An automated procedure for Red Blood Cell counting

Kiana Seraj

December 2023

1 Introduction

In healthcare, blood testing is as one of the most significant medical examination tests. In pathology labs, red blood cells are counted in microscopic images to diagnose the diseases. Counting RBCs (Red Blood Cells) in images of blood cells can play an important role in detection as well as to follow the treatment process of number of diseases [2]. RBCs are also known as erythrocytes. The function of RBC is to carry oxygen and collects carbon dioxide from a lungs to the cells of body. They contain protein called hemoglobin. The presence of inner and outer layers of protein gives red color to blood. In abnormal RBC counting, Low RBCs count can be a sign of Anemia, Leukemia(a type of blood cancer), Malnutrition and a high red blood cell count can be a sign of Dehydration, Heart disease and etc [4].

Counting and examination of blood cells manually from microscopic images is tedious, time intense and entails a lot of technical expertise. Hence arises a need to come across for automated blood cell detection and counting system that can facilitate physician for diagnosing diseases in fast and efficient way. In this project, a technique has been introduced to count the RBCs automatically. The major contribution of proposed work is the use of ImageJ [5] that can automate the manual method of detecting and counting RBCs. The most chief step in here is the segmentation of blood cells, based on a previous work by [1]. Additionally, the most important notion, which must be considered in the process of counting of cells, is that counting of the cells is built on the experimental evidence, in the absence of disorders, the discocyte shape of human RBCs is approximately 7.5 µm to 8.7 µm [3]. In this project all blood cells are assumed to have uniform sizes. The image processing includes extraction of cells by segmentation which is done by finding a threshold with which the standard deviation of the cell sizes become minimum. Next step which is also noteworthy is that exact cell count is based on the accurate segmentation of blood cells, the uncertainties intrinsic to the image processing procedure in order to classify the overlapped objects can obstruct the image analysis procedure, hence, for considering overlapped cells separately, watershed divides them into discrete cells. Eventually the process of counting happens quickly with the accuracy of around 90%.

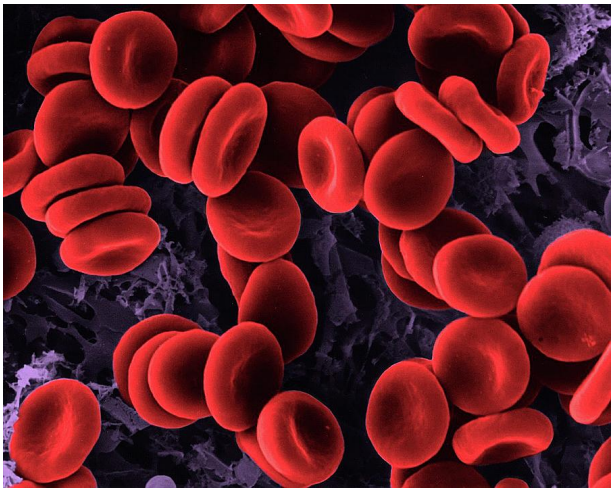


Figure 1: Human red blood cells in physiological isotonic saline solution (0.9% NaCl) imaged with scanning electron microscopy (SEM).

2 Methods

In this section the procedure that is implemented for the automated counting of red blood cells (RBCs) is explained: the procedure is addressed by newly developed ImageJ macros, which returns data that us then post-processed with a Python script.

As pointed out in the Sec.(1), one expects that in standard conditions RBCs have roughly the same areas. Therefore, RBC areas that are too large or too small with respect to the mean RBC area could be due to an incorrect segmentation. On the basis of that, an algorithm is developed that minimises the standard deviation of the RBC areas.

2.1 ImageJ Macros

The following procedure is implemented in the ImageJ macros "rbc_finder.ijm", which is fully reported in App.A. Let's consider the flowchart reported in Fig.(3): by assumption, the *Smear Image* is an RBG microscopic image of a group of RBCs, most often contaminated by the presence of other red cells such lymphocytes and platelets.
Initially, the image is split in the three color components. This is justified by the usual color of RBCs in microscopic images: as it can be seen in Fig.(2) the red component is the most saturated, both in the foreground and in the background, thus providing little signal to noise ratio. A similar argument goes for the blue component. On the contrary, the green component maximises the distance between background and foreground peaks, enhancing the signal to noise ratio.

