Digital Image Processing Techniques for Leukemia Detection

Mukesh Kumar Saini[1], Arun Saini[2], Sachin Gupta[3]

[1]PhD, MBA, MCA, Technologist

[2]MTech, Consulting Engineer

[3]MCA, Sr. Specialist Engineer

Abstract: ***This paper presents a methodology for the detection of leukemia using various digital image processing techniques. By identifying and analyzing red blood cells and immature white cells, diseases such as anemia, leukemia, malaria, and vitamin B12 deficiency can be diagnosed. The objective is to detect and quantify leukemia- affected cells, determining whether the condition is chronic or acute. Techniques such as histogram equalization, linear contrast stretching, morphological operations (area opening, area closing, erosion, dilation), watershed transform, K- means clustering, and shape-based features are employed. The accuracy of these methods is 72.2%, 72%, 73.7%, and 97.8%, respectively.***

Keywords: Leukemia detection, image processing, K-means clustering, watershed transform, histogram equalization, shape-based features, red and white cell counting

1. Introduction

There are various types of white blood cells in our body, and leukemia is a type of cancer that affects these cells. In leukemia, the number of white blood cells increases significantly, but these cells are immature and harmful, destroying other healthy cells. Diagnosing leukemia through laboratory tests currently takes a considerable amount of time, is prone to human error, and is often a tedious process.

In a healthy individual, the ratio of white blood cells to red blood cells is approximately 1:1000. This means there is one white blood cell for every thousand red blood cells. When the number of white blood cells increases significantly, it can indicate leukemia, which can be classified into two main types: acute and chronic. Both types involve the proliferation of immature white blood cells that damage the body's healthy cells.

The goal is to detect these immature cells using advanced image processing techniques and accurately count the total number of cells. Utilizing technology to identify different types of blood cells quickly is crucial, especially in emergency situations. It is also essential to study and understand how to differentiate between various cells and recognize immature cells to diagnose leukemia accurately.

Acute and chronic leukemia can each be further divided into two subtypes:1. Lymphocytic and 2. Myeloblastic. These subtypes are caused by the proliferation of immature lymphoid and myeloid cells, respectively.

By employing biomedical image processing, the task of detecting immature cells and diagnosing leukemia can be significantly improved and performed in near-real time.

The task of detecting immature cells and diagnosing leukemia can be significantly improved and performed in a near-real-time environment using biomedical image processing techniques. Here formation of lymphoid and myeloid cells of series is shown in figure 1. Figure 1 has been taken from web sources [22].

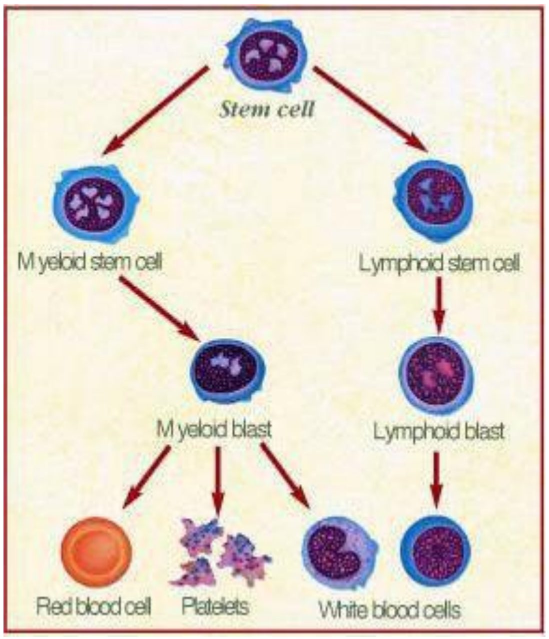


Figure 1: The Formation of Myeloid and Lymphoid Series of Cell

2. Rationale

Various methods have been developed and applied to automate the detection and counting of leukemia cells. These methods aim to improve the speed, accuracy, and efficiency of leukemia diagnosis, minimizing human error and reducing the time required for analysis. Here are some key techniques used.

2.1 Watershed Transform

The Watershed Transform is a powerful image segmentation technique used in various fields, including medical image analysis, to separate different objects within an image. It's particularly useful in situations where objects touch or overlap, such as identifying individual cells in a blood sample Lim Huey Nee et al. proposed methods for segmenting white blood cells using morphological operations, gradient magnitude, and watershed transformation. The process begins with image acquisition techniques, followed by segmentation to distinguish blast cells from the background. Initially, the RGB image is converted into the HSV color model, and the saturation component is extracted for further processing. The gradient magnitude of the saturation component is then calculated for edge detection. Sobel, Canny, and Prewitt operators are utilized for this purpose. After extracting the white cells and eliminating the background and red cells, dilation or erosion processes are performed. Finally, the watershed transform is applied to separate any connected cells. Thus, leukemic cell can be identified and this method gives very accurate result [2].

Steps in the Watershed Transform:

1) **Gradient Calculation:** Calculate the gradient of the image. This gradient image highlights the edges where there are significant changes in intensity, corresponding to the boundaries between different objects (e.g., between different cells).
2) **Marker-based Segmentation:** To improve the accuracy of the segmentation, marker-based Watershed Transform is often used. Markers are predefined points or regions that are known to be inside the objects of interest (foreground markers) and outside the objects (background markers). Markers can be obtained using techniques such as thresholding, morphological operations, or manual annotation.
3) **Flooding Process:** Starting from the markers, the algorithm simulates the flooding of the landscape. As the water level rises, it fills the basins two basins meet, a boundary (watershed line) is created to separate them.
4) **Resulting Segmentation:** The result is a segmented image where each object (e.g., each cell) is separated from the others by the watershed lines.

