# Leukemia Detection Algorithms: A Comparative Study

Sada Riad, Maryam Wazani, Aleen Abu Zant and Amjad Hawash*
Information & Computer Science Department
An-Najah National University
Nablus, Palestine, P.O. Box 7
Email: sadabasheer95@gmail.com, mariammohd890@gmail.com, aleenabuzant@gmail.com, amjad@najah.edu

***Abstract*—Leukemia is a group of blood cancers that usually begins in the bone marrow and is caused by a high number of abnormal blood cells. Yearly, the number of new cases of Leukemia increases with a high mortality rate due to the late detection of the disease. However, computerizing disease detection decreases the number of mortal cases by implementing several methodologies and algorithms. Image processing techniques are one of the major disciplines in detecting the disease early. A set of related algorithms were developed to help detect Leukemia depending on several blood components features like cells' properties (shape, color, size), cytoplasm features, and others. This work compares two approaches to Leukemia detection: Color K-means Clustering (CKC) and Acute Lymphoblastic Leukemia Subtypes (ALLS) detection algorithms. CKC algorithm depends on morphological filtering and segmentation using color k–means clustering and is tested with Nearest Neighbor (KNN) classifier. ALLS is based on detecting Leukemia subtypes by going through several image processing stages that lead to classifying the image as infected or not. In order to test the amount of accuracy difference between the two approaches, the extracted features of the testing set of images are processed using Weka software. The precision, recall, f-score, accuracy, and time measures were computed for comparison purposes.**

***Index Terms*—Leukemia Detection, Image Processing, Classification Algorithms.**

## I. Introduction

Leukemia is a type of cancer that affects the white blood cells (WBCs) where they become abnormal, divide and grow in an uncontrolled way [1]. These malfunctioned cells harm the blood and bone marrow, putting the immune system at risk, and imposing restrictions. These cancerous cells can leak into the bloodstream contaminating the blood and harming other organs of the human body such as the liver, kidneys, spleen, brain, and others, resulting in various lethal cancers.

Leukemia is classified according to the types of affected cells. In the case that the *Granulocytes* and *monocytes* cells are affected, it is classified as *myelogenous* (acute myeloid leukemia), and if the affected cells are *lymphocytes*, then the Leukemia is classified as *lymphoblastic* (acute lymphoblastic leukemia). According to French American British classification, it is categorized into 3 subtypes: *L1*, *L2*, and *L3* [2]. *L1* type shows a group of uniform, small blast cells with scanty cytoplasm; their nucleus is discoid and well structured. *L2* cells comprise larger blast cells with more prominent nucleoli and cytoplasm and with more heterogeneity also with small nucleoli inside the nucleus. However, *L3* type has large blasts with prominent nucleoli, strongly basophilic cytoplasm, and cytoplasmic vacuoles. These subtypes are shown in Fig. 1.

(a) L1 (b) L2 (c) L3

Fig. 1: *L1*, *L2* and *L3* cell types.

These varieties in shapes and components make it encouraging for a lot of image processing researchers to study the different attributes that appear in the images of cells in order to better detect the presence or absence of Leukemia [3]. Different image processing techniques are used to detect Leukemia such as manual morphological image analysis and complete blood count. However, these methods are time-consuming and less accurate. Conventional ways of detecting Leukemia like bone marrow examination in order to identify the type of Leukemia are also time-consuming and prone to some diagnosis errors since the procedure depends on the expertise of the medical professional [4].

On the other side, the latest computer vision techniques emerge a noticeable enhancement with respect to Leukemia detection by applying a set of image-related processes. These stages can be listed as: (1) Acquisition (2) Pre-processing Techniques (3) Segmentation (4) Features Extraction, and (5) Classification (presence or absence of Leukemia).

Our contribution in this work is related to implementing and executing a comparison between two different algorithms both related to examining a set of blood cells images depending on different extracted features. The comparison between the algorithms are based on a set of measures such as: *Precision*, *Recall*, *Accuracy*, *F-Score* and *execution time*. We can summarize our contribution in the following points:

- Prepare a set of images to be used as a data set.
- Extracting a set of features from the cells of the data set (CKC Algorithm).
- Implementing expert knowledge in detecting Leukemia

* Corresponding Author: amjad@najah.edu

cancer and categorizing images (data set) accordingly (ALLS Algorithm).
- Use Weka [1] to categorize the data set.
- Compare between all methods using a set of related measures.

The rest of this paper is organized as follows: Related Works is proposed in Section II. Section III discusses the implemented algorithms while Section IV discusses the conducted experimental tests .Finally, Section V concludes this paper.

## II. Related Works

During literature, image processing techniques were involved in several studies. This involvement led to an enhancement in both time and effort in extracting the desired features from images as usually experts do [5].

Remote sensing, image and data storage for transmission in business applications, medical imaging, acoustic imaging, Forensic sciences, and industrial automation are all examples of applications in which they rely on image processing techniques. The medical field is one of the important fields that employ image processing techniques due to the intensive exploration of features that can be gained. 3D models can be generated to better understand the human body activities [6].

The work published in [7] is related to inventing an Android application that is able to capture images of blood and analyze these images in order to detect Malaria, Leishmaniasis, and Acute Leukemia. Authors of the work deployed *K-nearest Neighbour (KNN)* algorithm to classify the samples of images.

Identifying red blood cells is the work published by the authors of [8] where the methodology applied is related to automatic identification and classification of red cells in different classes of interest for diagnosis using microscopic images of blood smear. The authors of the work have applied several image processing techniques like binarization, contrast enhancement, noise elimination, morphological operations (dilatation, erosion), labeling, and extraction of some features of interest (area, perimeter, diameter).

Machine learning is also involved in targeting blood-related diseases by continuously analyzing periodical images in order to better track the medical cases of patients. This is applied in the work published in [9] were authors of the work employed deep neural networks for the image classification and platelet count.