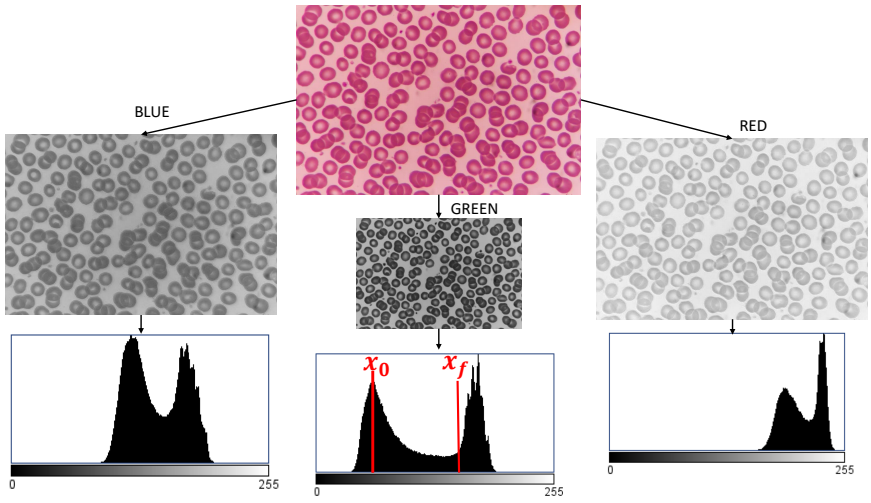


Figure 2: Splitting the original image into the three colors

To further increase the distance in grey level between the foreground and the background peaks, an histogram stretching (*Enhance Constrast* in ImageJ) is performed with a number of saturated pixel given in percentage of 1%, which is found to be a good compromise for our purposes.

Four parameters must be identified by the user, which will be the input of the procedure:

- x_0, x_f: the minimum/maximum grey level value of the green image histogram from which the threshold is performed
- **min_size, max_size**: the minimum/maximum size (in pixel^2) that is accepted for a RBC. Objects with smaller/larger area are assumed to be different cells and therefore are not counted.

The algorithm starts with a for loop construct, increasing the variable *grey_level* from x_0 up to x_f of unitary step and for each loop the following instructions are executed (see Fig. 4):

1. Threshold is performed at the value set by *grey_level*
2. Holes inside the cells are filled with the automatic ImageJ technique (*Fill Holes*). As the RBCs have a concave disc-shape, the interior has a brighter color with respect to the edges of the RBCs and hence is thresholded as the background. This step compensates for this effect.

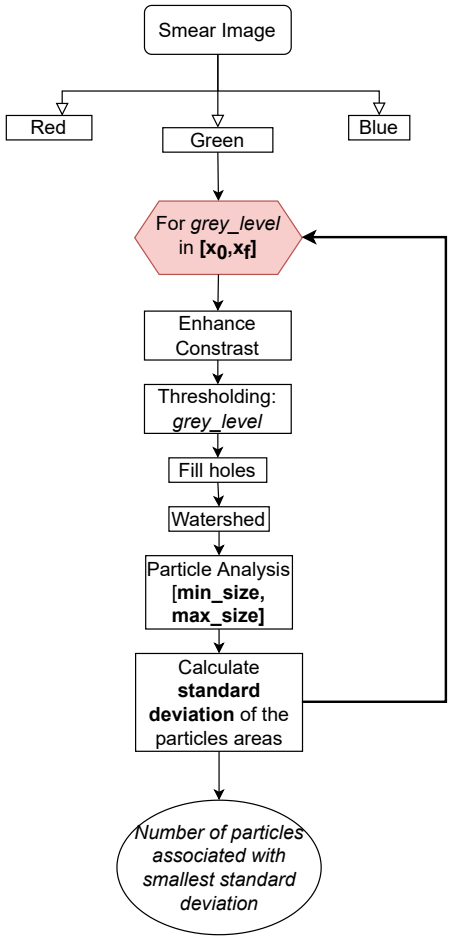


Figure 3: Flowchart of the algorithm that is implemented by ImageJ. Notice that the last step (written in italic font) is performed by the Python script.