Application in Leukemia Cell Detection:

1) **Preprocessing:** Convert the blood sample image to grayscale if it is in color. Apply noise reduction techniques such as Gaussian blur to smooth the image and improve the accuracy of gradient calculation.
2) **Gradient Calculation:** Compute the gradient of the preprocessed image to highlight the edges of the cells.
3) **Marker Generation:** Use morphological operations to generate foreground markers (areas likely to be inside cells) and background markers (areas outside cells). For example, you can use morphological dilation and erosion to isolate the centers of cells and the background.
4) **Watershed Segmentation:** Apply the Watershed Transform using the gradient image and the markers. The algorithm will segment the image, separating individual cells based on the watershed lines.
5) **Post-processing:** Refine the segmented regions by removing small artifacts and merging fragmented parts of the same cell if necessary.

Count the number of segmented regions to determine the number of cells. Here's a simplified workflow to implement the Watershed Transform in Python using OpenCV:

```
Import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load the image
image = cv2.imread('blood_sample.png') gray =
cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Apply a Gaussian blur to reduce noise blurred =
cv2.GaussianBlur(gray, (5, 5), 0)

# Compute the gradient of the image gradient =
cv2.morphologyEx(blurred, cv2.MORPH_GRADIENT,
kernel=cv2.getStructuringElement(cv2.MORPH_E    LLIPSE,
(3, 3)))

# Apply thresholding to create binary image

_, binary = cv2.threshold(gradient, 0, 255,
cv2.THRESH_BINARY | cv2.THRESH_OTSU)

# Perform distance transform and normalize the result
dist_transform = cv2.distanceTransform(binary,
cv2.DIST_L2, 5)
_, sure_fg = cv2.threshold(dist_transform, 0.7 *
dist_transform.max(), 255, 0)
sure_fg = np.uint8(sure_fg)

# Create markers for watershed sure_bg = cv2.dilate(binary,
kernel=cv2.getStructuringElement(cv2.MORPH_E LLIPSE,
(3, 3)), iterations=3)
unknown = cv2.subtract(sure_bg, sure_fg)
_, markers = cv2.connectedComponents(sure_fg) markers =
markers + 1
markers[unknown == 255] = 0

# Apply watershed
markers = cv2.watershed(image, markers) image[markers
== -1] = [0, 0, 255]

# Display the result plt.imshow(cv2.cvtColor(image,
cv2.COLOR_BGR2RGB))
plt.title('Watershed Segmentation') plt.show()
```

In this example, the Watershed Transform is applied to a blood sample image to segment individual cells. The markers are created using distance transform and morphological operations, ensuring accurate cell separation and counting.

But the exact separation of cells cannot be done using this method. The process flow is shown in figure 2. Figure 2(a) has been taken from web sources [23].

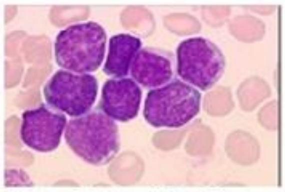


Figure 2(a) original image

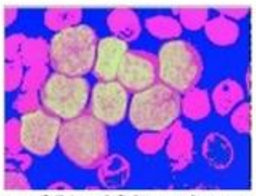


Figure 2(b) rgb2 hsv color model

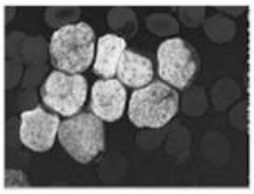


Figure 2(c) saturation component extracted

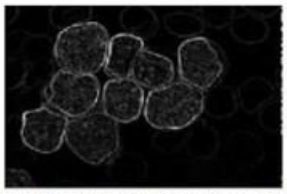


Figure 2(d) derivation of gradient magnitude

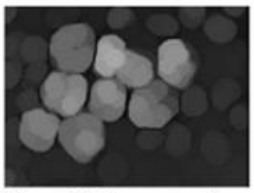


Figure 2(e) area opening: morphological operation

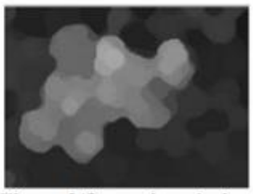


Figure 2(f) opening-closing :morphological operation

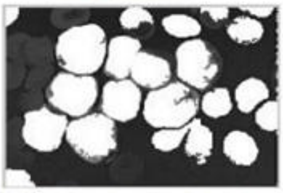


Figure 2(g) regional maxima superimposed on original image

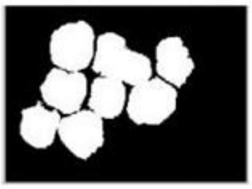


Figure 2(h) thresholded opening closing by reconstruction

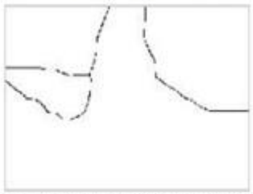


Figure 2(i) watershed ridge lines

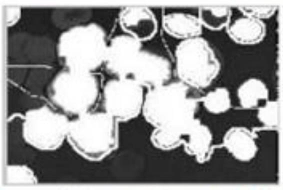


Figure 2(j) marker and object boundaries superimposed on original image

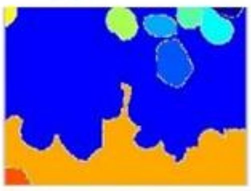


Figure 2(k) detect white cell using colored watershed label matrix

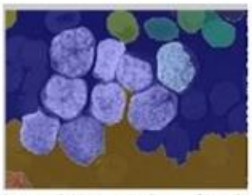


Figure 2(l) detection of white cell superimposed transparency on original image

2.2 K Means Clustering Technique

The K-Means Clustering Technique is a popular unsupervised machine learning algorithm used for partitioning a dataset into a specified number of clusters (K). Here's a breakdown of how the technique works and its steps:

1) **Initialization of K**: The algorithm starts by selecting K initial centroids, which represent the centers of the clusters.
2) **Assignment**: Each data point in the dataset is assigned to the nearest centroid based on a distance metric, often using the Euclidean distance.
3) **Update Centroids**: The algorithm calculates the new centroid for each cluster based on the mean of the data points assigned to that cluster. This often involves recalculating the centroid based on the new cluster assignments.
4) **Repeat**: The process of assigning data points to the nearest centroid and recalculating centroids continues iteratively until convergence, which usually means that the centroids have stabilized, and no further changes occur.
5) **Convergence Criteria**: The algorithm stops when the centroids no longer change significantly or a specified number of iterations have been completed.
6) **Result**: The output is a set of clusters, where each data point belongs to one of the K clusters.

Applications:

1) **Image Segmentation**: Used in image processing tasks to segment images into different regions.
2) **Customer Segmentation**: Helps businesses categorize customers into groups based on behavior and preferences.
3) **Document Clustering**: In natural language processing, K-Means can be used to group similar text documents.

This technique is efficient and straightforward, making it ideal for clustering applications where the number of clusters is known or can be reasonably estimated.

Mashiat Fatma and Jaya Sharma proposed the method of clustering techniques to identify the abnormalities in blood cells or to identify the lymphoblast, Initially, image pre-processing and feature extraction are performed to provide valuable insights into the image. Pre-processing primarily focuses on image enhancement and does not yield essential image-specific information. Consequently, acquisition is necessary, and contrast enhancement is employed to clarify the image. Subsequently, the RGB image is transformed into the HSI (Hue, Saturation, Intensity) color model, and the K-Means clustering technique is utilized for segmentation.

For example, K-means clustering on a synthetic dataset of blood cell features.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

# Generate synthetic blood cell data (features)
# This is just a simulation; real data would be more complex
and obtained differently np.random.seed(42)
n_samples = 300
n_features = 2  # Example features for blood cells: size and
shape or other metrics
n_clusters = 3  # Example number of clusters for different
cell types

X,        _        =        make_blobs(n_samples=n_samples,
n_features=n_features, centers=n_clusters, random_state=42)

# Apply K-Means clustering
kmeans = KMeans(n_clusters=n_clusters) kmeans.fit(X)

# Get the cluster labels and centroids labels = kmeans.labels_
centroids = kmeans.cluster_centers_

# Plot the data points with their respective clusters
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', s=50,
alpha=0.7, edgecolor='k')

# Plot the centroids
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=200,
alpha=0.7, label='Centroids', edgecolor='k')