Involvement of cells' boundaries and geometrical properties is also one of the aspects that stimulate image processing techniques to rely on [10]. Threshold-based methods such as Otsu and histogram identify the white blood cells (WBCs) directly from the blood smear image using the intensity level.

Contour-based methods to identify the irregularities of the nucleus boundary were related to the work published in [11]. Watershed segmentation algorithm and unsupervised color segmentation provided good segmentation of nuclei to detect acute leukemia.

[1] https://www.cs.waikato.ac.nz/ml/weka/

Some works are related to automating the detection the Leukemia from microscopic images. Kovalev [12] is one of these works in which the nuclei are first detected and region growing techniques are applied. However, Scotti [13] used some threshold operations, a low-pass filter for removal of background, and clustering for white blood cells segmentation. Moreover, Piuri [14] performed white blood cell segmentation using edge detection and trained a neural network by morphological features to recognize lymphoblast. Mohapatra [15] applied clustering for white blood cells segmentation and extracted some of the features like shape, color, texture, fractal, Fourier descriptors, and contour. Finally, Escalante [15] invented a scheme for classifying Leukemia using the swarm model. The Leukemia cells need to be isolated manually to make this system work. These isolated cells are then segmented by Markov random fields to find out features of the type of leukemia.

Some other works are related to microscopic approaches used for Leukemia imaging such as confocal microscopy (CM) and Optical coherence tomography (OCT). The works published in [16]–[18] are related to exploring the potential of optical coherence microscopy and the automatic calibration in Fourier-domain optical coherence tomography. The proposed works can be performed during scanning operation and does not require an auxiliary interferometer for calibration signal generation and an additional channel for its acquisition.

Despite the image processing-based algorithmic approaches to detecting Leukemia that relies on the different geometrical features detected in the images, considering other features that are attractive for blood pathologists is very important. However, it is quite useful to compare the accuracy of algorithms that involve the typical steps of image processing with respect to Leukemia detection and the manual detection of Leukemia that a blood pathologist usually follows. This is our contribution in this work, that is, comparing typical steps applied in computerized algorithms in detecting Leukemia and the procedure followed by a blood pathologist for the same purpose. To this, we implemented both methodologies and compared between them in terms of *precision*, *recall*, *F-score*, *accuracy* and *time*.

## III. Methodology

Images used in this study were obtained from *ALL-Image DataBase (IDB)* public data set available online for study purposes [19]. This data set (108 images) is related to acute *lymphoblastic leukemia-IDB*. These images are already categorized into *normal* or *leukemia* by an expert. Moreover, 8 microscopic blood images for typical *L1*, *L2*, and *L3* type were categorized by an expert oncologist. In total, we have 46 normal images and 71 cancerous ones. For classification studying purposes, we divided the data set into 60% as a training set and 40% as a testing set.

### A. Color K-means clustering (CKC)

This algorithm is based on analyzing several microscopic images (Fig. 2) of white blood cells to detect the presence or absence of Leukemia. As shown in Fig. 3, the flowchart

involves a set of steps for the detection process. These steps are discussed below:

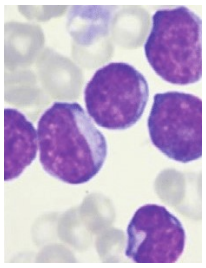


Fig. 2: A sample of microscopic blood images.

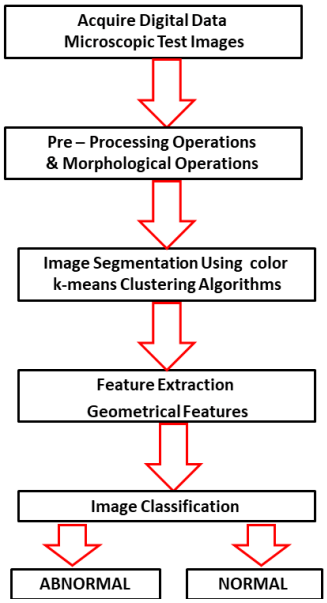



Fig. 3: Algorithmic steps for acute leukemia detection.

*1) Image Pre-processing:* This stage is related to removing image noises that might exist to prepare it for later processes. The image is converted to binary format with threshold=0.6 (estimated by testing) as in Fig. 4 (left) to be ready for the first morphological techniques (erosion using the function *imerode*[2]) that remove unnecessary pixels on object boundaries as shown in Fig. 4 (right). This procedure is used to isolate the nucleus.

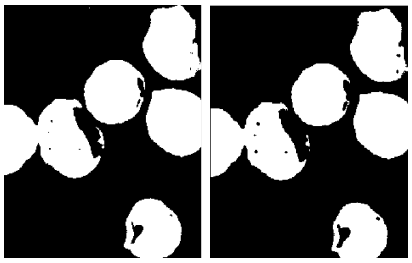


Fig. 4: Binary & Eroded images.

[2]This is a Matlab function. From now and on, we will notify the MatLab functions used to implement the used image processing techniques.

Open morphology operation using the function *bwareaopen* is then applied to smooth the contour of a given object, break narrow isthmuses, and eliminate thin protrusions [20] as shown in Fig. 5 (left). After that, the algorithm fills the holes inside the hollow shapes using the function *imfill*. We define these hollow shapes or holes as background pixels enclosed by a border of foreground pixels. This operation is useful in removing irrelevant artifacts from images. These techniques can also be used to find specific shapes in an image. Fig. 5 (right) depicts the image after it has been morphologically cleaned by removing irregular components that include the removal of leucocytes on the edges and irregular components [21].

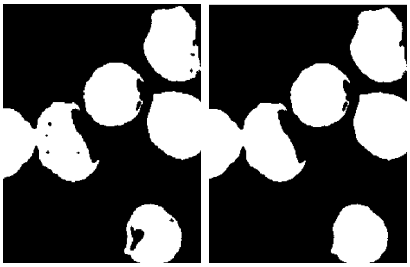


Fig. 5: Open morphology image & filling holes.

