. Thus, the Kaggle dataset required setting a higher α_E parameter value so the threshold would decay faster and allow the pixels to appear in the output. For classification, the V_E parameter was also reduced to allow more pixels to be captured in the output.

Future Work

Proposed improvements to the use of PCNN for segmentation, classification, and separation will be described in this section. The improvements to the segmentation are described first, followed by the improvements to classification, and ending with the improvements to the separation method.

Segmentation

The first improvement for segmentation relates to the preprocessing of the image before it is sent to PCNN segmentation. The ALL-IDB contained images of different sizes representing different resolutions and magnifications. The Kaggle dataset files were different sizes compared to the ALL-IDB files. This improvement would require resizing the images to a standard size to determine optimal values for PCNN parameters β and α_F for all images. Macawile et al. (2018) resized the ALL-IDB images to the required size for the different CNN models.

The second improvement to the standard PCNN is to add the linking control unit from Xu et al. (2018) to provide color input. The color option improved the region growing results compared to grey scale intensity and may provide an improvement in the standard PCNN segmentation.

Classification

Another set of proposals entail modifications to the PCNN classifier. The first is to add an average or mean color representation feature to the texture vector, such as average intensity or mean color value. Color features were used by Khobragade et al. (2015) and Alreza and Karimian (2016). Khobragade et al. (2015) converted RBG to HSV and captured the mean, and Alreza and Karimian (2016) captured from the grey scale value. The mean color or average intensity for color representation could be added as element(s) to the texture vector. This may improve classification between neutrophils, eosinophils, and basophils as they all have granules, but the eosinophils are a deep orange, and the basophils are a deep purple, so the color is an important distinguishing characteristic.

The second PCNN classifier improvement would be to use a PCNN variant called a Spiking Cortical Model (SCM). The SCM is a simplified PCNN with only 3 equations (Yang et al. (2017). Yang et al. (2017) found that for textural features the SCM showed an improvement over the standard PCNN.

Separation

The last set of proposed improvements are related to the *separate* method used to separate the cell objects from the segmented image. The current method finds the circles in the segmented image using CHT. It then processes each circle center found from CHT and determines if there is an acceptable object. The acceptance of an object is based on the matching criteria parameters. Once an object matches the acceptance criteria a cell object subimage is created. The cell object is then subsequently removed from the working image. The process continues with the next circle center. For the current separate method, CHT is run once on each segmented image.

The improvement to the separate method would start by executing CHT on the working image to find one circle at a time. The circle center found would be matched to the acceptance criteria to determine if there is an acceptable object. Like the current method a subimage would be created when an acceptable object is found, and that object is removed from the working image. The improvement process goes back to the working image and executes CHT to find the next circle. This continues until no more circles are found in the working image. The improvement flow is shown in Figure 39. An additional preprocessing step would remove small objects from the working image before finding the next circle with CHT. This should improve finding cells that are in clumps and reduce the number of background cells in the created object image improving classification.

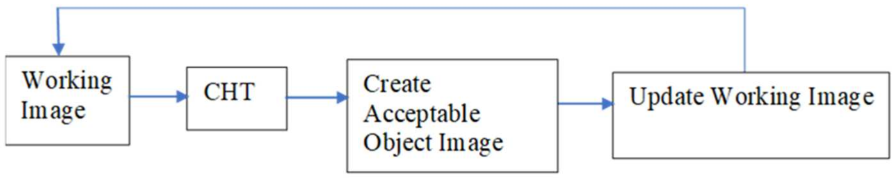


Figure 39 Separate Flow Improvement

Summary

The standard PCNN worked best on the segmentation and separation of RBC and WBC compared to the region growing versions. WBC segmentation and separation had significantly higher accuracy then RBC segmentation and separation. Two PCNNs were required for WBC segmentation and classification, one to segment and separate the WBC objects, and another to capture the texture vector for each object. Overall, using a standard PCNN to segment and classify WBCs provides comparable accuracy to the literature at 98% for segmentation, 97% for classification, and 96% for lymphoblast detection.