plt.title('K-Means Clustering of Synthetic Blood Cell
Features')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2') plt.legend()  plt.show()
```

Explanation of the Code:

1) **Generating Synthetic Data:** I generate synthetic data using make_blobs from sklearn.datasets. This simulates clusters that represent normal and abnormal blood cells.
2) **K-Means Clustering:** KMeans from sklearn.cluster is used to fit the data, determining the clusters.
3) **Plotting:** We plot the data points and centroids, using a colormap to differentiate clusters.

This approach can be adapted or extended to real blood cell datasets by replacing the synthetic data generation step with loading actual image or feature data from cytometry or similar sources.

A median filter is also applied to eliminate noise from the image. Following feature extraction, image classification is carried out using clustering techniques, with options for two approaches: 1) supervised learning and 2) unsupervised learning. The goal of feature extraction is to distinguish white cells or lymphoblasts within the image. So some features are used [3].

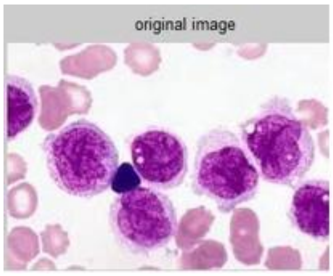


Figure 3 (a): Original Image

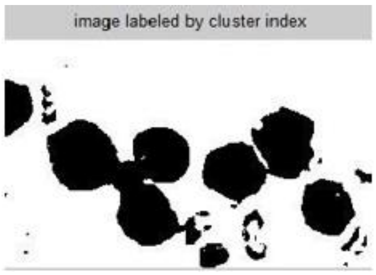


Figure 3 (b): Image labeled by cluster index

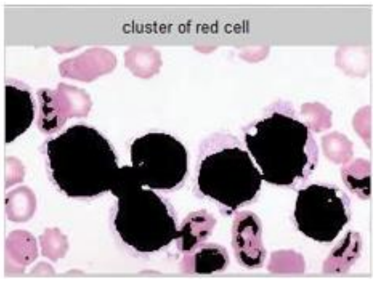


Figure 3 (c): Cluster of red cells

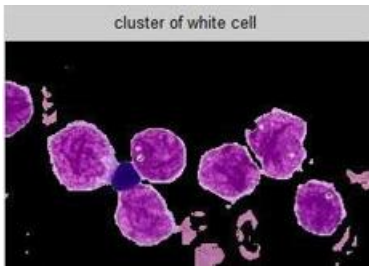


Figure 3 (d): Cluster of white cells

Figure 3: Process Flow for K means Clustering

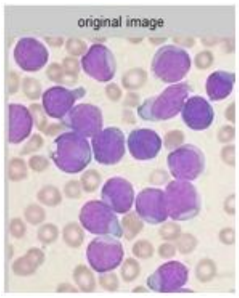


Figure 4(a): Original image

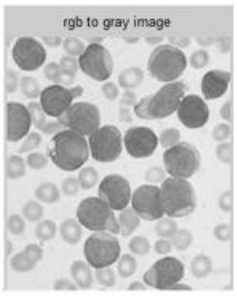


Figure 4 (b): rgb2 gray

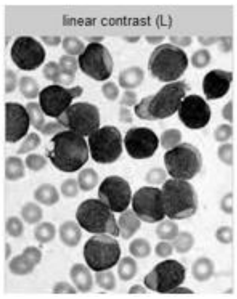


Figure 4 (c): Linear contrast

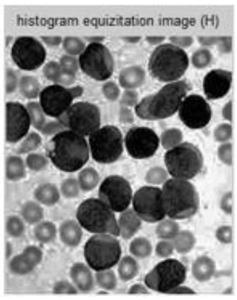


Figure 4 (d): Histogram equalization

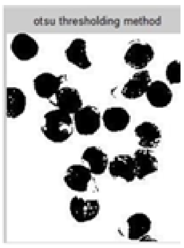


Figure 4(e): addition of 4 (c) and 4(d)

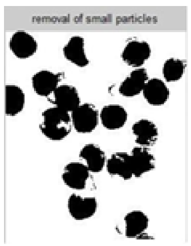


Figure4 (f): Subtraction of 4(c) and 4(d)

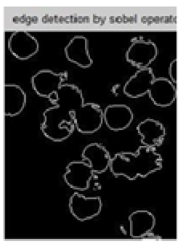


Figure4(g): Addition of fig, 4(e) and 4(f)

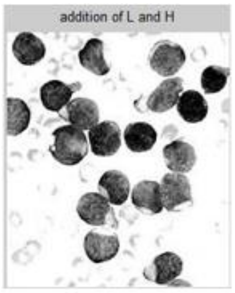


Figure 4(h): Thresholding method

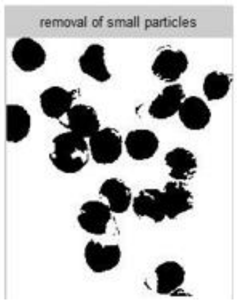


Figure 4(i) Removal of small particles

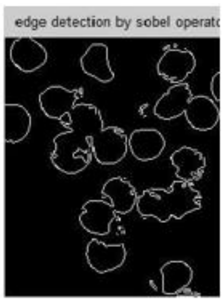


Figure 4(j) Edge detection using sobel operator

Figure 4: Process flow for histogram equalization and linear contrast stretching

# Display or save the result cv2.imshow('Equalized Image', img_equalized) cv2.waitKey(0)
cv2.destroyAllWindows()

Linear Contrast Stretching: This method enhances the contrast of the image by stretching the range of intensity values from the original minimum and maximum values to a new range. For example, if your original intensity range is [min_intensity, max_intensity], you can stretch it to [0, 255] for an 8-bit image.

- Compute the minimum and maximum intensity values of the image.
- Apply a linear transformation to stretch the intensity values to the desired range.

Python code snippet for linear contrast stretching: *# Calculate minimum and maximum intensity values*
min_intensity = np.min(img)
max_intensity = np.max(img) # Linear contrast stretching
*img_stretched = 255 * (img - min_intensity) / (max_intensity - min_intensity)*
img_stretched = np.clip(img_stretched, 0, 255).astype(np.uint8)
# Display or save the result cv2.imshow('Stretched Image', img_stretched) cv2.waitKey(0)
cv2.destroyAllWindows()-

Thresholding: After applying histogram equalization and/or contrast stretching, you can apply a thresholding technique to segment the white blood cells from the background. Thresholding sets all pixel values above a certain threshold to white (255) and all others to black (0), creating a binary image where white cells are highlighted.

- Choose an appropriate threshold value based on the histogram or experimentation.
- Apply the threshold to the image to create a binary mask.

Python code for thresholding
# Apply thresholding
_, thresh = cv2.threshold(img_equalized, thresh_value, 255, cv2.THRESH_BINARY)

# Display or save the thresholded image cv2.imshow('Thresholded Image', thresh) cv2.waitKey(0)
cv2.destroyAllWindows()

By combining these techniques (grayscale conversion, histogram equalization, contrast stretching, and thresholding), you can enhance the visibility of white blood cells in your images and facilitate their detection in subsequent image analysis tasks. Adjusting parameters such as the threshold value in thresholding or the transformation parameters in histogram equalization and contrast stretching may be necessary based on the specific characteristics of your images.

2.4 HSI Color Model based Segmentation

The HSI color model represents colors in terms of three components:

- **Hue (H):** The type of color such as red, green, blue, etc., represented as an angle in a color wheel.
- **Saturation (S):** The intensity or purity of the color, ranging from fully saturated (pure color) to unsaturated (gray).
- **Intensity (I) or Value (V):** The brightness of the color, ranging from dark (black) to light (white).

For segmentation of WBC and nucleus, linear contrast stretching techniques is used and for color-based segmentation HSI color model is used. For further segmentation, K means clustering techniques is used. For fully segmented WBC, extract the H component and to segment nucleus S component is necessary. For further segmentation, k means clustering is used. Then unwanted object or noise from image are removed using median filter. Thus, cytoplasm and nucleus can be extracted from image [15].

Steps for HSI Color Model Based Segmentation

Convert Image to HSI Color Space: Convert the RGB image to the HSI color space. In OpenCV, the HSI color space is represented as HSV (Hue, Saturation, Value), where Value corresponds to Intensity.

