Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 48 (2012) 9410-9413

ISSN: 2229-712X

Analyzing blood cell images to differentiate WBC and counting of linear & non-linear overlapping RBC based on morphological features

Arivu Selvan.K and Sathiya Moorthy
School of Information Technology & Engineering, VIT University, Vellore-632 014.

ARTICLE INFO

Article history:
Received: 3 April 2012;
Received in revised form:
15 June 2012;
Accepted: 9 July 2012;

Keywords
Image segmentation,
Cell counting, Noise Removal,
WBC,
RBC.

ABSTRACT

In this paper we propose a new set of features based on morphology for total RBC count in given blood sample is explained along with classification of WBC. For the diagnosis of any disease first requirement is complete blood count i.e total number of RBC, WBC and platelets in given blood sample. And if there are excess of any of these types or any of these is few in number then this gives the doctor a basic idea that the person is not healthy for sure. To do this manually is very tedious. Here RBC and WBC are first differentiated and then counted through some algorithm. Implementation here is done through MATLAB.

Introduction

Blood consists of majorly red blood cells (RBC), white blood cells (WBC), and platelets. Each has its own function in our body and is equally important. Function of WBC is to protect our body against diseases i.e. responsible for our immune system. Platelets help in clotting of blood. And RBC help in regulating oxygen. Here we will concentrate only on WBC and **RBC.**

Both WBC and RBC have fixed count in our body. If their count is less than the ideal count then it is an indication that our body is not healthy.

Basic difference between RBC and WBC is their size. This parameter is chosen to differentiate them and then counting them separately. By having a view on above picture we can say that RBC are generally round and WBC are hardly round. Also we can get an idea that how many pixels are their in the diameter of a cell. We set a threshold value for diameter or area of the cells. If the cell is greater than that particular area or diameter then it is WBC else RBC. First we focus on counting of RBC.

RBC is generally round in shape. Basic problem which we face during counting is the overlapping of the cells. Cells overlap each other and appear as one object when actually there is more than one. So sometimes even for a lab technician it becomes difficult to differentiate. And hence manual counting becomes error prone and tedious. The best solution is to develop software that can give more accurate counting result. Basically, that software should be able to distinguish overlapping cells. Here we have presented an algorithm to implement it. The accuracy of this algorithm declines when large number of cells that is more than 5 cells overlap. [NOTE: A good blood sample has not more than three cell overlapping.]

Classification of white blood cells:

Coming to the classification of WBC then they are majorly of 5 types:

a. lymphocyte-which has round shaped nucleus
b. monocytes-bean shaped nucleus
c. eosinophil-it has bilobed nucleus
d. neutrophil-it has multi lobed nucleus
e. basophil-it also has multi lobed nucleus but not visible due to the large dots in plasma area.

As coming from left to right in the following figure you can see neutrophil, monocytes, basophil, lymphocyte and eosinophil. Each has different function in our body.

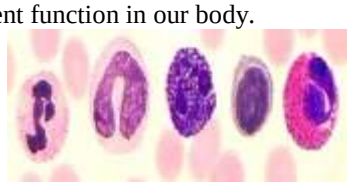

Fig 1. Classification of WBC

For RBC count we will be explaining through following image:

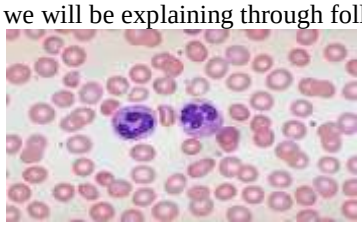

Fig 2. Sample image

Here as it is clearly visible that largely stained purple bodies are WBC. And also they are much bigger in size than RBC.

Once we recognize WBC on the basis of their shape we remove it from given picture and make it of background color. This can be done with MATLAB inbuilt function "imerode". Now we are left with only rbc.

Layout of the steps we will be following:

Segmentation→noise removal→removal of boundary elements→counting algorithm

Segmentation:

For proper segmentation following steps should be taken:

1. Convert the given image to gray image.
2. Increase the contrast of the image using imadjust and adapthisteq functions.
3. Then convert this image to binary using im2bw (f, graythresh(f)) command for any image f.

Tele:
E-mail addresses: arivupsg1@gmail.com

4.Use bwareaopen command to reduse noise that is small dots (generally salt and pepper noise).
5. fill the holes using imfill command.
6.remove the cells touching border as they are not completely included in it.
On the resultant image we will apply our logic explained in later sections.

Result after following above steps

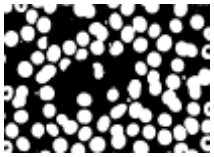

Fig 3. Grey Scale image

But this result is without using 'imclearborder' and 'bwareaopen'. And it is clearly visible that it is noisy.

Noise Removal:

In above figure you can easily find 'salt and pepper' noise. While counting even these will be considered as one object and hence giving wrong result by increasing total count. To avoid this we use 'bwareaopen' command which removes object of specific area. This command when applied to above figure gives following result:

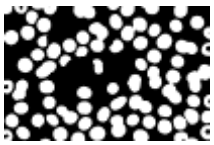

Fig 4. Image without noise

Now we should remove the objects touching border as they are not fully included in the sample. For this we should reset the value of pixel at first row, first column, last row, and last column which are white to zero along with their neighboring pixels which are white.