This research showed that a standard PCNN can be used to successfully segment RBC and WBC objects. The region growing PCNN was also successful in segmenting and separating WBCs but did not perform as well as the standard PCNN. The *separate* method introduced in this research facilitated cell counting and WBC classification with the creation of WBC subimages, along with detection of clumped cells. Using a standard PCNN as a WBC classifier was introduced with this research and proved to be a successful classifier and lymphoblast detector.

References

Abdul Nasir, A. S., Mashor, M., Y., Rosline, H. (2011). Unsupervised colour segmentation of white blood cell for acute leukaemia images. In *Proceedings of the IEEE International Conference on Imaging Systems and Techniques,* 142 - 145. doi:10.1109/IST.2011.5962188

Acharya, V., Kumar, P. (2018). Identification and red blood cell automated counting from blood smear images using computer-aided system. *Medical & Biological Engineering & Computing,* 56 (*3*). 483 - 489. doi:10.1007/s11517-017-1708-9

Adams, R., Bischof, L. (1994). Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 16 (*6*). 641 - 647. doi:10.1109/34.295913

Adagale, S. S., Pawar, S. S. (2013). Image segmentation using PCNN and template matching for blood cell counting. In *Proceedings of the IEEE International Conference on Computational Intelligence and Computing Research*. 1 - 5. doi:10.1109/ICCIC.2013.6724161

Alreza, Z. K. K., Karimian A. (2016). Design a new algorithm to count white blood cells for classification leukemic blood image using machine vision system. In *Proceedings of the 6th International Conference on Computer and Knowledge Engineering (ICCKE)*, 251 - 256. doi:10.1109/ICCKE.2016.7802148

Brown, B. A., (1980). *Hematology: Principles and Procedures (Third Edition).* Philadelphia, PA: Lea & Febiger.

Chacon, M. I., Mendoza, J. A. (2011). A PCNN-FCM time series classifier for texture segmentation. *2011 Annual Meeting of the North American Fuzzy Information Processing Society,* 1 - 6. doi:10.1109/NAFIPS.2011.5752019

Chouhan, S. S., Kaul, A., Singh, U. P. (2017). Soft computing approaches for image segmentation: a survey. *Multimedia Tools and Applications*, pp 1-55. doi:10.1107/s11042-018-6005-6

Dela Cruz, J. C., Valiente Jr., L. C., Castor, C. M. T., Mendoza, A. J. B., Song, C. J. L., Torres, B., B. P. (2017). Determination of blood components (WBCs, RBCs, and Platelets) count in microscopic images using image processing and analysis. In *Proceedings of the IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, 1 - 7. doi:10.1109/HNICEM.2017.8269515

Deng, X., Yan, C., Ma, Y. (2019). PCNN mechanism and its parameter settings. *IEEE Transactions on Neural Networks and Learning Systems*, 1 – 14. doi:10.1109/TNNLS.2019.2905113

Di Ruberto, C., Loddo, A., Putzu, L. (2016). A leucocytes count system from blood smear images. *Machine Vision and Applications*, 27(*8*), 1151 – 1160. doi:10.1107/s00138-016-0812-4

Eckhorn, R., Reitboeck, H. J., Arndt, M., Dicke, P. (1990). Feature linking via synchronization among distributed assemblies: simulations of results from cat visual cortex. *Neural Computation*, 2, 293 – 307.

