PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100 feet Ring Road, Bengaluru – 560 085, Karnataka, India

Report on

BLOOD CELL IDENTIFICATION USING IMAGE PROCESSING

Submitted by

KEERTHANA V BHAT (PES1201800549)

HIMAAVARSHINI N (PES1201802149)

SEM - 6 , SECTION - F

January – May 2021

Under the guidance of

Dr. SHIKHA TRIPATHI

Professor

Department of Electronics and Communication

PES University

Bengaluru – 560085

Faculty of Engineering

Department of Electronics and Communication

Program B-TECH

INTRODUCTION

The human blood contains the RBCs, WBCs, Platelets and Plasma. The entire blood count defines the state of health.

Blood could be a health indicator so segmentation and identification of blood cells is extremely vital as it helps doctors to diagnose various diseases such as anemia, leukemia etc.

The use of image processing techniques helps in improving the effectiveness of the identification in terms of accuracy and time consumption.

Analysis of blood smear is an important diagnostic test used in the diagnosis of an array of diseases.

Automation of this process ultimately narrows the scope of possible diseases saving a considerable amount of time.

Identification of red blood cells (RBCs) is carried out/done by the system using different techniques of image processing operations like pre-processing, operations for morphology, labelling and extraction of features.

The aim of this system is to assist the pathologist by giving quick results by analysing the smear samples.

THEORETICAL DETAILS

The main purpose of this paper is to segment WBCs from microscopic images. In the algorithm proposed, we implement suitable image segmentation and feature extraction techniques for blood cell identification on the obtained enhanced image.

Image Enhancement

This step is to improve the quality, contrast, brightness characteristics of an image and also to sharpen its details.

Pre-processing:

Pre-processing includes various steps like-

A) RGB to Grayscale:

The method of converting an RGB image to Grayscale using the command COLOR_BGR2GRAY.

B) Smoothening or Blurring

Smoothen the image for image enhancement and to reduce noise by the operations gaussian and median blur.

C) Canny Edge detection

The **Canny edge detector** is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. The steps are

1. Apply Gaussian / Median filter to smooth the image in order to remove the noise
2. Find the intensity gradients of the image
3. Apply gradient magnitude thresholding or lower bound cut-off suppression to get rid of spurious response to edge detection
4. Apply double threshold to determine potential edges
5. Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

Image Segmentation and Feature Extraction

Image segmentation is a method which can be used to understand **images** and **extract** information or objects. First step is to read the enhanced image and apply morphological operation.

A) **Morphology operations** process images based on shapes with a range of image processing operations like dilation and erosion. Addition of pixels is dilation and removal of pixels is erosion. These operations are performed on edge detected images

Dilation- Addition of pixels to the image object boundaries. In a binary image, a pixel is set to 1 if any of the neighboring pixels have value 1. This operation makes objects more visible and fills in small holes present in the objects.

Erosion- This operation removes pixels on object boundaries. In a binary image, a pixel is set to 0 if any of the neighboring pixels have value 0. This operation removes islands and small objects so that only substantive objects remain.

Dilation followed by erosion is Closing

B) **Thresholding or Binarization**

The method of converting black – white image from any grayscale image. Thresholding starts by comparing the threshold value for grayscale image with the intensity of gray scale value. If the value of pixel > the threshold, then the pixel is white, else the pixel is black when pixel < threshold. We apply Adaptive filtering using mean and gaussian filter and otsu thresholding and compare both results.

Adaptive thresholding is the method where the threshold value is calculated for smaller regions and therefore, there will be different threshold values for different regions.

