ABSTRACT

AREA AND PERIMETER MEASUREMENTS OF CELLS WITH BOUND FLUORESCENT ANTIBODIES

by
Aicha Weiss

The objective of this research was to geometrically analyze the characteristics of cells, specifically area and perimeter. The images were transferred from HP A900 system to an AVS work station. The images were visualized and manipulated by the AVS which permits the distinction between the cell body, fluorescent object, and background. The intrinsic geometry of the cell was determined by using image processing algorithms written in C. The images were then passed through a threshold filter, thereby yielding a black background and a white cell body. The perimeter calculation was then applied to the black and white images. Two test images were constructed to calibrate the software, both area and perimeter test results were satisfactory. A data set of 20 images of cells were analyzed for area and perimeter computation. The variation of the areas of the cells was 41% due to the variation of the areas of the fluorescent objects (correlation coefficient equal to 0.641126). The null hypothesis of no correlation was tested using statistical analysis and proved that the correlation was not by chance.

AREA AND PERIMETER MEASUREMENTS OF CELLS WITH BOUND FLUORESCENT ANTIBODIES

by
Aicha Weiss

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirement for the Degree of
Master of Science in Biomedical Engineering

Biomedical Engineering Committee

January 1995

APPROVAL PAGE

AREA AND PERIMETER MEASUREMENTS OF CELLS WITH BOUND FLUORESCENT ANTIBODIES

Aicha Weiss

Dr. Arthur B. Ritter, Thesis Advisor
Professor of Physiology, Division of
Microcirculation Research, New Jersey Medical School
University of Medicine and Dentistry of New Jersey

Date

Dr. Andrew Meyer, Committee Member
Professor of Control Systems,
Electrical and Computer
Engineering Department, NJIT

Date

Dr. David Kristol, Committee Member
Professor of Chemistry,
Director of Biomedical Engineering Program, NJIT

Date

BIOGRAPHICAL SKETCH

Author: Aicha Abou Weiss

Degree: Master of Science in Biomedical Engineering

Date: January 1995

Undergraduate and Graduate Education

- Master of Science in Biomedical Engineering
  New Jersey Institute of Technology, Newark, NJ, 1995

- Bachelor of Science in Electrical Engineering
  New York Institute of Technology, New York, NY, 1992

Major: Biomedical Engineering

This thesis is dedicated to my dear husband

ACKNOWLEDGMENT

The author wishes to express her sincere gratitude to her thesis advisor, Dr. Arthur B. Ritter, for his guidance, friendship, and moral support throughout this research. Dr. Ritter showed a great cooperation, flexibility, and readiness to help me obtain all the information necessary for this research.

Special thanks to Professor Andrew Meyer for serving as member of the committee.

I would like to express my sincere gratitude to Dr. David Kristol for his supervision in my graduate program, sparing his valuable time to help me in various issues, and serving as a member of the committee.

The author appreciates the timely help from the computer center consultants at UMDNJ who were very helpful in supplying me with all the relevant computer system information.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Morphology

The analysis of morphological data is one of the most widely used methods to understand diseases. This geometrical analysis helps the pathologist study and recognize structural characteristics that are considered different from normal [1]. Morphometry has a great use in evaluating changes that are unavailable by visual description.

1.2 Methodology

The cells that are used for computations are human lung embryonic cells that were labeled with fluorescent antibodies.

A few drops of suspended cells were placed on a microscope slide and covered with a glass cover slip. The cells were epiillumінated through the microscope objective with an incident light of 488nm. The fluochrome (fluorescein) attached to the antibody fluoresces at a peak wavelength of approximately 505nm (green). A dichrome mirror and optical filter pass light at the fluorescent wavelength (505nm) from the objective to a TV camera. The TV camera is situated on a trinocular mount at the eye piece of the microscope. The TV camera is a low light intensity device which allows a video recording to be made of each batch of cells. The video tape can then be played back, frame-by-frame for analysis of surface antigenic markers as measured by fluorescent intensity on the cell surface [2]. Images were recorded, and saved for further analysis.

1.3 The Objective

The images were first transferred from an HP A900 system to a workstation which supports the software called Application Visualization System (AVS). The images were visualized using the AVS workstation which has a high resolution monitor permitting the distinction between the background, cell body, and fluorescent object. The image processing techniques from the AVS menu were applied to the image for histogram evaluation. However, this image processing technique is insufficient for analyzing the geometrical properties of the cell, fluorescent body, as well as the corresponding areas. Therefore, area computation algorithms had to be constructed to perform this task.

In previous work [3], a different area computation technique was applied. In the "Quantitative Analysis of Nuclear Shape" paper, the method used the ratio of the computed nuclear area to the area of a rectangular figure that has sides tangent to the nuclear margin and parallel to the frame video monitor [3]. This method has the advantage of being independent of the magnification, but it is very time consuming and tedious, and the results may not be optimal. Therefore, a more practical approach is proposed. Compute the area of the raw image before magnification, utilizing an AVS histogram combined with specially constructed geometrical algorithms written in the C programming language.

In this thesis, the images were first visualized with the Application Visualization System software [4]. The purpose of using AVS is its computing platform (HP workstation) which has a high resolution monitor with sophisticated image processing and visualization capabilities. The histogram function display technique was applied to the images to differentiate the cell body, fluorescent object, and background. These results were used in conjunction with a differentiating algorithm (see Appendix personal program #1) written in the C programming language.

A threshold filter was applied (see Appendix personal program #3), to convert the images to a white cell body and a black background. An edge detection technique (see Appendix personal program #4) was used to compute the perimeter of the cells.

