
University of Algarve

Master Thesis

# Recognition of Leishmania Parasite and Macrophage Infection Rate using Image Processing Techniques

*Author:*
Ehsan Yazdanparast

*Supervisor:*
Prof. Hamidreza Shahbazkia

*A thesis submitted in fulfilment of the requirements for the degree of Master in Computer Science*

*in the*

Image Processing Group
Faculty of Science and Technology

**2013**

# Declaration of Authorship

I, Ehsan YAZDANPARAST, declare that this thesis titled, 'Recognition of Leishmania Parasite and Macrophage Infection Rate using Image Processing Techniques' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

# Declaracao de Autoria

Eu, Ehsan Yazdanparast declaro que esta tese intitulada "Reconhecimento do Parasita da Leishmaniose e Rácio de Infecção de Macrófagos usando Técnicas de Processamento de Imagem " e todo o trabalho nele desenvolvido é da minha autoria. Confirmo que:

- Este trabalho foi realizado no periodo de obtenção de grau de mestre nesta Universidade.
- A referência nesta tese a qualquer trabalho já submetido para um mestrado ou qualquer qualificação nesta Universidade ou noutra instituição foi claramente identificada.
- A consulta a trabalhos publicados por terceiros é sempre referenciada.
- Todas as citações, de fontes consultadas, incluídas na tese são reconhecidas e identificadas. Para além destas, todo o trabalho desenvolvido é exclusivamente da minha autoria.
- Reconheço todas as fontes principais de apoio.
- É devidamente identificado todo o trabalho realizado conjuntamente por mim e por terceiros, especificando as contribuições feitas por terceiros e aquelas realizadas única e exclusivamente por mim.

# *Abstract*

*Leishmaniasis* is an epidemic dangerous disease in tropical and sub tropical regions of the world and if it is not treated conveniently, it may cause several health infections and could even lead to the death of patients. Currently, there is no efficient vaccine for the disease and available treatments cause serious side effects such as toxicity and parasite resistance. Therefore, ongoing research in *Drug Discovery* and other related biological areas concentrate on finding adequate drug candidates for the disease. Drug discovery pipeline for Leishmaniasis disease facilitates so many biological techniques till now to understand level of effectiveness of different drug candidates for treating the disease. The accuracy and reliability of such techniques, however, highly depends on manual process of detecting, extracting, counting and analyzing components of interest such as cells, parasites and cytoplasm regions by expert biologists. Such kind of activities is subjected to so many human-made errors and is often considered to be too time and energy consuming. The existing computational based solutions, which are reduced, also suffer from limited level of analysis and in some cases inaccuracy of the results are evident. For instance, there is no package dedicated to *Intracellular Parasites Counting*, which is a central operation when researching drug candidates' effects. The current research aims at addressing the urgent need to find a solution to the problem of investigating *Infection Ratio* of cells more accurately, which is in fact still lacking. A computational framework which fulfills all those mentioned tasks automatically, in an acceptable time range with maximum possible accuracy of the results is suggested. The proposed solution mainly uses Image Processing approaches to process clinical images and analyze the results. The visual and statistical results then could be used independently or as a complementary tool for laboratories to investigate infection ratio of drug candidates and even at a higher level introduce vaccines for the disease.

**Keywords:** Image Processing, Leishmaniasis, Cells, Parasites, Cytoplasm Regions

# *Resumo*

A Leishmaniose é uma doença crónica perigosa, presente nas regiões tropicais e subtropicais. Provoca diversas infecções, podendo levar á morte se não for tratada convenientemente. Atualmente, não existe nenhuma vacina eficaz para a doença e os tratamentos administrados provocam efeitos secundários graves ao nivel da toxicidade e da resistência de parasitas. Desta forma, a investigação em áreas biológicas relacionadas com a doença centra-se na descoberta de fármacos novos, mais eficazes e eficientes no combate á doença. O processo de descoberta destes fármacos fornece ferramentas úteis à aplicação de muitas técnicas biológicas, avaliando-se, assim, o nivel de eficácia dos diferentes fármacos candidatos ao tratamento da doença. Porém, o rigor e a fiabilidade de tais técnicas dependem de processos manuais executados por biólogos para detetar, extrair, contar e analisar as componentes de interesse de tais células, parasitas e regiões de citoplasma. Esta atividade está sujeita ao erro humano sendo morosa com elevado desgate energético. O número reduzido de soluções computacionais que existem atualmente apresentam um nivel de análise limitado e, em alguns casos, com evidentes imprecisões nos resultados. Por exemplo, não existe qualquer pacote computacional dedicado à contagem de parasitas intracelulares, apesar desta etapa ser fundamental na pesquisa do efeito do fármaco candidato. Este trabalho tem como objetivo apresentar uma solução, ainda não existente, para o problema da investigação do rácio de infecção de células com maior rigor. Assim, propõe-se um ambiente computacional que executa eficientemente todas as tarefas acima mencionadas, de forma automatica, temporalmente aceitável e com a máxima precisão dos resultados. A solução proposta assenta numa abordagem de Processamento de Imagem, tratando imagens clínicas e analisando os resultados obtidos. Quer os resultados visuais, quer os estatísticos, podem ser utilizados posteriormente de forma independente ou como uma ferramenta complementar para a pesquisa laboratorial de rácios de infecção dos fármacos candidatos e, a um nivel mais elevado, introduzir uma vacina para a doença.

**Palavras-chave:** Processamento de Imagem, Leishmaniose, Células, Parasitas, Regiões de Citoplasma

# *Acknowledgements*

I wish to acknowledge following people for their role in the completion of this work:

- My supervisor, Prof. Hamidreza Shahbazkia, for introducing me to the Image Processing world, and his extreme generousity with his time, knowledge and ideas during the work. Thank you for your guidance, support, and patience throughout the course of the research and thesis writing.
- My biological guide and colleague, Prof. Baptiste Vergnes, for providing me biological inputs, guiding me with background related information, and for sharing his wealth of experience and knowledge to further my research. Thank you for your valuable collaboration and support during the work.
- My colleagues in Image Processing research group, especially Eng. Faroq Al-tam and Dr. Antonio Dos Anjos, for many productive knowledge exchanges, and for precious idea sharing. Thank you for helping me during this time.
- My thesis editor and peer reviewer, Denise Candeias, for her valuable edits and suggestions which makes my work more professional and unique. Thank you for your feedbacks.
- My dear parents and my lovely partner, Zeinab Doctor Arastooye Marandi, for their devoted love, encouragment and support, and for their collective energy which was a great motivation in my way. Without you, I could not do this work. Thank you, for being there for me, all the time.

# Contents

**Declaration of Authorship** i

**Declaração de Autoria** ii

**Abstract** iii

**Resumo** iv

**Acknowledgements** v

**List of Figures** x

**List of Tables** xiii

**1 Problem Definition and Background** 1
- 1.1 Motivations for Research . . . 1
  - 1.1.1 What is Leishmaniasis? . . . 2
  - 1.1.2 Clinical Background of Disease Research . . . 4
    - 1.1.2.1 Drug Discovery . . . 4
    - 1.1.2.2 High Throughput Screening . . . 5
    - 1.1.2.3 High Content Screening . . . 5
  - 1.1.3 How Image Processing techniques can help? . . . 6
- 1.2 Problem Formalization . . . 8
  - 1.2.1 Input Images Acquisition . . . 8
  - 1.2.2 Problem Definition . . . 10
  - 1.2.3 Challenges . . . 13
  - 1.2.4 Objectives of the Research . . . 15
- 1.3 Thesis Outline . . . 16

**2 Image Processing: Methods and Perspectives** 17
- 2.1 Introduction . . . 17
- 2.2 Advance Methods . . . 17
  - 2.2.1 Dynamic Mean Filter . . . 18

2.2.2 Median Filtering in Constant time . . . 19
2.2.3 Unsharp Mask . . . 19
2.2.4 Kernel Sub Division(KSD) Algorithm . . . 22
2.2.4.1 Contour Binning . . . 22
2.2.4.2 Kernel Subdivision . . . 24
2.2.5 Mean Adaptive Threshold . . . 25
2.2.5.1 Local Adaptive Thresholding Using Sauvola and Niblack Techniques . . . 27
2.2.5.2 Integral Images for computing local means and variances 28
2.2.6 Segmentation based on Morphological Watersheds . . . 29
2.2.6.1 Segmentation and Quantification of Two-Dimensional Gel Electrophoresis Images using WST . . . 30
2.2.7 Fast Two Scan Based Connected Component Labeling . . . 31
2.2.8 Efficient Distance Transform algorithm . . . 33
2.3 state-of-the-art . . . 34
2.3.1 Threshold for Decreasing PDF's . . . 34
2.3.2 Gradient Based Threshold . . . 37

**3 Experimental Setup and Results 41**
3.1 Cell Pipeline . . . 41
3.1.1 Smoothing . . . 44
3.1.2 Thresholding . . . 45
3.1.3 Correction . . . 46
3.1.4 Segmentation and Quantification . . . 47
3.2 Parasite Pipeline . . . 49
3.2.1 Smoothing . . . 49
3.2.2 Thresholding . . . 50
3.2.3 Feature Extraction and Quantification . . . 51
3.3 Cytoplasm Pipeline . . . 51
3.3.1 Smoothing . . . 53
3.3.2 Thresholding . . . 53
3.3.3 Correction . . . 55
3.4 Analytic section . . . 56
3.5 Putting it all together: The Algorithm . . . 62

**4 Evaluation and Comparison 63**
4.1 Computational Evaluation . . . 63
4.1.1 Comparing Results with Global Intensity-based Thresholds Results 65
4.1.2 Comparing Results with Local Intensity-based Thresholds Results 67
4.1.3 Comparing Results with non Intensity Based Thresholding methods Results . . . 70
4.2 Biological Evaluation . . . 72
4.2.1 Software validation versus Manual Giemsa Counting: susceptibility of intramacrophagic L. infantum towards glucantime . . . 72
4.2.2 Software validation towards intracellular Toxoplasma gondii parasite 74

**5 Conclusions and Further Research 76**
5.1 Thesis Summary and Concluding Remarks . . . 76

5.2 Recommendations for Future Works . . . . . 79

**A Image Processing: Notation and Terminology** **81**
A.1 What is Digital Image Processing? . . . . . 81
A.2 Image Enhancement . . . . . 82
A.2.1 Graylevel Transformations . . . . . 82
A.2.2 Histogram Processing . . . . . 83
A.2.3 Image Arithmetic . . . . . 83
A.2.4 Spatial Filtering . . . . . 86
A.2.4.1 Smoothing Spatial Filters . . . . . 86
A.2.4.2 Sharpening Spatial Filters . . . . . 88
A.3 Image Restoration . . . . . 88
A.3.1 Noise Models . . . . . 89
A.4 Morphological Image Processing . . . . . 92
A.4.1 Structuring Element(SE) . . . . . 92
A.4.2 Erosion . . . . . 93
A.4.3 Dilation . . . . . 94
A.4.4 Morphological Opening . . . . . 95
A.4.5 Morphological Closing . . . . . 95
A.4.6 Top Hat Transform . . . . . 97
A.4.7 Watershed Transform . . . . . 97
A.5 Image Segmentation . . . . . 98
A.5.1 Edge Detection . . . . . 98
A.5.1.1 Gradient Based Edge Detectors . . . . . 99
A.5.1.2 Laplacian Based Edge Detectors . . . . . 100
A.5.2 Thresholding . . . . . 102
A.5.2.1 Global vs Local Intensity-based Binary Thresholds . . . . 103
A.6 Other Methods . . . . . 104
A.6.1 Fill Holes . . . . . 104
A.6.2 Connected Component Labeling . . . . . 104
A.6.3 Distance Transforms . . . . . 105

**B Developed Software** **109**
B.1 Introduction . . . . . 109
B.2 Requirements . . . . . 110
B.3 Input Images' Format and Protocols . . . . . 110
B.4 Software's General Flow . . . . . 112
B.5 Cell Pipeline Parameters Setting . . . . . 114
B.5.1 Parameters subpanel . . . . . 116
B.5.2 Options subpanel . . . . . 117
B.6 Parasite Pipeline Parameters Setting . . . . . 118
B.6.1 Parameters subpanel . . . . . 118
B.6.2 Options subpanel . . . . . 119
B.7 Cytoplasm Pipeline Parameters Setting . . . . . 119
B.7.1 Parameters subpanel . . . . . 120
B.7.2 Options subpanel . . . . . 121

B.8 Outputs . . . . . 121

**Bibliography** **123**

# List of Figures


1.1 Geographical distribution of Visceral Leishmaniasis in the world . . . . . 2
1.2 Lifecycle of Leishmania . . . . . 3
1.3 Visceral and Cutaneous Leishmaniasis . . . . . 4
1.4 Image taken using DAPI technique . . . . . 9
1.5 Image taken using Phase Contrast technique . . . . . 10
1.6 Image taken using DIC technique . . . . . 11
1.7 Sample of intended result for the problem . . . . . 12

