

COMPUTER VISION

CSE 3007

MINI-PROJECT

TITLE : *Identification of human blood cells*

TEAM : Sheetal JANKEE
Véronique LEUNG

COURSE : BSc. (Hons) Computer Science and Engineering Level 3

5 November 2003

Section	Page
1. Introduction.....	1
2. Background study.....	2
3. Analysis.....	6
4. Design.....	7
4.1 Platform.....	7
4.2 Noise Removal.....	9
4.3 Image Segmentation.....	9
4.4 Morphological Operations.....	9
4.5 Image Labeling.....	10
4.6 Image Representation.....	10
4.7 Classification and counting.....	11
4.8 Training.....	12
5. Implementation.....	13
5.1 Noise Removal.....	13
5.2 Image Segmentation.....	13
5.3 Morphological Operations.....	14
5.4 Image Labeling.....	15
5.5 Image Representation.....	16
(i) Area.....	16
(ii) Perimeter and Circularity.....	17
5.6 Classification.....	18
5.7 Counting.....	20
5.8 Training.....	21
5.9 Class Description.....	22
Class Image	22
Class Object.....	23
5.10 Interface.....	25

6. Testing.....	30
7. Conclusion and Limitations.....	33
8. References.....	34
Appendix I - Blood cell values.....	35
Appendix II – Files structure.....	37

List of figures	Page
Figure 2.0: <i>A white blood cell (Lymphocyte)</i>	3
Figure 2.1: <i>The different types of blood cells</i>	3
Figure 2.2: <i>Abnormal red blood cells</i>	4
Figure 4.0: <i>Steps involved with process of classifying and counting blood cells in an image</i>	8
Figure 4.1: <i>Definition of circularity [3]</i>	11
Figure 5.0: <i>3 x 3 Filter</i>	13
Figure 5.1: (a) <i>Original Image</i> (b) <i>Output after image threshold</i>	14
Figure 5.2: (a) <i>A thresholded image</i> (b) <i>The corresponding labeled image</i>	16
Figure 5.3: <i>Direction notation in boundary tracing</i>	18
Figure 5.4: <i>Thresholding image subpart to get nucleus</i>	20
Figure 5.5: <i>Opening an image</i>	25
Figure 5.6: <i>The displayed image and the 'Auto Count' menu</i>	26
Figure 5.7: <i>Displayed results</i>	27
Figure 5.8: <i>Labeled image</i>	28
Figure 5.9: <i>System training</i>	29
Figure 6.0: <i>Correct identification of red blood cells, platelets and phagocyte</i>	30
Figure 6.1: <i>Correct identification of red blood cells, platelets, phagocyte and abnormal cells</i>	31
Figure 6.2: <i>Correct identification of red blood cells, platelets and lymphocyte</i>	31
Figure 6.3: <i>Correct identification of red blood cell, platelets, lymphocyte and phagocyte</i>	32
Figure 6.4: <i>The system takes touching or overlapping cells as one, yielding inaccurate results</i>	32
 List of tables	
Table 2.0: <i>Normal haematological values [4]</i>	5

1. INTRODUCTION

The identification of blood cells from blood samples is not a novel concept. Indeed it is being practiced since decades in the medical domain. Manual inspection required a human to visualize blood samples under a microscope to extract information about the shape, size and overall appearance of the blood cells. But now, with the advance in computer vision technology, automatic identification of the constituents of a blood sample is possible. The aim of our project is to automatically identify and classify the different types of blood cells from a given blood sample and to generate a complete blood count using computer vision techniques.

The background study introduces the medical facts about the characteristics of the different cells encountered in a blood sample. It is explained how the information gained from blood test helps to detect certain anomalies and diseases. In the analysis part, the need for system is justified and its objectives are outlined. The report also outlines the steps followed in design and implementation of system. Performance evaluation is based on results of tests carried out on system and the limitations of system are exposed.

2. BACKGROUND STUDY

The human blood consists of blood cells floating in a fluid called plasma, which contains several dissolved substances like proteins or mineral salts [2]. There are three types of blood cells: red blood cells (RBCs), white blood cells (WBCs) and platelets. WBCs are divided in three further categories: phagocytes, lymphocytes and basophils. There exist three types of phagocytes: neutrophils, monocytes, and eosinophils.

Red Blood Cells

Normal RBCs contain no nucleus and are biconcave in shape – the cell is thicker at the edge and thinner at its centre. Hence, RBCs appear to have a paler area at the centre [6]. Each RBC have an approximate diameter of 6.7 to 7.7 micrometers. There are around 5.5 millions of these cells per cubic millimeters (mm^3) in a healthy person.

White Blood Cells

WBCs are larger than RBCs and are less numerous – there are around 7,000 WBCs per mm^3 in a healthy person. Phagocytes have an irregularly shaped nucleus (which is lobed), while lymphocytes have a very much larger nucleus which nearly fills the cell. Lymphocytes are usually smaller in size than phagocytes but this variation is very small. Basophils are a rare type of white blood cells – they account for less than 1% of all WBCs (See Table 2.0 for an average count of these cells).

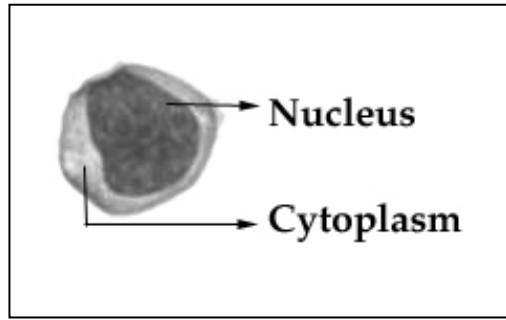


Figure 2.0 : A white blood cell (Lymphocyte). The cell consists of a central nucleus bathing in a liquid called the cytoplasm.

Platelets

Blood platelets are even smaller than WBCs and there are about 300,000 of platelets per mm^3 in a healthy person. Platelets also do not contain a nucleus.