Otsu's thresholding method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels each side of the threshold

C) Detection and Counting:(Feature extraction)

The **Hough Transform with circular parameters** (CHT) is a basic feature extraction technique used in digital image processing for detecting **circles** in imperfect images. This helps in cell detection and labelling. The last step is to count and display the number of white blood cells in the blood smeared microscopic image. In order to detect circles in images, we need to make use of the cv2.HoughCircles function

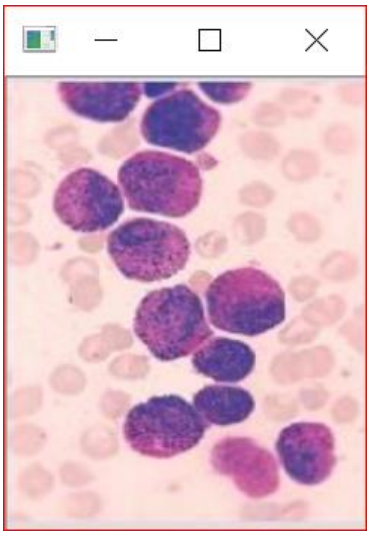
CODE

```
import numpy as np
import cv2
import matplotlib.pyplot as plt

#Read original image
image = cv2.imread("project.PNG")
cv2.imshow('Input image',image)
cv2.waitKey(0)
cv2.destroyAllWindows()

# convet to gray scale image
gray = cv2.cvtColor(image , cv2.COLOR_BGR2GRAY)
cv2.imwrite('gray.png', gray)

# Apply median filter for smoothning
blur_M = cv2.medianBlur(gray, 5)
cv2.imwrite('blurM.png', blur_M)

# Apply gaussian filter for smoothning
blur_G = cv2.GaussianBlur(gray, (9, 9), 0)
cv2.imwrite('blurG.png', blur_G)

#Display
blurHorti = np.concatenate((blur_M,blur_G), axis=1)
cv2.imshow('Smoothning Results- Median Blurring and Gaussian Blurring', blurHorti)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

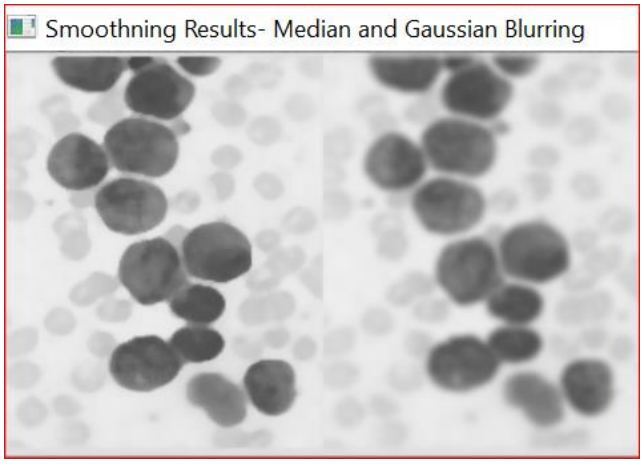
```
# Edge Detection using Canny Edge Detector
edge = cv2.Canny(gray, 100, 200)
edge_G = cv2.Canny(blur_G, 100, 200)
cv2.imwrite('edge_G.png', edge_G)
edge_M = cv2.Canny(blur_M, 100, 200)

#Display
edgeHorti = np.concatenate((edge,edge_G,edge_M), axis=1)
cv2.imshow('Canny edge detection Results- on Gray image,Gaussian and Mean Smoothened images', edgeHorti)
cv2.waitKey(0)
cv2.destroyAllWindows()

# Read enhanced image
img = cv2.imread('edge_G.png', 0)
kernel = np.ones((5, 5), np.uint8)

# Morphological Operations
dilation = cv2.dilate(img, kernel, iterations = 1)
closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)

# Adaptive thresholding using Mean and Gaussian filter
thresh1 = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 11, 2)
thresh2 = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
cv2.imwrite('thresh2.png',thresh2)

# Otsu's thresholding
ret3, thresh3 = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

#Display
threshHorti = np.concatenate((thresh1,thresh2,thresh3), axis=1)
cv2.imshow("Adaptive thresholding using Mean and Gaussian filter and Otsu's Thresholding Results", threshHorti)
cv2.waitKey(0)
cv2.destroyAllWindows()


# Initializing list
Cell_count, x_count, y_count = [], [], []

# Hough Transform with modified Circular parameters
display = cv2.imread("thresh2.png",0)
circles = cv2.HoughCircles(display, cv2.HOUGH_GRADIENT, 1.2, 20, param1 = 50, param2 = 28,
                           minRadius = 1, maxRadius = 20)