2.1 Dynamic Mean Filtering Example. $SF = 1$, $L = 2$, $Mean = 104$, $STDEV = 26.28$ and $numberofvalidPixels = 18$ . . . . . 19
2.2 Huang's $O(n)$ median filtering algorithm[1] . . . . . 20
2.3 Moving the window across the image using Huang median filter algorithm, the histogram is updated. At most $2r + 1$ pixel values are subtracted(red part) and at most $2r+1$ pixel values are added(green part). In the example $r = 2$. . . . . 20
2.4 Edge Enhancement using Unsharp Mask operator . . . . . 21
2.5 A representative object with contour bins assigned to it . . . . . 23
2.6 Subdivided kernels for circular SE with $d = 11$. Read 0-15, left to right, top to bottom . . . . . 25
2.7 Contour assignment to subdivided kernels . . . . . 26
2.8 Result of applying Otsu and Sauvola binarization algorithms on a camera captured document. $W = 15$ and $k = 0.2$ in experiments . . . . . 28
2.9 Segmentation based on Morphological Watersheds . . . . . 31
2.10 Example of Two scan Connected Component Labeling. a) input binary image. Gray pixels are foreground and white pixels are background b)pixel pre labeling c)pixel relabeling based on equivalency label set d)image with detected components . . . . . 33
2.11 Image with descending PDF . . . . . 35
2.12 Histogram of Fig. 2.11 . . . . . 36
2.13 Application of a proposed Binary Thresholding method on a sample image. Neighborhood Connectivity is assumed to be 8-connectivity and the different values on $n$ are tested . . . . . 38

3.1 DAPI test image . . . . . 42
3.2 Phase Contrast test image . . . . . 42
3.3 DAPI inverted input image . . . . . 43
3.4 Overlapping Cells in input image . . . . . 43
3.5 Cell Pipeline smoothing using Median Filter . . . . . 44
3.6 Cell Pipeline Thresholding using Mean Adaptive Threshold . . . . . 45

3.7 Cell Pipeline first Correction using Fill Holes Algorithm . . . . . 46
3.8 Cell scrap elimination using Morphological Closing . . . . . 47
3.9 Cell Pipeline second correction using Morphological Closing . . . . . 48
3.10 Euclidean Distance Transform of Thresholded image in Cell Pipeline . . . 48
3.11 Cell Pipeline Segmentation and Quantification Using Morphological Watersheds . . . . . 49
3.12 Parasite Pipeline Smoothing using Unsharp Mask filter . . . . . 50
3.13 Parasite Pipeline Thresholding step1: Black Top Hat Transform . . . . . 51
3.14 Parasite Pipeline Thresholding step2: subtraction of BTH transform from cell thresholded image . . . . . 52
3.15 Parasite Pipeline Thresholding step3: Threshold for Decreasing PDF's . . 52
3.16 Parasite Quantification using Connection Component Labeling . . . . . 53
3.17 Histogram of the sample Phase Contrast Image . . . . . 54
3.18 Cytoplasm Pipeline Smoothing using Dynamic Mean filter . . . . . 54
3.19 Edge Detection of smoothed Phase Contrast image using Sobel Edge Detector . . . . . 55
3.20 Cytoplasm Pipeline Thresholding using Gradient Based Threshold . . . . 56
3.21 Cytoplasm Pipeline first Correction using Morphological Closing . . . . . 57
3.22 Cytoplasm Pipeline second Correction using Median Filter with big kernel size . . . . . 57
3.23 Result Image with marked parasites and cells . . . . . 61
3.24 Result image with parasites assigned numerically to related cells . . . . . 61
3.25 Flowchart of the proposed solution . . . . . 62

4.1 Comparison of the cell segmentation quality on the test image between (a) our algorithm, and (b) a number of commonly used Global Intensity-Based Thresholds: Huang's (c) IsoData (d) Li (e) MaxEntropy (f) MinError (g)Otsu (h)Percentile . . . . . 66
4.2 Comparison of the parasite segmentation quality on the test image between (a) our algorithm, and (b) a number of commonly used Global Intensity-Based Thresholds: Huang's (c) IsoData (d) Li (e) MaxEntropy (f) MinError (g)Otsu (h)Percentile . . . . . 66
4.3 Comparison of the cytoplasm segmentation quality on the test image between (a) our algorithm, and (b) a number of commonly used Global Intensity-Based Thresholds: Huang's (c) IsoData (d) Li (e) MaxEntropy (f) MinError (g)Otsu (h)Percentile . . . . . 66
4.4 Comparison of the cell segmentation quality on the test image between (a) our algorithm, and (b) a number of commonly used Local Adaptive Intensity-Based Thresholds: Bernsen (c)Mean (d)Median (e)Midgray (f)Niblack . . . . . 68
4.5 Comparison of the parasite segmentation quality on the test image between (a) our algorithm, and (b) a number of commonly used Local Adaptive Intensity-Based Thresholds: Bernsen (c)Mean (d)Median (e)Midgray (f)Niblack . . . . . 68
4.6 Comparison of the cytoplasm segmentation quality on the test image between (a) our algorithm, and (b) a number of commonly used Local Adaptive Intensity-Based Thresholds: Bernsen (c)Mean (d)Median (e)Midgray (f)Niblack . . . . . 69

4.7 Comparison of the cell segmentation quality on the test image between (a) our algorithm, and (b) a number of state of the art non Intensity-Based Thresholds: k-means clustering (c)Level Sets (d)Trainable . . . . . 71
4.8 Comparison of the parasite segmentation quality on the test image between (a) our algorithm, and (b) a number of state of the art non Intensity-Based Thresholds: k-means clustering (c)Level Sets (d)Trainable . . . . . 72
4.9 Comparison of the cytoplasm segmentation quality on the test image between (a) our algorithm, and (b) a number of state of the art non Intensity-Based Thresholds: k-means clustering (c)Level Sets (d)Trainable . . . . . 72
4.10 Results of Software validation versus Manual Giemsa Counting . . . . . 74
4.11 Software validation towards intracellular Toxoplasma gondii parasite . . . 75

A.1 Background Illumination correction using Lightfield image . . . . . 84
A.2 Mean Filtering example . . . . . 87
A.3 Median Filtering example . . . . . 87
A.4 Model of the image degradation/ restoration process . . . . . 89
A.5 Some Important Probability Density Functions . . . . . 91
A.6 Examples of different Structuring Element shapes . . . . . 93
A.7 Erosion example for a binary image . . . . . 94
A.8 Dilation Example for a binary image . . . . . 95
A.9 Morphological Opening for a binary image . . . . . 96
A.10 Morphological Closing for a binary image . . . . . 96
A.11 Immersion simulation . . . . . 98
A.12 Example of various Edge Detector filters . . . . . 102
A.13 Binary Thresholding effect on an example image. The left colored image is converted to bi-level right image during binarization . . . . . 103
A.14 Pixels with 4 and 8 connectivity . . . . . 105
A.15 Effect of applying Distance Transform on two simple binary images . . . . 106
A.16 Euclidean Distance Transform for a sample input image . . . . . 107
A.17 City Block Distance Transform for a sample input image . . . . . 107
A.18 Chessboard Distance Transform for a sample input image . . . . . 108

B.1 Right order of putting images in Input Folder . . . . . 111
B.2 Main Panel of INsPECT . . . . . 112
B.3 Running Method Selection Panel . . . . . 112
B.4 Running Options Panel . . . . . 113
B.5 Ready to Run Panel . . . . . 115
B.6 Runtime Log Panel: . . . . . 115
B.7 Cell Pipeline Running Parameters Panel . . . . . 116
B.8 Parasite Pipeline Running Parameters Panel . . . . . 118
B.9 Cytoplasm Pipeline Running Parameters Panel . . . . . 120

# List of Tables


2.1 Subkernel assignment for Contour Bins[2] . . . . . 24
2.2 Equaivalecy Label set for input image Fig. 2.10 . . . . . 34

3.1 General Analytic report for the test image set . . . . . 58
3.2 Cells and Parasites report file for the test image set . . . . . 58

4.1 Comparison between global intensity based segmentation methods for cell pipeline in terms of defined measures . . . . . 67
4.2 Comparison between global intensity based segmentation methods for parasite pipeline in terms of defined measures . . . . . 67
4.3 Comparison between global intensity based segmentation methods for cytoplasm pipeline in terms of defined measures . . . . . 69
4.4 Comparison between local intensity based segmentation methods for cell pipeline in terms of defined measures . . . . . 70
4.5 Comparison between local intensity based segmentation methods for parasite pipeline in terms of defined measures . . . . . 70
4.6 Comparison between local intensity based segmentation methods for cytoplasm pipeline in terms of defined measures . . . . . 71
4.7 Comparison between non intensity based segmentation methods for cell pipeline in terms of defined measures . . . . . 73
4.8 Comparison between non intensity based segmentation methods for parasite pipeline in terms of defined measures . . . . . 73
4.9 Comparison between non intensity based segmentation methods for cytoplasm pipeline in terms of defined measures . . . . . 73

B.1 minimum requirements for running the INsPECT software . . . . . 110

***This Thesis is dedicated to the most important people in my life: my dear parents and my lovely partner***

*for their endless love, support and encouragement*

# Chapter 1

# Problem Definition and Background

The advent of high speed computers and high resolution imaging devices increasing the interest in computer-aided algorithms for a wide range of applications in industry, military, astronomy, sports and many others. *Medical Imaging* in recent years became one of the most important sub-fields in scientific imaging. This chapter is dedicated to formally explain the problem of *Leishmania Infection Ratio*, to discuss how Medical Image Processing can assist biologists and also provide motivations, objectives and challenges for solving the problem using computers.

## 1.1 Motivations for Research

Monitoring *components of the interest* in medical images has significantly drawn the attention of biologists in the recent decades. Classical approaches for such tasks mostly deal with manual processes such as generating *ground truth* by experts which is highly subjected to human-made errors and carelessness. In addition, such approaches are often time and energy consuming. The development of *image-based high content screening* has recently emerged as a new perspective for drug screening/discovery against different candidates. However, the cost of such equipment and the expertise required to develop sophisticated image analysis algorithm restricted those approaches to few laboratories all around the world. The motivation of this work is to propose an alternative solution based on Image Processing techniques for the analysis of Leishmaniasis disease image sets. The proposed solution should be accurate and fast enough and could compete with available academic and commercial packages. In this section, Leishmaniasis disease is introduced first, in section 1.1.1, then a brief biological research background for processing related images is explained in section 1.1.2. The advantages of using Image Processing techniques

as a complementary tool for accelerating processes and analysis is explored in section 1.1.3.

### 1.1.1 What is Leishmaniasis?

*Leishmaniasis* is a dangerous spreading disease which has considerable public health impact. The main cause of this disease is poverty. Malnutrition, displacement, poor housing, illiteracy, gender discrimination, weakness of the immune system and lack of resources are amongst the main reasons of disease spreading. The disease can be transmitted in tropical and subtropical regions of the world and currently 12 million people in 88 countries are estimated to be affected by the disease. It is also estimated that 2 million new cases of infection occurs each year and 350 million people are at the risk of influence [3].

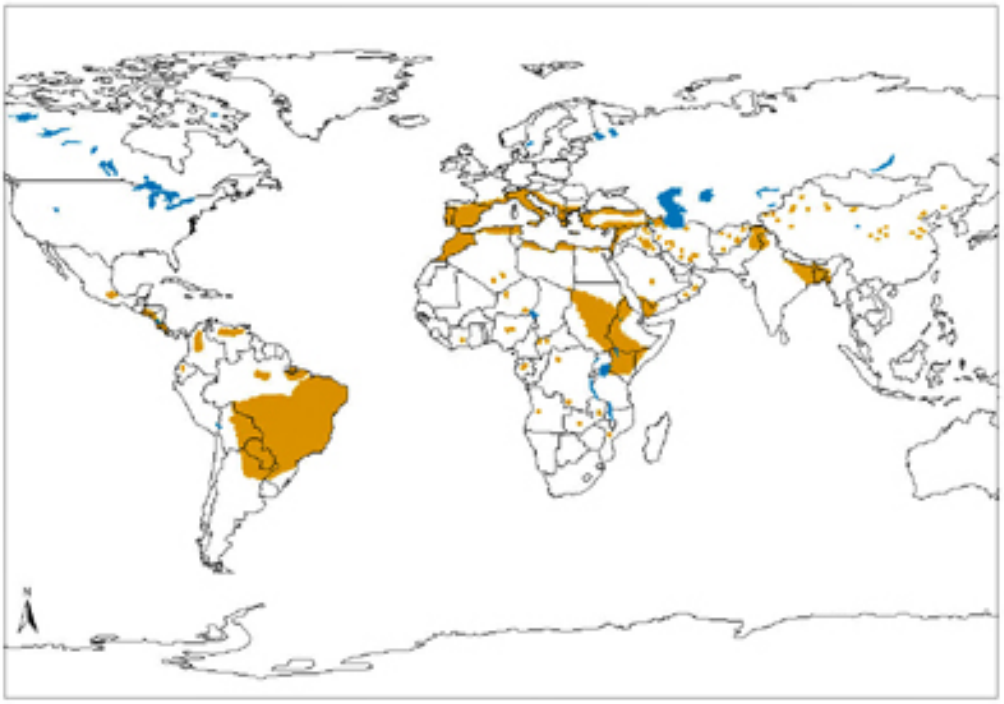


FIGURE 1.1: Geographical distribution of *Visceral Leishmaniasis* in the world[3]

The disease is caused by *Protozoan* Parasites which belongs to the genus *Leishmania* and is transmitted mainly through the bite of a special group of sandflies[3].