*2) Image segmentation:* Before segmentation, the image is converted to cluster index to facilitate the k-means clustering process to segment the image based on colors as shown in Fig. 6 (left). Segmentation is used to separate white blood cells from the image [22]. The color k-means method is used to divide the image into 3 clusters with k=3: (1) cytoplasm as in Fig. 6 (right), (2) nucleus of white blood cells as in Fig. 7 (left), and (3) background as in Fig. 7 (right).

Because the detection of Leukemia is just focused on white blood cells, we selected the cluster that includes them.

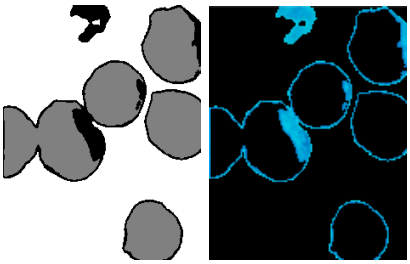


Fig. 6: Index image & Cytoplasm cluster.

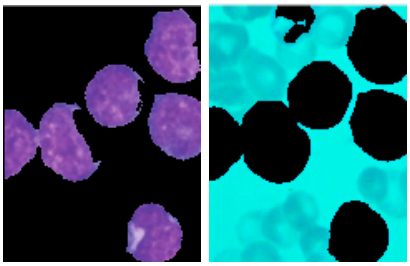


Fig. 7: Nucleus & Background clusters.

*3) Identification of grouped leucocytes:* Roundness has been used as a measure for dividing leucocytes [3]. Roundness can be computed by dividing the area of a circle by using the convex perimeter. If the cell is round, then the value of roundness is 1 (<1 otherwise). Fig. 8 shows cells with their roundness values where the ones with roundness near to 1 are close to circular shapes. Equation 1 illustrate the roundness formula where *area* is the circle area with radius computed from the Matlab method *regionprops* and *convex_perimeter* is the length of the cell perimeter computed by the same function [21].

$$Roundness = \frac{4 * \pi * area}{convex\_perimeter^2} \quad (1)$$

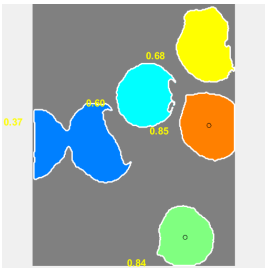



Fig. 8: Samples of Cells with their corresponding roundness values.

*4) Features extraction:* Features extraction technique converts some features in the image to numeric data to be used for image classification (infected or not). 10 features are extracted: (1) *Convexity*, (2) *Solidity*, (3) *Elongation*, (4) *Rectangularity*, (5) *Eccentricity*, (6) *Compactness*, (7) *Roundness*, (8) *Area*, (9) *Perimeter*, and (10) *Radius*.

The features *Perimeter*, *radius* and *area* are already implemented in Matlab using the method *regionprops*. In the following, we are clarifying the rest of used measures:

- *Convexity:* shows the relative amount of difference of an object from its convex object. The *convexity* measure is given by equation 2:

$$Convexity = \frac{perimeter\_convex}{perimeter} \quad (2)$$

- *Solidity:* is used to find out the density of a component. If the value is 1, then it can be identified as a solid object, or a component having irregular boundaries otherwise. The solidity measure is given by equation 3:

$$Solidity = \frac{area}{convex\_area} \quad (3)$$

- *Elongation:* indicates the object elongation towards a particular axis. Equation 4 illustrates the computation of this feature:

$$Elongation = 1 - \frac{major\_axis}{minor\_axis} \quad (4)$$

- *Rectangularity:* depicts how well the bounding box is filled. Equation 5 illustrates how this features is computed:

$$Rectangularity = \frac{area}{major\_axis * minor\_axis} \quad (5)$$

- *Eccentricity:* is the ratio of the major axis length and the foci of the ellipse. Equation 6 illustrates its computation:

$$Eccentricity = \frac{\sqrt{(major\_axis^2 - minor\_axis^2)}}{major\_axis} \quad (6)$$

- *Compactness:* is the ratio of the area of an object and the area of the circle having the same perimeter. Equation 7 reflects the computation of this feature:

$$Compactness = \frac{4 * \pi * area}{perimeter} \quad (7)$$

*5) Image classification:* K-nearest neighbors (*KNN*) is a supervised machine learning algorithm that can be used to solve both classification and regression tasks [3]. In our work, it is used to classify features of images to know if they are infected or not. Since we have 10 different features extracted from all of the images in the training set, we can consider we have a space of 10 dimensions, and using the *KNN* measure we can represent the 10 features extracted from the training set images for each cell in these images as points in that space each of which is composed of 10 indices. Moreover, the image under test is processed to extract the same features to be represented as points in the same space. The number of cells that appear in the image equals the number of points in the space. Now, *KNN* measure is used to compute the distance between all points of the image under test with all points in the space (those represent the cells of the training set images). After that, we ranked these distances from shortest to longest. Considering *k=1*, we checked the nearest point in the space to all points in the image under test. If we found one of the nearest points cancerous, we decide that the image under test is cancerous. However, equation 8 illustrates the formula for computing the distance between two points in a given space where $n$ is the dimension of the space.