3. Apply the automatic watershed segmentation technique of ImageJ (*Watershed*) [5]. This technique can be understood in a four steps procedure: (1) the image is mapped into an Euclidean Distance Map, which maps each pixel into the distance of the pixel to the nearest obstacle pixel, such as a boundary pixel. Then, (2) the features of the image are eroded until they all vanish. Eventually, (3) the features are grown until their edges touch; (4) edges are hence removed. It is important to note that is technique works best only with objects that do not overlap too much.

4. Count and measure the sizes of all the RBCs through the ImageJ function *Analyze Particles*. This routine scans the image looking for objects with sizes comprised in [**min_size**,**max_size**] and with a shape that ranges from a line to a perfect circle. Once it finds one object, it measures its dimensions (with the *Wand* tool) and covers it with same background color, so as not to count it again.

5. Save an array of data in which every line is associated with a cell and the columns are geometrical parameters, such as the area, the degree of circularity, etc. The first 6 rows associated to the first 6 RBCs identified by the *Analyze Particles* algorithm are reported in Tab.(1)

2.2 Python script for data handling

Data returned by the ImageJ macros can be used to identify the particular *grey level* value of threshold that minimises the standard deviation of the areas array (see the first column of Tab.1) and the respective number of RBCs (N_{min}^{rbc}). In addition to that, the Python code also evaluates the uncertainty associated with N_{min}^{rbc}. To do so, it is assumed that in correspondence of the minimum of the standard deviation, the distribution of the areas of the RBCs, namely the histogram, follows a Gaussian distribution, because only random (not systematic) variations of the areas are expected. Therefore, the histogram of the areas is interpolated with the Gaussian function:

$$N(A) = N_0 + Be^{-\frac{(A-A_0)^2}{2c^2}}, \tag{1}$$

Particle	Area	StdDev	Min	Max	Circ.	AR	Round	Solidity
1	290	0	255	255	0.829	1.457	0.686	0.949
2	119	0	255	255	0.363	1.672	0.598	0.642
3	338	0	255	255	0.442	1.200	0.834	0.687
4	154	0	255	255	0.340	2.153	0.465	0.665
5	681	0	255	255	0.876	1.164	0.859	0.947
6	468	0	255	255	0.898	1.204	0.831	0.942

Table 1: Table representing the results related to the first six particles identified by the ImageJ macros for a certain threshold value.