In the *infective stage*, the sand fly injects *Promastigotes* into the host's skin during the blood meal. Then these Promastigotes are phagocytized by *neutrophils* and infected ones release parasites which are then absorbed by macrophages. Subsequently the *diagnostic stage* begins, during which Promastigotes transform into *Amastigotes* and multiply themselves inside macrophages. To complete the cycle, the sand fly injects infected

macrophages again during a blood meal, injected Amastigotes transform into Promastigotes in the midgut, they divide themselves and finally migrate to the anterior midgut and foregut[4]. The life cycle of this process is shown in Fig. 1.2.

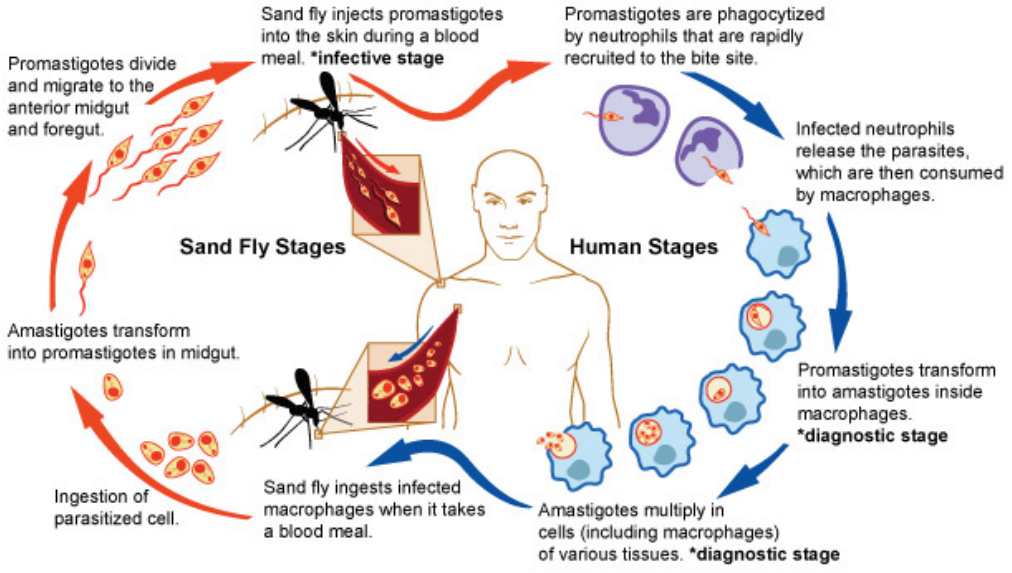



FIGURE 1.2: Lifecycle of Leishmania[4]

There are mainly two forms of disease[5]:

1. **Cutaneous Leishmaniasis** which is the most common type and is in the form of skin infection. It produces destructive and disfiguring lesions of the face which may not disappear from the skin if leaves untreated for a long time.

2. **Visceral Leishmaniasis(VL)** also called *kala-azar, black fever and Dumdum Fever* which is the more threatening type and infects integral organs such as liver, spleen and bone marrow. It is believed that this disease is the second largest parasite killer disease in the world after *Malaria*[6]. If left untreated, occasionally it will cause death of the host.

Unfortunately, there is no effective vaccine for the disease[7] and the existing common treatments which were developed decades ago have many drawbacks. They are often toxic and could even lead to patient's death. Moreover, in many cases, parasites show resistance when confronted with drugs[8, 9]. Therefore, new methods with minimum side effects are needed to treat the disease.

FIGURE 1.3: Visceral and Cutaneous Leishmaniasis

### 1.1.2 Clinical Background of Disease Research

*Leishmaniasis* disease is a a quite old Worldwide well-known disease. The disease mechanism, signs and symptoms, prevention methods and diagnosis are almost clear areas which have been well explored throughout the years. However, currently there is no routine treatment for the disease. Consequently, the main concentration of Leishmaniasis disease research during the last decades was on the Drug Discovery through screening processes. Most of the current antileishmanial drugs from the long time established *Antimonials* to the recently introduced *Miltefosine* had some drawbacks such as parasitic resistance, patient toxicity and other side effects[10]. So the main objective of the current research is to find new drugs with minimum side effects and disadvantages and maximum effectiveness.

#### 1.1.2.1 Drug Discovery

*Pharmaceutical drug*, also referred to as *medicine* or *medication* can be defined as any chemical substance which can be used for medical diagnosis, prevention or treatment of diseases. *Drug Discovery* is a general term assigned to the group of techniques which tries to discover new candidates for pharmaceutical drugs[11, 12].

The very first methods used for discovering drugs for diseases were based on traditional remedies or serendipitous discoveries. *Classical Pharmacology* also known as *Forward Pharmocology* relies on *Phenotypic Screening*(screening in intact cells or whole organisms) of chemical libraries of synthetic small molecules, natural products or extracts to detect substances which have intended therapeutic effect. On the other hand, in *Reverse Pharmocology* the assumption is that modulation of the activity of a specific protein target will have beneficial therapeutic effects and then in order to identify compounds that

bind with high affinity to the target, screening of chemical libraries of small molecules is used. The *Hits*(compounds with desired size of effects) from this screening process are then used as starting points for drug discovery. This method is widely used in drug discovery pipeline today[13]. Finally, scientists also use some features of biological molecules like their shape and volume to design drug candidates. Here, we briefly review the two most common screening methods which are widely used for drug discovery pipeline of Leishmaniasis disease.

#### 1.1.2.2 High Throughput Screening

*High Throughput Screening(HTS)* is the process of finding a new drug against chosen target for a specific disease. Generally, the Drug Targets are cellular or molecular structures which drug development techniques act upon to find drug candidates. During the process, large libraries of chemicals are tested on their ability to change the target. Usually the main objectives of HTS process are:

1. Identifying active compounds for drug target, feeding drug discovery pipelines with new candidates and optimizing both research costs and time.
2. Showing how selective the compounds are for the chosen target and minimizing off-target toxicity.

#### 1.1.2.3 High Content Screening

*High Content Screening(HCS)* is the process of detecting substances such as small molecules, *peptides* or *RNA*s that change the phenotype of the cells in desired manners[14, 15]. *Phenotypic changes* mainly include changes such as increasing or decreasing in the production of cellular products such as proteins and/or changes in the morphology (or visual appearance) of the cell. This class of screening methods may be used to determine whether a potential drug is disease modifying.

There are lots of Leishmaniasis Drug Discovery researches in literature which use different approaches of screening methods. To address some of them, Siqueiro-Neto et al. developed and implemented an automated high-throughput viability screening assay for the discovery of new drugs against Leishmania. Their HTS approach resulted in the discovery of two new antileishmanial compounds, bringing promising candidates to the leishmaniasis drug discovery pipeline[16]. Again, Siqueiro-Neto et. al. developed a high-content high-throughput image-based screening assay targeting the intracellular amastigote stage of different species of Leishmania in infected human macrophages. The

automated analysis generated parameters then used to quantify compound activities[17]. Aulner et. al. developed a biologically sound High Content Analysis assay, based on the use of homogeneous populations of primary mouse macrophages hosting *Leishmania Amazonensis amastigotes*. Their screening platform showed the ability to detect toxicity on macrophages, therefore leading to the discovery of compounds which are able to cross the membranes of the macrophage, thereby accelerating the hit to lead development process for compounds selectively targeting intracellular parasites. Furthermore, their assay aroused the discovery of anti-leishmanials that interfere with biological functions of the macrophage required for parasite development and growth[18].

Despite the fact that many drug candidates with different levels of effectiveness are detected and tested through screening assays of many researchers, new methods and candidates are still needed therefore, related researches towards finding new screening approaches and also drug candidates against the disease are ongoing.

### 1.1.3 How Image Processing techniques can help?

John Blume, Chief Science Officer for Applied Proteomics, Inc., mentioned an interesting description about HTS research as follows:

> *"Soon, if a scientist does not understand some statistics or rudimentary data-handling technologies, he or she may not be considered to be a true molecular biologist and thus will simply become a dinosaur[19]."*

The fact is that using diverse and improving methods for exploring biological structures lead to an explosion in the rate of data generated in recent years. Due to this, the development and adoption of appropriate experimental designs and analytic methods is so crucial for accelerating drug discovery development. As a result, Image Processing finds its central role in areas such as target identification, target validation, pathway analysis and *pharmacogenomics*. Here, we review some major benefits[20] which could be obtained by using Image Processing techniques in association with drug discovery pipelines.

1. **Improve Productivity in the Lab:** Diagnosis results, comparing results, peer review and preparing reports are just a few examples of many tasks a biologist should do to extract and use image data for specific purposes. There are lots of duplicated works and waste of time during these processes which can be reduced or eliminated using simple image management techniques.

2. **Fast and Effective Communication:** Classical sorting and correcting results by hand and cut-paste methods in reports can be efficiently replaced by automatic approaches which could then be shared promptly with colleagues, Worldwide.

3. **Facilitate interpretation of experiments:** Generally, comparing the results of multiple experiments on biological structures(targets) is considered to be the measure for understanding similarities and differences of responses of each structure(target) to specific compound(s). This type of comparison is extremely difficult without an organized informatics system to reduce the effort.

4. **More effective lead optimization through cellular screening:** In classic screening methods, occasionally hit selections were based on very few parameters such as measurement of binding or inhibition and yet the result hit selections were simple manual tasks. However, with the advent of techniques such as HTS, the nature of cells' response to compounds can be discerned; so hit selection process will be multi parametric. These cellular assay technologies require the quantitative analysis of up to 100,000 images a day. Surely, some image management and analysis system is needed for the processes and post screening analysis.

5. **Virtual screening:** Reanalyzing the existing results of cellular assays without needing to do the physical experiments again and again is a significant time and cost effective method in drug discovery processes. If experimental assays results are processed and analyzed in a good manner using image processing techniques, then subsequent questions of interest can be answered easily using stored data. For instance, quick search of large libraries of chemical structures to identify those structures which are most likely to bind to a drug target is a good example of Virtual Screening process[21, 22].

6. **Correlating results from preclinical experiments:** Drug discovery pipeline is often associated with preclinical research areas such as *pathology*, *toxicology*, *pharmacology* and *autoradiography* which are usually helpful to study the effects of a compound or family of compounds. Image Processing can help correlate data from these disciplines to gain a better understanding of biological mechanisms.

7. **Speeding up clinical trials:** The variety of image formats, media and delivery mechanisms are just some factors to name which makes analyzing image data complicated. An image database management system can address these issues and provide time and cost standard platform for researchers. The system is used as central repository for storing trials and distributing it between trial management team members.

What we can conclude is that Image Processing techniques can dramatically accelerate drug discovery and development through productivity improvement, better experiment interpretation and new insights derived within the experiments. Should these be used in association with knowledge of experienced experts, the outcome of experiments will be more efficient.

## 1.2 Problem Formalization

In this section, the exact definition of the problem, its objectives and challenges will be explained. It should be mentioned that the input images of the problem come from biological laboratories and also the final results of Image Processing pipeline will be validated with the help of biologist experts. So it is vital to also briefly review how input images are acquired and how final results will be analyzed. This section is organized as follows: Firstly, in section 1.2.1 we briefly describe how input data set was acquired in laboratories. In section 1.2.2, the exact problem definition is explored. Then, section 1.2.3 is dedicated to the challenges of the research and finally in section 1.2.4 objectives of the work are addressed thoroughly.

### 1.2.1 Input Images Acquisition

Generally, there are several steps with different levels of clinical details to acquire the Image set of our problem. Many of such details are out of scope of this report; thus in this section, we concentrate on general pipeline of input data acquisition and most relevant concepts to the report.

Here are the general steps for acquiring input image set of the problem:

1. Under study *Macrophage* cells are cultured in laboratory specific conditions.
2. Cultured macrophages are put to contact with disease parasites.
3. Different drug candidates with different concentrations will be added to the mixture.
4. Wait for 5 days.
5. The cells will be fixed.
6. Image acquisition is performed in the days that follow using microscopes.

There are different filters and options to capture images using microscopes; however three of them are of interest for our experiments:

### 4',6-diamidino-2-phenylindole or DAPI

The images taken of this method are used for processing cells and parasites. This filter is used extensively in *Fluorescence Microscopy*. All methods for fluorescent microscopy have the same principle. The sample is illuminated with light of a wavelength which excites fluorescence in the sample. The fluoresced light which normally has a longer wavelength than the illumination is then imaged through a microscope objective. DAPI can pass through an intact cell membrane therefore it can be used to stain both live and fixed cells[23]. You can see an example of such images in Fig. 1.4 .