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (8)$$

The following pseudocode represents how *KNN* works in finding the minimum distances between the features in the **10-D** space. The code starts by defining an empty array *distances[]* then the function *knn* is executed with two arrays as parameters: *features[]* and *cellsFeatures[]*. The first 2-D array contains the list of features extracted from all images in the test set, in which an entry is a 10 column of extracted features for each cell found in each image. The second array all features of all cells found in a given image, where each entry in this array is composed of the number of the cell and the list of 10 features extracted from that cell. The code that

[3]https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

evolves in the *features* array using the index *fchrs* and evolves in the *cellsFeatures* using the index *cFchrs*. By this, each represents a one-dimensional array that contains 10 extracted features. Both indexes are then sent to the function *distance()* that computes the distance between the two indexes and saves the computed distance in *dist* variable that is sent to the member function *add* that adds the data of both indices and their distance to the *distances* array. The code then returns the minimum distance found between a feature for one of the cells of the image being tested and all of the collected features in the testing set of images using the function *min()*.

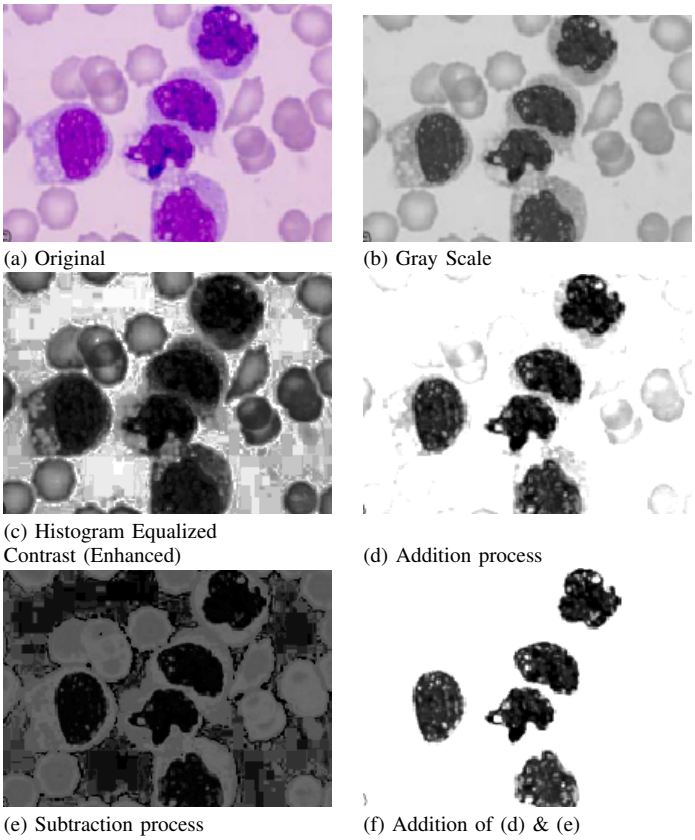
```
define distances[];
function knn(features[], cellsFeatures[]){
   foreach(features as fchrs){
       foreach(cellsFeatures as cFchrs){
           dist = distance(fchrs, cFchrs);
           distances.add(fchrs, cFchrs, dist);
           }
       }
   }
   return min(distances);
}
```

### B. Acute Lymphoblastic Leukemia Subtypes (ALLS)

In this algorithm, we relied on the visible characteristics in the image of the blood sample for the three types: *L1*, *L2* and *L3* as an expert recognizes them. To do this, we investigated a set of features with a pathologist who illustrates to us what features he searches for inside an image to decide if the image is infected or not. Type *L1* is distinguished by the presence of similarities between cells in terms of *size* and *shape*, while *L2* is characterized by the presence of small nuclei inside the nucleus and irregular cell shapes. Finally, *L3* is most distinguished by the presence of vacuoles in the cytoplasm. According to this, part of our contribution to this work is to implement the extraction of these features and to conduct a comparison between the results of ALLS with the results of CKC. The following steps are implemented:

*1) Image Pre-processing:* Image pre-processing passes through several stages in order to separate the components of the image. These steps are implemented according to the way the pathologist tests the absence or presence of Leukemia. Image pre-processing goes on with the following steps:

1) *Nuclei Separation*: nucleus needs to be separated from the cell using the pre-processing of microscopic blood image, by contrast, stretching enhances the global uniformity, local sensitivity, and geometry of the blood cells [15]. It is performed using the morphological addition and subtraction operations illustrated in the following steps: (1) Conversion of the original image to grayscale. (2) Histogram application using the values 0 and 255 as the lower and upper limits. (3) The application of contrast stretching. (4) Histogram equalization. (5) Addition then subtraction between contrast stretching image (step 3) with histogram equalized image (step 4). (6) addition between the addition and subtraction operations performed in 5.
   As a result of these steps, we get a contrast-enhanced image. Figures appear in the table of figures I are all examples of applying the 6 steps on the original image (a).
2) *Cytoplasm Separation*: After applying the previous steps, a separation of cytoplasm is executed to check if there is cytoplasmic vacuoles using flood fill image segmentation. This process starts by specifying a starting point in the image and then the method segments areas with similar intensity values. A sample result is shown in Fig. 9.



(a) Original (b) Gray Scale (c) Histogram Equalized Contrast (Enhanced) (d) Addition process (e) Subtraction process (f) Addition of (d) & (e)

TABLE I: A sample figure (a) goes on with the 6 steps.

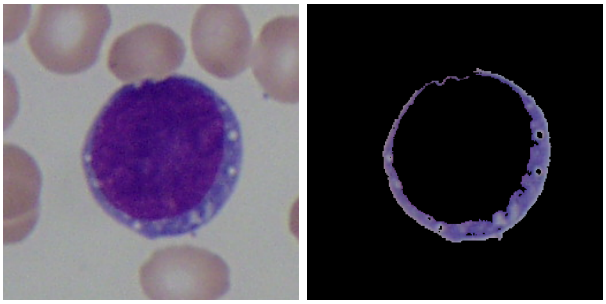


Fig. 9: (a) original image & (b) Separation of the cytoplasm.