# Circle Detection and labeling using Hough Transformation
if circles is not None:
        circles = np.round(circles[0, :]).astype("int")

        for (x, y, r) in circles:
                cv2.circle(display, (x, y), r, (0, 0, 0), 2)
                cv2.rectangle(display, (x - 2, y - 2), (x + 2, y + 2), (0, 0, 0), -1)
                Cell_count.append(r)
                x_count.append(x)
                y_count.append(y)

        cir = thresh2-display
        img = cv2.add(cir,gray)
        cv2.waitKey(0)

# Display the count of White Blood Cells
Horti1 = np.concatenate((gray,blur_G,edge_G), axis=1)
Horti2= np.concatenate((dilation,closing,thresh2, cir, img), axis=1)


cv2.imshow('Gray_Img ,Blurred_Img ,Canny_edge', Horti1)
cv2.waitKey(0)
cv2.destroyAllWindows()
cv2.imshow('Dilation, Closing, Thresholding, Detection, Counting', Horti2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
print('number of cells =' ,len(Cell_count))
print('Radius of cells =', Cell_count)
print('x-axis =', x_count)
print('y-axis =', y_count)
```

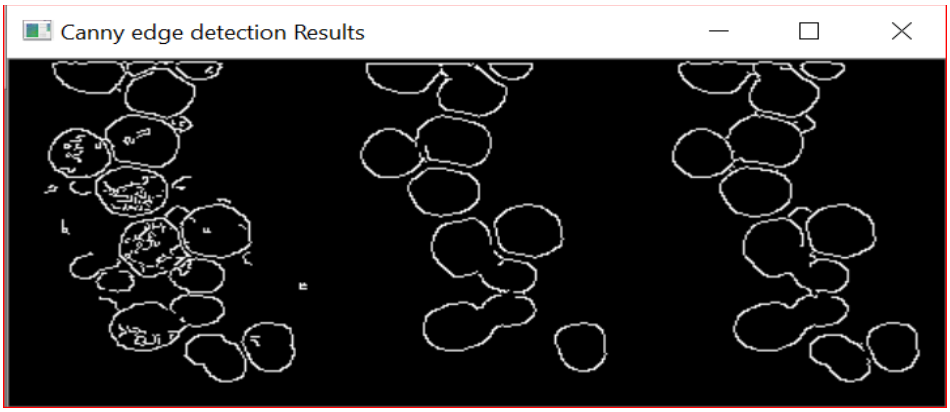
INPUT:



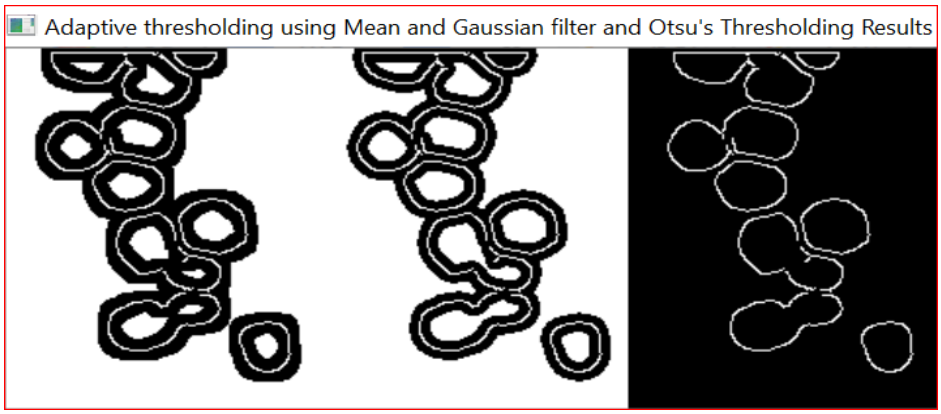
OUTPUTS:



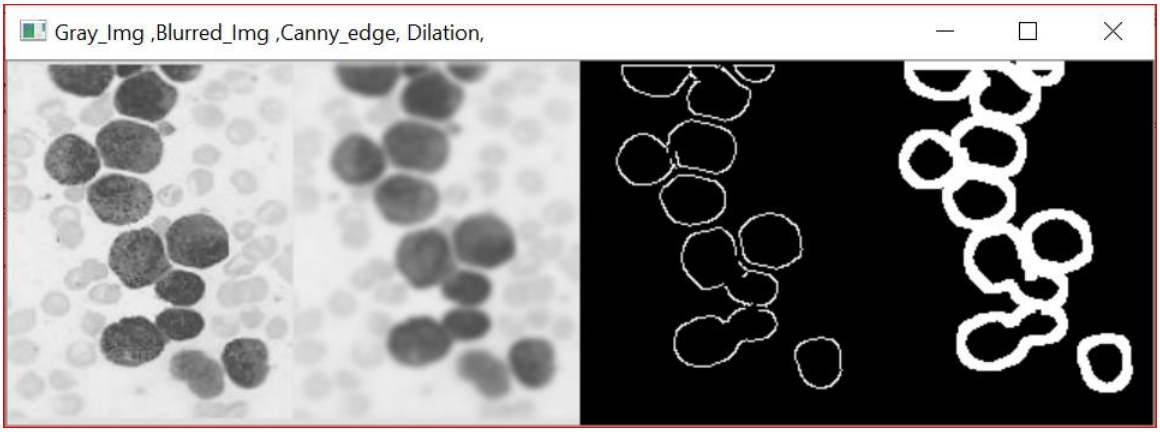
Median blur **Gaussian blur**



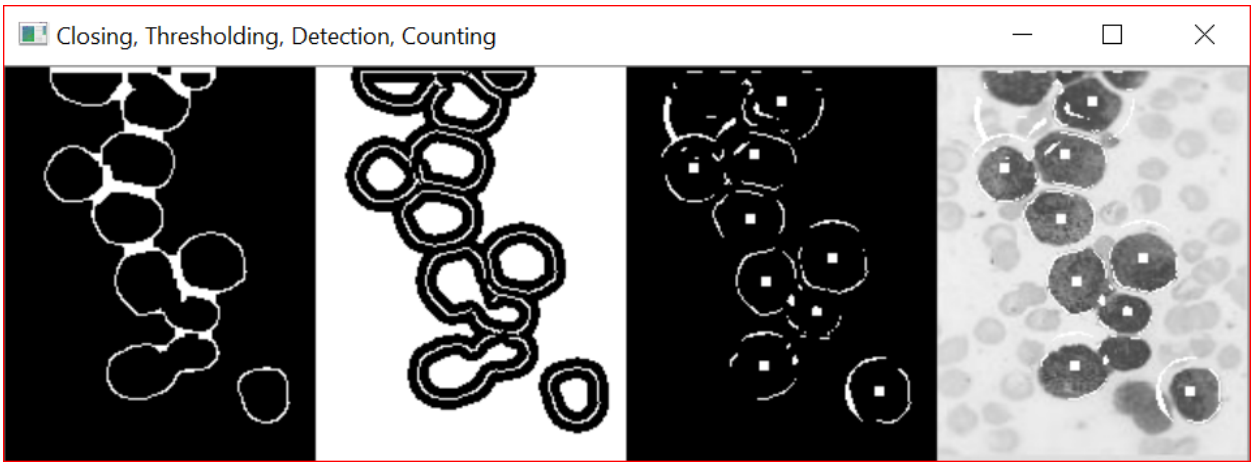
Edge of grayscale image **Edge of Median blurred Image** **Edge of Gaussian blurred Image**



Mean adaptive threshold **Gaussian adapting threshold** **Otsu threshold**



Grayscale image **Gaussian blur** **Gaussian canny edge** **Dilation**



Closing **Adaptive mean threshold** **Detected cells** **Counting of cells**

```
Number of cells = 10
Radius of cells = [19, 19, 18, 17, 14, 18, 19, 19, 11, 10]
x-axis = [61, 65, 106, 133, 35, 68, 79, 71, 97, 99]
y-axis = [49, 74, 97, 169, 53, 109, 19, 149, 124, 145]
```

From the outputs that the Number of cells , each cell Radius and their Center position (x and y coordinate points) of each cell can be observed.