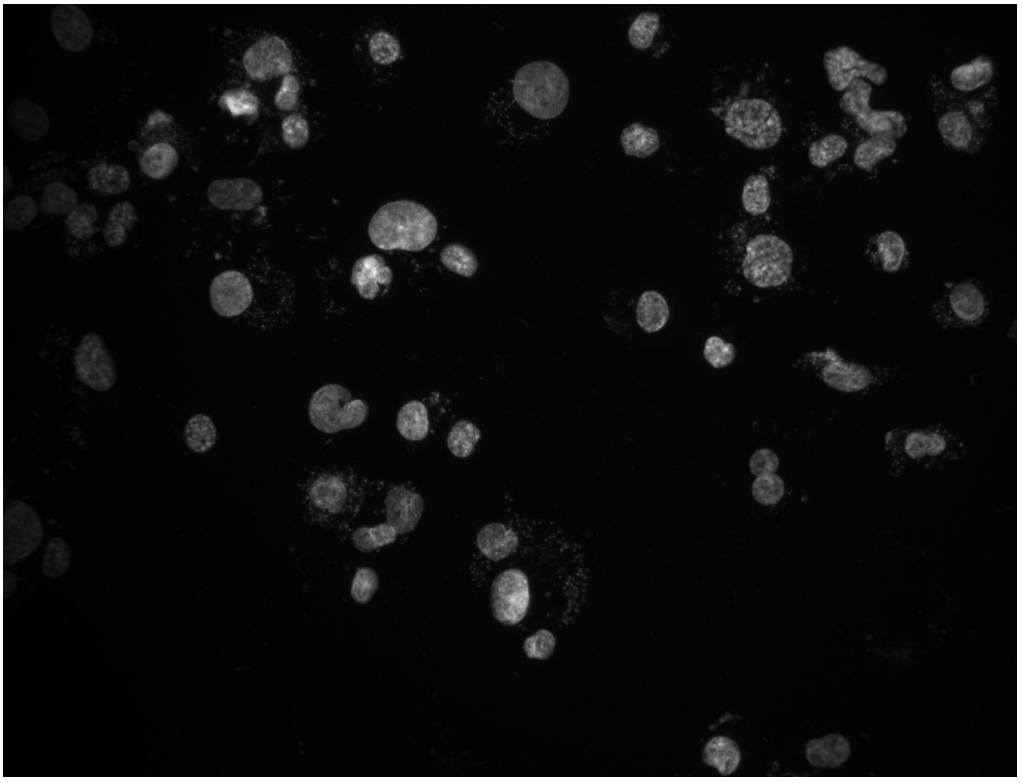


FIGURE 1.4: Image taken using DAPI technique

### Phase Contrast Imaging

In this method, the images taken are used for processing cytoplasm regions. It is a light microscopy imaging method which has a range of different applications. The idea behind this technique is that different structures in images have different refractive indexes (a dimensionless number that describes how light, or any other radiation, propagates through that medium) bending light and delaying its passage through the sample by different amounts. The delaying of the light results in *out of phase waves* compared to others. For the human eye, a microscope in phase contrast mode effectively darkens or

brightens particular areas to reflect this change[24]. You can see an example of such images in Fig. 1.5.

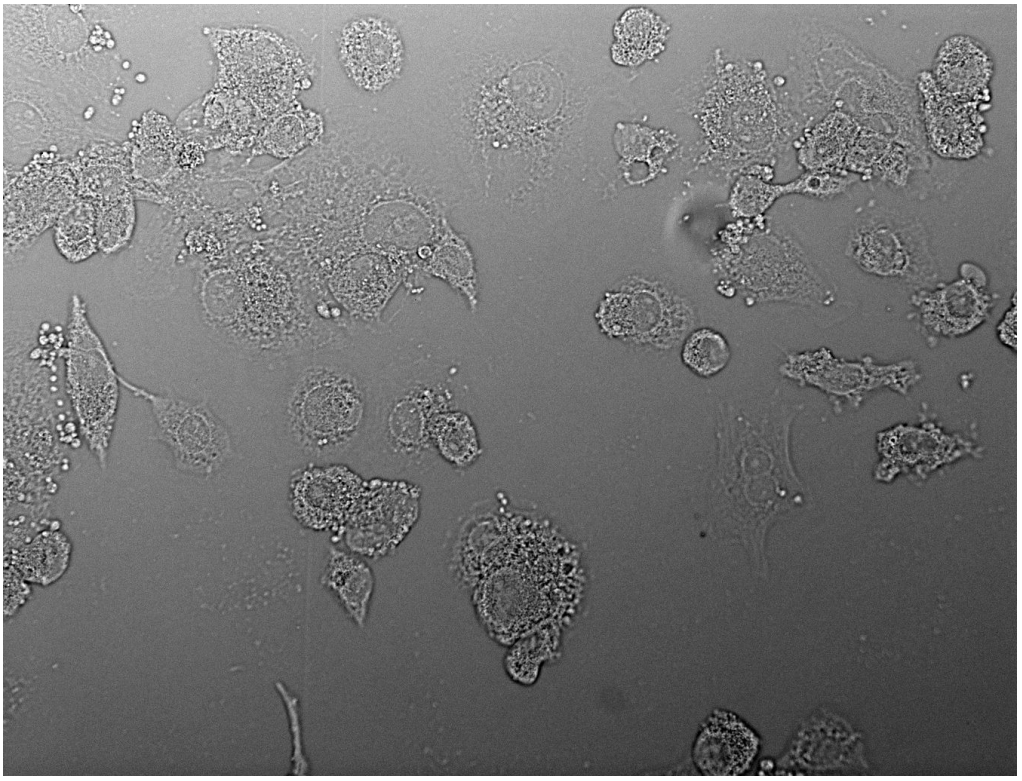


FIGURE 1.5: Image taken using Phase Contrast technique

### Differential Interference Contrast Microscopy or DIC

The taken images of this method are very similar to the images of Phase Contrast imaging method. The major difference is that it gives you more 3D view of the images capturing the sense of objects' volume. Therefore the image set of this category can be used directly for processing cytoplasm regions or alternatively as a complementary data for Phase Contrast image sets. DIC is an optical microscopy illumination technique mainly used to enhance the contrast in unstained, transparent samples. DIC works on the principle of interferometry to gain information about the optical path length of the sample to see invisible features[25]. You can see an example of such images in Fig. 1.6.

## 1.2.2 Problem Definition

Having different sets of microscopic images, the Image Processing part of the problem is formalized as follows:

On the one hand, in DAPI images, cell parts are circular, quite bright areas and the parasites are very small bright portions around them. On the other hand, in the light

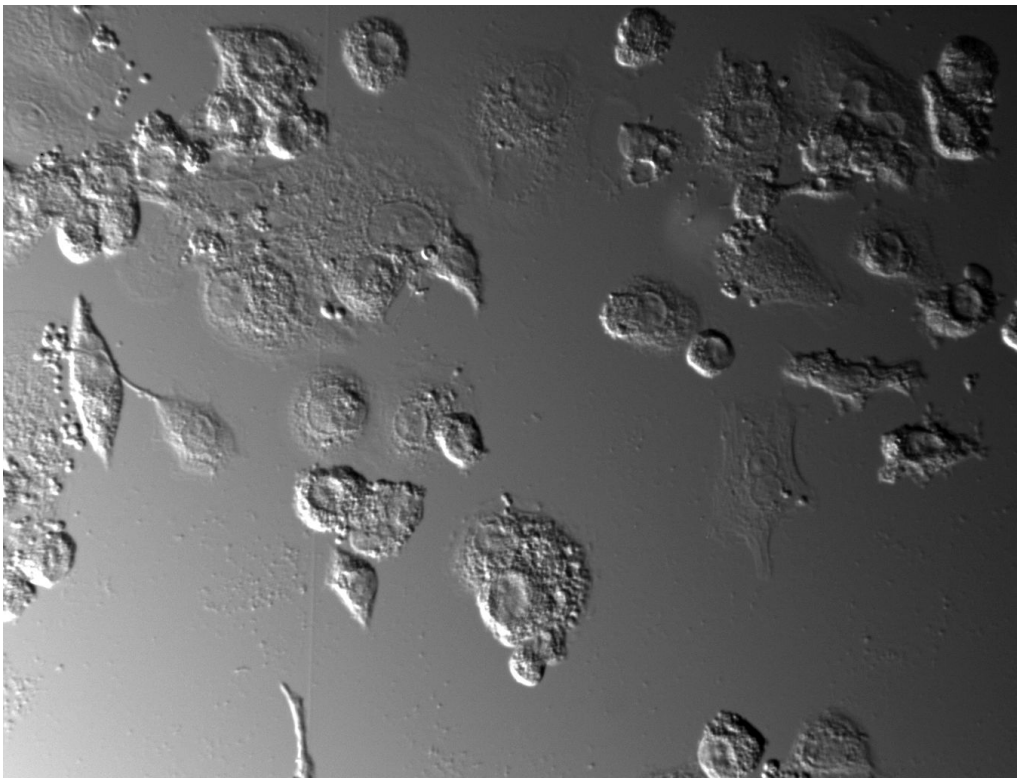


FIGURE 1.6: Image taken using DIC technique

microscopy images (DIC and/or Phase Contrast) cell parts are visible as a stain and around them one can see the trace of the cytoplasms. The main objective of this work is to: firstly detect and count the number of parasites and cells in the DAPI image, secondly detect the outline of the cells' cytoplasms in the light microscopy images and finally, combine the two images allowing us to count the number of intra cellular and extra cellular parasites. The intended result of such algorithm is something like what is shown in Fig. 1.7.

According to the above explanations, we can conclude that the problem could be divided to into four main sections:

1. **Cell Pipeline:** detection of cell regions using DAPI images. The information of detected cells such as their location, center of cells, area of cells and others should then be recorded for further visualization and analysis.

2. **Parasite Pipeline:** extraction of parasites around cells using DAPI images. Again, the information of detected parasites should be recorded for upcoming uses.

3. **Cytoplasm Pipeline:** detecting cytoplasm regions of the images using DIC and/or Phase Contrast images. The exact result of detected cytoplasm regions should also be saved.

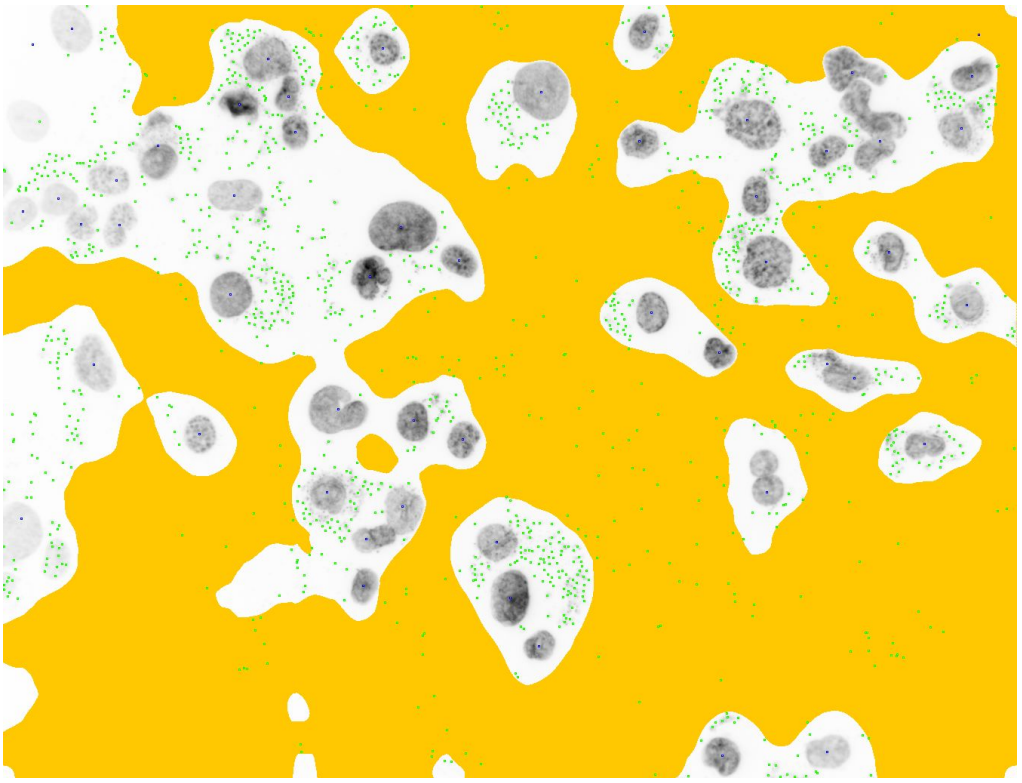


FIGURE 1.7: Sample of intended result for the problem

4. **Analytic Section:** The main objective of this step is to judge about infection ratio of cells using interpretation of extracted data from the first three pipelines. These interpretations will then be used by biologists and other experts as the measure for determining effectiveness of drug candidates.

   The level of infection can be calculated in several ways. Some are just counting the mean number of parasites per cell. However for the purpose of our experiments, is to calculate the infection ratio from a parameter called *parasitic index(PI)*. The following equation represents the PI :

$$ParasiticIndex(PI) = \%ofinfectedcells \times meannumberofparasitespercell \tag{1.1}$$

   To find the above mentioned statistics, the analytic section should also suggest methods for assigning detected parasites to most related cells, and also discriminating between intra cellular parasites and extra cellular parasites of each cell.

