Determining Numbers of Red Blood Cells in Blood Smear Images Using Blob Detection

Asst.Prof.Dr.Nidaa Flaih Hassan **Asma Ibrahim Hussein**

Computer Science Department
University of Technology/Baghdad

Ministry of Higher Education and
Scientific Research /Baghdad

Abstract

Counting of red blood cells (RBCs) in blood smear images is significant to detect and follow the process of treatment in some blood diseases like Anemia, malaria, leukemia …. etc. The determination and counting of RBCs manually is boring and time-consuming, but it can be simplified by means of automatic analysis. In this paper, a new algorithm is proposed to segment and count of RBCS in microscopic blood cell images using Blob detection, the proposed algorithm consists of three stages: preprocessing, edges detection and segmentation. Determining the number of RBCs is made by using (65) samples microscopic images and the best counting rates (93.6 %) is achieved when using Blob detection method compared with conventional manual counting method.

Keywords: Blood smear, Blob detection, RBC, Counting, segmentation.

تحديد عدد كريات الدم الحمراء في صور مسحة الدم باستخدام تحديد الفقاعة

أ.م.د. نداء فليح حسن
الجامعة التكنولوجية / قسم علوم الحاسوب

م.م. اسماء ابراهيم حسين
وزارة التعليم العالي والبحث العلمي

المستخلص:

إن تحديد عدد خلايا الدم الحمراء (كريات الدم الحمراء) في صور مسحة الدم أمر مهم لكشف ومتابعة عملية علاج بعض الأمراض مثل فقر الدم وسرطان الدم ... الخ , وان تحديد عدد كريات الدم الحمراء يدويا هي عملية مملة ومستهلكة للوقت ولكن يمكن أن تكون مبسطة عن طريق التحليل الآلي . في هذا البحث تم اقتراح خوارزمية جديدة لتقسيم وتحديد عدد كريات الدم الحمراء في الصور المجهرية لخلايا الدم باستخدام تحديد الفقاعة ، تتكون الخوارزمية المقترحة من ثلاث مراحل وهي عملية تجهيز و معالجة الصور ، تحديد الحواف ومرحلة التقسيم ، في تحديد عدد كريات الدم الحمراء تم استخدام (٦٥) عينة من الصور المجهرية لخلايا الدم وأفضل معدلات العد هي (٩٣,٦%) قد تحققت عند استخدام طريقة تحديد الفقاعة مقارنة مع الطريقة التقليدية للعد والفرز اليدوي.

1. Introduction

One of the significant information which physicians used to diagnose some blood diseases is the numbers of RBC. In fact the determination of RBCs aids the body to im plement the tasks involved with continue to live or exist, also it's so expensive and time - consuming. In addition, it can be a hard job to obtain accurate result by using manual counting of red blood cells [1].

In the lab counting the number of RBCs for each cubic millimeter from blood smear using a microscope is performed where millions of cells have been found and the range of: males, 5.4 ± 0.8; females, 4.8 ± 0.6. The result reports of counting RBCs by laboratories is based on doctors'

recommendation so as to helps the diagnosis of the blood diseases that infect patients [2].

In the following sections, canny edge detection is presented which will be used for edge detection; in addition Blob detection which is considerd as one of the segmentation methods is also presented.

2. Canny Edge Detection

This mechanism of detecting the edge is proposed by John Canny in 1986, it has been greatly beneficial for the edge detection operations. Actually, J. Canny detailed a mechanism to generate the edge, the main point that it states is noting the fact that the three most influential implementation factors for the successful edge detector must be as follows:

1. Providing a high quality detecting that has high possibility to report an edge, where there isn't an edge in case of low possibility of reporting.
2. Providing quite a precise indexing, which means that the pixels that are denoted to be edge pixels must have high approximation to the real index of that edge.
3. Responding just for one time to a single edge (in a 1-D signal) [3].

The following describe Canny's edge detection steps:

Step 1 : Reducing Noise through smoothing**.**
Step 2 : Finding gradients**.**
Step 3 : Non maximum suppressions.