3) *Morphological operations on segmented nuclei and cytoplasm*: Morphological operations on nuclei and cytoplasm are executed on the input image $C[1..X, 1..Y]$ to produce an output image $M[1..X, 1..Y]$ as follows:
   a) The Contrast manipulated image $C[1..X, 1..Y]$ is initially converted to a binary image using the Matlab

function $BlackAndWhite[1..X, 1..Y]$.

b) An erosion and dilation process are executed to remove the neglecable tiny objects in the image to reduce its noise by applying the following Matlab functions:

i) $erode = imerode(BlackAndWhite, se)$

ii) $dilate = imdilate(erode, se)$

where $es$ is the amount of erosion used to separate the cells. The results of morphological operations are shown in Fig. 10.

Fig. 10: (a) binary image of nuclei & (b) the same image after morphological operations.

*2) Features Extraction:* After applying necessary image pre-processing to separate nuclei from the cytoplasm and after executing morphological operations, features extraction operations are executed in order to go on with the categorization process and determine the absence or presence of Leukemia. The following steps are executed:

1) *Nuclei features*: We relied on the visual features that appear in the cell to determine acute Leukemia based on the three previously mentioned subtypes *L1*, *L2* and *L3*. We will examine the most outstanding feature of each type: similarity of cells(uniform cells) for *L1*, the existence of prominent nucleoli inside the nucleus there are prominent nucleoli inside the nucleus for *L2*, and check cytoplasmic vacuoles for *L3*. Therefore, we handled three different cases:

a) *Roundness and Area check*: The first test is to check the *L1* type as a *uniform cells* using the morphological processed image. The idea is to search for other similar spots taking into account the other two characteristics: *roundness* and *similarity in size*. Initially, the code handles the image resulting from the morphological operations, starting with *roundness* check for each cell using threshold 0.8 to decide whether the cell is a circle or not by applying equation 1. So, if there is a group of cells that has roundness values of more than 0.8 the roundness check decides them as similar circles. Fig. 11 below depicts the idea.

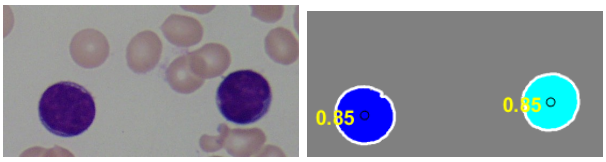



Fig. 11: (a) The original & (b) The *roundness* check images.

After checking the *roundness* feature, the code checks the *area* feature. This is done by comparing the area of the region under check with the area of the last region found and decided as a cell. A threshold value of 1000 pixels is used for this check so that the amount of area difference between two regions is less or equal to 1000 pixels, then the checked regions are considered as similar area cells. Finally, if both features are set, the sample is considered as an infected with *L1* type. The following pseudocode illustrates the previous steps where the code initiates the checking process by executing the function *checkShape* that takes the morphological image variable *Image* as a parameter. The code then executes the member function *Image.getCells()* that returns the list of cells in the image to be parameterized to the function *checkRound* that checks the amount of roundness for each cell (threshold = 0.8).

```
function checkShape(morphImage Image[][]){
    define result[] = checkRound(Image.getCells());
    define size = checkArea(result);

    if(result.size()>1 and size>1) return true;
        else return false;
}
```

b) *Searching for enclosed regions*: the second test is to check if there are prominent nucleoli inside the nucleus in order to verify the second type *L2* of acute Leukemia. Nuclei appear as gaps or holes inside the cell being checked (the human eye). The checking process starts by searching for any white spots in the cells. If found, the code selects an arbitrary white pixel in a given spot and starts the checking process by selecting the 8 surrounding pixels of the selected pixel to form the initial checking region. The code then gradually increases the size of the checked region by taking into account all surrounding pixels (up to 10 in all directions) and checks if these pixels form a closed region to decide if it finds nuclei or not. If yes, the color of the white pixels is converted to red (in the spot being checked) to be distinguished from the green background. If the checked pixels did not form a closed region, then the color of these pixels is converted to green as a sign that they are not nuclei. If the red pixels are greater than 20, then the code decides they are forming a nucleus. The numbers fixed in this code are considered as thresholds found after we executed experimental tests on a set of images. Fig. 12 shows the effect of applying the code on an original image (left) and how the white spots are converted to red (right).

The following pseudocode represents the previous functionality in checking the presence of nuclei. The code starts by executing the function *checkStatus* that accepts the *GapsDetectImage* array which contains the binary data for the image being tested. The variable *countRed* is used to count the number of pixels that are converted to red color. After that, the code checks every pixel in the array by checking the

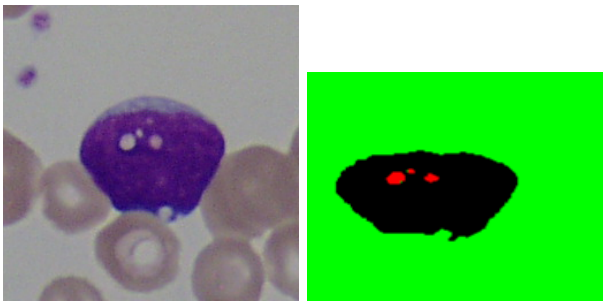


Fig. 12: (a) original & (b) nuclei detection images.