### 1.2.3 Challenges

Although the problem is well-defined and objectives are clear, there are some challenges during the experiments which should be addressed. The final solution tries to decrease the side effects of the problems caused by these challenges to the highest degree.

**Image Acquisition standards**

Frequently, even for well-defined Medical Image Processing problems, input data(medical image set) is generated in different laboratories and for each set of experiments, different instruments and methods may be used. This variation of Image Acquisition methods creates islands of data.

The laboratory conditions in which image capturing devices(cameras, microscopes, etc) acquire images are very important parameters once they directly influence the quality of Image Processing pipeline results. Fluorescent Staining methods, microscopic filters and the accuracy of capturing process are between some important issues which influence the outcome of future Image Processing pipeline. For instance, consider capturing Cytoplasm image set of our problem, once using phase contrast (real light flat image) filter of used microscope and the other time using DAPI filter and fluorescent light which is called DIC(Differential Interference Microscopy) image. Both methods are common in laboratories and the major difference between the two sets is that DIC images give 3D alike images with emphasis on volume of cytoplasms. Now, the challenge is to ensure that the program is able to endeavor such variations of clinical experimental conditions with some level of tolerance.

Another important problem arises from photography conditions namely uneven illumination of the image. Usually inhomogeneity caused by the imaging equipment will lead to smoothly varying non-linear illumination across the image and if not conveniently treated, it will have significant unwelcomed effects on the results. Two main categories of Image Processing operations which deal with this problem are *Segmentation* and *Restoration*.

*Image Segmentation* is generally the name of the operations which try to extract regions of interest from the raw input images and specific further analyses is problem dependent. This process is considered to be one of the most basic and important components of Image Processing packages. For our problem, segmenting cells areas, parasites and cytoplasm regions are the instances of using such operators. Despite the fact that the objectives of these operations are easy to state, they are usually difficult to achieve accurately due to several difficulties. Image Segmentation approaches mainly depend on some input image features such as pixel intensities, edge information and texture, etc.

However such algorithms are very sensitive to random noises and also variations in illumination. Non uniformity of illumination can disturb histogram distributions, causing several problems such as significant overlaps between intensity peaks and thus leading to substantial misclassification in traditionally intensity based classification methods[26]. The proposed algorithm's components should be in such a way that they handle these possible variations with some degree of flexibility.

*Image Restoration* is the set of techniques to model the degradation caused by random noises of uneven illumination and it tries to recover the original image using smoothing and denoising algorithms. However, using such methods have some disadvantages such as losing very weak parts of regions of interest during degradation processes. Specifically, for the problem of this report, if such methods are used abnormally, then some portions of cells, parasites or cytoplasm areas may be destroyed along degradation process. Thus, smoothing filters and degradation models should be used wisely and with some prior knowledge of possible effects.

**Large Data Management**

Managing higher resolution input images is often a challenging task. On the one hand, this kind of images is welcome for most of the Image Processing operations which do pixel manipulation tasks. In fact, higher resolution means generally higher accessibility level to minor details in images. This is specifically important when some parts of Image Processing pipeline includes operations which work with such details. On the other hand, processing and rendering of data on a high resolution format is a costly task which requires scalable algorithms to reduce processing time and increase memory efficiency[27].

The image set of our problem contains high-resolution images with detail preserving formats (such as TIFF) and in different points of the algorithm, it contains operations which deal with individual pixel's details. For instance, the parasites could be considered as very small portions of the images; so the accuracy of detecting and extracting them highly depends on the resolution of input images, in other words, the level of details' accessibility. Therefore, having both accuracy in results and efficiency in computational complexity is crucial for the proposed solution.

**Parameter Estimation**

Generally, many Image Processing operators which have contribution in solving a problem work with one or more parameters. Just to name a few, consider the window(or kernel) size for smoothing filters, structure element radius for morphological operators or threshold value set for different segmentation operators. Considering the fact that Image Processing is not a single-step process and that several steps should be executed consecutively to finally obtain the information of interest from images, then for a large

problem such as our problem, we will have a parameter set of different operators. The ideal case occurs when we can have parameter estimation for all parameters subjecting the problem to the algorithm, running totally automatically and estimating each parameter's value during run time. However, this hypothesis is not very realistic due to the fact that parameterizations can be as diverse as the application areas. The approaches for parameter estimation encompass a wide range of techniques, often tuned to the application, the underlying data and viable assumptions. However, there are some techniques such as *Optimization of Filter Kernels*, *Optical Flow Estimation*, *Noise Modeling* and so on which try to eliminate or reduce the parameter domain of related operators and could be used to decrease the dependability of the solution to parameters[28].

**Validation**

Till the date of publishing this report, there is no package dedicated to intracellular parasites' counting. Biologists generally use *ImageJ* software to manipulate images(free software) and some are writing complementary pluggins. While some work well to determine number of cells by DAPI staining, none are dedicated to intracellular parasites counting. Some rich laboratories use *IN Cell Analyzer*[29]. Nothing really exists for the rest and small scale screening is the activity of many other laboratories. So, in what result validation is concerned, the only way beyond manual counting done by biologist experts is to cross check results with existing but yet incomplete softwares with quite the same functionalities.

Due to the above challenges associated with solving the problem and the fact that the problem has not got an unique solution, the main objective of the proposed solution is to give good enough results in terms of accuracy and cost and meanwhile build a comprehensive framework with the highest degree of interaction with user for result validation.

### 1.2.4 Objectives of the Research

The few computational based academic and commercial existing solutions for processing clinical Leishmaniasis image sets suffers dramatically from inaccuracy of the results and also none of them are dedicated to the major problem of intracellular parasite counting. Therefore, the main objective of this research is to develop a competent framework for processing such clinical images to assist biologists and other experts visualize and analyze results easily. The given solution is claimed to process images with acceptable computational complexity and very good accuracy of results. Furthermore, it tries to address the challenges mentioned in the previous subsection, as follows:

- **Image Acquisition standards:** set some minimum standards and requirements for clinical image acquisition which should be obeyed Worldwide to gain more efficient results.
- **Large Data Management:** decreasing time and memory complexities of the solution as much as possible while preserving maximum accuracy in results.
- **Parameter Estimation:** the policy is to make the framework work in a more automatic fashion by estimating parameter values of different Image Processing operators, resulting in the elimination of parameter setting by user.
- **Validation:** the framework itself can be considered as a validation tool for future researches in association with manual validations of biologist experts.

## 1.3 Thesis Outline

This thesis is organized in five chapters. The present chapter explains the problem of interest, its biological and computational research background, motivation and objectives for the research and the encountered challenges. Chapter 2 gives a comprehensive literature review which covers Image Processing approaches for solving the problem. Chapter 3 describes proposed solution and experimental setup and results. Chapter 4 is dedicated to validation of the project outcomes, evaluating and comparing results from both computational and biological perspectives. Chapter 5 summarizes the achievements of the thesis and points out possible research issues to be addressed in the future by other researchers.

# Chapter 2

# Image Processing: Methods and Perspectives

## 2.1 Introduction

The main objective of this work is to introduce a fast and dependable framework for manipulating medical images of Leishmaniasis parasitic disease. The primary tool used here is Image Processing systems which tries to both enhance images' information for human interpretation and also process images' data for later storage, analysis and transmission. To have an efficient solution in terms of time complexity and accuracy, firstly operators which are most fitted to the needs of problem in hand should be used, and secondly, novel methods should be proposed to help the pipeline better process the images. For this purpose, in this chapter, the most advanced and hi-tech related perspectives which are used during the pipeline is explained. Moreover, state-of-the-art proposed perspectives are described and proofs of efficiency of such methods are given. Appendix A also provides the readers with the basic related Digital Image Processing concepts used or referenced during this work.

## 2.2 Advance Methods

In this section, the operators which help proposed algorithm to process images faster and more accurate is explained. It should be noticed that many of these algorithms have classic or modified versions. However, the used version of such operators for this report is either more fitted to the nature of input images or has the same effect but shown to be faster in comparison with other versions.

### 2.2.1 Dynamic Mean Filter

Although the classic implementation of *Mean Filter*[30] is good enough in many denoising and smoothing image processing tasks, the filter suffers from some problems which sometimes cause very poor results. First of all, using large filters will cause intensive smoothing and losing many useful details. Secondly, single pixels with very different values from their surroundings can significantly affect the mean value of all of the pixels in the neighborhood and finally, if sharp edges are required in the output, then blurring effects of mean filter are unwelcome[31, 32]. Some people use some relaxed versions of these filters to overcome some parts of the problem.

An algorithm proposed by Vijaykumar, Vanathi and Kanagasabapathy[33] and showed to be a fast and efficient one when dealing with *Gaussian Noises* in the images. It removes noises effectively while preserving edges and the computational complexity of the algorithm is low. The algorithm works with a sliding window and is as follows:

Let the size of the window be $2L+1$ by $2L+1$. Then:

1. Standard deviation of the image is computed($STDEV$) and $L$ is set to 1.
2. Calculate the absolute value of the difference between the current pixel in process $P$ and each of the window element pixels $w(i,j)$ as $Diff(i,j) = |P - w(i,j)|$
3. For each calculation, if $Diff(i,j) < STDEV \times SF$, which $SF$ is a smoothing factor, then the pixel located at $i,j$ is considered as a valid pixel.
4. If the number of valid pixels is more than $2 \times (2L+1) - 1$ or the maximum value for $L$ is reached, then the mean value is calculated for the window and the $P$ value is replaced by that value. Otherwise $L$ is incremented and the process will be repeated from 2 until proper window size is found for current pixel.

Let's show this with an example. In the Fig. 2.1, we want to apply the filter for the center pixel. Let $SF = 1$. Then the Mean Value of neighbor pixels becomes 113 and $STDEV$ is 33.78. Number of valid pixels will be 5, then. So, $L$ is incremented and the new values will be Mean = 104, $STDEV = 26.28$ and valid pixels = 18. Therefore, the value of the center pixel is changed to 104.

Vijaykumar showed in his work that his proposed algorithm removes noise with edge preservation for low to high Gaussian noise corrupted images. Moreover, in terms of peak signal to noise ratio($PSNR$) and the mean absolute error($MAE$), the algorithm excels *mean filter*, *wiener filter*, *alpha trimmed mean filter*, *K-Means filter* and *bilateral filter*[33].

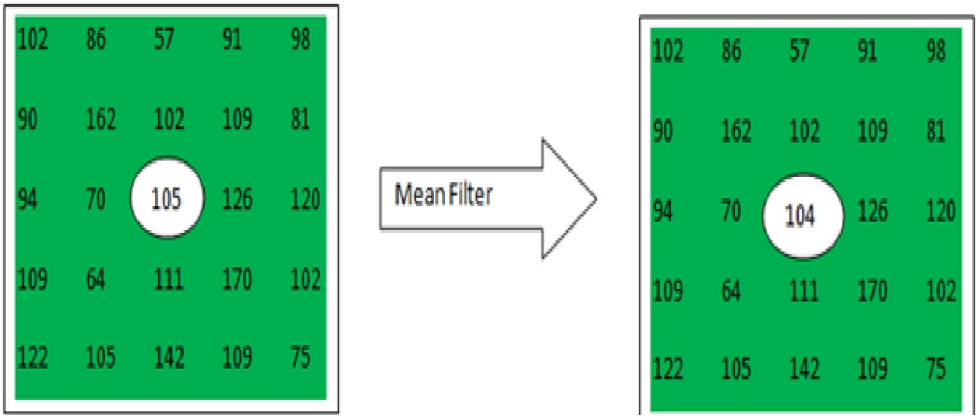



FIGURE 2.1: Dynamic Mean Filtering Example. $SF = 1$, $L = 2$, $Mean = 104$, $STDEV = 26.28$ and $numberofvalidPixels = 18$

### 2.2.2 Median Filtering in Constant time

The classic implementations of *Median Filter*[34] are usually considered to have high computational costs since it needs to sort sliding window elements' intensities for each pixel. As a result, with increasing window size and also for larger images, while this algorithm generates smoother noise-free images, on the other hand it takes more time to accomplish the task.

There are several implementations of Median Filter in literature with lower runtime complexities. An efficient and well known algorithm is the *Fast two dimensional Median Filtering Algorithm*, proposed by Huang[35] which is based on *Running Median Windows* and has $O(n)$ complexity(see Fig. 2.2 ).

The algorithm uses a kernel of radius $r$ and a histogram for holding pixel intensities of the current moving window. When we move the window from one pixel to the next one, the window only shifts one column. So in order to update the values of the histogram, we just need to throw away the passed column values and add new column values to the histogram. So for each pixel, $2r + 1$ additions and $2r + 1$ subtractions should be done to update the histogram. The median value of the window elements intensities, can then be computed easily from the histogram in constant time by adding the values from one end and stopping when the sum reaches $(2r + 1)^2/2$[1].