Neighborhood:

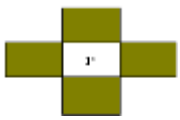

Fig 5. 4*4 neighbours

Consider above figure. Let p be a pixel. Then all green shaded portions are its neighbors. This is said to be 4*4 neighborhoods. While if it's diagonal positions are also included as its neighbors then it is said to be 8*8 neighborhoods as shown below.

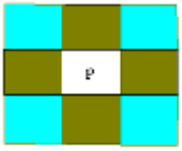

Fig 6. 8*8 neighbours

Whenever we mean neighborhood then it means 8*8 neighborhood else it would be explicitly mentioned.

Proposed Logic:

Divide the given image into blocks of equal size preferably. Then for every block do the following:

Scan every pixel such that first row is scanned first then second and so on. And whenever it finds white pixel it label that pixel. If that pixel is the neighbor of already scanned pixel then whatever the label is given to its neighbor, same label is given to that pixel. Every time when new pixel is found which is not the neighbor of already scanned pixel then label count is increased by 1 and assigned to it. Now the question is how overlapping is overcome by this logic? This can be understood by taking following two cases of overlapping. (NOTE: there are even more cases of overlapping but if these two are understood then rest can easily be implemented.)

Case 1:

Fig. 7

Here we see the first row for which white pixel comes. It is taken as a new object. In the same row one more white pixel is there, since both are not the neighbors of each other so labeling is done with different digit. This is the condition corresponding to line 1 in figure given below. Then the row where these circles intersect is scanned (line 2 in given figure). For the pixel which is at the intersection point of these shapes its neighbors have different labels. Its left neighbor has label value as 1 and right neighbor has value as 2. So this is surely the case of overlapping and is overcome if you write f(a,b)= = 0 where a,b is the position where these circles intersect. This condition is checked for every PIXEL in a row.

Fig. 8

Case 2: In some cases above logic fails. For example we consider the following image.

Fig. 9

This condition is checked for all objects of a row. We have the basic idea about the width of the cell. Now if the width of the cell is greater than that then it is another case of overlapping. And is simply overcome by increasing the label count by 1 or 2 according to the diameter relation.

Similarly we have an idea that at what vertical length does maximum width of the cell occurs?

If it is greater than that or maximum repeats itself then also overlapping occurs. And we can increment the label count accordingly. This is done in case of following type of overlapping.

Fig. 10

Proposed Algorithm for RBC counting:

Assumption: f is the image and [m n] is its size.

Main function:

begin

1. set the value of threshold.
2. set value of variable t ← 0 and label← 1.
3. M← zeros (m, n).
4. Flag3 ← flag () and flag4 ← flag ().
5. repeat for a=2 to m-1
6. repeat for b=2 to n-1
7. if f(a,b) = =1 % white pixel is found
8. if f(a,b-1)= =0 and b+1<n and f(a,b+1)= =0 % new object which doesn't have any

% neighboring white pixel

9. label← label +1
10. M(a,b)← label
11. q=b
12. repeat while q-1>0 and a+1<m and f(a+1,q-1)= =1
13. q← q-1
14. else
15. call linear_overlapping()
16. if b<n
17. if f(a,b+1)= =0 %end of that object in that particular row
18. call dia_overlapping()

end

Linear_overlapping ()

Begin

1. set variables i , r ,count3 and q to 0.
2. set matrix K such that K←zeros (4)
3. if f(a+1, b-1)= =1
4. M(a+1, b-1)←M(a, b)
5. q←b-1
6. repeat while count3<=20 and q>1 and f(a+1, q-1)= =1
7. M(a+1, q-1)←M(a,b)
8. q←q-1
9. count3←count3+1
10. if f(a-1, b-1)= =1
11. i←i+1
12. K(i)← M(a-1, b-1)
13. if f(a-1,b)= =1
14. i←i+1
15. K(i)←M(a-1,b)
16. if f(a-1, b+1)= =1
17. i←i+1
18. K(i)← M(a-1, b+1)
19. if f(a, b-1)= =1
20. i←i+1
21. K(i)← M(a, b-1)
22. if i= =1 and M(a,b)= =0
23. M(a,b)←K(i)
24. else
25. u=max(K(1:i))
26. v=min(K(1:i))
27. if u = =v
28. M(a,b)←i
29. else
30. f(a,b)←0
31. r←a

end

dia_overlapping()

begin

1. s←b
2. repeat while s-1>0 and f(a,s-1)= =1
3. s←s-1
4. y←flag3(label)
5. if f(a, b-1)= =1
6. if b-s>threshold
7. c←(b-s)/threshold
8. if c>1.4
9. if y= =1
10. flag3(label)= =2
11. t←t+1

end

flag()

begin

1. one←ones(300)

end

(Note: here sum of t and label gives total count.)