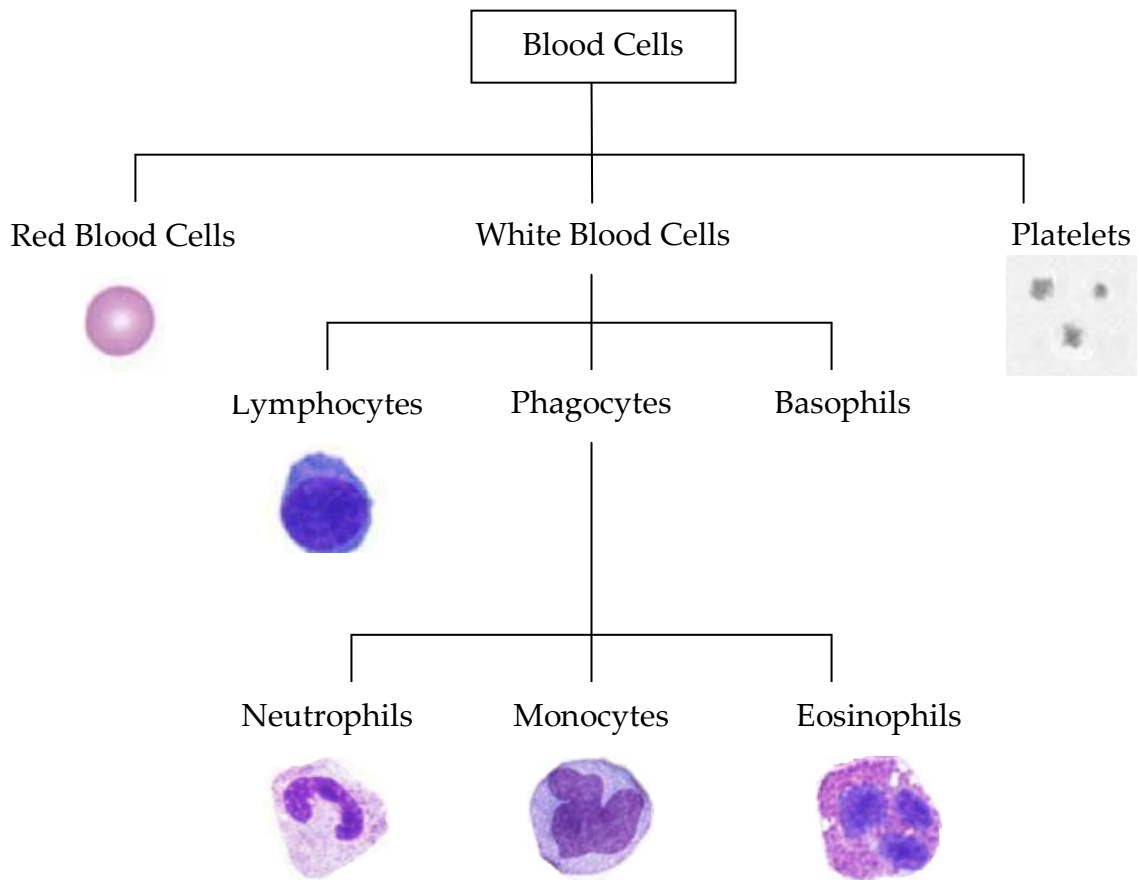


Figure 2.1: The different types of blood cells

Abnormal Cells

The above three categories of blood cells may also be affected by diseases. In certain cases, this distorts their shapes. For example, normal red blood cells are circular in shape, but infected red cells may have a tear drop shape (found in patients with pernicious anaemia) or a rod-like shape (elliptocytes - found in various anaemias) or a sickle/crescent shape (found in sickle cell anaemia). The other types of blood cells also have abnormal shapes when affected by particular diseases [5].



(a) *Teardrop shaped*



(b) *Burr*



(c) *Schistocyte*



(d) *Elliptocytes*



(e) *Acanthocyte*



(f) *Sickle*

Figure 2.2: *Abnormal red blood cells*

Blood Cell Count

The 'normal' values for blood cell count is difficult to state since this amount depends on the age, sex, physique, occupation, genetic and ethnic background or the geographical location of the person among several other factors [4]. But it has been observed that the normal count of each type of blood cell varies within a known range for healthy people. Hence, a blood cell count can be used to detect certain diseases.

An abnormal number of any of the blood cells may indicate a disease. For example, anaemia is characterized by an abnormally low RBC count, leukaemia by an abnormally higher WBC count and haemophilia by a low platelet count. Abnormal blood cell counts can also point to a defect in blood cell production.

Blood Cell Type	Number per mm ³ *
Red Blood cell	5.5 million
White Blood Cell	7000
Lymphocytes	2000
Phagocytes (neutrophils + eosinophils + monocytes)	4500
Basophils	25
Platelets	300000

* Figures are for a healthy adult male

Table 2.0: *Normal haematological values* [4]

3. ANALYSIS

A manual blood count is possible but tedious and prone to errors especially if there is a high number of cells to be counted; hence a computer vision application capable of automatically identifying and counting the different blood cells would simplify this task and would provide quicker results.

The aim of our system is thus to:

- (1) Detect and classify each blood cell in a digital image into one of these categories:
 - (i) Red blood cells
 - (ii) White blood cells (differentiating between lymphocytes and phagocytes)
 - (iii) Blood platelets
 - (iv) Other (Abnormal cells)

It should be noted that for the first three categories, we are counting only the *normal* blood cells. The fourth category (Other) may thus contain *abnormal* cells, but which might also belong to any one of the three other categories.

- (2) Count the number of each of the above blood cell types present in the image and give an average count of the number of each of these blood cell types in one cubic millimeter of blood in an adult male.

4. DESIGN

4.1 Platform

A small amount of blood is taken from the patient. A thin film of this blood is spread on a glass slide, diluted to reduce its concentration and stained with a special blood stain. The sample is then placed under a special digital camera which will magnify and capture a microscopic image of the blood sample. All images should be taken at the same magnification. The image should also be taken where illumination is controlled so that all images will be exposed to the same amount of illumination.

The application is implemented in Visual C++ since Visual C++ provides a flexible and effective environment for the image processing required by the system. The different tools available provide an easy interface and makes for easy learning.

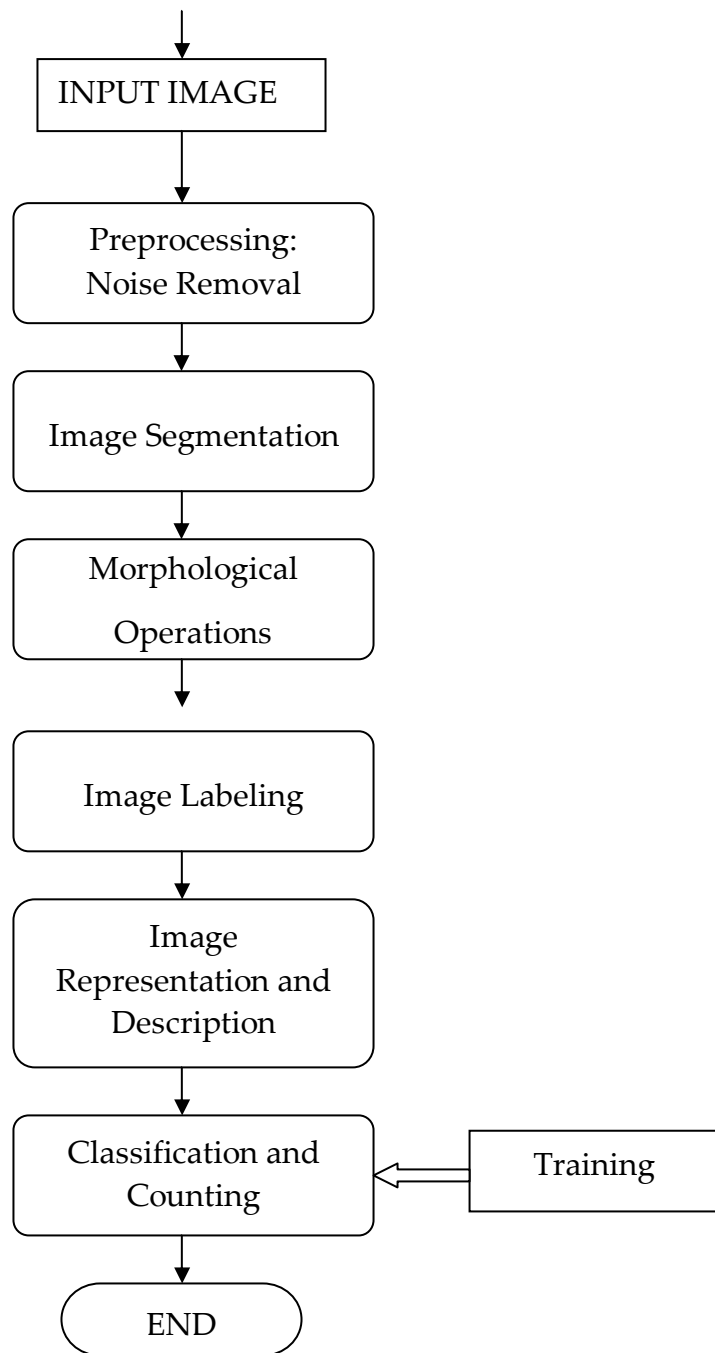


Figure 4.0: Steps involved with process of classifying and counting blood cells in an image

4.2 Noise Removal

Before extraction of the objects in the image, some pre-processing techniques need to be applied to the image to increase the chances for success of further processing like image analysis. The image needs to be filtered with a noise removal filter to remove any unwanted information that might occur in the image. This noise might occur during image acquisition by the camera or because of the dissolved substances found in the blood plasma.