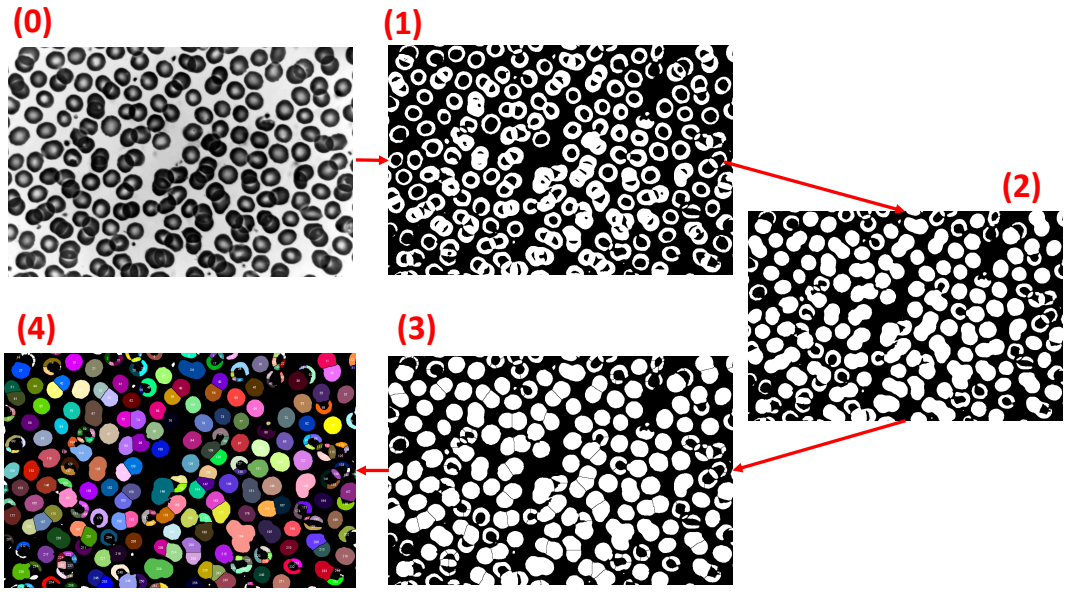


Figure 4: Steps of the algorithm are graphically described: (0) green component of the image, (1) threshold of the image at a given threshold value (*grey_level*), (2) fill holes and (3) watershed and eventually (4) particles are found and separated by different colours.

where A is the area of the RBC, A_0 the mean area, c the standard deviation and N_0 the offset in number of RBCs (expected zero). Hence, the number of RBCs whose areas are found to be beyond three times c is defined as the uncertainty on N_{min}^{rbc}. This definition is based on the idea that for these outlier RBCs the algorithm did not work properly: they are either too small (for instance because watershed has split two overlapping RBCs in more than two pieces) or too large (watershed failed to separate two or more overlapping cells). Therefore, the overall uncertainty is defined as the number of RBCs that the algorithm failed to segment in a proper way. The result of the procedure is graphically reported in Fig.(5): the two areas highlighted in yellow are beyond the three standard deviations of the Guassian function that interpolates the histogram data. As it can be seen, the more cells with the sizes in these areas, the higher the uncertainty.

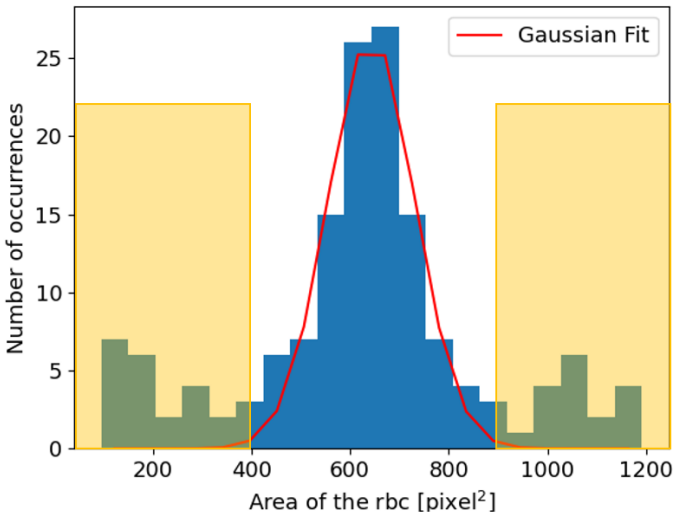


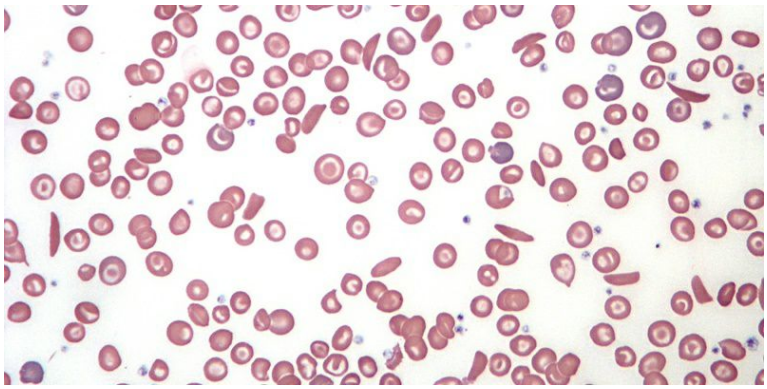
Figure 5: Graphical explanation of the uncertainty evaluation: all the cells that fall outside the three standard deviations (yellow regions) are summed to form the overall uncertainty.