### 2.2.3 Unsharp Mask

*Unsharping Mask(UM)* is the main operation in Image Processing to emphasize edges and differences between light and dark areas in the images[36]. Although more pixels

**Input:** Image $X$ of size $m \times n$, kernel radius $r$
**Output:** Image $Y$ of the same size as $X$
  Initialize kernel histogram $H$
  **for** $i = 1$ to $m$ **do**
    **for** $j = 1$ to $n$ **do**
      **for** $k = -r$ to $r$ **do**
        Remove $X_{i+k,j-r-1}$ from $H$
        Add $X_{i+k,j+r}$ to $H$
      **end for**
      $Y_{i,j} \leftarrow \text{median}(H)$
    **end for**
  **end for**

FIGURE 2.2: Huang's $O(n)$ median filtering algorithm[1]

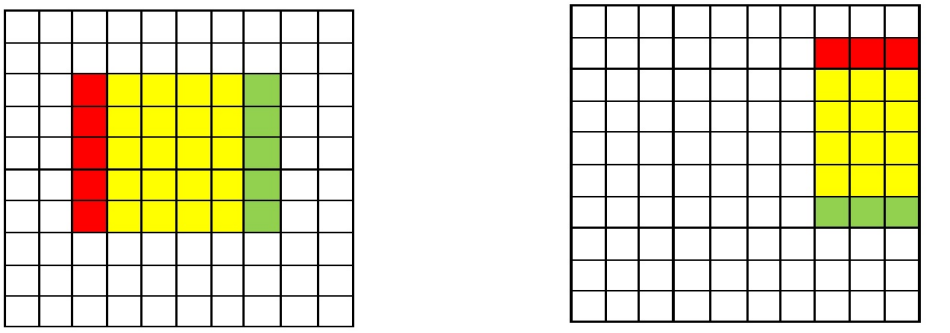



FIGURE 2.3: With moving the window across the image using Huang median filter algorithm, histogram will be updated. At most $2r+1$ pixel values will be subtracted(red part) and at most $2r + 1$ pixel values will be added(green part). In the example $r = 2$

help us to see more details in the images, neither does it highlight edges nor can it sharpen the image to viewer's eye. So the task of Sharpening operators such as UM is to enhance high frequency components like edges and produce clearer images[37].

Unsharping Mask process has two steps. In the first step, a smoothed version of the original image is subtracted from it to highlight the edges and other high frequency components of the images. Then in the second step, a constant of the resulting image of the first step is added to the original one. What is actually done in this process is the elimination of *lowpass* components of the image using subtraction first and then adding high pass detected components to the original one, resulting in an image where high frequency components are amplified. We can also see the philosophy behind the name of the operator from its definition. The name comes from the fact that this operator sharpens some structures in images with the help of an unsharped or smoothed version of the image. Let $f$ be the original image, $g$ the resulting image of step 1 and $k$ the constant which is multiplied by $g$ in step 2. Then the formal mathematical definition of UM is:

$$g(x, y) = f(x, y) - f_{smooth}(x, y) \tag{2.1}$$

$$f_{sharp}(x, y) = f(x, y) + k \times g(x, y) \tag{2.2}$$

Fig. 2.4 shows a sample using of UM operator. The Left image is the original image which is captured by the camera. However, it has blurred edges. In order to have sharp edges, mean kernel of size 3 by 3 is subtracted from the original image(middle image) and then by choosing $k = 0.7$, we multiply this image to the original one, recovering it with highlighted edges.

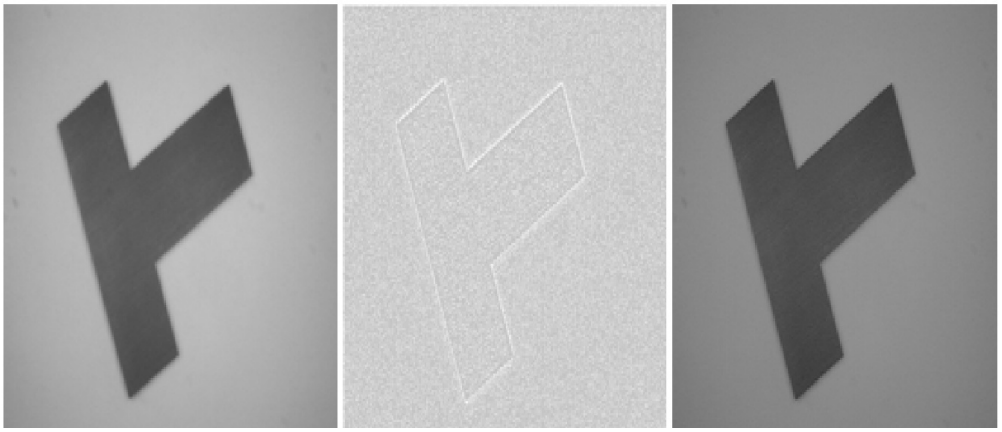


FIGURE 2.4: Edge Enhancement using Unsharp Mask operator[38]

Finally, there are some notes that we should consider for implementing UM efficiently:

1. Smoothing function $f_{smooth}$ should be chosen and used wisely; otherwise subtracting a bad smoothed image from the original one will not contain edges and other high frequency elements anymore. *Mean Filter*, *Gaussian Blur Filter* and other blurring based filters are occasionally good choices for this purpose.

2. The reasonable values for parameter $k$ are between 0.2 and 0.7 and higher values will generate sharper and more unrealistic images.

3. Presence of low noise levels will affect the results of the operator. So it is highly recommended to use the filter after removing such kind of noises.

### 2.2.4 Kernel Sub Division(KSD) Algorithm

For an image of size $m \times n$ and structuring element with area $a$, computational complexity of *Erosion*[39] and *Dilation*[39] operators is $O(m \times n \times a)$ since for each $m \times n$ pixel of the image, the algorithm should do one translation along the area of the *Structuring Element*[39]. Decomposition of structuring elements could be used to achieve a speed up for the algorithm. However the results of the operations are not exactly the same with the classical implementations and have some errors. Another speed up can also occur if we just consider erosion and dilation of edge pixels and then add the results to the original image. The computational complexity, in this case, then reduces to $O(l \times d^2)$ where $l$ is number of edge pixels and $d$ is side size of the SE. Narayanan proposed a method called $KernelSubDivision(KSD)$ Algorithm which has the same results as classic implementations for binary dilation and erosion but also speed ups that version dramatically. It does this by fractioning both $l$ and $d^2$ terms by decomposing Structuring Elements to several subsets and also working only on image contours.

The proposed algorithm[2] does the speed up by changing both $l$ and $d^2$ terms. In order to accomplish this, two different processes called *Contour Binning* and *Kernel Sub Division* are used. It should be noted that a circular kernel is used in next sections to illustrate purposes. However, the algorithm works for any type of the SE shape.

#### 2.2.4.1 Contour Binning

In *Contour Binning* part of the algorithm, all the pixels of the image are divided into contour bins according to their 4-connected neighborhood pixels. There will be 5 bins as follows:

$L_1$: Set of pixels which have one neighbor with different value in 4-connected neighborhoods

$L_{2adj}$: Set of pixels which have two perpendicular neighbors with different value in 4-connected neighborhoods

$L_{2n-adj}$: Set of pixels which have two non-perpendicular neighbors with different value in 4-connected neighborhoods

$L_3$: Set of pixels which have three neighbors with different value in 4-connected neighborhoods

$L_4$: Set of pixels which have four neighbors with different value in 4-connected neighborhoods

Under such assignment, we have:

$$L = L_1 + L_{2adj} + L_{2n-adj} + L_3 + L_4 \tag{2.3}$$

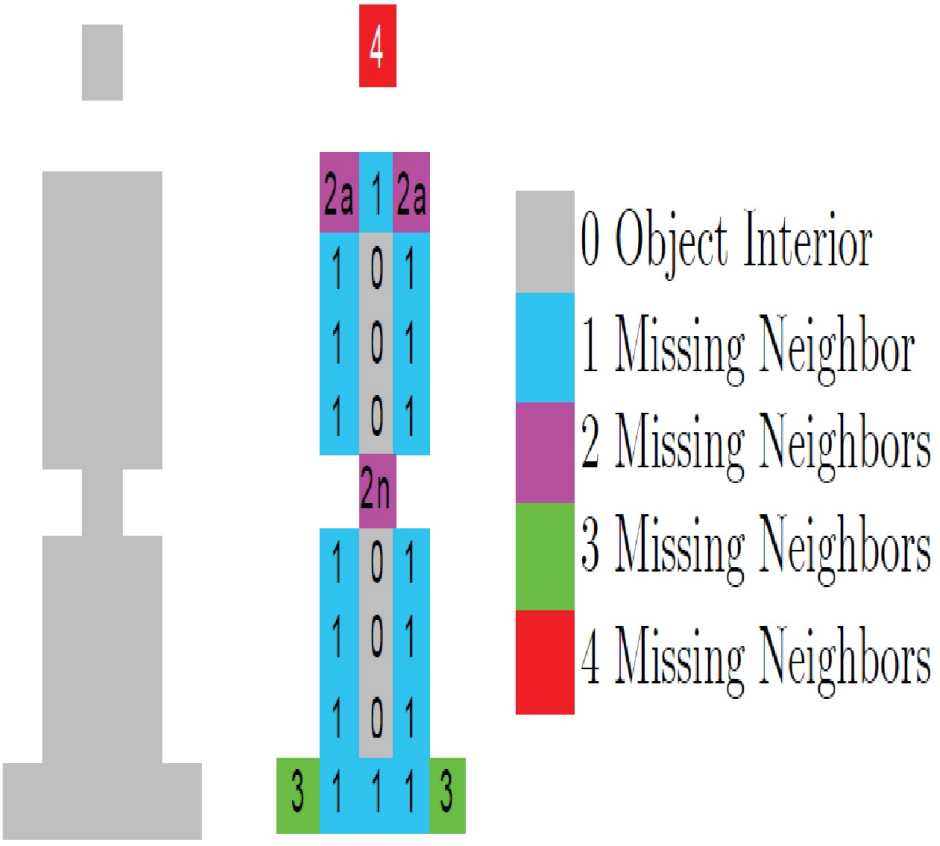



FIGURE 2.5: A representative object with contour bins assigned to it[2]

#### 2.2.4.2 Kernel Subdivision

In this part, original kernel (which is a circular Structuring Element in our illustration) is divided into different subkernels in such a way that each subkernel can be assigned to one contour bin. What we do in this algorithm is actually decide which subkernel is appropriate to use with which contour bin. Then, the subkernels are stored as a lookup table and will be used when needed during binary erosion or dilation. So the main task here is to define a proper subdivided kernel set and then find a mapping between the set's elements and correlated contour bins.

For instance, you can see the subdivided kernel set for circular SE in Fig. 2.6. Moreover, the mapping relation between this set's elements and contour bins link up as is shown in Table 2.1.

TABLE 2.1: Subkernel assignment for Contour Bins[2]

| Contour Bin | subkernels |
|---|---|
| $L_1$ | 1,2,4,8 |
| $L_{2adj}$ | 3,6,9,12 |
| $L_{2n-adj}$ | 5,10 |
| $L_3$ | 7,11,13,14 |
| $L_4$ | 15 |

Now, if we calculate the computational complexity for the proposed algorithm, we have:

For bin 1: $complexity = o(l_1 \times (\frac{(d-1)}{2}))$

For bin 2adj: $complexity = o(l_{2adj} \times (\frac{l \times A}{4}))$

For bin 2n - adj: $complexity = o(l_{2n-adj} \times d)$

For bin 3: $complexity = o(l_3 \times (\frac{l \times A}{2}))$

For bin 4: $complexity = o(l_4 \times A)$

Where $A = areaofkernelforeground = \pi(\frac{(d^2)}{4})$

Total complexity is the sum of bins' complexities and as we can see, the complexity of the algorithm is a direct function of distribution of edge pixels in the image. Furthermore, the algorithm reduces this complexity significantly by exploiting data redundancy in the image.