```
import cv2
import numpy as np # Load an image
img = cv2.imread('your_image.jpg') # Convert to HSV color space
img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

Thresholding Based on Intensity (Value):
White blood cells typically appear bright in intensity compared to the background. Therefore, we can use the Value (V) channel of the HSV image for thresholding.

```
# Extract the Value channel (Intensity) intensity_channel =
img_hsv[:, :, 2]
# Define a threshold to segment bright regions (white cells)
_, thresh = cv2.threshold(intensity_channel, thresh_value, 255,
cv2.THRESH_BINARY)
# Optionally, apply morphological operations to enhance the
segmentation
kernel = np.ones((5, 5), np.uint8) thresh =
cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel)
```

Adjust thresh_value according to the intensity characteristics of your image. This thresholding operation will create a binary mask (thresh), where white pixels correspond to regions that likely contain white blood cells.

Considerations

- **Threshold Selection**: The choice of thresh_value in step 2 is critical and should be adjusted based on the intensity distribution of your specific images.
- **Morphological Operations**: These can help in smoothing the segmented regions and connecting nearby pixels to form cohesive white blood cell regions.
- **Validation**: Always validate the segmentation results visually or using other metrics to ensure the accuracy of the detected white blood cells.

By leveraging the HSI color model and focusing on the intensity (Value) component for segmentation, you can effectively detect white blood cells in images. Adjusting parameters and applying appropriate post-processing techniques will help refine and improve the segmentation results based on the characteristics of your images.

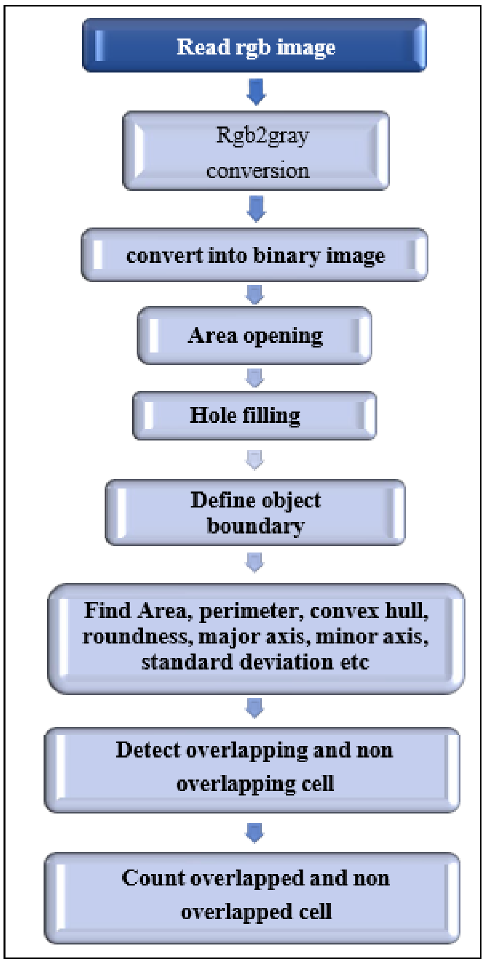
Process flow is shown in figure 5. Figure 5(a) has been taken from web sources [23]

3. Methodology

After gaining knowledge of these techniques, it has been determined that shape-based features contribute significantly to achieving better results and accuracy. These features are employed to identify various shapes such as circles, rectangles, ellipses, and squares. Given that blood cells exhibit diverse sizes and shapes, employing shape-based features proves invaluable for detecting the geometric shapes of cells.

In Section 3.1, an algorithm is presented for performing various image processing operations on images depicting leukemia. This algorithm proves beneficial for detecting the number of overlapping and non-overlapping cells, as well as for counting red and white cells. Initially, the RGB image is converted to grayscale to reduce its dimensionality. Subsequently, the image undergoes thresholding to convert it into a binary format for more precise analysis; Otsu's method is particularly effective for this purpose. Following thresholding, several morphological operations such as area opening, closing, erosion, and dilation are applied. Area opening eliminates connected components, dilation adds pixels to object boundaries, and erosion removes pixels from object boundaries. Once the object boundaries are detected, a hole filling operation is performed to accurately identify intact cells.

3.1 Algorithm to Count the Cells


Read rgb image
Rgb2gray conversion
convert into binary image
Area opening
Hole filling
Define object boundary
Find Area, perimeter, convex hull, roundness, major axis, minor axis, standard deviation etc
Detect overlapping and non overlapping cell
Count overlapped and non overlapped cell

According to the values of major axis and minor axis, it is possible to detect the number of overlapping and non-overlapping cells. This detection process facilitates accurate cell counting.

Figure 6(a) displays the original image, while Figure 6(b) shows its grayscale version. Figure 6(c) presents the binary image derived from the original, followed by Figure 6(d) where area opening has been applied. Figure 6(e) illustrates the hole filling operation, and Figure 6(f) delineates the boundaries of each object. Figure 6(g) highlights overlapped cells with major axis values. Figures 6(h) and 6(i) depict the radius and roundness of cells, respectively. Figures 6(j), 6(k), and 6(l) show features such as centroid, standard deviation, and bounding box of cells.

This algorithm was implemented using MATLAB. MATLAB's 'regionprops' function was employed to extract properties such as area, roundness, centroid, major axis, minor axis, standard deviation, etc.

In the analysis, the total number of actual objects or cells (including red and white cells) is 92. Among these, 68 are identified as non-overlapping cells and 11 as overlapping cells. It is noted that overlapped cells count as two instances each, hence (11 * 2 = 22) cells are overlapped. Thus, according to the algorithm, the total number of cells detected is 90. This result is highly accurate, closely matching the actual count of 92. This demonstrates that shape-based features provide a high level of accuracy, achieving 97.8% accuracy as previously stated.