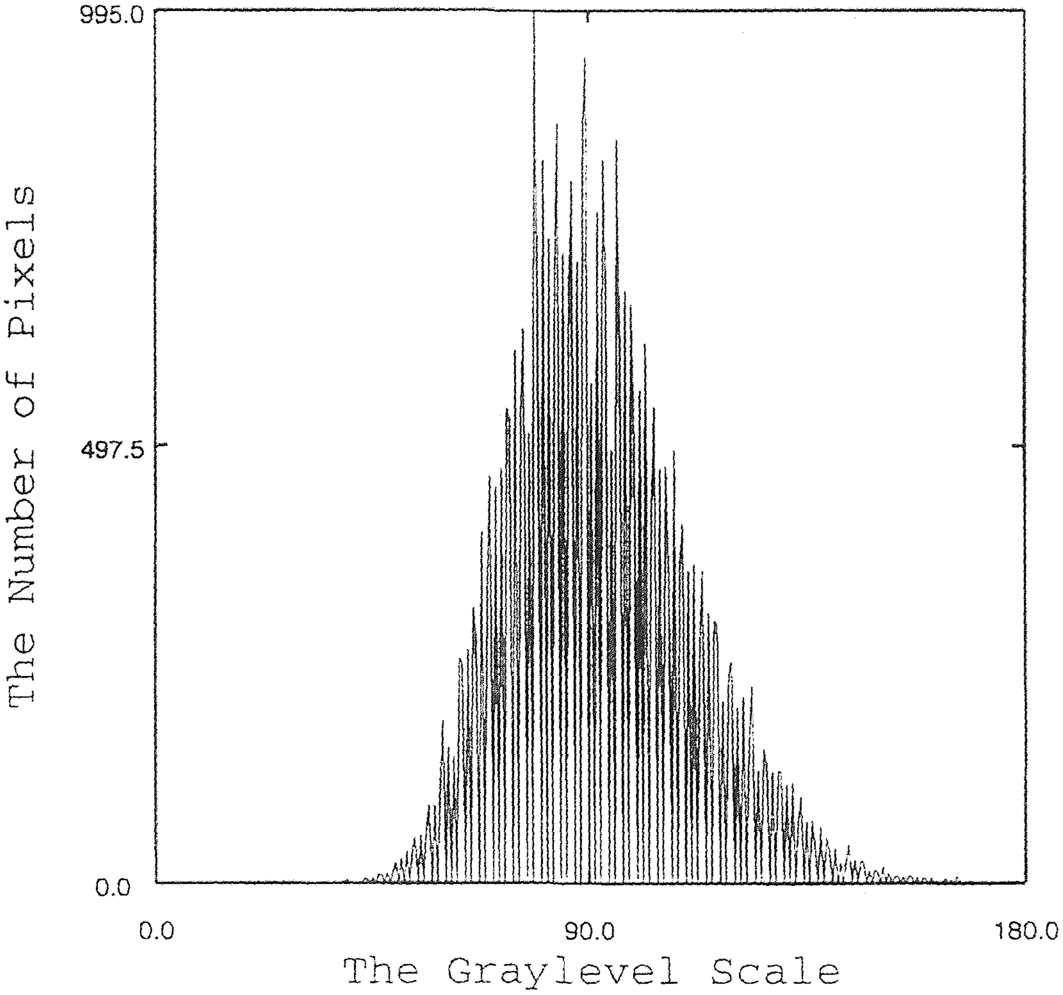

Figure 2.1
The Histogram Display

CHAPTER 2

IMAGE MEASUREMENT

2.1 Introduction

Image measurement deals with image analysis including the brightness value of a pixel or set of pixels. The pixels in the images used in this thesis are one byte representing 256 graylevels, the white value is typically 255, while the black is represented as zero. The graylevel value of a pixel increases going from the background, to the cell body, to the fluorescent antibody.

The images are stored in binary files and handled as a width value (xdim) by length value (ydim) matrices.

2.2 Histogram Processing

The histogram of an image is a function giving the number of pixels of a particular graylevel versus the graylevel. The histogram does not provide information about the location of pixels or one pixel's proximity to another. The images contain (xdim)x(ydim) pixels with 256 graylevel values in the range from 0 to 255. Each of these values can be described by an eight bit unsigned character. The histogram of one of the images is visualized in Figure 2.1.

CHAPTER 3

MENSURATION

Mensuration is the process of evaluating geometric information of imaged objects. These measurements are used to classify objects for recognition schemes.

3.1 Area Calculations

Area algorithms (see Appendix personal program #1)are used to measure the number of pixels contained within the cell body and antibodies. The region wherein the object of concern lies, is visualized and the pixel values are separated into groups by using the histogram technique. For the images used, the graylevel of the cell body, antibodies, and background each lie between maximum and minimum threshold values. The values of the maximum and minimum threshold were determined by zooming on different parts of the image and studying their specific histogram very closely. Figure 3.1 shows an illustration of various histograms.

The physical size of each pixel is known, and the actual area in size units can be calculated. The algorithm (see Appendix personal program #1) accepts an image that is assumed to be an 8-bit unsigned character pixel. The maximum and minimum thresholds values of the cell body, antibodies,

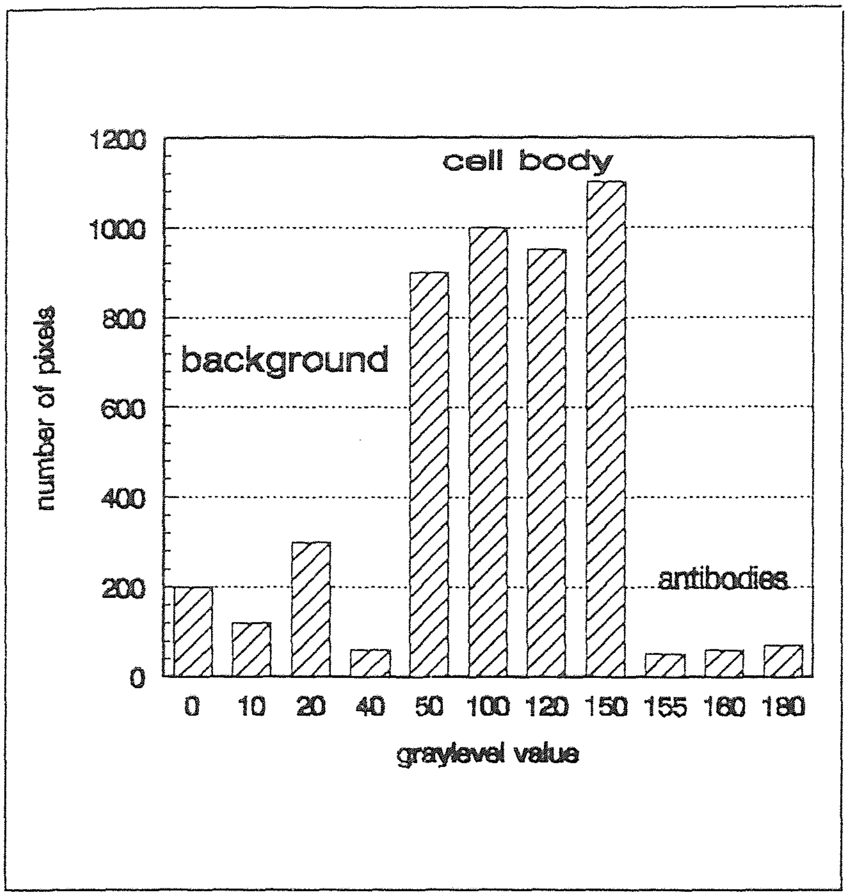

Figure 3.1
The Graylevel Threshold Representation

and background are also passed to the algorithm. The area of the cell body, and antibodies are computed by counting respectively the number of pixels that lie in the corresponding graylevel value range. These values, multiplied by the physical area of the pixel are returned by the algorithm as the areas of concern.

3.2 Perimeter Calculations

The perimeter is the number of pixels of an object's boundary. The boundary is computed using an edge detector (see Appendix personal program #4). First the images were converted from a 256 shades of gray to black and white pictures. Hence; only two graylevel values, 255 for the white body cell, and 0 for the black background. The fluorescent antibodies were merged with the body cell since they are contained within the cell.

The edge was then computed by examining the difference between adjacent pixels close to the boundary. Using two counters, one counter increments each time it encounters a difference between the maximum and minimum graylevel values in a line, while the other counter increments each time it encounters the same difference between adjacents pixels in each column. The sum of these two counters is then multiplied by the average side of the pixel (length plus width divided by 2).