Step 4 : Hysteresis thresholding **[4].**

3. Segmentation using Blob Detection

It means to segment cells (normal or abnormal) from RBC image using Blob detection technique , this mechanism is rapid and simple that can be used to segment object from image by using connected components labeled each object and give different color for each object about other. [5]

Blobs can be of different colors, shapes, areas and perimeter. Before actual Blob detection ,the image undergoes various processes such as preprocessing and edge detection, image consisting of various objects.Through all these objects, deteching one object needs that its own color is different from other objects. In Blob detection, the objects can be segmentation from others in the same image, after the image passes many processes [6]

4. The Proposed Algorithm

RBCs in digital images could be counted in this proposed algorithm by detecting Blob. The following Algorithm (1) illustrated the proposed algorithm. Figure (1) illustrated block diagram of the proposed algorithm which is composed of four stages; the following sections illustrated the task of each stage.

Algorithm (1): Proposed Algorithm for Detection Cells of RBC Images

Input : *Color image RBC image*
Output : *Segmented cell image and counted cell*

Begin
Step1: *OpenFileDialog pic = new OpenFileDialog (); //open image of RBC.*
if (pic.ShowDialog() == DialogResult.OK)
Input. Image = Image.FromFile (pic. Filename);
Step2: *Remove noise using Median Filter.*
Step3: *Edge detection of cell using sobel by apply algorithm (3.2).*
Step4: *Convert RBC image into black and white image by apply in algorithm (3.4).*
Step5:*Detec each cell with particular color using function ConnectedComponentsLabeling();*
Step6: *Segment each cell from image by using blob detection*
Int point min xy , max xy ; //get bounding rectangle of the points list.
Points cloud, Get Bounding Rectangle (edge points, out min xy , out max xy);
Int point cloud size = max xy – min xy; // get cloud s size.
Double point center =min xy+(Double point) cloud size/2; //calculate center point.
Float radius= (float) cloud size.x+ cloud size.y)/4; // calculate radius.
Float mean Distance =0;
Step7: F*or (int I =0, n = edge points. Count; i<n; i++) // Calculate mean distance between provided edge points and estimated circles edge.*
{Mean Distance+= Math. Abs (float) center. mean Distance to (edge points [i]) –radius);
} End I.
Step8: *Blob Counter blob counter = new Blob Counter ( ); // locate object using blob counter*
Blob counter. Process Image (bitmap);

Blob [ ] blobs= blob Counter. Get object Information ( );
Graphics g = Graphics.form Image (bitmap); // create Graphics object to draw on the image and a pen.
Pen red pen = new Pen (color.Red, 2);

Continue of Algorithm (1)
***Step 9: For** (int I = 0; n = blobs.lenght; i<n; i++); // check each object and draw circle around objects, which are recognized as circles.*
{ List< Int Point> edge points = blob counter .Get blobs Edge points (blobs[i]);
Point center;
Float radius;
If (shape Checker .Is Circle (edge points , out center, out radius))
{ g. Draw Ellipse (red pen,
(int) (center.x- radius),
(int) (center .y- radius),
*(int) (radius*2),*
*(int) (radius*2), }*
} End for
End

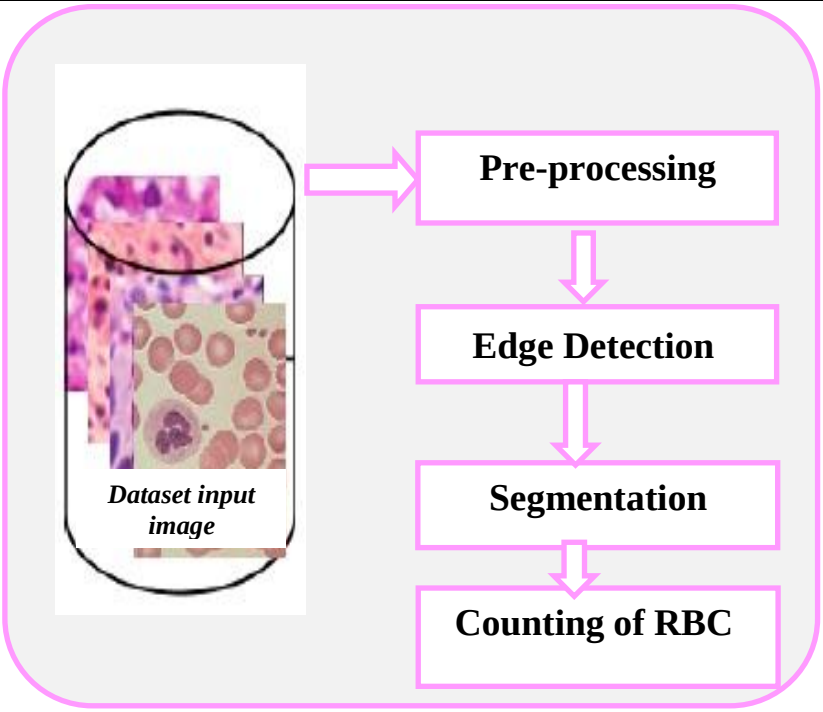


Figure (1): Block diagram of the Proposed Algorithm

4.1 Preprocessing Stage

This process is considered the first stage that consists of the following steps:

- **Acquisition of RBC images:** RBC images acquisition from hospitals (Al-karma Teaching Hospital, Ibn Al-Balady Hospital and Al-Yarmouk Teaching Hospital). It is captured with high resolution and JPEG format**.**
- **Reduction of Noise:** in this step noise is reduced by using mean filter and median filter based on C#.NET programing to implement these filters to get better image quality, with preserving on detailed data and integrity of edges image. Algorithm (2) illustrates median filter.

Algorithm (2): Median filter

Input : *Color image RBC image*
Output : *cell image removed noise.*

```
Begin
Step1: OpenFileDialog pic = new OpenFileDialog ();          //open
 image of RBC.
      if (pic.ShowDialog() == DialogResult.OK)
       Input. Image = Image.FromFile (pic. Filename);
Step2: if (input.Image != null)
      {
         Bitmap sorim = (Bitmap)input.Image;
         Bitmap outim = new Bitmap(sorim.Width, sorim.Height);
         BitmapData bmd1, bmd2;
         Rectangle rect = new Rectangle(0, 0, sorim.Width, sorim.Height);
         bmd1 = sorim.LockBits(rect, ImageLockMode.ReadOnly,
 PixelFormat.Format24bppRgb);
         bmd2 = outim.LockBits(rect, ImageLockMode.WriteOnly,
 PixelFormat.Format24bppRgb);
         unsafe
         {
Step3: for (int i = 1; i < bmd1.Height - 1; i++)
          {
```

```
byte* ptr1 = (byte*)bmd1.Scan0 + (i - 1) * bmd1.Stride;
byte* ptr2 = (byte*)bmd1.Scan0 + (i) * bmd1.Stride;
byte* ptr3 = (byte*)bmd1.Scan0 + (i + 1) * bmd1.Stride;
byte* ptr4 = (byte*)bmd2.Scan0 + (i) * bmd1.Stride;
```

Step4:

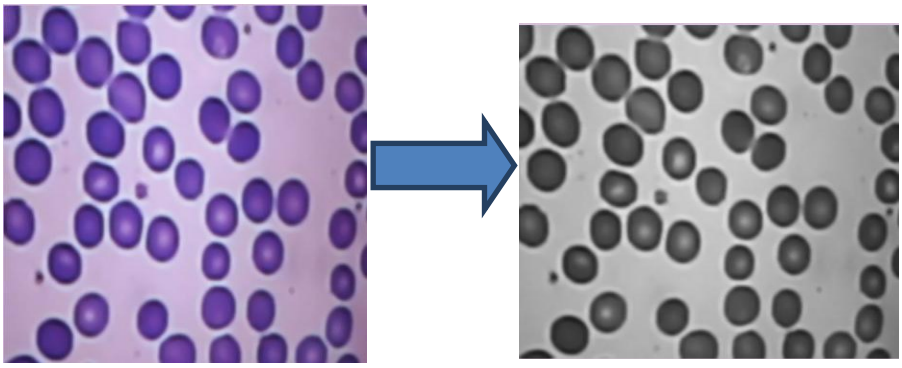
```
for (int j = 3; j < bmd1.Stride - 6; ++j)
{
   List<int> red, blue, green;
   red = new List<int>();
   blue = new List<int>();
   green = new List<int>();
   blue.Add(ptr1[j - 3]);
   green.Add(ptr1[j - 2]);
   red.Add(ptr1[j - 1]);
```