3 Results

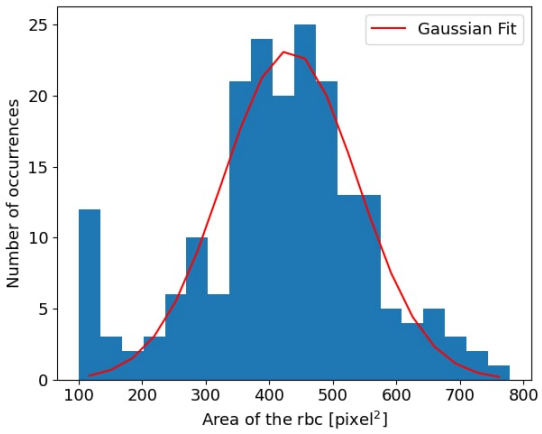
In this section we get to see the outcomes of applying the process defined above, the macros and then the python script. In order to evaluate the performance of the suggested algorithm, we applied it to three sample images with different conditions (e.g. different sizes, shapes of cells and extent of overlapping).

3.1 RBC Sample 1

As shown in Fig 6a we can see there is not much overlapping among the blood cells. They have a vast extent of circularity that is from exact round to elliptical shaped. We also find many small cells (platelets) which are not to be considered in the counting procedure. After performing the algorithms. The following Standard Deviation and Number of cells graphs are obtained for a range of different gray level thresholds Fig 7. Based on the Fig 6b we observe that the variation of the RBC area follows a Gaussian Distribution. Hence, for this threshold we can analytically deduce that at the lowest Standard Deviation we get the optimal result of the number of RBC cells.
We can see that the Standard deviation first decreased to the minimum number of cells and then increased slightly. So the number of cells doesn't change drastically.
The final number of RBC's estimated by algorithm (having lowest Standard Deviation as represented with red point in Fig 6b) is 199 $\pm$ 10, whereas the cells manually counted are 216 $\pm$ 2.



(a) The first RBC microscopic image sample.



(b) The variation of the histogram of the first RBC microscopic image sample.

Figure 6: The RBC microscopic image 6a and the variation histogram 6b.

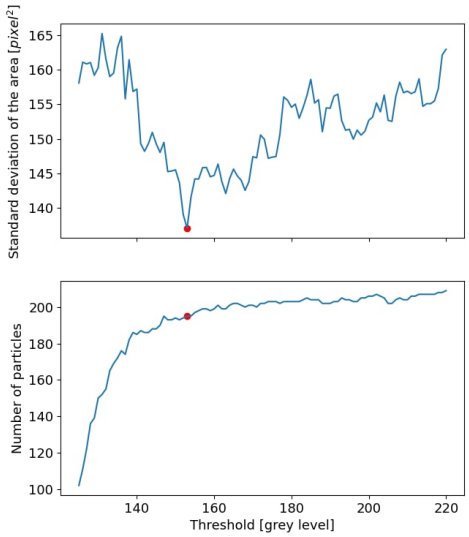
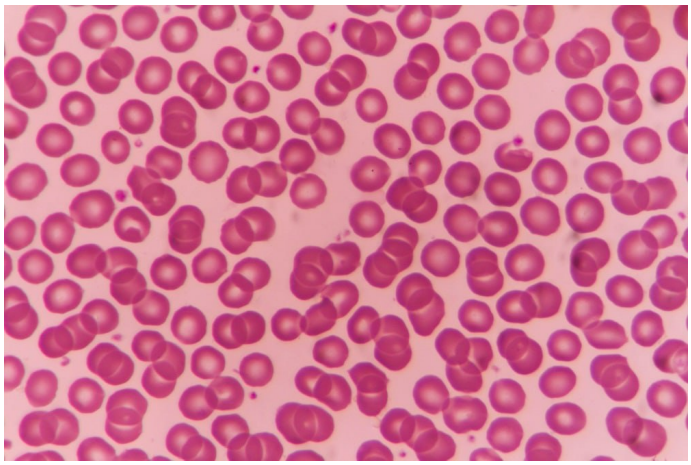