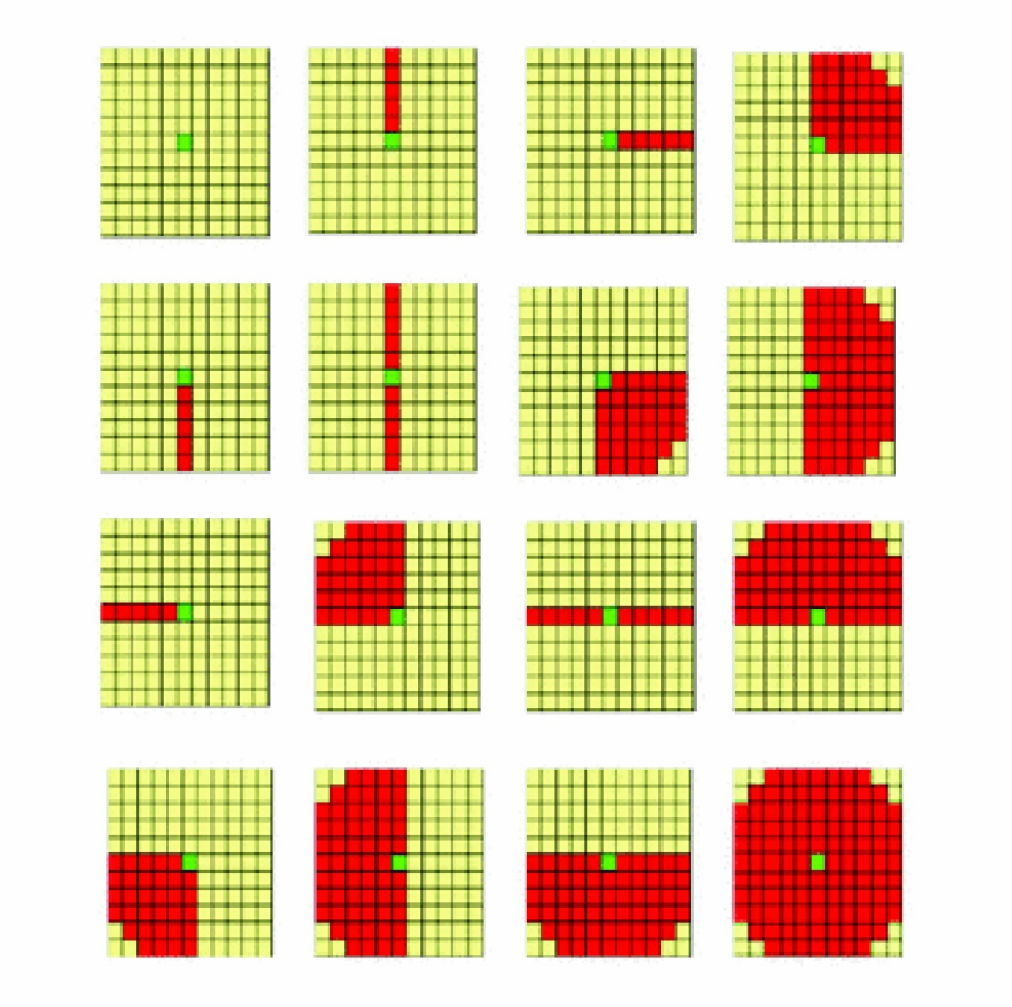


FIGURE 2.6: Subdivided kernels for circular SE with $d = 11$. Read 0-15, left to right, top to bottom[2]

### 2.2.5 Mean Adaptive Threshold

There are two main categories of solutions to find the thresholds per pixel in local thresholding methods. (i) *Chaw and Kaneko* approach and (ii) *local statistical* approach. In the first approach, the image is divided to array of overlapping subimages and then using the histogram of these subimages, thresholds are calculated for each subimage. Finally, for each pixel, results of subimages' thresholds are interpolated and the final threshold is then calculated. This approach is efficient yet expensive computationally, so people usually prefer to use the other category, called *local statistical* approach. In this approach based on the nature of input images, intensity values of some neighbourhood for each pixel is examined and thresholds are assigned to pixels based on this investigation. To clarify this idea, some simple functions which could be used as the measure are listed

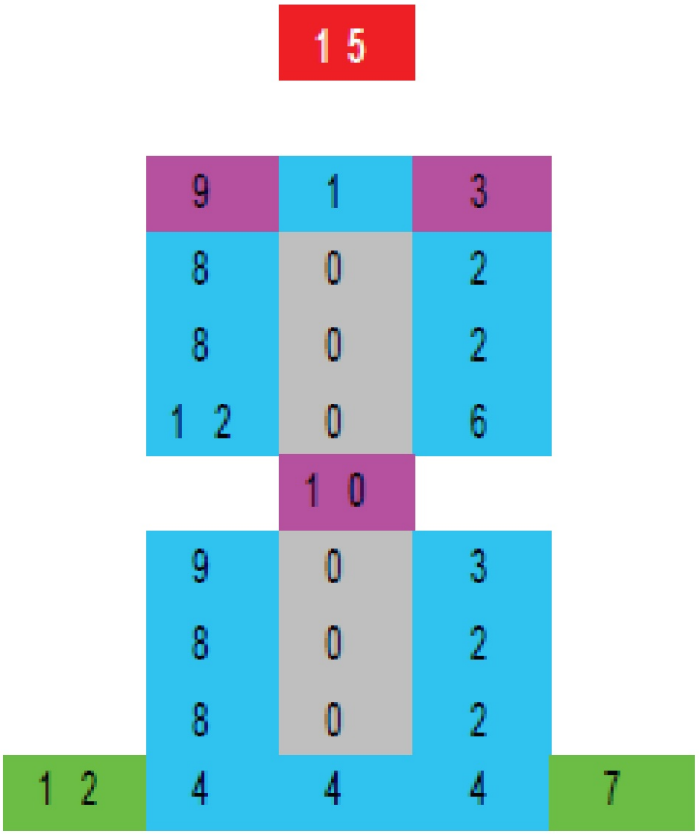



FIGURE 2.7: Contour assignment to subdivided kernels[2]

below[31, 36].

$$T = mean \tag{2.4}$$

$$T = median \tag{2.5}$$

$$T = \frac{max + min}{2} \tag{2.6}$$

In all functions and for each pixel $P$ in the image, $T$ value for neighborhood pixels in a window of size $m$ by $n$ is calculated. If the intensity value of the $P$ is lower than this value, then it sets to background, otherwise it is a foreground pixel. Also it should be noted that window size should be large enough to fairly consider both background and foreground pixels. Moreover, we can use a $c$ offset value for each function to hint the algorithm on how much the current pixel value could differ from the mean.

Among the above simple functions, *local mean* function has received lots of attention in recent years and because of its strength in segmenting images, most researchers use this parameter as a base for local binarization processes. However, local mean alone

occasionally cannot fulfill all of the segmenting requirements. So different techniques and approaches are tested and analyzed to verify with which parameters, local mean, can perform well. Some of these evaluations can be found here[40]. Based on these findings, we can conclude that two binarization techniques, called *Niblack's binarization* and *Sauvola's binarization* methods could be considered the state-of-the-art techniques for local binarization.

#### 2.2.5.1 Local Adaptive Thresholding Using Sauvola and Niblack Techniques

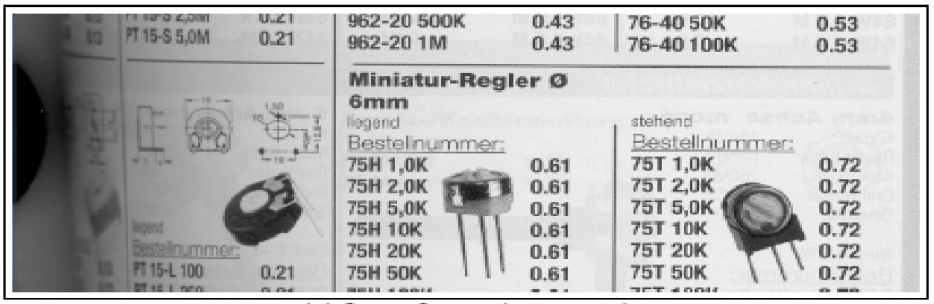
The idea behind both methods to calculate threshold values is to use *mean* value and *standard deviation* of window pixels simultaneously. Let $m$ be the mean value and $s$ be the standard deviation of the window pixels in which $P$, the current pixel in process exists. Moreover $k$ is a constant which takes positive values and $R$ is the maximum value of the calculated standard deviations for the pixels. The formula for each technique is given below.

$$Niblack : T = m + k \times s \tag{2.7}$$

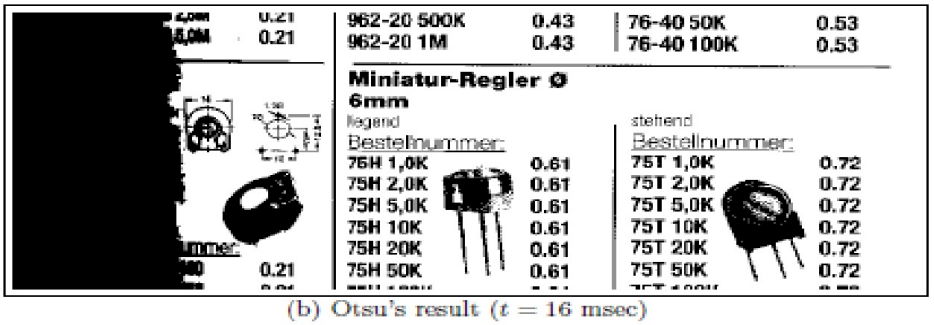
$$Sauvola : T(x, y) = m(x, y)[1 + k(\frac{s(x, y)}{R} - 1)] \tag{2.8}$$

The drawback of the Niblack's method is that if by any chance the background area contains some foreground noise or vice versa, then the grey values of these unwanted details will exceed threshold value and as a result will generate poor segmented areas. Sauvola's Method is actually a generalized formula of the Niblack's method and overcomes drawbacks of this method. To prove this, first consider that we want to find the threshold for a high contrast region. For such region, $s(x, y) \cong R$ and as the result, $T(x, y) \cong m(x, y)$ which is quiet the same result with Niblack's method. On the other hand, assume we want to set the threshold for a low contrast region. In such region, $s(x, y)$ is quiet low and so the $t(x, y)$ is lower than $m(x, y)$ and causes the removal of dark regions of the background correctly.

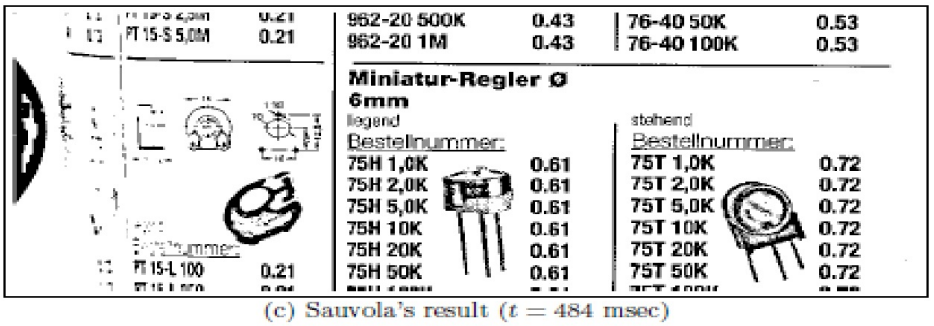
The only thing that makes this methods computationally expensive is that for all pixels we need to compute mean($m$) and standard deviation($s$). If the size of the window is $W$ by $W$ and the image is $N$ by $N$, then the complexity of the algorithm is $O(W^2N^2)$. Sauvola proposed a speedup for the computation of $m$'s and $s$'s. They proposed to calculate the threshold for a pixel, then interpolate the results for the next $n$ pixels and so on. While this approach speeds up the computation to some factor, on the other hand it risks losing the accuracy of computation of threshold values. Shafait proposed another




(a) Input Image ($842 \times 324$)




(b) Otsu's result ($t = 16$ msec)




(c) Sauvola's result ($t = 484$ msec)

FIGURE 2.8: Result of applying Otsu and Sauvola binarization algorithms on a camera captured document. $W = 15$ and $k = 0.2$ in experiments[40]

method based on integral images to speed up the computations. This method is accurate and works independently from window size.

#### 2.2.5.2 Integral Images for computing local means and variances

Let $g$ be the input image. To comput the corresponding pixel intensities of integral image $i$[40, 41], we need to sum up the pixels' intensities located above and left of the pixel in the original image $g$. The formula is like this:

$$I(x, y) = \sum_{i=0}^{x} \sum_{j=0}^{y} g(i, j) \tag{2.9}$$

$I(x, y)$ can be computed in a single pass using:

$$s(x, y) = s(x, y - 1) + i(x, y) \tag{2.10}$$

$$ii(x, y) = ii(x - 1, y) + s(x, y) \tag{2.11}$$

where $s(x, y)$ is the cumulative row sum, $s(x, -1) = 0$ and $ii(-1, y) = 0$.

Now that we have the integral image, mean value for each pixel in the original image is calculated as follows:

$$m(x, y) = (I(x + w/2, y + w/2) + I(x - w/2, y - w/2) - \tag{2.12}$$

$$I(x + w/2, y - w/2) - I(x - w/2, y + w/2))/w^2 \tag{2.13}$$

And for local standard deviation, we have:

$$s^2(x, y) = \frac{1}{w^2} \sum_{i=x-w/2}^{x+w/2} \sum_{j=y-w/2}^{y+w/2} g^2(i, j) - m^2(x, y) \tag{2.14}$$

What we should do here is merely compute integral image for $g^2$ and the rest of the process follows the same method used to calculate $m(x, y)$.

So we see that we can compute $m(x, y)$ and $s(x, y)$ for the original image without considering the size of the window in which we work and this will help us reduce the computational complexity from $O(W^2N^2)$ to $O(W^2)$.

### 2.2.6 Segmentation based on Morphological Watersheds

There are several approaches for segmenting an image. Segmentation based on Edge Detection and Thresholding(locally and globally) is the most common approaches. However, each of these approaches have some advantages and disadvantages. For instance, global thresholding methods are quiet fast yet inaccurate in many situations. On the other hand, local thresholds are slower but also more accurate and they compensate the problems regarding illumination non uniformity to some extent. Recently, a third approach towards segmentation is explored which roots in *Mathematical Morphology* concepts and is called *Segmentation Based on Morphological Watersheds*[42]. The segmentation methods of this category often produce more stable segmentation results including connected segmentation boundaries.