color of the current pixel whether it is white or not. If yes, the code executes the function *CheckGaps* that gradually checks the existence of white pixels around the current white one. The check of white pixels goes as follows: The 8 pixels around the current one are checked whether they contain black pixels or not. If not, the 8 pixels above, under, to the right, and to the left of the current pixel are also checked if they count 10 or more white pixels. If yes, the code moves to the upper, lower, left, and right pixels of the current one and repeats the same check for all of these pixels.

```
function checkStatus(GapsDetectImage[][]){
   define countRed=0;
   for (i=1 to GapsDetectImage.rows){
       for (j =1 to GapsDetectImage.columns){
           if GapsDetectImage[i][j] == 255{
               define outletFound = true;
               outletFound = CheckGaps(
                    GapsDetectImage,i,j)
               if  outletFound == true
                   GapsDetectImage[i][j].color = "
                        green";
               else{
                   GapsDetectImage[i][j].color = "
                        red";
                   countRed++;
               }
            }
       }
   }
   if countRed > 20
       return true; else return false;
}

function CheckGaps(image,rowPixel,columnPixel){
    define stageNumPixcel = image[i][j]*8; defect
        =0; outletFound=0;
    for (i=rowPixel-stage; i<=rowPixel+stage; i=i
        +8){
       for (j =columnPixel-stage; j<=columnPixel+
            stage; j=j+8){
           if (i or j) within stage
               if image[i][j].color== "black"
                       defect++;
               else outletFound=1;
           if stage < 10
               stage++;
       }
   }
   return defect;
}
```

c) *Cytoplasm Check*: the third test is to check the cytoplasmic vacuoles *L3* type considering the image of cytoplasm after it is being converted to binary format and the code then fills the holes that might appear in the image. As a result, all cells with their appearing holes become black. The code then tracks the original binary image by comparing the white pixels that appear in the cells with the same positions as the pixels of the converted one to fill image holes. If there are white pixels, then they are converted to red ones. Fig. 13 depicts the idea.

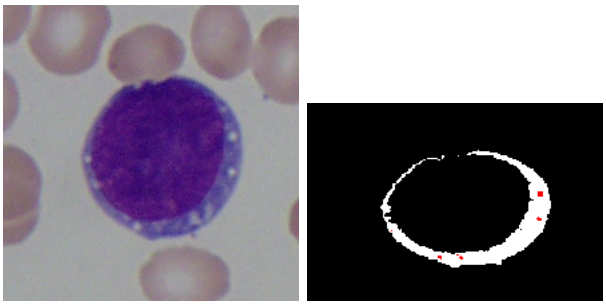


Fig. 13: (a) original & (b) cytoplasm vacuoles detection images.

The following pseudocode illustrates this check. The code starts by executing the function *cytoplasmCheck* that takes *Image* as a parameter. The array *outputImage* is then defined to save the image result from this code. The function *fillHoles* is then executed to fill the possible holes in the image and after the conversion, all not black pixels are converted to red ones. At the end of the code, the function *checkStatus* discussed in the second test is executed.

```
function cytoplasmCheck(binaryImage Image[][]){
     def outputImage[][];
     def afterFillHoles = fillHoles(Image);
     afterFillHoles.convertToRGB();
     for(i as rows)
         for(j as columns)
             if(Image[i][j] == afterFillHoles[i
                 ][j] == "black") outputImage[i
                 ][j]="black";
             else outputImage[i][j]= "red";
     checkStatus(outputImage);
}
```

## IV. Experimental Tests

The main purpose of the conducted experimental tests is to compare between the two implemented algorithms: **CKC** and **ALLS**. The comparison takes into account the following measures: *Precision*, *Recall*, *Accuracy*, *F-measure* and *Speed*. Table II below reflects the results of this comparison. It is clear that **ALLS** outperforms **CKC** for that values of *Recall*, *Accuracy* and *Speed*, while both nearly equal in *Precision* and *F-measure*. Despite that both algorithms are involved image processing techniques, **ALLS** has better performance (in general) than **CKC**.

TABLE II: Comparison results of the two algorithms.

| **Measurements** | **Algorithm Name** | |
|---|---|---|
| | **ALLS** | **CKC** |
| **Precision** | 0.90716 | 0.910714 |
| **Recall** | 0.98387 | 0.809524 |
| **Accuracy** | 0.93103 | 0.822917 |
| **F-measure** | 0.94 | 0.953271 |
| **Speed** | 7.5 minutes | 10 minutes |

The amount of accuracy for the two algorithms is also measured using three different classifiers: *Random Forest*, *KStar* and *REPTree*. We found that **ALLS** has in general more accuracy than **CKC** as Table III indicates.

TABLE III: Algorithms Accuracy.

| Classifier | Algorithm | |
|---|---|---|
| | ALLS | CKC |
| **RandomForest** | 93.9% | 83.5% |
| **KStar** | 93.1% | 86.1% |
| **REPTree** | 90.5% | 78.6% |

Moreover, we relied on **Weka** to compute the set of previous measures as well as additional ones (*TP*: True Positive, *FP*: False Positive, *MCC*: Quality of Binary Classifications, *ROC*: Receiver Operating Characteristics, *PRC*: Precision Recall Area, *TN*: True Negative, and *FN*: False Negative). To do so, we fed the software with the set of features of both algorithms and used **Weka** to compute the measures that appear in Tables IV and V. If we compare the values in both tables as a whole, we can conclude that **ALLS** performs better than **CKC**, in which this compiles with the same conclusion we got by our implementation and conducted results we did and reflected in Tables II and III. We can see that the values of *Precision*, *Recall*, *TP*, *F-Measure*, *MCC*, *ROC*, *TN* and *PRC* are all with high values, while the values of *FP* and *FN* measures are very low.

*ROC* measure is of one of the important **Weka** measures that gives an idea of how the classifiers are performing in general. However, Figures 14 and 15 depict the visualized *ROC* curves of the highest accuracy classifier for each algorithm. From the images, we can conclude that the classifier performance is very good.

Although comparing **ALLS** and **CKC** with other image processing Leukemia detection-based algorithms is important, the comparison must take into account the same sample images in order to have accurate comparison. However, and for clarification purposes, Table VI compares **ALLS** and **CKC** with other 3 algorithms in terms of accuracy, Recall and Precision where **ALLDSVM**, **ALLDDNN** and **ALLKNN** are acronyms for Acute Lymphoblastic Leukemia Detection using Support Vector Machine, Acute Lymphoblastic Leukemia Detection using Deep Neural Network and Acute Lymphoblastic Leukemia Detection using K Nearest Neighbor, respectively [4].