4.3 Image Segmentation

Segmentation subdivides the image into its constituent parts or objects [1]. One approach to image segmentation is through thresholding. We need to threshold the image to get a binary image so that visible parts of object have a significantly different intensity from that of the background [7]. However, thresholding does not yield accurate results if the objects in the image overlap or touch each other. However, dilution of the blood sample has been done prior to taking the image, therefore the cells will be further apart from each other and has a lower probability of overlapping. Also, for image thresholding to work well, the brightness of every visible part of the object needs to be significantly different from that of the background. Thus, the image is assumed to be taken under the same controlled conditions i.e. there is control of illumination.

4.4 Morphological Operations

Morphological opening is applied to the binary image to remove any bridges that might connect close cells. Opening is the process of erosion followed by dilation. This operation is done to separate any close cells that might be connected by unwanted 'bridges', and thus be taken as only one cell after image labeling, instead of separate cells. Erosion decreases the size of objects while dilation increases their size. Since we are applying dilation on an

eroded image, the objects regain their original size, but bridges which might have connected close objects will not be present after opening is performed.

4.5 Image Labeling

We now have to assign a unique label to each distinct object found in the image. We also need to discard objects touching the border of the image since most probably, the visible parts of these objects will not have the same characteristics as the whole object.

4.6 Image Representation

Each object identified with the previous method needs to be classified into one of the five types of blood cells, namely

- (i) Red blood cells
- (ii) Lymphocytes
- (iii) Phagocytes
- (iv) Platelets
- (v) Other (Abnormal cells)

To be able to distinguish between the different blood cell types, we will use their area and shape (circularity) as a classification factor. Hence, we need to extract these features from each object present in the image.

The area of an object (a cell) is defined as the number of pixels contained within its boundary [1]. White blood cells (lymphocytes and phagocytes) have the largest areas while red blood cells have far smaller areas. Blood platelets have even smaller areas than red blood cells. Thus, for each object previously identified and labeled, we need to compute its corresponding area.

Shape description is also required since there could be abnormal cells present in the blood. These abnormal cells might have the same area as other normal

cells of other categories, and hence, might be erroneously identified as such. For example, a sickle-shape red blood cell has a substantially smaller size (area) than a normal red blood cell, and might be taken as a blood platelet if only the area is being considered. Thus, we also need to ascertain that a cell has a circular shape to be classified as a normal blood cell.

We use the circularity (C_0) of an object, defined in Figure 4.1 to determine the shape of a blood cell. A circle-shaped object will have a circularity of 1. Thus any normal blood cell belonging to any category will have a circularity of around 1 while non-circular cells (like sickle-shape red blood cells) will have a circularity of much less than 1. Therefore, we can use the circularity value of a cell to determine whether it is not a normal cell (does not have a round shape).

$$C_0 = \frac{4\pi A}{P^2}$$

A: area of object
P: perimeter of object

Figure 4.1: *Definition of circularity* [3]

As can be seen in the definition, circularity involves the computation of the perimeter of the cell. The perimeter of a region is the length of its boundary [1]. Hence, to determine the perimeter, we need to count the number of pixels which make up the boundary of the cell. Since circularity is a dimensionless quantity, it does not take into account the size of the object.

4.7 Classification and counting

Now that we have the area and circularity values of each object in the image, we can classify each object into one of the five categories mentioned above. To determine if the cell is a normal blood cell, we check its circularity value. If

the cell is a normal cell, we use its area to determine to which category it belongs. Mean values of the areas are read from a file and the minimum distance value between the area of the cell and the mean areas are computed.

If the minimum distance value indicates that the cell might be a white blood cell (lymphocyte or phagocyte – both have approximately the same area), then we further need to distinguish between them. A differentiating feature which we use is the **nucleus area to whole area ratio** of the cell, defined as the ratio between the area of the nucleus and the area of the whole cell. We then need to determine to which class of blood cell type this ratio is nearest to.

As each cell is classified, we need to keep track the number of each of the different types of blood cells that are present in the image.

4.8 Training

Before any classification can be done, we need to train the system, which involves the selection of the particular features which best describe the blood cells. Since we are using the area as classification factor and also the nucleus area to whole area for white blood cells, we need to store a mean vector that will represent each of the blood cell type in a file. If the cell is a white blood cell, we also need to calculate its nucleus area to whole area ratio. The values obtained for each sample image needs to be added to the values already found in the training file. We also assume that the image used for training will contain only one blood cell, and the cell does not touch any of the image borders.

5. IMPLEMENTATION

5.1 Noise Removal

To remove any abrupt change in gray level, we use median filtering. This filter works by replacing a gray level value of a pixel by the median value of the pixels in its neighborhood. Since we are using a 3x3 filter size, in this case, the response at a pixel is the median value of the intensities of the 8-connected neighbors of the pixel along with the value of the pixel intensity itself. For the pixel with intensity value I_4 in Figure 5.0, the intensity of this pixel will be replaced by the median value of these intensities: $I_0, I_1, I_2, \dots, I_8$. The mask is applied to each pixel within the image.

I_0	I_1	I_2
I_3	I_4	I_5
I_6	I_7	I_8

Figure 5.0: 3 x 3 Filter

If the mask is applied on pixels that lie on the border of the image, part of the mask will lie outside the image. Hence, we apply the mask only where the whole mask will lie within the image, which also means that border pixels of the image are not processed by the mask.

5.2 Image Segmentation

Image segmentation is performed through a global thresholding technique: all pixels having a gray level value above a certain value (the threshold value) are assigned a value of 255 (or the maximum gray level), and pixels with gray level values lower than the threshold value are assigned a value of 0. The threshold value was determined empirically i.e. we tested several values until

a suitable threshold value was observed that produces the best image in the sense that regions which are blood cells in the resultant binary image have a different intensity than regions belonging to the background. However, after image thresholding, the central pallor inside some red blood cells is regarded as another object – the red blood cell seems to contain a hole inside it.

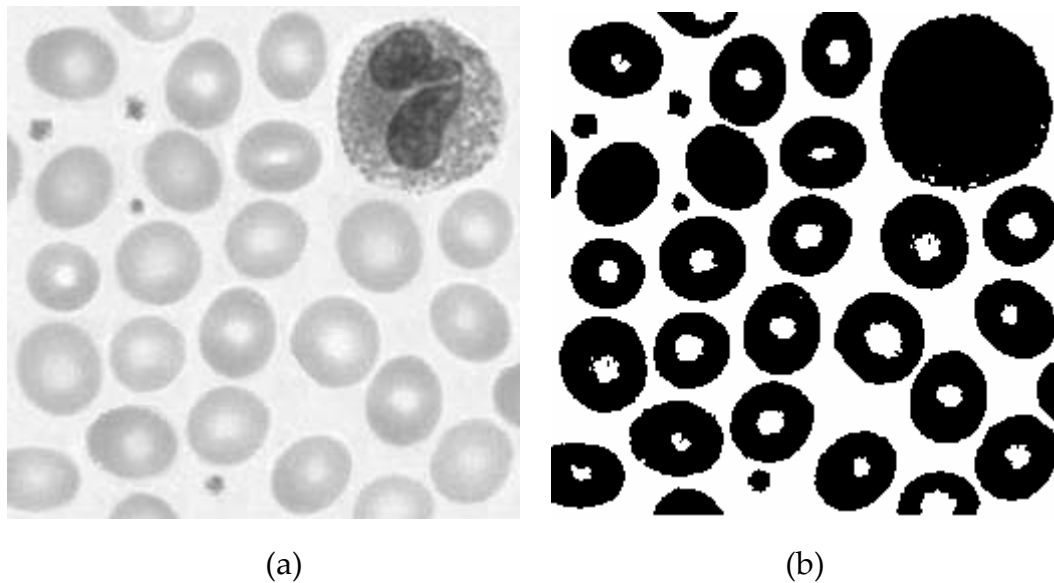


Figure 5.1: (a) *Original Image* (b) *Output after image threshold*

5.3 Morphological Operations

Opening is then applied to the thresholded image. Opening consists of two steps: erosion, followed by dilation. Erosion is the process of replacing the value of a pixel in an image with the maximum value (since our image consists of black blobs over a bright background) in a neighborhood which is represented by a 3x3 pixel square structure (similar to the filter shown in Figure 5.0). The resultant eroded image is an eroded image consisting of objects shrunk by one pixel. Dilation is the opposite process of erosion as each pixel is assigned the minimum value in the neighborhood. Again, border pixels are not processed during the opening operation. The result of the opening operation is an image where any 'bridges' of one pixel wide have