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Testing of Area Computation Algorithm

4.1.1 Test 1

This test was performed to check if the algorithm counts correctly. The area computation program was tested on constructed images (Appendix D) with known dimensions. The first image shown in Figure 4.1 (Appendix A) represents a rectangle 4 pixels wide and 18 pixels long. The pixels in the rectangle have a graylevel value of 122. The rectangle is enclosed in matrix that is 80 (xdim) by 7 (ydim). The background pixels graylevel value is 65.

The calculated number of pixels in the rectangle is 72 (18 by 4). If the physical dimensions of each pixel are assumed to be unity, and the rectangle representing the cell body then the calculated areas are:

area of cell body = 72 (18 by 4).

area of the background =488 (80 by 7 minus 72).

The thresholds values for the rectangle (minimum threshold =maximum threshold =122) and the background (minimum threshold =10, maximum threshold =65) were passed to the algorithm (Appendix personal program #1). The output of the program was:

cell area = 72.

background area =488.

The obtained results are exactly the same as the calculated ones.

4.1.2 Test 2

This test was performed to check if the program counts with differentiating different threshold values. The constructed image (Appendix D) for this task is shown in Figure 4.2 (Appendix A). The image is a simulation of the fluorescent object being the brightest part of the image, the cell body being the second brightest part of the image, and the background as the darkest of all.

The dimensions of the total image are 80 (xdim) by 23 (ydim). The graylevel value of the pixels within the cell body is between threshold1 =64, and threshold2 = 95. The number of pixels in the cell body is 740 (including the fluorescent object).

The graylevel value of the pixels in the fluorescent object is between threshold3 = 120, and threshold4 = 152. The number of pixels within the fluorescent object is 20.

The background was constructed with pixels having a graylevel values between threshold5 = 33, and threshold6 = 63. The minimum threshold value of the background is chosen to be at least 10 because the last column of the image is darker than the background due to the carriage return

Table 4.1 The Threshold Values

Cells Labels	Cell Body Threshold Values		Antibody Threshold Values		Background Threshold Values	
	Thres1	Thres2	Thres3	Thres4	Thres5	Thres6
cell1	103	153	154	180	0	102
cell4	103	153	154	166	0	102
cell8a	103	153	154	212	0	102
cell8b	103	153	154	213	0	102
cell11a	103	153	154	191	0	102
cell11b	103	153	154	243	0	102
cell12	103	153	154	166	0	102
cell18	103	153	154	188	0	102
cell20	103	204	205	217	0	102
cell27	103	204	205	255	0	102
cell28	103	204	205	255	0	102
cell31	103	204	205	233	0	102
cell32	103	204	205	231	0	102
cell34	103	204	205	217	0	102
cell38	103	153	154	196	0	102
cell43	103	153	154	231	0	102
cell45	103	153	154	209	0	102
cell48	103	153	154	196	0	102
cell49	52	102	103	165	0	51
cell55	103	153	154	224	0	102

graylevel value, for best results the minimum threshold value for the background (threshold5) was chosen to be zero. The number of pixels in the background is 1100 (80 by 23 minus 740).

The above threshold values were passed to the algorithm with threshold5=0 instead of 33 and assuming the physical dimensions of the pixel are unities. The output was as follow:

cell area =740.

antibodies area = 20.

area ratio = 0.026.

the background area = 1100.

The above results are identical to the calculated ones.

4.1.3 Results of Area Computation

A group of 20 images of cells were analyzed. First each image was visualized by the AVS system, various histograms of different part of the image were studied to extract the graylevel threshold values for the cell body, fluorescent object, and the background. The results for all 20 data sets are shown in Table 4.1.

The threshold values obtained were then passed to the algorithm for areas computations and the results are recorded in Table 4.2. Table 4.2 includes the background area which was used to check if the computed number of bytes in the image that is (background area +cell body ares)

Table 4.2 The Computed Areas

Cells Labels	cell body area (in micrometers square)	antibody area (in micrometers square)	background area (in micrometers square)
cell1	67031.586	704.22	188883.442
cell4	59728.023	1742.4	66051.484
cell8a	89457.727	6534	86960.281
cell8b	107215.68	14955.601	67024.32
cell11a	44670.781	529.98	79112.219
cell11b	63677.461	2998.38	60105.541
cell12	102482.164	130.68	194088.844
cell18	35174.699	152.46	208761.312
cell20	59299.684	479.16	49600.32
cell27	216449.641	8269.141	103716.359
cell28	185296.984	3833.28	79148.523
cell31	154521.844	1422.96	53114.16
cell32	170718.906	1001.88	42725.102
cell34	93879.062	87.12	47690.941
cell38	153919.266	6955.08	78669.359
cell43	143544.719	10693.98	34325.281
cell45	96470.883	2250.6	25497.121
cell48	38296.5	2555.52	215835
cell49	181144.266	16741.561	46456.742
cell55	173710.031	7514.1	52801.98

divided by pixel area is similar to the xdim times ydim given. In all cases the computed number of bytes in the image is the same as the known image total number of bytes.

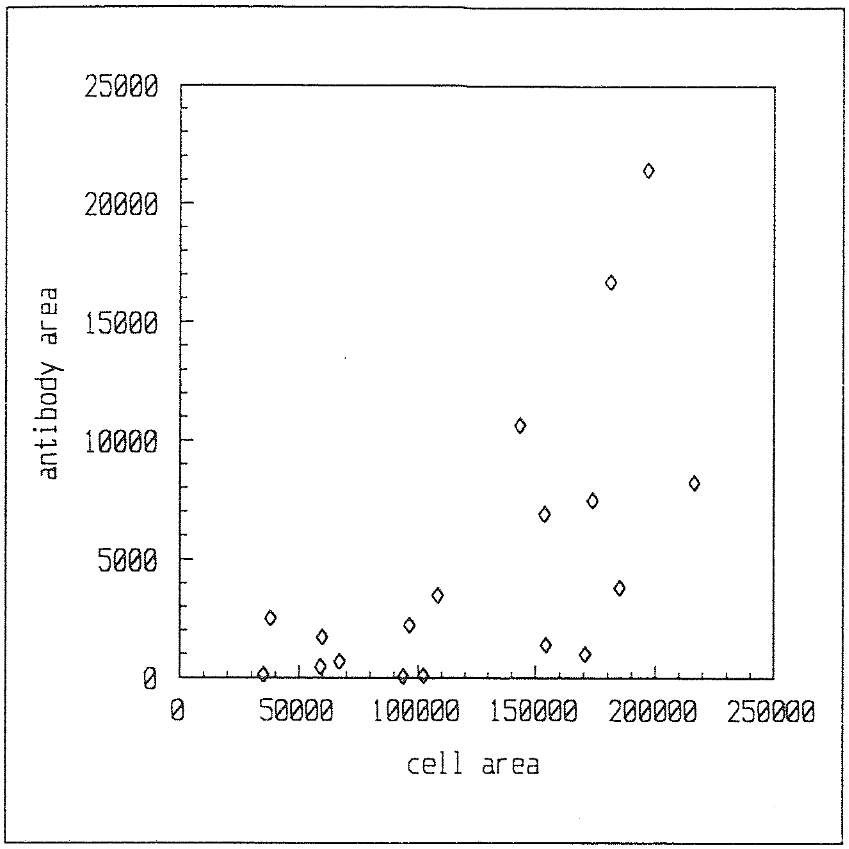

Figure 4.3
Cell Area Versus Antibody Area

Figure 4.3 is a plot of the cell body area and the antibody area with all 20 data sets taken in consideration. This figure shows a very poor correlation between the two quantities. The following formulas were used to compute the correlation coefficient.