Gatc, J., Maspiyanti, F. (2016). Red blood cell and white blood cell classification using double thresholding and BLOB analysis. In *Proceedings of the 4th International Conference on Information and Communication Technology (ICoICT)*, 1 - 5. doi:10.1109/ICoICT.2016.7571900

Gautam, A., Bhadauria, H. (2014). Classification of white blood cells based on morphological features. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2363 - 2368. doi:10.1109/ICACCI.2014.6968362

Ghosh, A., Singh, S., Sheet, D. (2017). Simultaneous localization and classification of acute lymphoblastic leukemic cells in peripheral blood smears using a deep convolutional network with average pooling layer. In *Proceedings of the International Conference on Industrial and Information Systems (ICIIS)*, 529 - 534. doi:10.1109/ICIINFS.2017.8300425

Gonzalez, R. C., Woods, R. E. (2002). *Digital Image Processing (Second Edition)*. Upper Saddle River, NJ: Prentice Hall.

Jagadev, P., Virani, H. G. (2017). Detection of leukemia and its types using image processing and machine learning. In *Proceedings of the International Conference on Trends in Electronics and Informatics (ICEI)*, 522 - 526. doi:10.1109/ICOEI.2017.8300983

Khobragade, S., Mor, D. D., Patil, C. Y. (2015). Detection of leukemia in microscopic white blood cell images. In *Proceedings of the International Conference on Information Processing (ICIP)*, 435 - 440. doi:10.1109/INFOP.2015.7489422

Kim, K. S., Kim, P. K., Song, J. J., Park, Y. C. (2000). Analyzing blood cell image to distinguish its abnormalities. In *Proceedings of the eighth ACM International Conference on Multimedia*, 395 - 397. doi:10.1145/354384.354543

Kolhatkar, D., Wankhade, N. (2016). Detection and counting of blood cells using image segmentation: a review. In *Proceedings of the World Conference on Futuristic Trends in Research and Innovation for Social Welfare (WCFTR)*, 1 - 5. doi:10.1109/STARTUP.2016.7583931

Kuntimad, G., Ranganath, H., S. (1999). Perfect image segmentation using pulse coupled neural networks. IEEE Transactions on Neural Networks, 10*(3)*. 591 - 598. doi:10.1109/72.761716

Labati, R. D., Piuri, V., Scotti, F. (2011). ALL_IDB: The acute lymphoblastic leukemia image database for image processing. In *Proceedings of the 18th IEEE International Conference on Image Processing*, 2045 - 2048. doi:10.1109/ICIP.2011.6115881

Le, D. T., Bui, A. A., Yu, Z., Bui, F. M. (2015). An automated framework for counting lymphocytes. In *Proceedings of the International Conference and Workshop on Computing and Communication (IEMCON)*, 1 - 6. doi:10.1109/IEMCON.2015.7344535

Li, H., Zhou, H., Chen, Y., Shi, X. (2010). Image segmentation by using grayscale iteration threshold pulse couple neural network. In *Proceedings of the 2nd International Conference on Information Engineering and Computer Science (ICIECS),* 1 - 4. doi:10.1109/ICIECS.2010.5678152

Liang, G., Hong, H., Xie, W., Zheng, L. (2018). Combining convolutional neural network with recursive neural network for blood cell image classification. *IEEE Access, 6,* 36188 - 36197. doi:10.1109/ACCESS.2018.2846685

Lindblad, T., Kinser, J. M. (2013). *Image Processing Using Pulse-Coupled Neural Networks. Applications in Python (Third Edition).* New York, NY: Springer. Doi:10.1007/978-3-642-36877-6

Liu, Z., Wang, F., Yan, S., Huang, R. (2016). Blood cell segmentation based on improved pulse coupled neural network and fuzzy entropy. *International Journal BioAutomation*, 20(*4*), 471 - 482.