Therefore, it can be concluded that shape-based features are indeed more accurate for cell counting and analysis, as evidenced by this algorithm's performance.

3.2 Experimental Results

After that hole filling operation, boundary is detected around the cells. To detect object boundaries many operators are used like sobel, prewitt, canny, etc. after define the boundary of object, many shape-based features, like major axis, minor axis, area, perimeter, standard deviation, radius, roundness can be found. Radius, roundness, standard deviation can be found by given formula:

$$\text{Radius} = \frac{\text{Roundness}}{\text{Standard Deviation}}$$

Where X= (Mean)

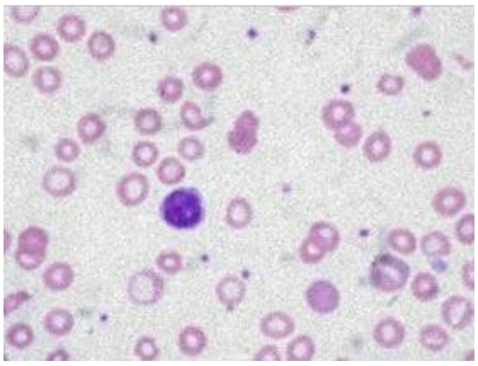


Figure 6 (a): original image

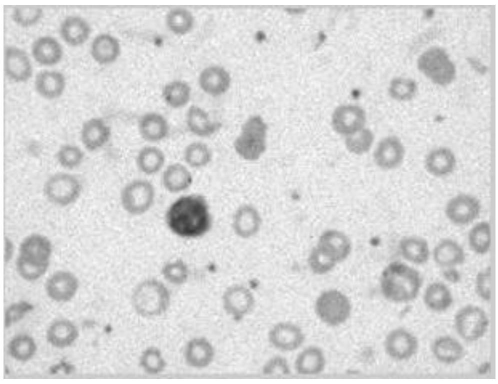


Figure 6 (b): rgb2gray

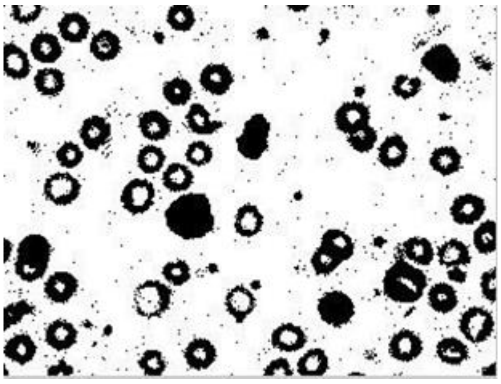


Figure 6(c): Thresholding

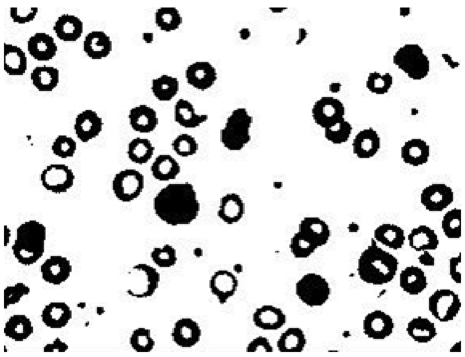


Figure 6 (d): Area Opening

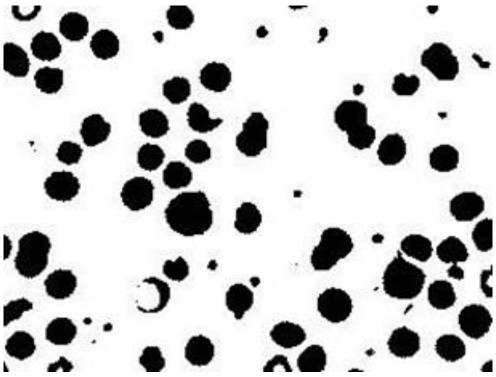


Figure 6 (e): Hole Filling

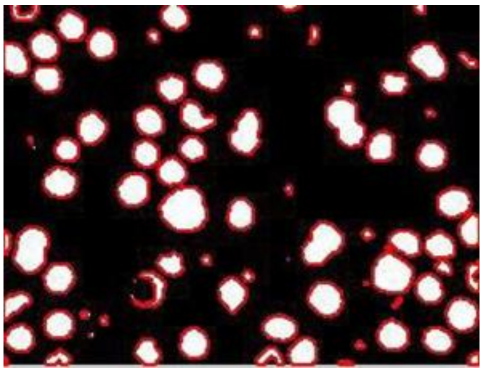


Figure 6 (f): Detected boundary

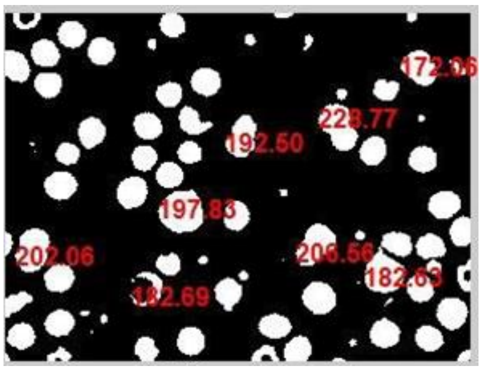


Figure 6 (g): Detected overlapping cell

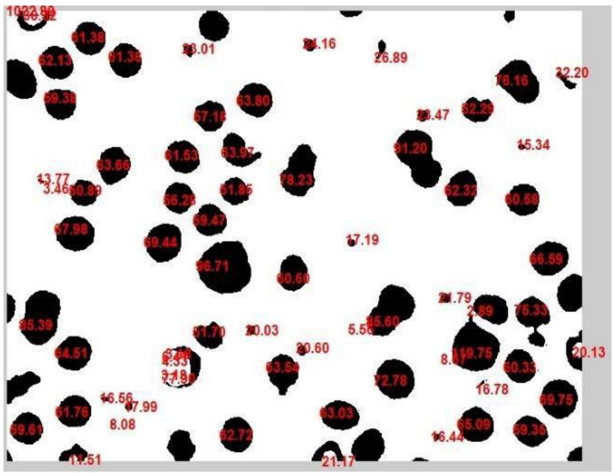


Figure 6 (h): Radius of cell

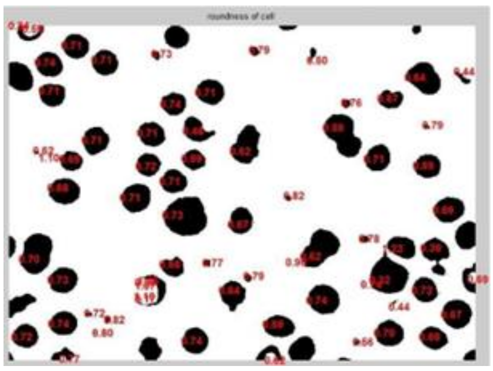


Figure 6 (i): Roundness of cell

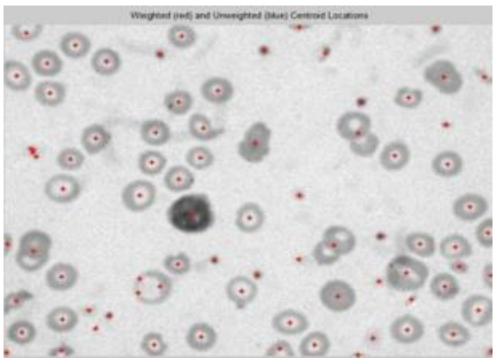


Figure 6 (j): Centroid of cell

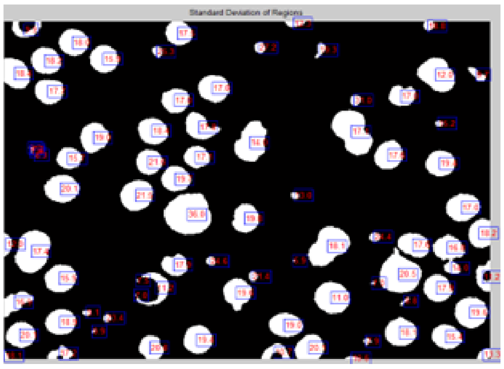


Figure 6 (k): standard deviation

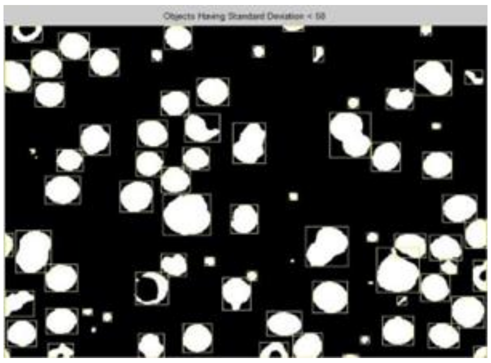


Figure 6 (l): bounding box Figure 6 Process Flow of Shape based Features

4. Conclusion

Several methods have been examined and assessed, with their merits and demerits derived from experimental outcomes. The conclusion drawn is that the proposed method, focusing on shape-based feature detection, offers superior accuracy compared to other techniques for counting leukemic cells, achieving a high accuracy rate of 97.8% as depicted in the table below. Shape-based features are utilized to detect various geometric shapes of cells such as basophils, eosinophils, lymphocytes, and monocytes. The count of immature cells aids in diagnosing diseases. Limitations are identified for watershed transform, K-means clustering, and histogram equalization methods.