been eliminated. The elimination of such 'bridges' is particularly important for the boundary tracing algorithm which is discussed in section 5.5 (ii) since this algorithm works for regions having a width of greater than one pixel.

5.4 Image Labeling

To label each object, we scan all the pixels in the original image, starting from the top left corner and going row by row until we reach the bottom right corner. Initially, all pixels in the output image (or labeled image) are labeled with a value of -1 (background). As an object is identified, it is assigned an ObjectID which is unique for each object. In the original image, all pixels belonging to the background have intensity values 255, while objects have intensity values 0. Thus, when we encounter a pixel with intensity value 0 in the original image, we assign the corresponding pixel in the output image the value of the objectID. All pixels that are connected to this initial pixel are then assigned the same ObjectID in the output image.

To determine connectivity, we use 8-connectedness. Starting from the initial pixel, we check if each of the 8-connected neighbors of this pixel has an intensity value of 0 in the original image. If this is true for one of the neighbors, this pixel becomes the starting pixel and the whole process of looking and labeling an 8-connected neighbor is repeated. This process stops when either we reach the image border or an intensity value of 255 is encountered in the original image (this means that we have cross over to the background).

When the whole object has been labeled, we continue scanning the pixels in the original image to label any further object until all pixels in the original image have been processed.

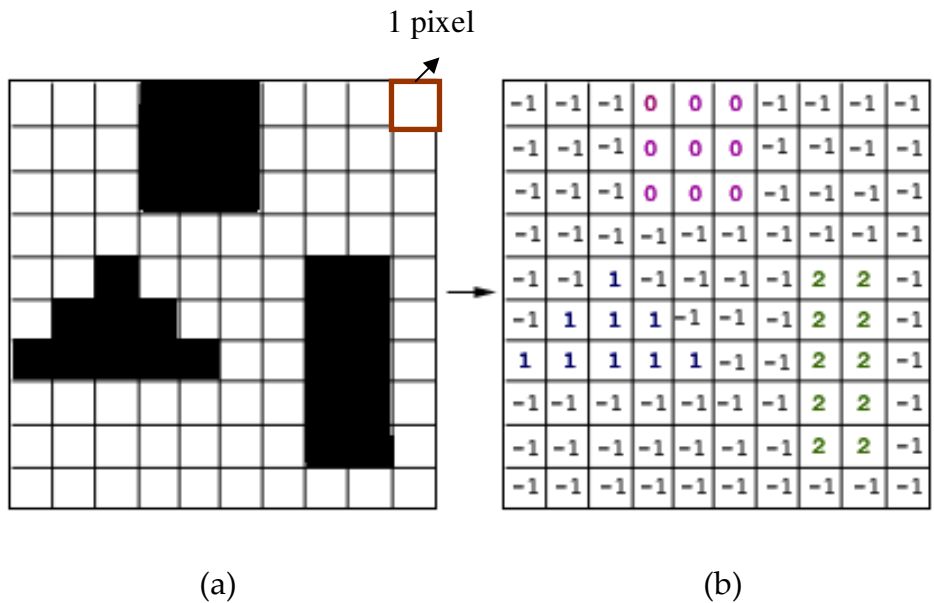


Figure 5.2: (a) A thresholded image (b) The corresponding labeled image

After all objects in the image have been assigned a particular ObjectID, an array containing all the identified objects in the image is created. Then, we need to discard objects touching the border of the image. This is done by scanning each of the four outermost rows or columns in the output image. If a pixel in the output image is found to have a value other than -1 (the pixel is not part of the background), the pixel is assigned a value of -1 and all pixels in the output image having the same labeled value as the initial pixel are assigned a value of -1 . The status of the object being discarded is also modified to indicate that the object is not going to be considered for further processing.

5.5 Image Representation

(i) Area

To determine the area of each object in the image, we scan the output image from top to bottom, left to right. For each pixel value encountered with a label

other than -1 , we increment by 1 the area of the object having the same ObjectID as the pixel labeled value.

Note that for red blood cells containing a 'hole' in the thresholded image (Figure 5.1), the area of the hole will not be included in the total area of the red blood cell. Thus, the total area of this cell will not be accurate. However, this does not affect the results of classification since the area of a red blood cell minus the area of its hole is still much greater than the area of a blood platelet.

(ii) Perimeter and Circularity

To compute the circularity of a cell, we need to determine its perimeter. Hence, for each object labeled previously, we extract the pixels making up the inner boundary of the object. The boundary tracing algorithm [8] works as follows:

1. The starting pixel (P_0) is initially the topmost pixel in the object, and we define a variable *direction* whose initial value is 7. The direction notations are defined in Figure 5.3. We use a counter to keep track of the number of pixels encountered, and a suitable structure to store those pixels.
2. We locate the first 8-connected neighbor (P_1) of P_0 which has the same labeled value (which is also the ObjectID) as that of P_0 . The search is done in an anti-clockwise direction, where the beginning search direction is:
 - a. $(direction + 7) \bmod 8$ if *direction* is even (rotate direction by 45° clockwise)
 - b. $(direction + 6) \bmod 8$ if *direction* is odd (rotate direction by 90° clockwise)

The counter is incremented, and the pixel P_1 is stored.

3. The direction of P_1 with respect to P_0 is the new value of *direction*. Step 2 is repeated with P_1 as starting pixel.

- The searching process is stopped when the starting pixel coincides again with its initial value (i.e. the topmost pixel).

It should be noted that the above algorithm works only for regions larger than one pixel.

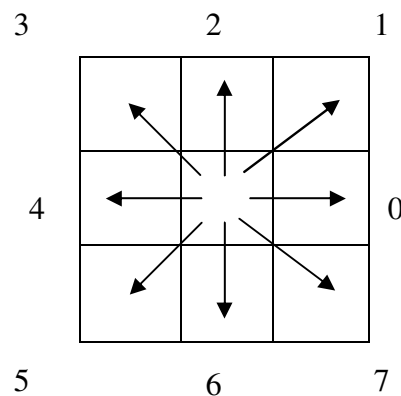


Figure 5.3 : *Direction notation in boundary tracing*

5.6 Classification

To classify each object in the image, we first check the circularity value of the cell under consideration to see if it is a normal blood cell. If the circularity value is above a certain threshold value, the cell is assumed to be normal. The threshold value was obtained by comparing the circularity values of sample blood cells. Normal blood cells were observed to have a circularity of 1.0 ± 0.2 , while abnormal cells have a circularity value lower than this range. Therefore, we have set the threshold value to 0.75: a cell is normal if its circularity is above 0.75. Sample circularity values can be found in Appendix I.