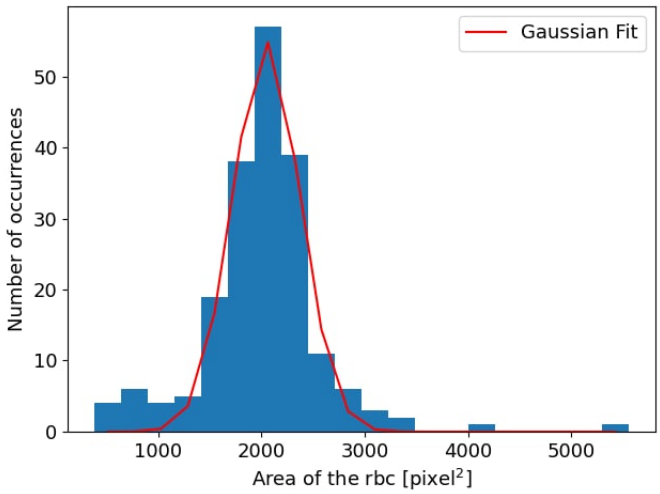
Figure 7: The image shows the Standard deviation and the number of particles for various thresholds. The red point denotes the lowest possible SD for first RBC sample.

3.2 RBC Sample 2

According to Fig 8a the conditions are different from the previous figure such as the number of overlapped cells are much more and the background and foreground have approximately the same colour range. Due to the similar colour the histogram peaks related to the background and foreground are really close together which makes operations for image processing difficult in this limited region. By enhancing the contrast, we can increase the distance between the two peaks. That allows better implementation because it extends the range over which we can scan with algorithm. This allows better image segmentation because the probability of assigning a pixel to a wrong group is decreased. As depicted in Fig 9 the Standard deviation first decreases to the minimum and then increase. The number of cells also follow the similar trend as the Standard deviation which comes from the fact that threshold might not be much effective in this case. However, as shown in the Fig 8b the error related to counting the number of cells is high because there are many cells with an area of greater than three times the standard deviation. So a possible interpretation can be that the cells are highly overlapped.
The final number of RBC's estimated by algorithm (having lowest Standard Deviation as represented with red point in Fig 8b) is 196 ± 21. The number of cells manually counted is of 198 ± 2.



(a) The second RBC microscopic image sample.



(b) The variation of the histogram of the second RBC microscopic image sample.

Figure 8: The RBC microscopic image 8a and the variation histogram 8b.

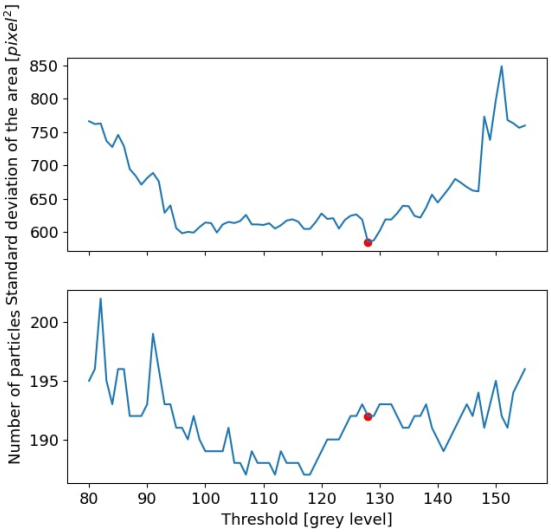
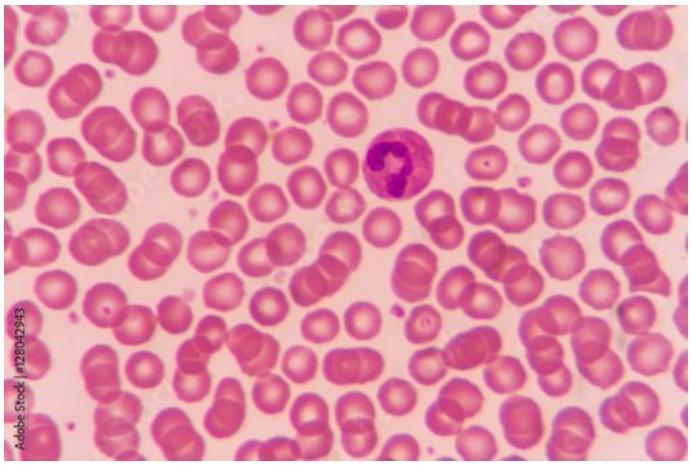