Continue of Algorithm (2)

```
   blue.Add(ptr1[j]);
   green.Add(ptr1[j + 1]);              //sum first row
   red.Add(ptr1[j + 2]);
   //.........
   blue.Add(ptr1[j + 3]);
   green.Add(ptr1[j + 4]);
   red.Add(ptr1[j + 5]);
   //*****************************************************
   blue.Add(ptr2[j - 3]);
   green.Add(ptr2[j - 2]);
   red.Add(ptr2[j - 1]);
   //.........
   blue.Add(ptr2[j]);
   green.Add(ptr2[j + 1]);              //sum of second row
   red.Add(ptr2[j + 2]);
   //.........
   blue.Add(ptr2[j + 3]);
   green.Add(ptr2[j + 4]);
   red.Add(ptr2[j + 5]);
   //*************************************************
   blue.Add(ptr3[j - 3]);
   green.Add(ptr3[j - 2]);               // sum of third row
   red.Add(ptr3[j - 1]);
   //.........
   blue.Add(ptr3[j]);
   green.Add(ptr3[j + 1]);
   red.Add(ptr3[j + 2]);
   //.........
   blue.Add(ptr3[j + 3]);
   green.Add(ptr3[j + 4]);
   red.Add(ptr3[j + 5]);
   blue.Sort();
```

```
                red.Sort();
                green.Sort();
                int b = blue[4], r = red[4], g = green[4];
                ptr4[j] = (byte)b;
                ptr4[j + 1] = (byte)g;
                ptr4[j + 2] = (byte)r;
              }
            }
            sorim.UnlockBits(bmd1);
            outim.UnlockBits(bmd2);
          }
          outgry.Image = outim;
        }
      }
} End for
```

- **Conversion of Image to Gray Level:** the color images are converted to gray scale to simplify process edge detection illustrated in Figure (2).



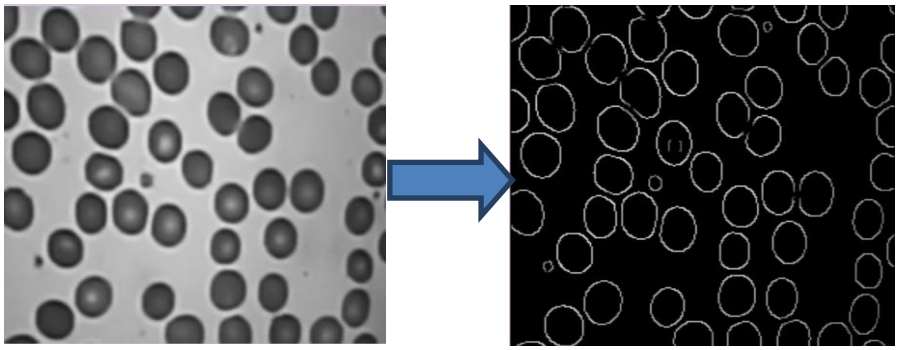
(a) (b)

Figure (2): Convert of RBG image to gray level. a) Original image. b) Gray image.

4.2 Edge Detection Stage

Edge detection for Image reduces the amount of data and preservs the structural characterishc of object in the image. In this paper Canny Edge detector is used to detect edges.

This edge filter includes four steps (smoothing, computing gradients, non-maximum suppression and hysteresis thresholding). Figure (3) shows the canny edge detection of RBC image where edge detection of all cells passes in four steps.

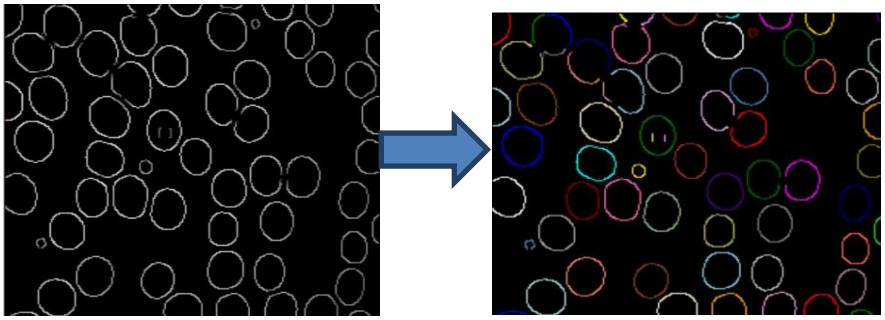


(a) (b)

Figure (3): Canny Edge Detector a) RGB gray image b) RBC edges

4.3 Image Segmentation

In this stage, RBC images are segmented to determine the numbers of regions, each region represents one cell from smear of blood, the concept of blob detection supported on the presumption of all cells edge have the same distance to center, which represent radius. In RBC image some distortion in shape of cells and some edge pixels may be near or remote to center, but this difference in the distance to the center must ation be not big. If is big the shape it is not RBC.Figure (4) shows segmentation using Blob detection.