If the cell is a normal blood cell, we use the training values read from the file ("data.txt") to determine to which category the cell belongs. This file contains the mean values of the areas of each blood cell type, and the nucleus area to

whole area ratio for white blood cells (Details about the file structure can be found in Appendix II).

The distance measure, d , between the area of the cell under consideration (previously obtained) and each of the four mean values is computed and the minimum distance value is found.

$$d = | \text{area of cell} - \text{mean area} |$$

If the minimum distance value indicates that the cell might be a white blood cell (lymphocyte or phagocyte), we use the nucleus area to whole area ratios read from the file to distinguish between the two cells. To count the area of the nucleus, we need to extract only the cell under consideration from the whole image. This is done by finding the leftmost and rightmost column value and the topmost and bottommost row value of the cell. With these values, we can form a rectangle (a subpart of the image) which encloses the whole cell. We apply median filtering and threshold this subpart of the image so that now it is the nucleus which is the object that needs to be distinguished from the background (Figure 5.4). In this case, the background is now the area surrounding the cell plus the cell cytoplasm. Again, the thresholding value has been determined empirically until a suitable value is observed. The nucleus is given the lowest intensity value while the background is given the highest intensity value. To obtain the area of the nucleus, we simply need to count the number of pixels having the lowest intensity value in the subimage. The minimum distance between the ratio obtained and the ratios read from the file determines whether the cell is a lymphocyte or a phagocyte.

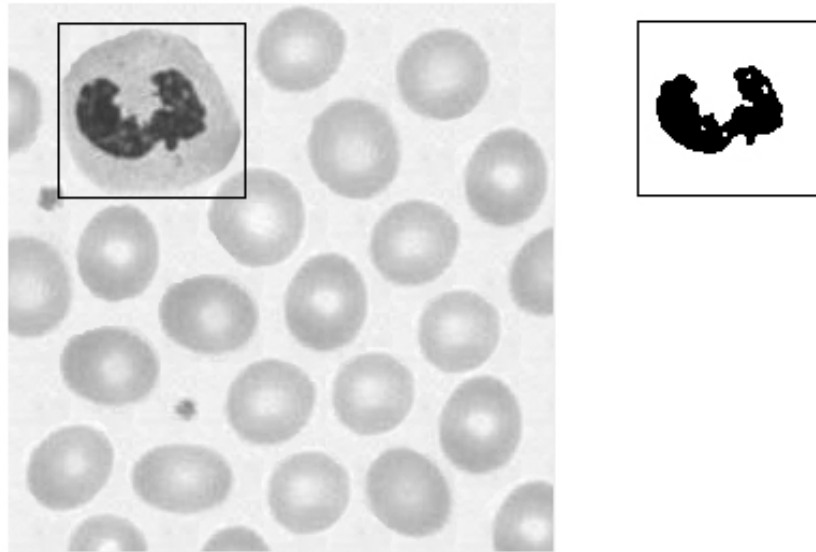


Figure 5.4 : *Thresholding image subpart to get nucleus*

5.7 Counting

As each cell is classified, we need to keep track of the number of each of the different types of blood cells that are present in the image. Each cell type is associated with a variable which keeps track of the count for this particular type of cell. Initially all count variables are set to zero. Each time a cell is identified into its proper category, the associated count is incremented.

The results are displayed to the user and also written to a file “cellcount.txt” to allow subsequent analysis and comparisons. The results displayed to the user include the number of each cell type found in the image, and also the average number per mm^3 of each cell type found in a normal adult man. We calculate the latter values as shown below:

$$\text{Average number of cell type / mm}^3 = \frac{\text{Number of cell type in image} * 10^9}{\text{Volume of blood shown in image}}$$

Note that the volume of blood shown in image is in cubic micrometers (μm^3), where $1 \text{ mm}^3 = 1,000,000,000 \mu\text{m}^3$.

5.8 Training

Training involves determining a mean value for the area of each type of blood cell and if the cell is a white blood cell, to determine its nucleus area to whole area ratio. For each image input for training, we applied median filtering and thresholding is then done with the same threshold value as determined in section 5.2. We do not need to label the training image as there would be a single cell present, and we can consider that the process of thresholding yields a labeled image. The area and circularity values for the cell are then counted and calculated using the same methods as discussed in section 5.5.

To calculate the *mean* area for the cell type under consideration, we multiply the corresponding current mean area value for that cell type by the number of samples used for training for that cell type (both values are stored in the file “train.txt” – the structure of this file is described in Appendix II). To this sum, the current area obtained for the cell in the training image is added, and the new mean can thus be calculated. The number of images used for that cell type is also incremented and stored in the file, along with the new mean area value. The same process is applied if the cell is a white blood cell to get the mean nucleus area to whole area ratio.

Values obtained during the training process are stored to three files:

- (i) “data.txt” file, which is used for classification purposes.
- (ii) “train.txt” file, which contains the same values as “data.txt”, and also the mean circularity and the number of training images used.
- (iii) “details.txt” file, which stores all values obtained while a training image is processed.

5.9 Class Description

Class Image

Attributes	Description
int width	Width of image
int height	Height of image
int maxgray	Maximum graylevel in image
double **matrix	Graylevel values of pixels in image

Operations

(i) void MedianFilter ()

- Applies median filter to image.

(ii) void Threshold (int thresholdval)

Input: a threshold value.

- For each pixel in image, assign lowest intensity to pixel having intensity less than threshold value, else assign highest intensity.

(iii) void Erosion ()

- Performs morphological erosion to a binary image.

(iv) void Dilation ()

- Performs morphological dilation to a binary image.

(v) int labelImage (int **labeling)

Input: An array of same size of image to be labeled, with all its values initially set to -1.

- Label each object found in image with a unique identifier.

Output: Number of objects in image.

(vi) **void removeObj (int i, int j, int **labeling, int ID)**

Input: a pixel location (i, j) in the labeled image (represented by matrix labeling) and the ObjectID (ID) of the object to which the pixel (i,j) belongs to.

- Label all 8-connected neighbors of pixel (i,j) that have a label of value ID, with value -1.

(vii) **void checkConnectivity (int i, int j, int **labeling, int ID)**

Input: a pixel location (i, j) in the labeled image (represented by matrix labeling) and the ObjectID (ID) of the object to which the pixel (i,j) belongs to.

- Label all 8-connected neighbors of pixel (i,j) that have intensity 0 with value ID.

Class Object

Attributes	Description
int area	Area of object
int ID	Unique identifier for each object
int x,y;	(x,y) is topmost pixel location in image
char * type	Type of blood cell: "R" - RBC "L" - Lymphocyte "H" - pHagocyte "P" - Platelet "O" - Other (abnormal cell)
int ** pixlist	List of pixels making boundary of object
int perimeter	Length of boundary
double circularity	Circularity of object
bool status	status=false: object touches image border else status = true

Operations

(i) **void initpixlist** (int x)

Input: Number of pixels x

- Initializes array of pixels that make up object boundary.

(ii) **void countPerimeter** (int **labelimg)

Input: A labeled image, *labelimg*

- Count the number of pixels that make up the object, updating values of pixlist and perimeter.

(iii) **void calculateCirc** ()

- Compute circularity value of object based on formula in Figure 4.1.

(iv) **int classify** (Image img)

Input: The image in which the object to be classified belong

- Determines to which category the cell belongs.

Output: The type of blood cell (0 for RBC, 1 for lymphocyte, 2 for phagocyte, 3 for platelet, 4 for other (abnormal)).

Class cellView

Operations

(i) **void OnAutocount()**

Extract area, and circularity of each object present in image displayed and classify and count each object in image displayed.

(ii) **void OnTraining()**

Extract area, circularity, and nucleus area to whole area ratio for a sample image for training, and save values to a file.

5.10 Interface

When a user opens an image, he has to input the image path name and the volume of blood that is going to be processed. This volume is the amount *before* the dilution of the blood sample.