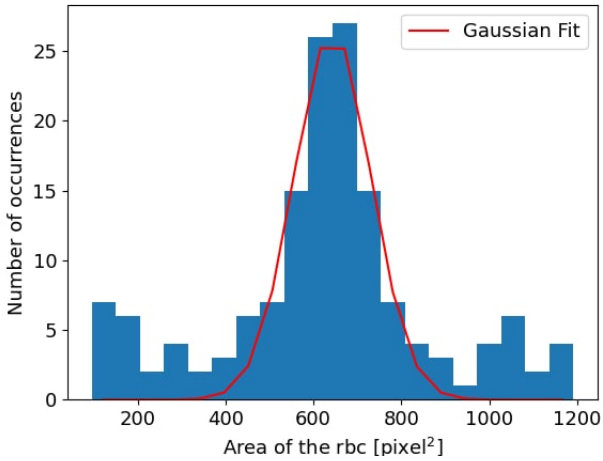
Figure 9: The image shows the Standard deviation and the number of particles for various thresholds. The red point denotes the lowest possible SD for second RBC sample.

3.3 RBC Sample 3

Lastly, as seen in this image 10a we find that not only small cells but also cells which are larger than blood cells can create problem in counting the number of cells. There are also other challenges such as the high number of overlapped, unfilled cells and the colour similarity in background and the foreground of the image. In order to solve the size problem we used maximum area threshold along with the smallest area threshold. The low signal to noise ratio can be sloved by using contrast enhancement. After the processing we got the minimum Standard deviation as in 11 which erratically changed as well as the number of particles, we can see a huge variation in the number of particles. We can see there are a lot of outliers in the histogram as seen in the Fig 10b.
The final number of RBC's estimated by algorithm (having lowest Standard Deviation as represented with red point in Fig 10b) is 151 ± 30. On counting with eyes the number of Red Blood Cells came out to be 158. The error came out to be 4.6%.



(a) The third RBC microscopic image sample.



(b) The variation of the histogram of the third RBC microscopic image sample.

Figure 10: The RBC microscopic image 10a and the variation histogram 10b.

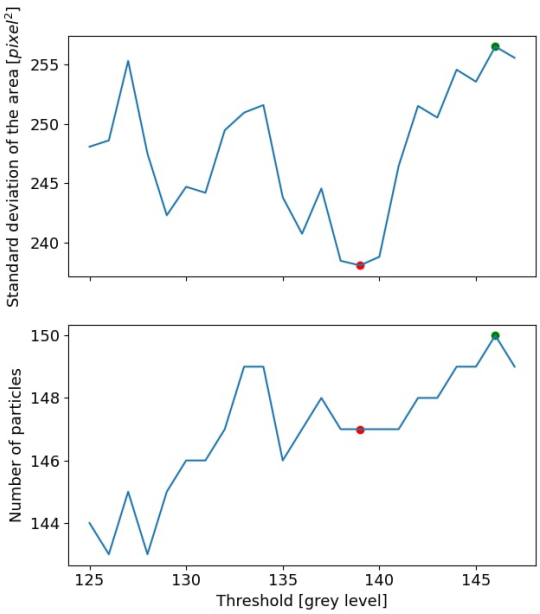


Figure 11: The image shows the Standard deviation and the number of particles for various thresholds. The red point denotes the lowest possible SD for third RBC sample.

4 Conclusions

As we can see from the standard deviations reported in the Sec.(3), the procedure we have developed has been able to count the number of cells in three different images with a relative error of 10 % on average.
In order to check the results we counted RBCs with our eyes in each image, obtaining the following values:

Sample	Cells counted by hand	Cells counted by algorithm
1	216 ± 2	199 ± 1
2	198 ± 2	196 ± 21
3	158 ± 2	151 ± 30

Table 2: Summary of the number of RBCs counted in each image both manually and using the algorithm

The results obtained counting the cells manually agree with the number of cells obtained with the algorithm in sample 2 and 3. In sample 1 we obtained an higher number of RBCs when counting manually; this is due to the presence in the image of some smaller cells that are not counted by the algorithm but may have been considered in the manual count.
It's also important to highlight the fact that, as stated in the previous sections, we based the optimization criterion on a well known biological principle: healthy RBCs have almost the same size [3]. Starting from this, we managed to optimize the counting by choosing the output value of the counting associated with the minimum standard deviation of the area.
Furthermore, we managed to write our own macro in order to make the analysis automated. Using the macro make it possible to analyze different images requiring little knowledge about the image and the number of cells contained in it.