Method	Pros/Cons	Accuracy
Watershed transform	Pros: Easy method for detection of white cell Cons: It cannot give accurate result and cannot implement on each and every image	72.2 %
K means clustering	Pros: It is used for clustering and separate the data based on value of K. Cons: It does not give classification with labeled data and also not applicable on incremental data.	72 %
Edge detection using histogram equalizing method and linear contrast stretching	Pros: This is very useful method to detect white cell and for contrast enhancement. Cons: It is hard to define boundary of overlapping cell.	73.7 %
Shape based features	Pros: Very easy for the detection of white and overlapping cell and shape of cell Cons: This is based on statistics so can get approximate result.	97.8 %

References

[1] Mohapatra, Subrajeet, Sushanta Shekhar Samanta, Dipti Patra, and Sanghamitra Satpathi. "Fuzzy-based blood image segmentation for automated leukemia detection." In Devices and Communications (ICDeCom), 2011 International Conference on, pp. 1-5. IEEE, 2011.

[2] Lim, Huey Nee, Mohd Yusoff Mashor, and Rosline Hassan. "White blood cell segmentation for acute leukemia bone marrow images." In Biomedical Engineering (ICoBE), 2012 International Conference on, pp. 357-361. IEEE, 2012.

[3] Fatma, Mashiat, and Jaibir Sharma. "Identification and classification of acute leukemia using neural network." In Medical Imaging, m- Health and Emerging Communication Systems (MedCom), 2014 International Conference on, pp. 142-145. IEEE, 2014.

[4] Madhloom, H. T., S. A. Kareem, H. Ariffin, A. Zaidan, H. O. Alanazi, and B. B. Zaidan. "An automated white blood cell nucleus localization and segmentation using image arithmetic and automatic threshold." (2010).