$$r = \frac{COV(x, y)}{\sigma_x \sigma_y}$$

$$COV(x, y) = \frac{\sum_{i=1}^{N} (x_i y_i - N\overline{yx})}{N}$$

$$\sigma_y = \sqrt{\frac{\sum_{i=1}^{N} (y_i - \overline{y})^2}{N}}$$

$$\sigma_x = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \overline{x})^2}{N}}$$

Where r is the correlation coefficient, cov(x,y) is the covariance of x and y, σ_x and σ_y are the standard deviations of x and y. And x and y represents the cell body area and the antibody area respectively. The calculated correlation statistics are:

correlation coefficient = 0.641126

mean of x = 124261

mean of y = 4975.12

standard deviation of x = 56215.3

standard deviation of y = 5914.69

variance of x = 3.16017E+009

variance of y = 3.49835E+007

covariance of x and y = 2.13172E+008

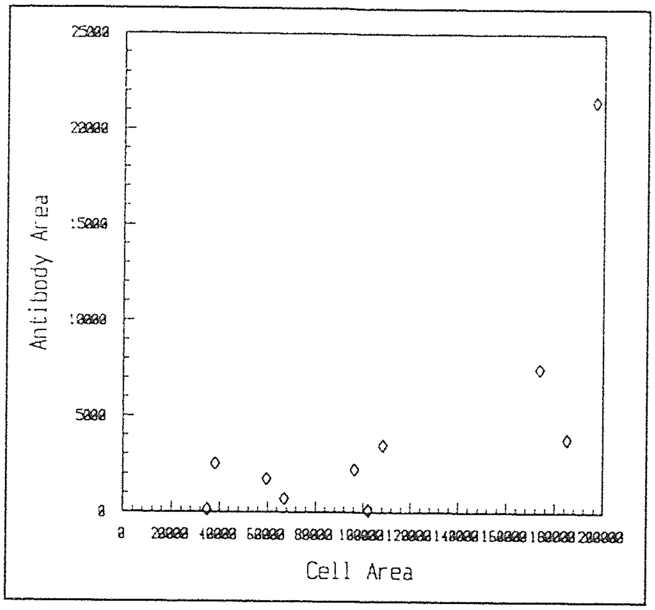

Figure 4.4
Cell Area Versus Antibody Area

Figure 4.4 shows the relation between cell body area and antibody area of a set of 10 data cell taken out of the total data, they represent the images with the least noise and the best distinction between cell body, fluorescent object, and the background. The correlation statistics on this set are :

correlation coefficient = 0.706595

mean of x = 106321

mean of y = 4390.12

standard deviation of x = 56975

standard deviation of y = 6065.02

variance of x = 3.24615E+009

variance of y = 3.67844E+007

covariance of x and y = 2.44167E+008

4.1.4 The Interpretation and Analysis of The Correlation

For the set containing all the data, the computed correlation coefficient is r = 0.641126. Which means that 41% (correlation squared times 100) of the variation of the y's (the antibody area) is accounted for by the relationship with x (the cell area) [12]. For the selected 10 data cells having the best distinction between the cell body, and fluorescent object, the correlation coefficient r = 0.706595 this means that about 50% of the variation of the antibodies areas is due to the variation in the cells areas. The correlation of this set is 1.22 times (50% divided by 40%) greater than the previous set.

The computed value of the correlation coefficient does not confirm the existence of a relationship between the x's and y's. In fact we may get a strong positive or negative correlation purely by chance [12]. To test whether the computed value of r is by chance, the null hypothesis of no correlation is used [12]. A change of scale from r to Z is given with the following formulas:

$$Z=\frac{1}{2}\ln\frac{1+r}{1-r}$$

$$\mu_z=\frac{1}{2}\ln\frac{1+\rho}{1-\rho}$$

$$\sigma_z=\frac{1}{\sqrt{n-3}}$$

$$z=\frac{(Z-\mu_z)}{\sigma_z}$$

Where r is the correlation coefficient, ρ is the population correlation coefficient, Z is the Fisher Z Transformation , μ_z is the mean in the Z scale, and σ_z is the standard deviation in the Z scale [12].

Using α =0.05 the level of significance to test the null hypothesis of no correlation (the null hypothesis ρ = 0). Since μ_z = 0 for ρ = 0, reject the null hypothesis if z $\leq$ -1.96 or z $\geq$ 1.96, where 1.96 is $z_{\frac{\alpha}{2}}$ with $z_{\frac{\alpha}{2}}$ denoting the value of z for which the standard-normal-curve area to its right is equal to $\frac{\alpha}{2}$ and -1.96 is the same value to the left of the standard-normal-curve. With n = 18 (set of 20 data with 18 cells where cell8a, cell8b form one cell and cell11a, cell11b is one cell) , r = 0.641126, the computed Z = 0.76 and z = 2.9437 which exceeds 1.96, the null hypothesis must be rejected. This means that the value obtained for r is significant and was not by chance.

The same test (null hypothesis) was performed on the 10 data cell with 0.05 level of significance, and the results with r = 0.706595 are as follow : Z = 0.8803, z = 2.3291 which is greater than 1.96. The null hypothesis must be rejected as expected.

4.2 Testing of Perimeter Computations Algorithm

4.2.1 Test 1

This test was performed on the same image in Figure 4.1. This image was first put trough a white and black filter converting the rectangle to white and the background to black. The rectangle dimensions are known (see Appendix D) 18 pixels length and 4 pixels width.

If the physical dimensions of the pixels are unity, then the expected perimeter is 44, with the horizontal counter being 8, and the vertical counter being 36.

The image in Figure 4.1 was first converted to black and white using the black and white filter (see Appendix personal program #3) and Figure 4.5 shows the corresponding black and white version of the same image in Figure 4.1. The latter image was passed to the perimeter calculation algorithm (Appendix personal program #4). The output of the algorithm is as follow:

```
horizontal count =9 (instead of 8)
vertical count   =38 (instead of 36)
perimeter        =47 (instead of 44)
```

The computed results differ by a 6.18% error from the expected ones, and this is the best error minimization that was done on the software, due to the fact that when the image files are converted from raw to white and black format, few unexpected characters appear on the white and black images. And the presence of these characters increments both counters to slightly higher values than the expected ones.

4.2.2 Test 2

This test was performed on the image in Figure 4.2 which does not have a specific shape. This image was constructed (Appendix D)with the cell body having 150 pixels on the boundary, 32 pixels counting by line, and 118 pixels counting by column.