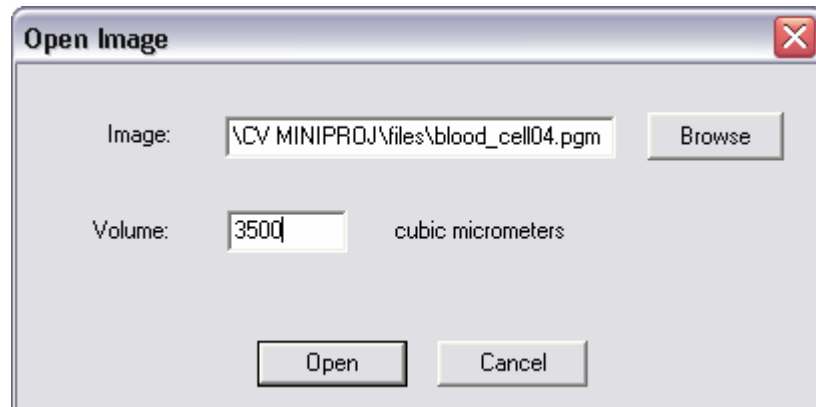


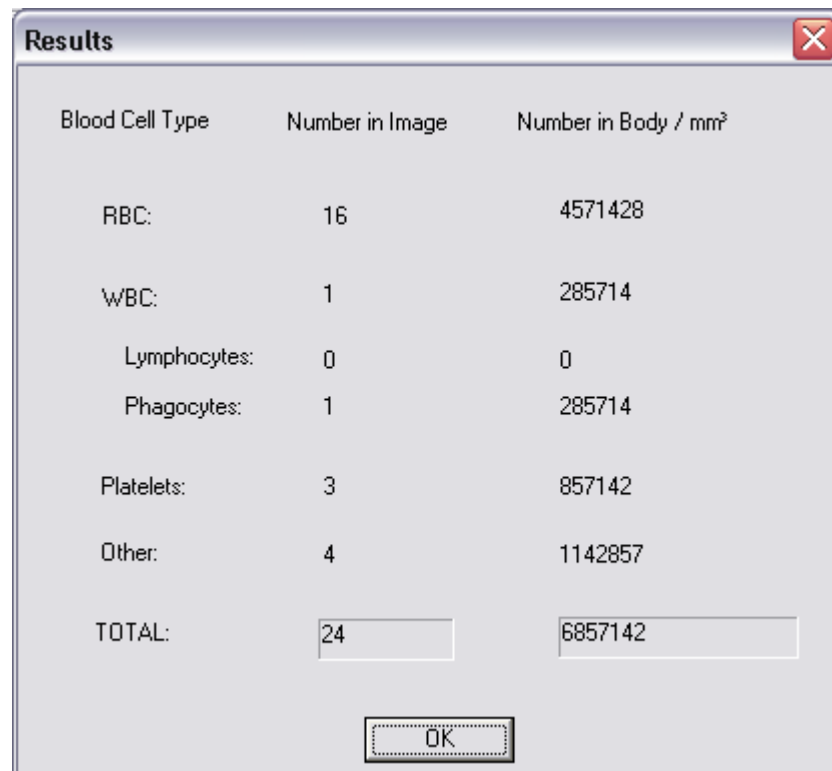
Figure 5.5: *Opening an image*

Once the image is displayed, the user needs to choose Auto Count from the menu to obtain the count of the number of each cell type in the image.



Figure 5.6: *The displayed image and the 'Auto Count' menu*

The results are displayed on a dialog box.



Blood Cell Type	Number in Image	Number in Body / mm ³
RBC:	16	4571428
WBC:	1	285714
Lymphocytes:	0	0
Phagocytes:	1	285714
Platelets:	3	857142
Other:	4	1142857
TOTAL:	<input type="text" value="24"/>	<input type="text" value="6857142"/>

OK

Figure 5.7: *Displayed results*

Each cell in the original image is labeled according to which category the cell belongs to. 'R' denotes a red blood cell, 'L' a lymphocyte, 'H' a pHagocyte, 'P' a platelet, and 'O' for other abnormal cells.

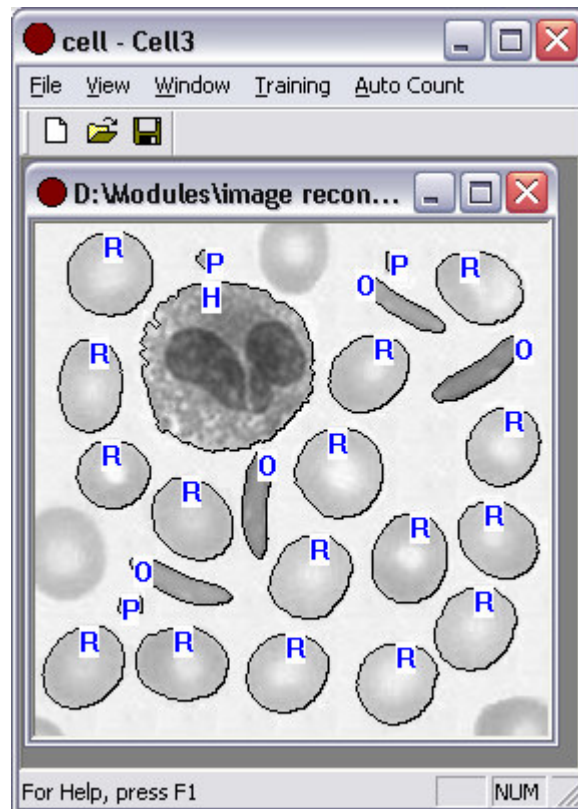


Figure 5.8: *Labeled image*

To input an image for training, the user chooses 'Training' from the menu. The image path is specified and the user needs to choose the type of blood cell from the drop-down list.

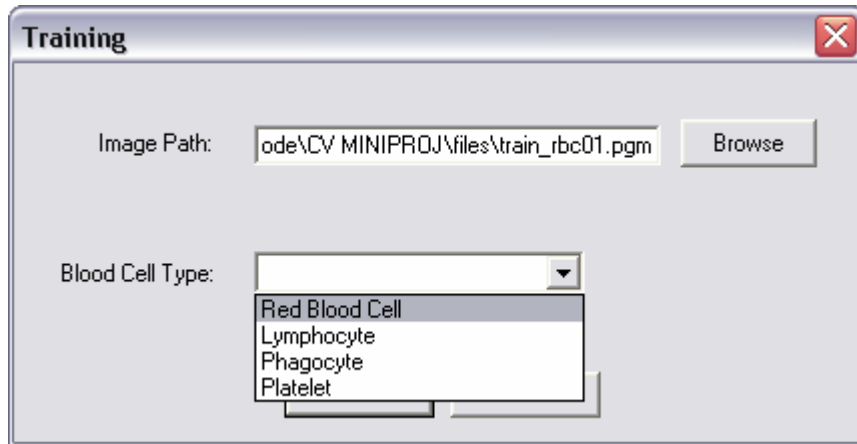


Figure 5.9: *System training*

6. TESTING

Testing involves running the program with the aim to find any existing faults and deficiencies. The system has been tested using a number of sample blood images using the black box approach. The outputs were checked for errors against the expected results. For the set of test images used, the system yields a very good performance, provided that our initial assumptions and conditions are respected. The different types of cells were correctly identified and labeled and the blood counts correctly tabulated.

The figures below present some of the results obtained during testing. Figures (a) are the original images while figures (b) are the images labeled by the system.