[5] Halim, NH Abd, M. Y. Mashor, A. S. Abdul Nasir, N. R. Mokhtar, and H. Rosline. "Nucleus segmentation technique for acute leukemia." In Signal Processing and its Applications (CSPA), 2011 IEEE 7th International Colloquium on, pp. 192-197. IEEE, 2011.

[6] Raje, Chaitali, and Jyoti Rangole. "Detection of Leukemia in microscopic images using image processing." In Communications and Signal Processing (ICCSP), 2014 International Conference on, pp. 255-259. IEEE, 2014.

[7] Mohapatra, Subrajeet, and Dipti Patra. "Automated leukemia detection using Hausdorff dimension in blood microscopic images." In Emerging Trends in Robotics and Communication Technologies (INTERACT), 2010 International Conference on, pp. 64-68. IEEE, 2010.

[8] Berge, Heidi, Dale Taylor, Sriram Krishnan, and Tania S. Douglas. "Improved red blood cell counting in thin blood smears." In Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on, pp. 204-207. IEEE, 2011.

[9] Mazalan, Siti Madihah, Nurul H. Mahmood, and Mohd Azhar Abdul Razak. "Automated Red Blood Cells Counting in Peripheral Blood Smear Image Using Circular Hough Transform." In Artificial Intelligence, Modelling and Simulation (AIMS), 2013 1st International Conference on, pp. 320-324. IEEE, 2013.

[10] Akrimi, Jameela Ali, Azizah Suliman, Loay E. George, and Abdul Rahim Ahmad. "Classification red blood cells using support vector machine." In Information Technology and Multimedia (ICIMU), 2014 International Conference on, pp. 265-269. IEEE, 2014.

[11] Mohapatra, Saurav, Dipti Patra, Sudhakar Kumar, and Siddhartha Satpathi. "Kernel-induced rough c-means clustering for lymphocyte image segmentation." In Intelligent Human Computer Interaction (IHCI), 2012 4th International Conference on, pp. 1-6. IEEE, 2012.

[12] Ge, Jia, Z. Gong, Jiann-Jong Chen, Jiangchuan Liu, John Nguyen, Z. Y. Yang, Chingyue Wang, and Yue Sun. "A system for automated counting of fetal and maternal red blood cells in clinical KB test." In Robotics and Automation (ICRA), 2014 IEEE International Conference on, pp. 1706-1711. IEEE, 2014.

[13] Supardi, N. Z., M. Y. Mashor, N. H. Harun, F. A. Bakri, and R. Hassan. "Classification of blasts in acute leukemia blood samples using k-nearest neighbour." In Signal Processing and its Applications (CSPA), 2012 IEEE 8th International Colloquium on, pp. 461-465. IEEE, 2012.

[14] Mohammed, Emad, Mostaja MA Mohamed, Christopher Naugler, and Behrouz H. Far. "Chronic lymphocytic leukemia cell segmentation from microscopic blood images using watershed algorithm and optimal thresholding." In Electrical and Computer Engineering (CCECE), 2013 26th Annual IEEE Canadian Conference on, pp. 1-5. IEEE, 2013.

[15] Abdul Nasir, A. S., M. Y. Mashor, and H. Rosline. "Unsupervised colour segmentation of white blood cell for acute leukaemia images." In Imaging Systems and Techniques (IST), 2011 IEEE International Conference on, pp. 142-145. IEEE, 2011.

[16] Rawat, Jyoti, A. Singh, H. S. Bhadauria, and I. Kumar. "Comparative analysis of segmentation algorithms for leukocyte extraction in the acute Lymphoblastic Leukemia images." In Parallel, Distributed and Grid Computing (PDGC), 2014 International Conference on, pp. 245-250. IEEE, 2014.

[17] Saini, Mukesh Kumar; Singh, Jaibir. "Big data analytics and machine learning: Personalized, predictive health and boost exactitude medicine research" In Big data analytics and machine learning, 2021 International Journal of Management IT and Engineering, pp. 47-56. International Journal of Management IT and Engineering 2021.

[18] Das, Biplab Kanti, Krishna Kumar Jha, and Himadri Sekhar Dutta. "A New Approach for Segmentation and Identification of Disease- Affected Blood Cells." In Intelligent Computing Applications (ICICA), 2014 International Conference on, pp. 208-212. IEEE, 2014.

[19] Mohapatra, Subrajeet, and Dipti Patra. "Automated cell nucleus segmentation and acute leukemia detection in blood microscopic images." In Systems in Medicine and Biology (ICSMB), 2010 International Conference on, pp. 49-54. IEEE, 2010.

[20] Madhloom, Hayan T., Sameem Abdul Kareem, and Hany Ariffin. "A Robust Feature Extraction and Selection

[21] Method for the Recognition of Lymphocytes versus Acute Lymphoblastic Leukemia." In Advanced Computer Science Applications and Technologies (ACSAT), 2012 International Conference on, pp. 330-335. IEEE, 2012.

[22] Putzu, Lorenzo, and Cecilia Di Ruberto. "White blood cells identification and counting from microscopic blood images." World Academy of Science, Engineering and Technology 7, no. 1 (2013): 363-370.

[23] (2015) Figure 1 website. [Online] Available at: www.ufrgs.br

[24] (2015) Figure 2(a) website. [Online] Available at: www.doctortipster.com

[25] (2015) Figure 3(a) website. [Online] Available at: www.pathologystudent.com

[26] (2015) Figure 4(a) website. [Online] Available at: www.medicalxpress.com

[27] (2015) Figure 5(a) website. [Online] Available at: www.prezi.com

[28] (2015) Figure 6(a) website. [Online] Available at: www.commons.wikimedia.org