Despite the effort to separate overlapping cells, the area of two (or more) overlapped cells is calculated as the union of them and not as the sum. Since the optimization principle is based on cells' dimension, this fact may affect the analysis as we can see from the higher values of the uncertainty in the number of cells in the images where more overlapping cells are present. Sample 1 (3.1) is the one with the smallest relative error (0.5 %) since few cells are overlapped in the image, so the problem of considering the area of overlapped cells as the union of the two isn't present. The situation is different in sample 2 (3.2) and sample 3 (3.3) where many cells are overlapped and their area is not properly calculated, resulting in both smaller and bigger values for the cell's surface: as shown in figure(8) and figure(10) many cells have and area that falls beyond three standard deviations. The presence of many outliers lead in both case to a larger uncertainty than the one of Sample 1.
In addition to the macro we used python to choose the optimal value of the threshold, as explained in Sec.(2). This

means that two different programs are needed to perform the analysis, the only use of ImageJ is not sufficient; moreover the analysis requires a large computational effort.
Despite the limitation related to the overlapping cells and computational effort our algorithm provides an automated way to count RBCs in images based on a verified biological principle with a relative error around 10 %.

Appendix

Appendix A: ImageJ Macros

It is herein reported in the macros "rbc_finder.ijm".

```
// open the image_name at path-to-dir
open("C:/path-to-dir/image_name");

// select the image window and split it into three channels
selectWindow("image_name");
run("Split Channels");

// close red and blue images
selectWindow("image_name (blue)");
close();
selectWindow("image_name (red)");
close();
selectWindow("image_name (green)");

// stretch the histogram with 1% of saturated pixels
run("Enhance Contrast...", "saturated=1");

// produce the histogram of the image
run("Histogram");

// input the data
x0 = getNumber("Please, type the initial threshold",1);
xf = getNumber("Please, type the final threshold",2);
min_size = getNumber("Please, type the minimum area [pixel] to be analyzed",3);
max_size = getNumber("Please, type the maximum area [pixel] to be analyzed",4);



// start of the for loop (from x0 to xf, with a step 1)
for (grey_level=x0; grey_level < xf + 1 ; grey_level++)
{
    open("C:/path-to-dir/image_name");
    run("Split Channels");
    selectWindow("image4.jpg (blue)");
    close();
    selectWindow("image4.jpg (red)");
    close();
    selectWindow("image4.jpg (green)");
    run("Enhance Contrast...", "saturated=px_sat");

    // set the threshold value "grey_level"
    setThreshold(0, grey_level);
    // keep the background in back (cells are white)
    setOption("BlackBackground", true);
    // converting the thresholded image into a mask
    run("Convert to Mask");

    // while gaps inside the cells
    run("Fill Holes");
    // separate overlapping cells
    run("Watershed");
    // count particles and measure the area
    run("Analyze Particles...", "size=min_size-max_size display");
    // Saving results (create "results" dir before executing)
    saveAs("Results", "C:\\path-to-dir\\results\\Results_"+i+".csv");
    // Close Results window
    close("Results");
    close();
}
```

References

[1] Sherif Abbas. Microscopic images dataset for automation of rbcs counting. *Data in brief 35-40*, 2015.

[2] Gulpreet Kaur Chadha, Aakarsh Srivastava, Abhilasha Singh, Ritu Gupta, and Deepanshi Singla. An automated method for counting red blood cells using image processing. *Procedia Computer Science*, 167:769–778, 2020.

[3] Han J, Lim CT, Suresh S Diez-Silva M, Dao M. Shape and biomechanical characteristics of human red blood cells in health and disease. *MRS Bull*, 35(5):382–388, 2010.

[4] Naotaka Hamasaki and Masaaki Yamamoto. Red blood cell function and blood storage. *Vox sanguinis*, 79(4):191–197, 2000.

[5] Wayne Rasband Tiago Ferreira. Imagej user guide.