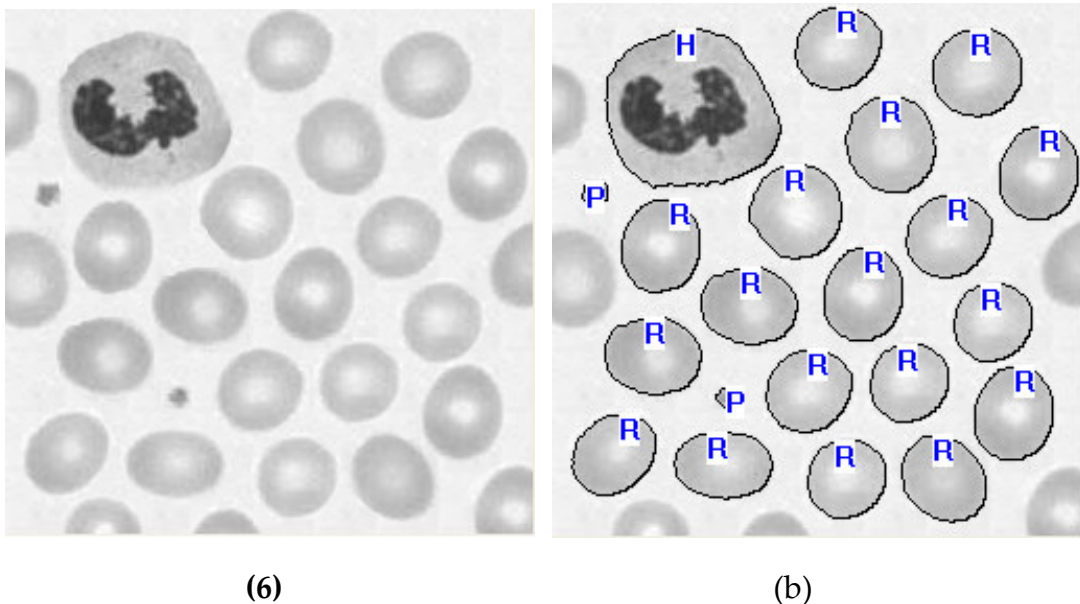
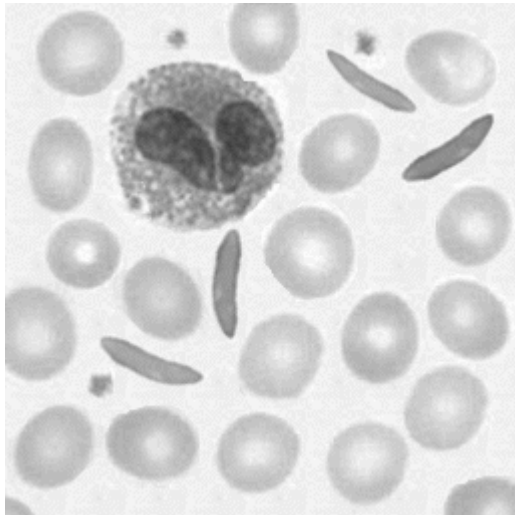
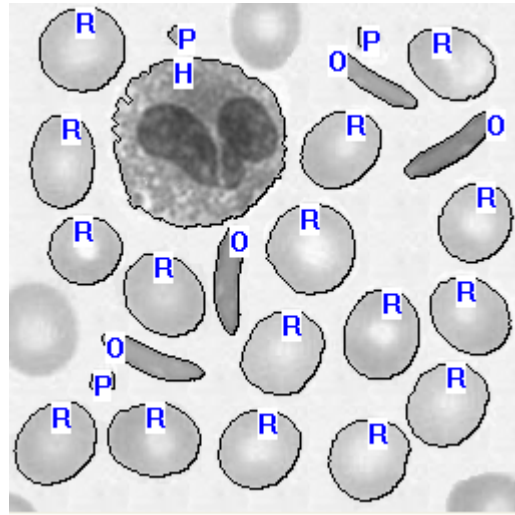


Figure 6.0: *Correct identification of red blood cell, platelets and phagocyte*

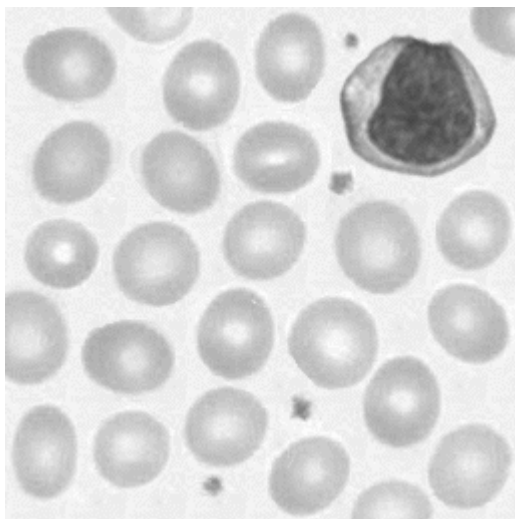


(6)

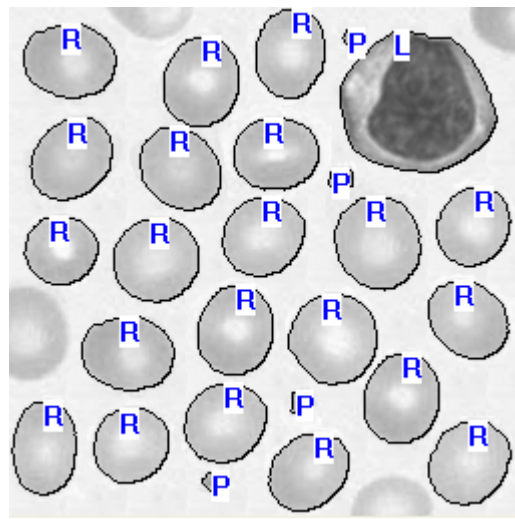


(b)

Figure 6.1: Correct identification of red blood cells, platelets, phagocyte and abnormal cells



(6)



(b)

Figure 6.2: Correct identification of red blood cells, platelets and lymphocyte

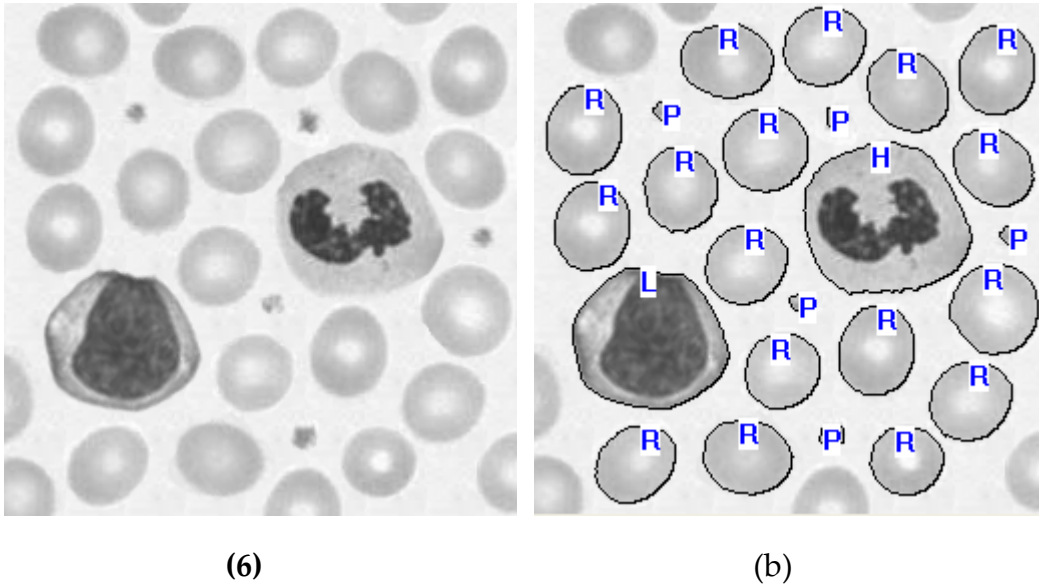


Figure 6.3: *Correct identification of red blood cells, platelets, lymphocyte and phagocyte*

As we have said earlier, the image should not contain touching or overlapping cells. Figure 6.4 shows how the results would be distorted if this were the case.