Similarly threshold condition for vertical overlapping (case 3 in above examples of overlapping) can be written and implemented.

Classification of WBC (for the purpose of counting):

Initially we removed WBC from given sample to compute RBC count. Now we consider the original image. First we classify that given cell is WBC or not on the basis of shape and size. If it is WBC then its type is identified and then corresponding counter is increased by 1.

CONVEX -HULL' property

Here to classify lymphocytes we have used 'convex hull' property. An object is convex if any two vertices in it when joined then form an edge which is contained within it.

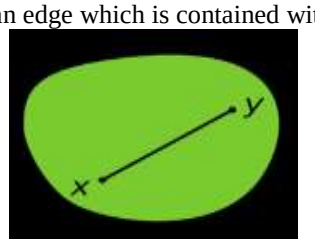

Fig. 11 Convex Shape

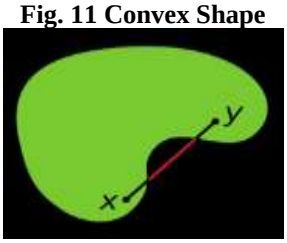

Fig. 12 Convex Shape

RBC's Fig 13. Counting of Overlapped

Proposed Algorithm for WBC classification

Following steps are to be followed:

Segmentation of wbc (image) → basophil or not → lymphocyte or not → monocyte or not → eosinophil or neutrophil.

Step 1: Segmentation

Nucleus should be clearly visible in comparison to the background that is nucleus is needed to be extracted. For that you can use normal boundary conditions or the given commands

```
I_eq =adapthisteq(f)
I_adjust=imadjust(I_eq)
Mask=imextendedmax (I_adjust , 23)
Mask=imfill(Mask, 'holes')
Mask=bwareaopen(Mask, 8 )
Se=strel ('periodicline',1,[1 -2] )
Eroded=imerode (Mask ,Se)
Perimeter=bwperim (eroded)
F(perimeter)=255
```

When these steps are applied to the image on which we are working (shown in the beginning), we get the following image:

Fig. 14

Now the resultant image is ready for the classification. We check out no. of objects in nucleus, if it is more than one then it means that they are dots of plasma area so it is basophil else proceed with next tests.

If no. of objects = = 1

Get convex image of nucleus using regionprops command. Subtract original image and convex image. Resultant image will have objects only when object is not circular. If the object is circular then its convex image is same as the object itself. So if no object in resultant matrix then it is 'lymphocyte'.

Else

Calculate area using regionprops command. If area > 60% of area of the cell then it is 'monocytes'.

Else

Remove small linkages (very thin strip) between lobes.

If no. of objects after removing small linkages = =2

Compare the area of two objects. If they are approximately same then it is 'eosinophil'.

Else if area of one is double or greater than double then it is 'neutrophil'.

Else if no. of objects >2 or = =1

Then it is 'neutrophil'

Conclusion:

Above proposed rbc count algorithm when compared with existing neural network algorithm then they both have approximately same accuracy. Neural network gives 81% accuracy. And above algorithm has 75-80% accuracy. As far as wbc classification is concerned the neural network gives better solution because above algorithm is applicable for healthy wbc that is in the case of variation of shape of nucleus than ideal shape it may fail.

References

[1] D.H.Tycko, S.Anbalagan, H.C.Liu and Leonard Ornstein "Automatic Leukocyte classification using Cytochemically stained smears", The journal of histochemistry and cytochemistery,Vol.24,No.1.178-194,1976

[2] Nipon Theera-Umpon, "Automatic White Blood Cell classification using biased-output neural networks with morphological features", Thammasat Int.J.Sc.Tech., Vol.8, No.1, January-March 2003

[3] Su Mao-Jun, Wang Zhao-bin, Zhang Hong-Juan, Ma Yi-de, "A New method for blood cell image segmentation and counting based on PCNN and Autowave", ISCCSP 2008, Malta, 12-14 March 2008

[4] Saif Zahir, Rejaul Chowdhury and Geoffery W.Payne, "Automated assessment of erythrocyte disorders using Artificial neural network", IEEE International symposium on signal processing and information technology 2006

[5] Mihir S.Sewak, Narender P.Reddy and Zhong-Hui Duan," Gene Expression based leukemia sub-classification using committee neural networks",Bioinformatics and Biology Insights 2009:3 89-98

[6] H.Ranganathan and N.Gunasekaran," Simple Method for Estimation of Hemoglobin in Human Blood Using Color Analysis", IEEE Transactions on Information Technology In Biomedicine, Vol.10, No.4, October 2006

[7] Wei He, M.S.,Joseph Wilder, " Nucleus Shape Recognition for An Automated Hematology Analyzing System", Proceedings of the second joint EMBS/BMES Conference, Houston,TX,USA-October 23-26,2002

[8] Tobias Bergen, Dirk Steckhan, Thomas Wittenberg and Thorsten Zerfab ,"Segmentation of Leukocytes and Erythrocytes in Blood smear images", 30th Annual international IEEE EMBS Conference , Vancouver, British Columbia, Canada, August 20-24, 2008