This image file was passed to the black and white filter and the resulting image is shown in Figure 4.6. The perimeter calculation algorithm was performed on the black and white image and the actual output of the program is as follow:

Table 4.3 The Computed Areas and Perimeters

Cells Labels	Cell Body Area (in micrometers square)	Cell Body Perimeter (in micrometers)
cell1	67031.586	21268.625
cell4	59728.023	14911.2
cell8a	89457.727	19178.55
cell8b	107215.68	10643.85
cell11a	44670.781	15532.5
cell11b	63677.461	19230.325
cell12	102482.164	41697.95
cell18	35174.699	13573.225
cell20	59299.684	11126.175
cell27	216449.641	19979.7
cell28	185296.984	22846.4
cell31	154521.844	24233.425
cell32	170718.906	13199.9
cell34	93879.062	16554.37
cell38	153919.266	33525.67
cell43	143544.719	12657.62
cell45	96470.883	13186.27
cell48	38296.5	14973.87
cell49	181144.266	12281.57
cell55	173710.031	25511.44

horizontal count = 33 (instead of 32)

vertical count =118 (same as expected)

the perimeter =151 (instead of 150)

The actual perimeter for this image ,if the physical dimensions of the pixel are unity, is slightly different from the expected one by a 0.67%. Similar to image in Figure 4.5 the black and white image file contained foreign characters which increment one or both of the counters to slightly higher values.

4.2.3 Results of Perimeter Computation

A group of 20 images of cells were analyzed. The images were first converted to black and white. Figure 4.7 shows an original image along with the corresponding black and white conversion. The perimeter computation algorithm (Appendix personal program #4) outputs an approximation of the perimeter of the white and black images knowing the physical dimensions of the pixel. The perimeter is computed in micrometers and the collected data is given in Table 4.3.

CHAPTER 5

CONCLUSION

Images of human embryonic lung cells were transferred from HP A900 system to a Unix based system and converted into the software called Application Visualization System (AVS) [5] compatible format. Histogram techniques were introduced for area calculation using a thresholding counter to compute areas of both cell body and fluorescent antibodies. The images were passed as raw binary files but were visualized through AVS. The image processing techniques of AVS allowed a display of the histogram of the entire image and the algorithm used these information to compute the pertinent areas.

Before the perimeter calculations was applied, the images were passed through a white and black filter. Then an algorithm was written to count the transition from the maximum graylevel value (255) to a minimum graylevel value (0) or vice versa in each line and column, the sum of the horizontal and vertical counters was multiplied by the average physical size of the pixel.

Different images were used to test the algorithms, and the results showed a consistency. The algorithms include the option of changing the threshold level as well as the pixel's physical size.

Further research is required to investigate the error committed using this method compared to other techniques such as subtracting the one pixel eroded image from the original one and count the pixels left as the perimeter of the image. The area algorithm needs to be investigated especially for the case where the threshold values overlap.

This error could be reduced if the threshold values were more precise for the total cell body, therefore differentiating them from the background and the noise present in the image.

One way of yielding better results is to increase the size of the acquisition matrix precisely the xdim and ydim, thereby yielding an increased resolution. Another way of acquiring images with the least noise is to adjust the magnification in the microscope for each image which could be tedious and very time consuming.