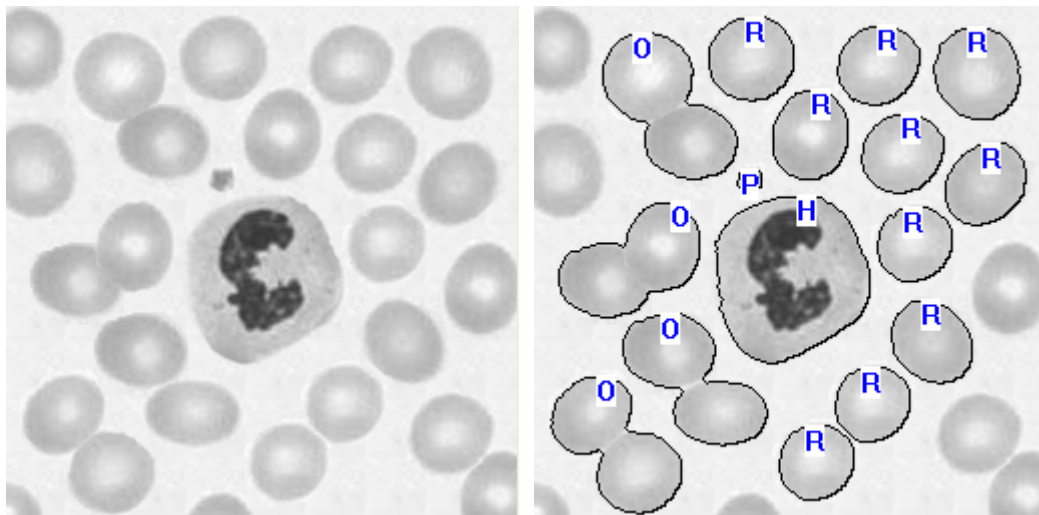


Figure 6.4: *The system takes touching or overlapping cells as one, yielding inaccurate results.*

7. CONCLUSION AND LIMITATIONS

The developed system satisfies the initial set of objectives. It is able to successfully distinguish between the different types of blood cells and correctly identify and classify them. However, the system is not a perfected version. There is scope for lots of improvements to increase efficiency of the system. Due to lack of time and resources, we have taken some assumptions which means that the program may not yield satisfactory results in all cases. Some identified limitations of our system are described below.

We have identified only normal blood cells belonging to these categories: red blood cells, lymphocytes, phagocytes and platelets; and we have only detected abnormal blood cells. However, we have not identified to which category the abnormal blood cell belongs, and we have not classified all the subcategories of white blood cells (we have taken only the two main categories into consideration).

We have used images where the blood cells do not overlap or touch each other. This factor can otherwise substantially affect our results (as shown in Figure 6.4). In that case, we will need to determine whether two or more cells are overlapping and to separate them so that classification is possible.

Due to time constraint, system training has been done on only a small sample of images just to show that the application works. However, to yield better precision during classification, the system needs to be trained with a large number of samples. But we have written the training part such that further training is possible.

8. REFERENCES

- [1] Gonzalez R.C. & Woods R.E 1992, '*Digital Image Processing*', Addison-Wesley Publishing.
- [2] TO Y. K. 1994, '*Basic Principles in Biology (Book 2)*', 5th Edn., Hung Fung Book, pp. 1-11.
- [3] Pratt W. K. 1991, '*Digital Image Processing*', 2nd Edn., John Wiley & Sons.
- [4] Patel A. H. 1994, '*A Manual of Medical Laboratory Technology*', Navaneet Prakashan Limited.
- [5] Parums D. V. 1996, '*Essential Clinical Pathology*', Blackwell Science, pp. 191 – 223.
- [6] Mukherjee K. L. 1988, '*Medical Laboratory Technology*', Volume I, Tata McGraw-Hill.
- [7] Lewis R.. 1990, '*Practical Digital Image Processing*', Ellis Horwood.
- [8] Sonka M., Hlavac V. & Boyle R. 1999 , '*Image Processing, Analysis and Machine Vision*', 2nd Edn., PWS Publishing, chapter 5 p. 142.

APPENDIX I - Normal blood cell values obtained during training

Image Number	Cell Type ¹	Area ²	Circularity	Nucleus to whole area ratio
0	1	4361	0.920484	0.487044
1	1	4248	1.05446	0.501412
2	1	4007	0.812139	0.436985
3	1	4358	0.950768	0.486691
4	2	4435	0.943824	0.373619
5	2	5079	0.93693	0.262059
6	2	4887	0.929792	0.290771
7	2	4331	0.906705	0.382129
8	2	4334	0.929968	0.383018
9	2	5075	0.965561	0.262266
10	3	118	0.974906	-
11	3	130	1.07405	-
12	3	60	1.20637	-
13	3	130	1.07405	-
14	3	133	1.15743	-
15	0	1363	0.953885	-
16	0	1354	0.906541	-
17	0	935	0.95362	-
18	0	1183	0.893337	-
19	0	1344	0.969307	-
20	0	1294	0.947549	-
21	0	1209	0.956963	-
22	0	1502	0.936059	-
23	0	1162	0.891243	-
24	0	1363	0.953885	-
25	0	1327	1.0178	-
26	0	1362	0.953185	-
27	0	1363	0.953885	-
28	0	1641	0.994474	-

Mean Values

Cell Type	Area	Circularity	Nucleus to whole area ratio
0	1311	0.948695	–
1	4243	0.934462	0.478033
2	4689	0.935463	0.325644
3	113	1.09736	–

Abnormal Cells

Cell Type	Area	Circularity
Acanthocyte	1651	0.34848
Elliptocyte	1282	0.716004
Sickle	1062	0.467263

¹ Cell Type

0: Red blood cell

1: Lymphocyte

2: Phagocyte

3: Platelet

² Area

Area is for images magnified x1000.

APPENDIX II




Structure of file “data.txt”

This file contains the mean area values for each type of blood cell in the first four lines in that order: red blood cells, lymphocytes, phagocytes, and platelets. The next two lines represent values for nucleus area to whole area ratio for lymphocytes and phagocytes respectively. This file is used during classification and updated during training.

Structure of the file “train.txt”

This file contains information gathered during system training. Its structure is basically the same of that of the file “data.txt” except that for each of the first four lines, not only the mean value is stored, but also the mean circularity and number of images used for training in that particular cell category.

A typical row for red blood cells in the file might look like this:

1314.43	0.948695	14
		
Mean Area	Mean Circularity	Number of RBC images used

The last two lines represent values for nucleus area to whole area ratio for lymphocytes and phagocytes respectively as in the file “data.txt”.

Structure of the file “details.txt”

This file stores all values computed during training i.e. for each training image, the type of blood cell under consideration, its area, its circularity and its nucleus area to whole area ratio (where applicable) is stored in one row of the file.

Structure of the file "cellcount.txt"

This file contains the results obtained after an image is analyzed for classification. The pathname of the image analyzed, and the average count of each type of blood cell per mm³ is appended to the file. Below is a typical record that is stored after an image is analyzed.

```
-----  
Image: D:\MINIPROJ\files\blood_cell101.pgm  
  
RBC:                5142857  
LYMPHOCYTES:        0  
PHAGOCYTES:         285714  
PLATELETS:          571428  
OTHER:              0  
-----
```

Note that the file "data.txt" and any image to be analyzed need to be in the same directory. The files "data.txt", "train.txt" and any image used to train the system needs to be in the same directory too.