Loddo, A., Putzu, L., Di Ruberto, C., Fenu, G. (2016). A computer-aided system for differential count from peripheral blood cell images. In *Proceedings of the 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, 112 - 118. doi:10.1109/SITIS.2016.26

Ma, R., Liang, Y., Ma, Y. (2016). A self-adapting method for RBC count from different blood smears based on PCNN and image quality. In *Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 1611 - 1615. doi:10.1109/BIBM.2016.7822760

Macawile, M. J., Quinones, V. V., Ballado Jr., A., Dela Cruz, J., Caya, M., V. (2018). White blood cell classification and counting using convolutional neural network. In *Proceedings of the 3rd International Conference on Control and Robotics Engineering (ICCRE)*, 259 - 263. doi:10.1109/ICCRE.2018.8376476

Manik, S., Saini, L., M., Vadera, N. (2016). Counting and classification of white blood cell using Artificial Neural Network (ANN). In *Proceedings of the IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, 1 - 5. doi:10.1109/ICPEICES.2016.7853644

Mao-jun, S., Zhao-bin, W., Hong-juan, Z., Yi-de, M. (2008). A new method for blood cell image segmentation and counting based on PCNN and autowave. In *Proceedings of the 3rd International Symposium on Communications, Control and Signal Processing (ISCCSP)*, 6 - 9. doi:10.1109/ISCCSP.2008.4537182

Mohamed, M., Far, B., Guaily, A., (2012). An efficient technique for white blood cells nuclei automatic segmentation. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 220 - 225. doi:10.1109/ICSMC.2012.6377703

Mooney, P. (2018). Kaggle Blood Cell Images Version 6 [Webpage]. Retrieved from https://www.kaggle.com/paultimothymooney/blood-cells

Ongun, G., Halici, U., Leblebicioglu, K., Atalay, V., Beksac, M., Beksac, S. (2001). An automated differential blood count system. In *Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEMBS)*, 3, 2583 - 2586. doi:10.1109/IEMBS.2001.1017309

Puttamadegowda, J., Prasannakumar, S. C. (2016). White blood cell segmentation using fuzzy c means and snake. In *Proceedings of the International Conference on Computational Systems and Information Systems for Sustainable Solutions (CSITSS)*, 47 - 52. doi:10.1109/CSITSS.2016.7779438

Quinones, V. V., Macawile, M. J., Ballado Jr., A., Dela Cruz, J., Caya, M. V. (2018). Leukocyte segmentation and counting based on microscopic blood images using HSV saturation component with blob analysis. *In Proceedings of the 3rd International Conference on Control and Robotics Engineering (ICCRE)*, 254 - 258. doi:10.1109/ICCRE.2018.8376475

Rashid, N. Z. N., Mashor, M. Y., Hassan, R. (2015). Unsupervised color image segmentation of red blood cell for thalassemia disease. In *Proceedings of the 2nd International Conference on Biomedical Engineering (ICoBE),* 1 - 6. doi:10.1109/ICoBE.2015.7235892

Rawat, J., Singh, A., Bhadauria, H. S., Kumar, I. (2014). Comparative analysis of segmentation algorithms for leukocyte extraction in acute lymphoblastic leukemia images. In *Proceedings of the International Conference on Parallel, Distributed and Grid Computing,* 245 - 250. doi:10.1109/PDGC.2014.7030750

Razzak, M. I., Naz, S. (2017). Microscopic blood smear segmentation and classification using deep contour aware CNN and extreme machine learning. In *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 801 - 807. doi:10.1109/CVPRW.2017.111

Ruifrok, A. C., Johnston, D. A. (2001). Quantification of histochemical staining by color deconvolution. *Analytical and Quantitative Cytology and Histology,* 23 (*4*), 291 - 299.

Savkare, S. S., Narote, A. S., Narote, S. P. (2016). Automatic blood cell segmentation using k-mean clustering from microscopic thin blood images. *In Proceedings of the Third International Symposium on Computer Vision and the Internet.* 8 - 11. doi:10.1145/2983402.2983409

Savkare, S. S., Narote, S. P. (2015). Blood cell segmentation from microscopic blood images. *In Proceedings of the International Conference on Information Processing (ICIP).* 502 - 505. doi:10.1109/INFOP.2015.7489435

Seth, S., Palodhi, K. (2017). An efficient algorithm for segregation of white and red blood cells based on modified hough transform. In *Proceedings of the 2017 IEEE Calcutta Conference (CALCON)*, 465 - 468. doi:10.1109/CALCON.2017.8280777