APPENDIX A

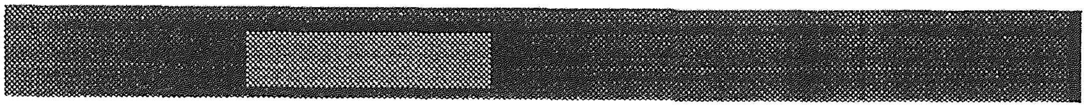

Figure 4.1
Test 1 Image Display

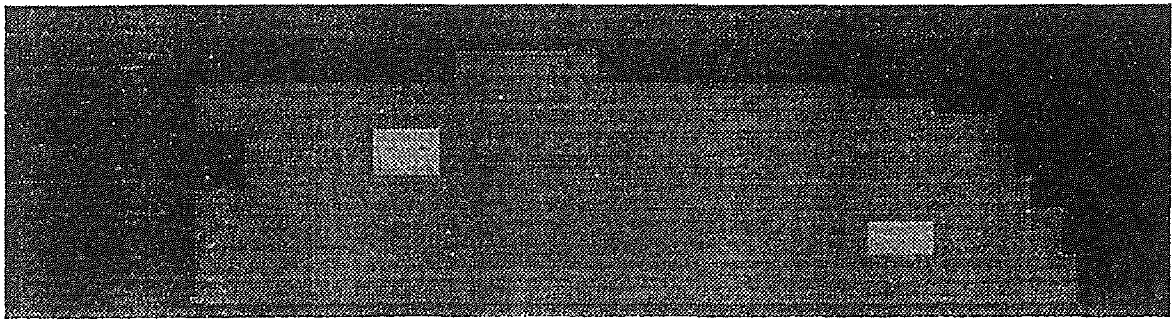

Figure 4.2
Test 2 Image Display

APPENDIX B

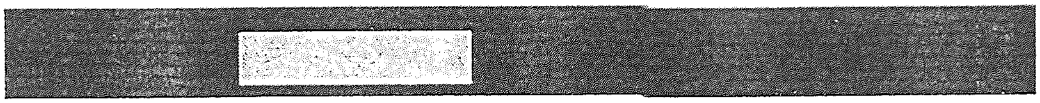

Figure 4.5
Test 1 Image Display in Black and White

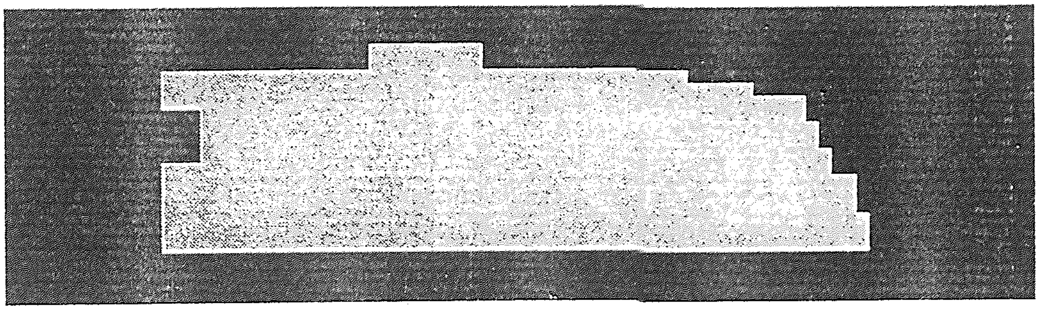

Figure 4.6
Test 2 Image Display in Black and White

APPENDIX C

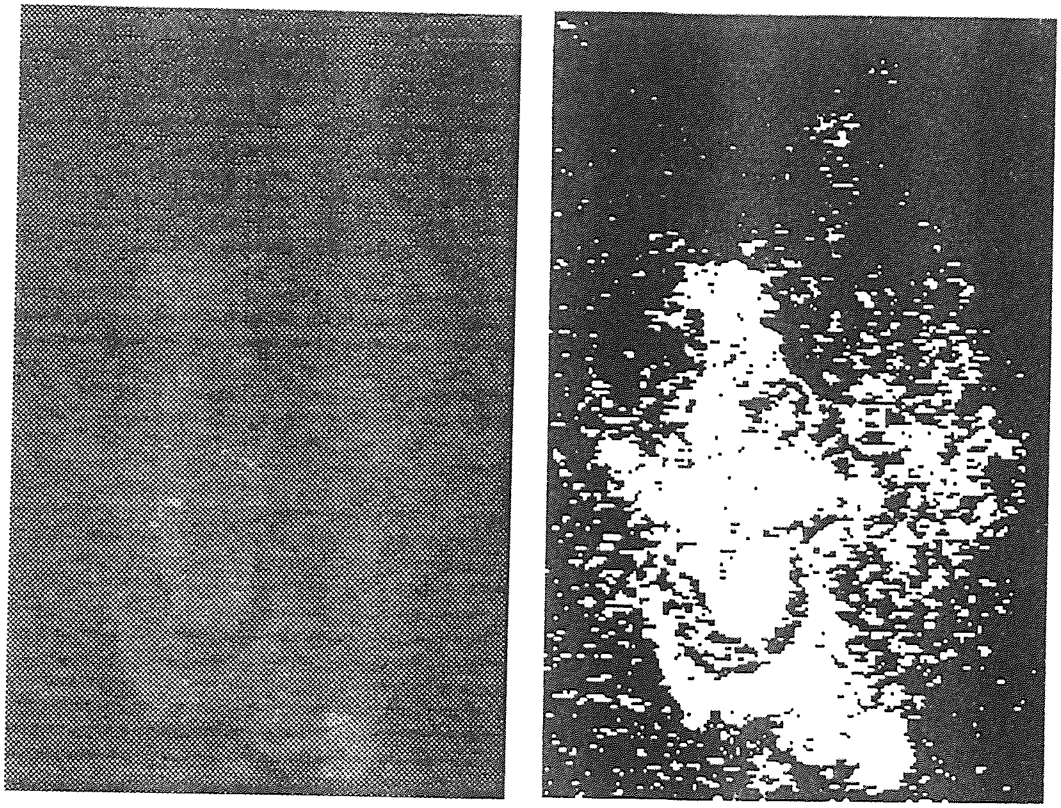

Figure 4.7
Original Image Versus Black and White Version

APPENDIX D

TEST IMAGES

```
IMAGE FILE
TEST 1 IMAGE
 80 7
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAzzzzzzzzzzzzzzzzzzAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAzzzzzzzzzzzzzzzzzzAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAzzzzzzzzzzzzzzzzzzAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAzzzzzzzzzzzzzzzzzzAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
IMAGE FILE
TEST 2 IMAGE
 80 23
!!!!!!!!!!!!#######!!!!!!!!&&&&&&&&&&$$$$$$$$$$$$$$$$%%%%%%%%%%%%%%%%%%%%%%%%%%%
!!!!!!!!!!!!#######!!!!!!!!&&&&&&&&&&$$$$$$$$$$$$$$$$%%%%%%%%%%%%%%%%%%%%%%%%%%%
!!!!!!!!!!!!#######!!!!!!!!&&&&&&&&&&$$$$$$$$$$$$$$$$%%%%%%%%%%%%%%%%%%%%%%%%%%%
!!!!!!!!!!!!#######!!!!!!!!&@@@@@@@@@$$$$$$$$$$$$$$$$%%%%%%%%%%%%%%%%%%%%%%%%%%%
!!!!!!!!!!!!#######!!!!!!!!&@ABCDEFG@$$$$$$$$$$$$$$$$%%%%%%%%%%%%%%%%%%%%%%%%%%%
!!!!!!!!!!!!@@@@@@@@@@@@@@@@@ABCDEFG@@@@@@@@@@@@@@@@@@%%%%%%%%%%%%%%%%%%%%%%%%%%
!!!!!!!!!!!!@ABCDEFGABCDEFGABCEFGABCEDFGABCEDEGABCDEF@@@@@@%%%%%%%%%%%%%%%%%%%%%
!!!!!!!!!!!!@@@@ABCEDFGABCEFGABCDEFGABCEDFGABCDEFGABCDEFGA@@@@@%%%%%%%%%%%%%%%%%
!!!!!!!!!!!!###@ABCDEFGxxxxFGABCDEFGABCDEFGABCDEFGABCDEFGABCDE@%%%%%%%%%%%%%%%%%
!!!!!!!!!!!!###@ABCDEFGxxxxFGABCDEFGABCDEFGABCDEFGABCDEFGABCDE@@%%%%%%%%%%%%%%%%
!!!!!!!!!!!!###@ABCDEFGxxxxFGABCDEFGABCDEFGABCDEFGABCDEFGABCDEF@%%%%%%%%%%%%%%%%
!!!!!!!!!!!!###@ABCDEFGABCDEFGABCDEFGABCDEFGABCDEFGABCDEFGABCDE@@%%%%%%%%%%%%%%%
!!!!!!!!!!!!@@@@ABCDEFGABCDEFGABCDEFGABCDEFGABCDEFGABCDEFGABCDEF@%%%%%%%%%%%%%%%
!!!!!!!!!!!!@ABCDEFGABCDEFGABCDEFGABCDEFGABCDEFGABCDEFGABCDEFGAB@@@%%%%%%%%%%%%%
!!!!!!!!!!!!@ABCDEFGABCDEFGABCDEFGABCDEFGABCDEFGABCDEFGxxxxEFGABCD@%%%%%%%%%%%%%
!!!!!!!!!!!!@ABCDEFGABCDEFGABCDEFGABCDEFGABCDEFGABCDEFGxxxxEFGABCD@%%%%%%%%%%%%%
!!!!!!!!!!!!@ABCDEFGABCDEFGABCDEFGABCDEFGABCDEFGABCDEFGABCDEFGABCD@@%%%%%%%%%%%%
!!!!!!!!!!!!@BCDEFGABCDEFGABCDEFGABCDEFGABCDEFGABCDEFGABCDEFGABCDEF@%%%%%%%%%%%%
!!!!!!!!!!!!@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@%%%%%%%%%%%%
!!!!!!!!!!!!#######!!!!!!!&&&&&&&&&&$$$$$$$$$$$$$$$$$%%%%%%%%%%%%%%%%%%%%%%%%%%%
!!!!!!!!!!!!#######!!!!!!!&&&&&&&&&&$$$$$$$$$$$$$$$$$%%%%%%%%%%%%%%%%%%%%%%%%%%%
!!!!!!!!!!!!#######!!!!!!!&&&&&&&&&&$$$$$$$$$$$$$$$$$%%%%%%%%%%%%%%%%%%%%%%%%%%%
!!!!!!!!!!!!#######!!!!!!!&&&&&&&&&&$$$$$$$$$$$$$$$$$%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

APPENDIX E

PROGRAM LISTING

The algorithms written in this paper are listed as follow:

Program 1: area.c

This program computes the area of a particular part of the image given the thresholds levels values.

Program 2: bw4avs.c

It converts an image from 256 shades of gray to a white cell body and black background, and makes the image ready to be visualized in AVS.

Program 3: bw4perim.c

It converts the image file to black and white, outputs the xdim, ydim, and saves the black and white image in a file with a .b extension.

Program 4: perimcal.c

This is the algorithm used to compute the perimeter of the white and black image.

Program 1: area.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define maxline 51
#define pix_area 7.26
```

```
main(argc,argv)
     int argc;
     char *argv[];
{
int i,j,xdim,ydim,xydim,cell_pix=0,THRES1,THRES2,THRES3,
    THRES4,THRES5,THRES6;
int antib_pix=0,back_pix=0,gr_level;
char line[maxline],ch1,ch2;
unsigned char *matrix;
float cell_area,antib_area,back_area,ratio_area;
FILE *fpt;
   if(argc!=2)
    {
       printf("compute area of a cell with fluorescent
               antibodies\n");
       printf("Usage: area [input-file]\n");
       exit(1);
    }
fpt = fopen(argv[1],"rb");
   if (!fpt)
    {
       printf("Error in reading file %s \n",argv[1]);
       exit(1);
    }
for (i=1;i<=2;i++)
  fgets(line,maxline-1,fpt);
fscanf(fpt,"%s",line);
fscanf(fpt,"%s",line);
xdim=atoi(line);
fscanf(fpt,"%s",line);
ydim=atoi(line);
fgets(line,maxline-1,fpt);
xydim = xdim*ydim;
matrix = (char *)calloc(xydim,sizeof(char));
   if(!matrix)
    {
       printf("memory request failed\n");
       exit(1);
    }
   if(fread (matrix,sizeof(char),xydim,fpt)!=xydim)
    {
       printf("Input file error! \n");
       exit(1);
    }
printf("do you want to change the threshold values?(y or n)
       \n");
scanf("%c",&ch1);
   if (ch1 == ’y’)
    {
       printf("enter the threshold values for the cell
       body\n");
       printf("enter the minimum threshold value : \n");
       scanf("%d",&THRES1);
       printf("enter the maximum threshold value : \n");
       scanf("%d",&THRES2);
       printf("enter the threshold values for the
               antibodies\n");
       printf("enter the minimum threshold value : \n");
       scanf("%d",&THRES3);
       printf("enter the maximum threshold value : \n");
       scanf("%d",&THRES4);
       printf("enter the threshold values for the background
              \n");
       printf("enter the minimum threshold value : \n");
       scanf("%d",&THRES5);
       printf("enter the maximum threshold value : \n");
       scanf("%d",&THRES6);
    }
    else
    {
       THRES1=103;
       THRES2=153;
       THRES3=154;
       THRES4=255;
       THRES5=0;
       THRES6=102;
    }
for (j=0; j<=(ydim-1); j++)
   for (i=0; i<=(xdim-1);i++)
    {
       gr_level =*(matrix + j*xdim +i);
       if(THRES1<= gr_level && gr_level<= THRES2)
         cell_pix=cell_pix+1;
       if(THRES3<= gr_level && gr_level<= THRES4)
         antib_pix=antib_pix+1;
       if(THRES5<= gr_level && gr_level<= THRES6)
         back_pix=back_pix+1;
    }
cell_area = pix_area *(cell_pix+antib_pix);
antib_area = pix_area * antib_pix;
ratio_area = antib_area / cell_area;
back_area = back_pix * pix_ara;
printf("The cell area in micrometers is : %5.3f\n",
        cell_area);
printf("The area of antibodies in micrometers
        is:%5.3f\n",antib_area);
printf("The background area is : %5.3f\n",back_area);
printf("The area ratio is:%f\n",ratio_area);
printf("Program completed!\n");
fclose(fpt);
}
```