(a) **(b)**

Figure (4): Segmentation of cells in RBC image, a) Entire image. b) Detection cell using Blob detection

5. Experimental Discussion of the Results

Here we estimate the results of the proposed algorithm on four stages. The total number of RBC sample images is (65). Measurement accuracy of the proposed algorithm depended on the result of the cells counted using the Blob detection compared with the traditional counting; equation (1) is used to calculate the accuracy of the proposed algorithm:

$$\text{Accuracy} = \frac{\boldsymbol{RBCcount}}{\boldsymbol{Actual}} \text{ x } \quad 100\ \% \ldots\ldots (1)$$

Based on 65 RBC images using microscop, the rate of accuracy obtained by comparing the actual number of cells in the image with the image after implementation proposed algorithm. This rate did not reach 100% , because some of the

cells are not counted correctly to complete the cell shape ,and some cells have radius out of the range. captured. Table (1) shows the accuracy rate of counting RBC using Blob detection. The optimal results of counting are achieved with rate (93.6%) compared with conventional manual counting method.

Table (1): Result of RBC Counting using Blob detection

Sample images	***Actual number***	***Number Using algorithm***	***Accuracy***
1	88	84	95%
2	35	33	94%
3	43	39	93%
4	55	53	96%
5	32	29	90%
6	47	45	95%
7	22	20	90%
8	65	63	96%
9	95	90	94%
10	86	82	95%
11	50	46	92%
12	79	77	97%
13	33	31	93%
14	84	80	95%
15	42	37	88%
16	77	70	89%
17	22	20	90%
18	33	32	96%
19	40	35	94%
20	31	29	95%

21	33	30	90%
22	20	19	96%
23	60	56	93%
24	50	49	95%
25	73	70	88%
26	31	27	89%
Sample images	***Actuale number***	***Number Using algorithm***	***Accuracy***
27	44	42	90%
28	54	50	96%
29	20	18	94%
30	66	64	95%
31	87	85	85%
32	23	20	90%
33	50	49	96%
34	42	40	94%
35	24	20	95%
36	30	29	90%
37	25	24	86%
38	43	39	93%
39	55	53	96%
40	32	29	90%
41	47	45	95%
42	22	20	90%
43	65	63	96%
44	95	90	94%
45	86	82	95%
46	50	46	92%
47	79	77	97%
48	88	84	95%

49	35	33	94%
50	43	39	93%
51	55	53	96%
52	32	29	90%
53	47	45	95%
54	22	20	90%
55	65	63	96%
56	95	90	94%
57	95	90	94%
58	86	82	95%
59	50	46	92%
60	79	77	97%
61	79	77	97%
62	33	31	93%
63	84	80	95%
64	42	37	88%
65	77	70	89%

6. Conclusion

In this paper, new proposed algorithm for counting RBCs in images is presented; this algorithm includes four stages: at the first stage preprocessing is applied RBC images, then edge is detected at the second stage , in the third stage the RBC images are segmented using Blob detection, at the last stage the counting of RBC cells in blood images is achieved . Accuracy results indicate that the best accuracy rate (93.6 %) is obtained when using Blob detection compared with conventional manual counting method.

References

[1] Siti Madihah M.,et al. "**Automated Red Blood Cells Counting in Peripheral Blood Smear Image Using Circular Hough Transform**", IEEE, First International Conference on Artificial Intelligence, Modelling & Simulation , 2013.

[2] Alaa H., et al. , "**Automated Red Blood Cell Counting**" , international journal of computer science, VOL. 1, NO. 2, 2012.

[3] Gerhard X. and Joseph N.," **computer vision algorithms in image algebra**", Boca Raton London New York Washington, 2nd Ed, 2000.

[4] Sarah J.," **Iris Recognition System Using Statistical Measurement ",** M.Sc. Thesis, Department of Computer Science, University of Technology, 2014.

[5] Sanjeevi .C, et al., " **Morphology Based Automatic Disease Analysis Through Evaluation of Red Blood Cells"**, Fifth International Conference on Intelligent Systems, Modeling and Simulation, 2014.

[6] Roy F.," Red **Blood Cells and White Blood Cells Detection, Differentiation and Counting using Image Processing**", Presented at the DLSU Research Congress 2015 De La Salle University, Manila, Philippines March 2-4, 2015.