Shankar, V., Deshpande, M. M., Chaitra, N., Aditi, S. (2016). Automatic detection of acute lymphoblastic leukemia using image processing. In *Proceedings of the IEEE International Conference on Advances in Computer Applications (ICACA)*, 186 - 189. doi:10.1109/ICACA.2016.7887948

Sinha, N., Ramakrishnan A. G. (2003). Automation of differential blood count. In *Proceedings of the Conference on Convergent Technologies for Asia-Pacific Region (TENCON)*, 2, 547 - 551. doi:10.1109/TENCON.2003.1273221

Stewart, R. D., Fermin, I., Opper, M. (2002). Region growing with pulse-coupled neural networks: an alternative to seeded region growing. *IEEE Transaction on Neural Networks*, 13 (*6*), 1557 - 1562. doi:10.1109/TNN.2002.804229

Syahputra, M. F., Sari, A. R., Rahmat, R. F. (2017). Abnormality classification on the shape of red blood cells using radial basis function network. In *Proceedings of the 2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT)*, 1 - 5. doi:10.1109/CAIPT.2017.8320739

Yu, W., Chang, J., Yang, C., Zhang, L., Shen, H., Xia, Y., Sha, J. (2017). Automatic classification of leukocytes using deep neural network. In *Proceedings of the 12th International Conference on ASIC (ASICON)*, 1041 - 1044. doi:10.1109/ASICON.2017.8252657

Van der Walt, S., Schonberger, J. L, Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T., the scikit-image contributors. (2014). scikit-image: image processing in Python. *PeerJ*, 2, e453. doi:10.7717/peerj.453.

Varoquaux, G., Louppe, G., Pedregosa, F., Buitinck, L., Grisel, O., Mueller, A. (2015). scikit-learn: Machine learning without learning the machinery. *GetMobile: Mobile Computing and Communications,* 19 (*1*), 29 - 33. doi:10.7717/peerj.453.

Vogado, L. H. S., Veras, R. M. S., Andrade, A. R., de Araujo, F. H. D., e Silva, R. R. V., de Medeiros, F. N. S. (2016). Unsupervised leukemia cells segmentation based on multi-space color channels. In *Proceedings of the 2016 IEEE International Symposium on Multimedia (ISM)*, 451 - 456. doi:10.1109/ISM.2016.0103.

Xu, G., Li, S., Lei, B., Lv, K. (2018). Unsupervised color image segmentation with color-alone feature using region pulse coupled neural network. *Neurocomputing*, 306, 1 - 16. doi:10.1016/j.neucom.2018.04.010

Yang, L., Meer, P., Foran, D. J. (2005). Unsupervised segmentation based on robust estimation and color active contour models. *IEEE Transactions on Information Technology in Biomedicine*, 9 (*3*), 475 - 486. doi:10.1109/TITB.2005.847515

Yang, R., Lyu, C., Liu, Y., Zhou, W., Chen, C., Jiang, X., Li, P., Chen, H., Xu, R., Wang, Y. (2017). Spiking cortical model for geometry invariant and antinoise texture retrieval. In *Proceedings of the 2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 645 - 650. doi:10.1109/RCAR.2017.8311936

Zhan, K., Zhang, H., and Ma, Y. (2009). New spiking cortical model for invariant texture retrieval and image processing. *IEEE Transactions on Neural Networks*, 20 (*12*), 1980 - 1986. doi:10.1109/TNN.2009.2030585

Zhang, C., Xiao, X., Li, X., Chen, Y.J., Zhen, W., Chang, J. (2014). White blood cell segmentation by color-space-based k-means clustering. *Sensors*, 14, 16128 - 16147. doi:10.3390/s140916128

Zhou, D., Shao, Y. (2018). Region growing for image segmentation using an extended PCNN model. *IET Image Process*, 12 (*5*), 729 - 737. doi:10.1049/iet-ipr.2016.0990