TABLE IV: Weka Results for CKC.

| Measurements | REP Tree | K-Star | Random Forest |
|---|---|---|---|
| **Precision** | 0.829 | 0.904 | 0.864 |
| **Recall** | 0.899 | 0.912 | 0.925 |
| **TP Rate** | 0.899 | 0.912 | 0.925 |
| **FP Rate** | 0.545 | 0.286 | 0.429 |
| **F-Measure** | 0.863 | 0.908 | 0.894 |
| **MCC** | 0.391 | 0.632 | 0.539 |
| **ROC Area** | 0.830 | 0.893 | 0.825 |
| **PRC Area** | 0.920 | 0.949 | 0.901 |
| **Confusion Matrix** | | | |
| **TP** | 204 | 207 | 210 |
| **FP** | 42 | 22 | 33 |
| **TN** | 35 | 55 | 44 |
| **FN** | 23 | 20 | 17 |

TABLE V: Weka Results for ALLS.

| Measurements | REP Tree | K-Star | Random Forest |
|---|---|---|---|
| **Precision** | 0.982 | 0.983 | 0.984 |
| **Recall** | 0.848 | 0.894 | 0.909 |
| **TP Rate** | 0.848 | 0.894 | 0.909 |
| **FP Rate** | 0.020 | 0.020 | 0.020 |
| **F-Measure** | 0.911 | 0.937 | 0.945 |
| **MCC** | 0.821 | 0.866 | 0.882 |
| **ROC Area** | 0.902 | 0.906 | 0.912 |
| **PRC Area** | 0.928 | 0.907 | 0.941 |
| **Confusion Matrix** | | | |
| **TP** | 56 | 59 | 60 |
| **FP** | 1 | 1 | 1 |
| **TN** | 49 | 49 | 49 |
| **FN** | 10 | 7 | 6 |

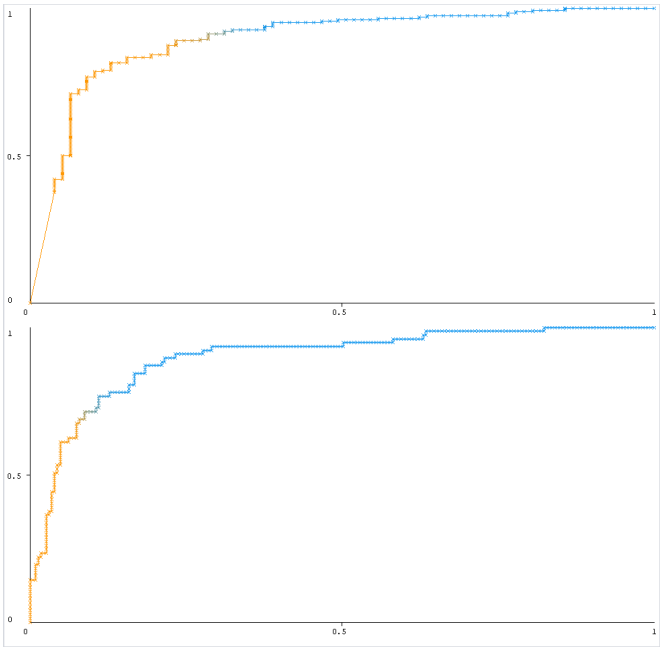


Fig. 14: (a) CKC ROC curve of defected cells. & (b) CKC ROC curve of normal cells.


## ACKNOWLEDGEMENT

The authors of the work would like to thank both Dr. Majdi Dwaikat and Dr. Fadi Draidi (An-Najah National University)


[4]We would like to draw the reader's attention that the values of the compared algorithms are taken from more than one work, each of which has its own sample images.

TABLE VI: Comparison between ALLS and CKC with other 3 algorithms.

| Algorithm | Accuracy | Recall | Precision |
|---|---|---|---|
| **ALLS** | 93% | 98% | 90% |
| **CKC** | 92% | 80% | 91% |
| **ALLDSVM** | 99.5% | 8.89% | 87.9% |
| **ALLDDNN** | 93% | 91% | 98.5% |
| **ALLKNN** | 82% | 96% | 96% |

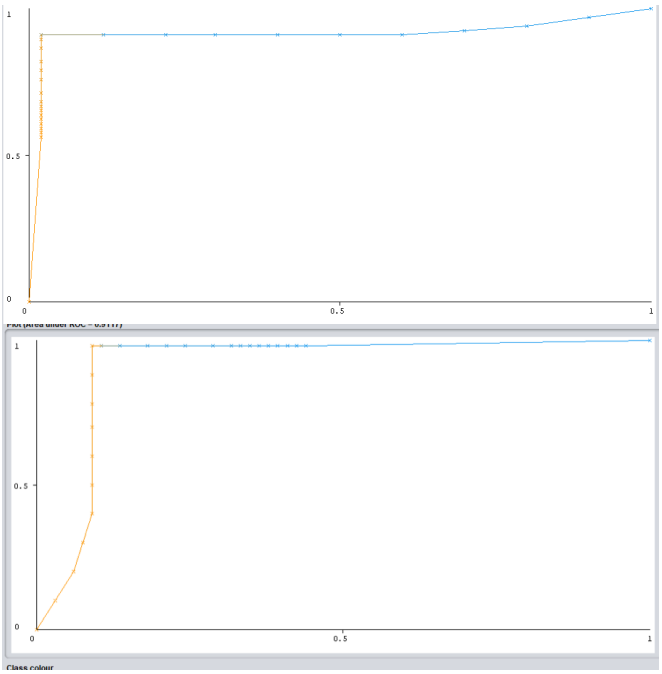


Fig. 15: (a) ALLS ROC curve of defect cells & (b) ALLS ROC curve of normal cells.

for their efforts and valuable information about the expert procedures in detecting Leukemia disease implemented and tested in this work and for the directions in implementing some image processing techniques.