Program 2: bw4avs.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define black 0
#define white 255
#define maxline 51

main(argc,argv)
     int argc;
     char *argv[];
{
   int i,j,index,xdim,ydim,xydim,avsdim,nln,THRES6;
   char line[maxline],ch;
   unsigned char  *matrix,*avsmatrix,*strpt,pix;
   FILE *fpt,*avsfpt;

   if(argc!=2)
    {
       printf("Convert raw file to black and white in AVS
               format\n");
       printf("Usage: bw4avs [input-file]\n");
       exit(1);
    }
fpt = fopen(argv[1],"rb");
   if (!fpt)
    {
       printf("Error in reading file %s \n",argv[1]);
       exit(1);
    }
for (i=1;i<=2;i++)
   fgets(line,maxline-1,fpt);
fscanf(fpt,"%s",line);
fscanf(fpt,"%s",line);
xdim=atoi(line);
fscanf(fpt,"%s",line);
ydim=atoi(line);
fgets(line,maxline-1,fpt);
xydim = xdim*ydim;
avsdim =4*xydim;
matrix = (char *)calloc(xydim,sizeof(char));
avsmatrix = (char*)calloc(avsdim,sizeof(char));
   if((!avsmatrix)||(!matrix))
    {
       printf("memory request failed\n");
       exit(1);
    }
printf("change in max threshold value of the backgound?
       (y or n)\n");
scanf("%c",&ch);
```

```
   if (ch == ’y’)
    {
       printf("enter the maximum threshold value for the
               background\n");
       scanf("%d",&THRES6);
    }
   else
       THRES6 = 102;

   if (xdim<=170)
    {
       nln = (xdim+1)/2;
        for (j=0,index=0; j<=(ydim-1); j++)
         for (i=0; i<=(xdim-1);i++,index+=4)
     {
       pix=getc(fpt);
       avsmatrix[index]=255;
         if (pix <= THRES6)
           pix = black;
         else
           pix = white;
       avsmatrix[index+1]=pix;
       avsmatrix[index+2]=pix;
       avsmatrix[index+3]=pix;
         if (i==(nln-1))
           pix=getc(fpt);
     }
    }
   else
    {
       if(fread (matrix,sizeof(unsigned
          char),xydim,fpt)!=xydim)
     {
        printf("Input file error! \n");
        exit(1);
     }
       for (j=0,index=0; j<=(ydim-1); j++)
        for (i=0; i<=(xdim-1);i++,index+=4)
        {
          avsmatrix[index]=255;
           if( matrix[j*xdim+i] <= THRES6)
              matrix[j*xdim+i] = black;
           else
              matrix[j*xdim+i] = white;
          avsmatrix[index+1]=matrix[j*xdim+i];
          avsmatrix[index+2]=matrix[j*xdim+i];
          avsmatrix[index+3]=matrix[j*xdim+i];
        }
    }
strpt = strchr(argv[1],’.’);
   if (strpt)
    *strpt = ’\0’;
```

```
strcat(argv[1],".x");
printf("Output file name : %s\n",argv[1]);
avsfpt = fopen(argv[1],"wb");
   if (!avsfpt)
    {
       printf("Error in writing file %s \n",argv[1]);
       exit(1);
    }
   if((fwrite (&xdim, sizeof(int),1,avsfpt)!=1)||
      (fwrite (&ydim, sizeof(int),1,avsfpt)!=1))
     {
       printf("Output file error \n");
       exit(1);
     }
   if(fwrite (avsmatrix, sizeof(unsigned
      char),avsdim,avsfpt)!=avsdim)
     {
       printf("Output file error \n");
       exit(1);
     }
printf("Program completed!\n");
fclose(fpt);
fclose(avsfpt);
}
```

Program 3: bw4perim.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define black 0
#define white 255
#define maxline 51

main(argc,argv)
     int argc;
     char *argv[];
{
   int i,j,index,xdim,ydim,xydim,avsdim,nln,THRES6;
   char line[maxline],ch;
   unsigned char  *matrix,*avsmatrix,*strpt,pix;
   FILE *fpt,*avsfpt;

   if(argc!=2)
     {
       printf("Convert raw image file to black and
               white\n");
       printf("Usage: bw4perim [input-file]\n");
       exit(1);
```

```
  }
fpt = fopen(argv[1],"rb");
if (!fpt)
  {
    printf("Error in reading file %s \n",argv[1]);
    exit(1);
  }
for (i=1;i<=2;i++)
  fgets(line,maxline-1,fpt);
fscanf(fpt,"%s",line);
fscanf(fpt,"%s",line);
xdim=atoi(line);
fscanf(fpt,"%s",line);
ydim=atoi(line);
fgets(line,maxline-1,fpt);
xydim = xdim*ydim;
avsdim =xydim;
printf("the x value is : %d \n",xdim);
printf("the y value is : %d \n",ydim);
matrix = (char*)calloc(xydim,sizeof(char));
avsmatrix = (char*)calloc(avsdim,sizeof(char));
if((!avsmatrix)||(!matrix))
  {
    printf("memory request failed\n");
    exit(1);
  }
printf("change in max threshold value of the backgound?
    (y or n)\n");
scanf("%c",&ch);
if (ch == ’y’)
 {
    printf("enter the maximum threshold value for the
            background\n");
    scanf("%d",&THRES6);
 }
else
    THRES6 = 102;
if (xdim<=170)
  {
    nln = (xdim+1)/2;
    for (j=0,index=0; j<=(ydim-1); j++)
   for (i=0; i<=(xdim-1);i++,index+=1)
     {
       pix=getc(fpt);
          if (pix <= THRES6)
              pix = black;
          else
              pix = white;
       avsmatrix[index]=pix;
       if (i==(nln-1))
         pix=getc(fpt);
     }
     }
   else
     {
       if(fread (matrix,sizeof(unsigned
          char),xydim,fpt)!=xydim)
      {
        printf("Input file error! \n");
        exit(1);
      }
       for (j=0,index=0; j<=(ydim-1); j++)
      for (i=0; i<=(xdim-1);i++,index+=1)
        {
             if( matrix[j*xdim+i] <= THRES6)
                 matrix[j*xdim+i] = black;
             else
                 matrix[j*xdim+i] = white;
          avsmatrix[index]=matrix[j*xdim+i];
        }
     }
   strpt = strchr(argv[1],'.');
   if (strpt)
     *strpt = '\0';
   strcat(argv[1],".b");
   printf("Output file (in black and white)name
           :%s\n",argv[1]);
   avsfpt = fopen(argv[1],"wb");
   if (!avsfpt)
     {
       printf("Error in writing file %s \n",argv[1]);
       exit(1);
     }
   if((fwrite (&xdim, sizeof(int),1,avsfpt)!=1)||
      (fwrite (&ydim, sizeof(int),1,avsfpt)!=1))
     {
       printf("Output file error \n");
       exit(1);
     }
   if(fwrite (avsmatrix, sizeof(unsigned
char),avsdim,avsfpt)!=avsdim)
     {
       printf("Output file error \n");
       exit(1);
     }
   printf("Program completed!\n");
   fclose(fpt);
   fclose(avsfpt);
 }
```

Program 4: perimcal.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

main(argc,argv)
     int argc;
     char *argv[];
{
int i,j,counth=0,countv=0,xdim,ydim,xydim,nln;
float perim;
unsigned char *matrix;
FILE *fpt;

   if(argc!=2)
     {
       printf("compute the perimeter of a cell\n");
       printf("Usage: perimcal [input-file.b]\n");
       exit(1);
     }
   fpt = fopen(argv[1],"rb");
   if (!fpt)
     {
       printf("Error in reading file %s \n",argv[1]);
       exit(1);
     }
   printf("enter the x value : ");
   scanf("%d",&xdim);
   printf("enter the y value : ");
   scanf("%d",&ydim);
   xydim=xdim*ydim;
   matrix =(char*)calloc(xydim,sizeof(char));
    if(!matrix)
     {
       printf("memory request failed\n");
       exit(1);
     }


    if(fread (matrix,sizeof(char),xydim,fpt)!=xydim)
      {
        printf("Input file error! \n");
        exit(1);
      }
   for (j=0;j<= (ydim-1);j++)
     for (i=0;i<= (xdim-1);i++)
        {
         if (matrix[j*xdim+i] != matrix[j*xdim+i+1]
             &&matrix[j*xdim+i]!=’\n’ &&matrix[j*xdim+i
             +1]!=’\n’ &&matrix[j*xdim+i]!=EOF &&matrix[
             j*xdim+i+1]!=EOF &&matrix[j*xdim+i]!=’\0’
             &&matrix[j*xdim+i+1]!=’\0’ &&i!=xdim-1)
         counth = counth+ 1;
        if (matrix[j*xdim+i] != matrix[(j+1)*xdim+i]
            &&matrix[j*xdim+i]!=’\n’ &&matrix[(j+1)*xdim+i
            ]!=’\n’ &&matrix[j*xdim+i]!=EOF &&matrix[
            (j+1)*xdim+i]!=EOF &&matrix[j*xdim+i]!=’\0’
            &&matrix[(j+1)*xdim+i]!=’\0’ &&j!=ydim-1)
        countv = countv+ 1;

       }

  printf("number of horizontal counts is : %d\n",counth);
  printf("number of vertital counts is : %d\n",countv);
  perim = (counth +countv)*2.725;
  printf("the perimeter in micrometer is : %f\n",perim);
  fclose(fpt);
}
```

REFERENCES

1. Pesce, C. M. "Biology of Disease Defining and Interpreting Diseases through Morphometry." Lab. Investigation 56, 1987, pp. 568-573.

2. Goldman, M. "Fluorescent Antibody Methods." Bionetics Research Lab. , Inc. Falls Church, Virginia. Academic Press, New York and London, 1968.

3. Wydner, K. S., J. J. Godyn, M. L. Lee and L. J. Sciorra. "A New Approach to The Computer-Assisted Quantitative Analysis of Nuclear Shape." Modern Pathology 4, 1991, pp. 154-159.

4. Application Visualization System User's Guide. Advanced Visual System, Inc., 1992.

5. Wang, Kuen. "Blur Identification Restoration of Image of Coronary Microvessel." Thesis, New Jersey Institute of Technology, Newark, NJ, May 1993.

6. Myler, H. R. and A. R. Weeks. Computer Imaging Recipes in C. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1993.

7. Embree, P. M. and B. Kimble. C Language Algorithms for Digital Signal Processing. Prentice-Hall, Inc., EngleWood Cliffs, NJ, 1991.

8. Myler, H. R. and A. R. Weeks. The Pocket Handbook of Image Processing Algorithms in C. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1993.

9. Kernigham, B. W. and D. M. Ritchie. The C Programming Language. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1978.

10. Schildt, H. Teach Yourself C. McGraw-Hill, Inc., Berkeley, California, 1990.

11. Weibel, E. R. Stereological Methods. Academic Press, Inc., New York, NY, 1979.

12. Freund, J. E. and G. A. Simon. Modern Elementary Statistics. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1992.