## V. CONCLUSION

Early detection of the Leukemia disease allows for quicker action with respect to patients. Image processing techniques are one of the methods used to detect Leukemia from images. This work investigates two Leukemia detection algorithms based on image processing techniques. Several image processing steps are implemented for both algorithms. The conducted experimental tests indicate that **ALLS** outperforms **CKC** in the most of measures executed. **Weka** software is also used as a part of the experimental tests in order to be sure of the results we gained from our own experimental tests.

For future works, authors are planning to investigate more image processing algorithms to study their performance and their accuracy like Marr–Hildrethm, Canny edge detector, and Hough transform algorithms.

## REFERENCES

[1] D. D. Nawgaje and R. D. Kanphade, "Implimentation of adaptive neuro fuzzy inference system for cancer detection using tms320c6711 dsp," in *Proceedings of the International Conference & Workshop on Emerging Trends in Technology*, ICWET '11, (New York, NY, USA), p. 608–612, Association for Computing Machinery, 2011.

[2] S. Shafique and S. Tehsin, "Acute lymphoblastic leukemia detection and classification of its subtypes using pretrained deep convolutional neural networks," *Technology in Cancer Research & Treatment*, vol. 17, 2018.

[3] N. Patel and A. Mishra, "Automated leukaemia detection using microscopic images," *Procedia Computer Science*, vol. 58, pp. 635–642, 2015. Second International Symposium on Computer Vision and the Internet (VisionNet'15).

[4] K. Anilkumar, V. Manoj, and T. Sagi, "A survey on image segmentation of blood and bone marrow smear images with emphasis to automated detection of leukemia," *Biocybernetics and Biomedical Engineering*, vol. 40, no. 4, pp. 1406–1420, 2020.

[5] A. Kumar and F. Shaik, *Importance of Image Processing*, pp. 5–7. Singapore: Springer Singapore, 2016.

[6] Y. M. Y. Abdallah and T. Alqahtani, "Research in medical imaging using image processing techniques," in *Medical Imaging* (Y. Zhou, ed.), ch. 5, Rijeka: IntechOpen, 2019.

[7] K. Jewani, K. Boddu, P. Gumani, and K. Solapure, "Detection of diseases via blood analysis using image processing techniques," in *2018 International Conference on Smart City and Emerging Technology (ICSCET)*, pp. 1–4, 2018.

[8] N. Safca, D. Popescu, L. Ichim, H. Elkhatib, and O. Chenaru, "Image processing techniques to identify red blood cells," in *2018 22nd International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 93–98, 10 2018.

[9] B. Rajithkumar, D. Shilpa, and B. Uma, "Detection of blood-related diseases using deep neural nets," in *Handbook of Research on Deep Learning Innovations and Trends*, pp. 1–15, IGI Global, 2019.

[10] S. S. Agaian, K. Panetta, S. C. Nercessian, and E. E. Danahy, "Boolean derivatives with application to edge detection for imaging systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, pp. 371–382, 2010.

[11] S. Mohapatra and D. Patra, "Automated cell nucleus segmentation and acute leukemia detection in blood microscopic images," *2010 International Conference on Systems in Medicine and Biology*, pp. 49–54, 2010.

[12] X. Zheng, Y. Wang, G. Wang, and J. Liu, "Fast and robust segmentation of white blood cell images by self-supervised learning," *Micron*, vol. 107, pp. 55–71, 2018.

[13] F. Scotti, "Robust segmentation and measurements techniques of white cells in blood microscope images," in *2006 IEEE Instrumentation and Measurement Technology Conference Proceedings*, pp. 43–48, 2006.

[14] V. Piuri and F. Scotti, "Morphological classification of blood leucocytes by microscope images," in *2004 IEEE International Conference onComputational Intelligence for Measurement Systems and Applications, 2004. CIMSA.*, pp. 103–108, 2004.

[15] S. Mohapatra, D. Patra, and S. Satpathy, "An ensemble classifier system for early diagnosis of acute lymphoblastic leukemia in blood microscopic images," *Neural Computing and Applications*, vol. 24, 06 2013.

[16] R. K. Meleppat, C. Shearwood, L. K. Seah, and M. V. Matham, "Quantitative optical coherence microscopy for the in situ investigation of the biofilm," *Journal of Biomedical Optics*, vol. 21, no. 12, pp. 1 – 9, 2016.

[17] K. M. Ratheesh, L. K. Seah, and V. M. Murukeshan, "Spectral phase-based automatic calibration scheme for swept source-based optical coherence tomography systems," *Physics in Medicine and Biology*, vol. 61, pp. 7652–7663, oct 2016.

[18] K. M. Ratheesh, P. Prabhathan, L. K. Seah, and V. M. Murukeshan, "Gold nanorods with higher aspect ratio as potential contrast agent in optical coherence tomography and for photothermal applications around 1300 nm imaging window," *Biomedical Physics & Engineering Express*, vol. 2, p. 055005, sep 2016.

[19] R. D. Labati, V. Piuri, and F. Scotti, "All-idb: The acute lymphoblastic leukemia image database for image processing," *2011 18th IEEE International Conference on Image Processing*, pp. 2045–2048, 2011.

[20] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. USA: Prentice-Hall, Inc., 2006.

[21] S. Kumar, S. Mishra, P. Asthana, and Pragya, "Automated detection of acute leukemia using k-mean clustering algorithm," *CoRR*, vol. abs/1803.08544, 2018.

[22] R. Bagasjvara, I. Candradewi, S. Hartati, and A. Harjoko, "Automated detection and classification techniques of acute leukemia using image processing: A review," in *2016 2nd International Conference on Science and Technology-Computer (ICST)*, pp. 35–43, 2016.