Detecting and counting particles in medical image sets and particularly for *Two-Dimensional Gel electrophoresis* images has so many complexities. The ideal solution should be able to detect spots, aligns multiple 2-DE images together, quantifies *protein* spots and meanwhile do not need manual edition and supervision by user. However, many existing techniques suffer from underestimating or overestimating the particles' areas due to existence of background noise or non-uniformity of particles distribution. For instance, some early approaches like Olson and Miller[43] or Appel et. al[44] for spot detection and segmentation relies on using *Laplacian of Gaussian(LoG)* techniques which is too sensitive to noise and needs pre-filtering of those noises which itself causes elimination of weakest parts of the proteins. Moreover, clinical image acquisition conditions like *photographic staining* methods, degree of particle separation and so on will affect the segmentation and counting results dramatically. Because of these limitations, today the most efficient way for segmentation and counting such particles is using *Watershed Transform(WST)*. Shahbazkia et. al[45] proposed an approach towards the problem using WST which is explained in this section.

#### 2.2.6.1 Segmentation and Quantification of Two-Dimensional Gel Electrophoresis Images using WST

The proposed solution uses two images for the purpose of segmentation and quantification. The first image is the original image and the second one is the Gradient Magnitude image which could be obtained using *Gradient Magnitude Function*(see section A.5.1.2 ). The proposed technique works as follows:

1. Pre-filtering the images using a Gaussian Kernel($0.0 \leq S \leq 4.5$) to remove high-frequency noise.

2. **Initial watershed on the original image:** under the assumption that images do not suffer from saturation, limitations of the scanner or staining methods, then the output of this watershed will be a set of basins which each of them contains maximum one cell(some basins do not include any spots and will be ignored in next steps as background area).

3. **Automatic Basin Validation**: the spots which do not have any cell inside them will be rejected in this step. This elimination has two phases. First, basins that their areas are smaller than a threshold value $t$ will be ignored. Next, using watershed lines of last step, for each remained basin, a *synthetic background* is generated using *interpolation*(*bi linear*, *bi cubic* or other types of interpolation). If the difference between standard deviation of this interpolated areas with their corresponding

basin's standard deviation is more than a threshold $t_2$, then that basin contains a valid protein spot, otherwise it will be rejected. The less the value of $t_2$, the more the valid protein spots.

4. **Placing Spot Center Markers on topological surface of gradient magnitude image**: for each valid spot identified in last step, a marker is put in its called centroid pixel. This centroid pixel could be minima of the component or the better choice could be the pixels which are central to the spot and are not neighbor to plateus or watershed lines. The only limitation is that markers should not be on or outside of the borders(the watershed lines to be found).

5. **Final Watershed on Gradient Magnitude image**: the flooding procedure of this watershed will be started from Marked centroid pixels of the spots and watershed lines of the WST of step 2 and the edges of the spots in this situation will be detected when water coming from different sides meet each other. The only point is that Gradient Magnitude image will contain many other minimas except watershed lines and marker positions, which if not considered well, then flooding procedure will also start from them and cause unwelcome results. The solution as authors mentioned is to morphologically reconstruct the Gradient Magnitude image by minima imposition. In other words, using marker image, masked Gradient Magnitude image is reconstructed with new sets of minimas.

In Fig. 2.9 , you can see the result of applying this method to a sample image.

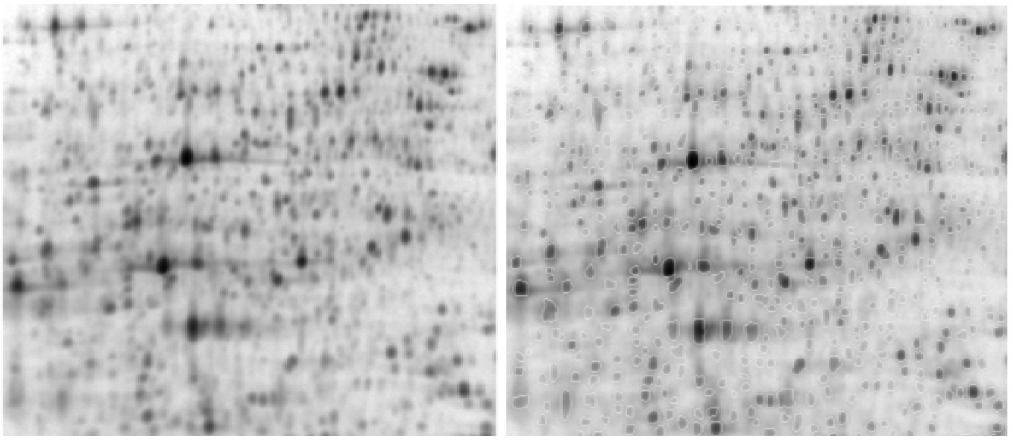


FIGURE 2.9: Segmentation based on Morphological Watersheds[45]

### 2.2.7 Fast Two Scan Based Connected Component Labeling

There are mainly four approaches for labeling the components in the literature[46].

- **Multi Scan Algorithms**: scans images in both forward and backward directions and updates label equivalences until no labels are changed. The number of scans depends on geometric complexity of the input image. Maximum complexity of such algorithms for input images of size $n$ by $n$ is $O(n^3)$.
- **Two Scan Algorithms**: scans images two times. In the first scan, temporary labels are assigned to all pixels and equivalent labels found for each pixel. Then in the second scan, based on equivalent label set, temporary label of each pixel is replaced by minimum equivalent label. Complexity of such algorithms usually depends on policies for finding equivalent labels and resolving them to the pixels.
- **Hybrid Algorithm**: is a combination of first two categories of algorithms. Like Multi Scan algorithms, it scans images in forward and backward direction several times and like Two Scan algorithms, it updates and resolves label equivalency set. It is faster than the previous two categories of algorithms and it also uses at most 4 scans during its run.
- **Tracing Type Algorithms**: works independently from label equivalency set and instead it traces contours of the objects usually in a recursive manner. The performance of such algorithms is highly dependent on the shape of objects and their contours and so it is not so suitable for hardware and parallel implementation.

For the purpose of this report, we explain a version of the most common and fast enough algorithm from the category of Two Scan Algorithms[46]:

The algorithm has two phases. In the first phase, labels are assigned to the foreground pixels and equivalencies between labels are found. Then in the second phase, foreground pixels are relabeled to their lowest equivalent label. The exact algorithm to perform Connected Component Labeling (CCL) is explained below.

Starting from the first pixel, located at the utmost top left part of the image, for each foreground pixel, we check its neighbors at west, north west, north and north east. If any of these pixels are foreground and already have some labels, then we pick the minimum label and put it as the label for the current pixel. Otherwise we assign a new label to the pixel. Then, we update the equivalency label set and move on to the next pixel in the image and do the same labeling and equivalency set updating for this one too. Now we have foreground labeled pixels and also labelled equivalency set. We start the second phase of the algorithm by going through foreground pixels, one by one, and relabel them with minimum label of the corresponding entry of label equivalency set. You can trace the flow of the algorithm for an example binary image in Fig. 2.10 . Equivalency label set for this example can be found in Table B.1.

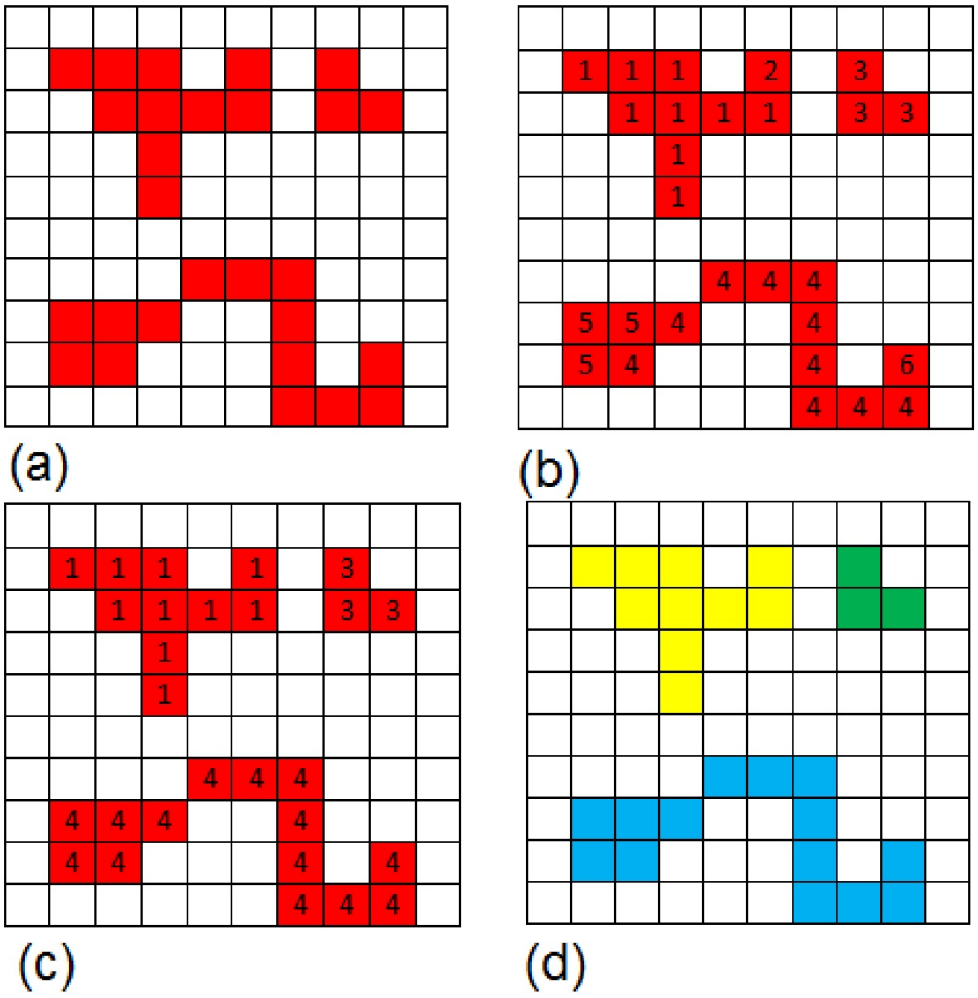



FIGURE 2.10: Example of Two scan Connected Component Labeling. a) input binary image. Red pixels are foreground and white pixels are background b)pixel pre labeling c)pixel relabeling based on equivalency label set d)image with detected components

### 2.2.8 Efficient Distance Transform algorithm

Based on the *Distance Metric* and *Region of Interest* definition for the problem, there will be different versions of Distance Transform(DT) and even when we define these concepts for the problem, there will be lots of algorithms to compute the transform. However, an efficient and basic algorithm which is fast enough and shows good experimental results is explained here. This method is based on multiple use of erosion using some structuring element SE. We perform multiple successive erosions using SE until all foreground regions of the image have been eroded away. The Distance Transform grayscale value for each pixel will then be the number of erosions that had to be performed before the pixel disappears. The nature of structuring element here defines which distance metric we choose. If we choose a 3 by 3 square SE for erosion, then distance metric is *Chessboard*. A cross shaped SE will be equivalent with choosing *City Block* distance metric

TABLE 2.2: Equaivalecy Label set for input image Fig. 2.10

| Label | Equivalency Label Set |
|---|---|
| 1 | 1,2 |
| 2 | 1,2 |
| 3 | 3 |
| 4 | 4,5,6 |
| 5 | 4,5,6 |
| 6 | 4,5,6 |

and disk shaped element gives the *Euclidean Distance* Transform(the problem in hand alternative). The final note is that distance transform is so sensitive to the noise or small changes in the images intensities and it should be used wisely when dealing with such kind of intensity distribution.

## 2.3 state-of-the-art

Some operators which are used in proposed solution is first introduced in this work and could be considered as novel methods. These operators, as will be explained in chapter 4, excel all of the existing methods for defined tasks. Furthermore, in this section, exact proofs will be given to prove efficiency of such methods.

### 2.3.1 Threshold for Decreasing PDF's

*Binary Thresholding* an image which has descending *Probability Density Function*(PDF) is sometimes a challenging task. Fitting a proper global threshold to such images is not always a trivial task. For instance, in Fig. 2.11, you can see an example of such images. As you can see in the image, there are lots of pixels with dark values and brighter pixels are in minority. The histogram of this image is also given in Fig. 2.12 where we can see the descending nature of image's PDF.

For this class of images, occasionally both local and global thresholding methods suffer from bad separation of pixels to foreground and background regions.

For global methods, first, such class of PDF's do not have multiple peaks and as a result, thresholding methods like *Otsu*, *Max Entropy*, *Intermodes* or *Minimum* which rely on the assumption that histogram is *bimodal* fail to give good thresholds. The members of other category of auto thresholding which often use some statistics of histogram like mean value of intensities as in *Mean* Auto threshold method or fraction of foreground pixels as in *Percentile* Auto threshold method also fail to suggest good automatic thresholds

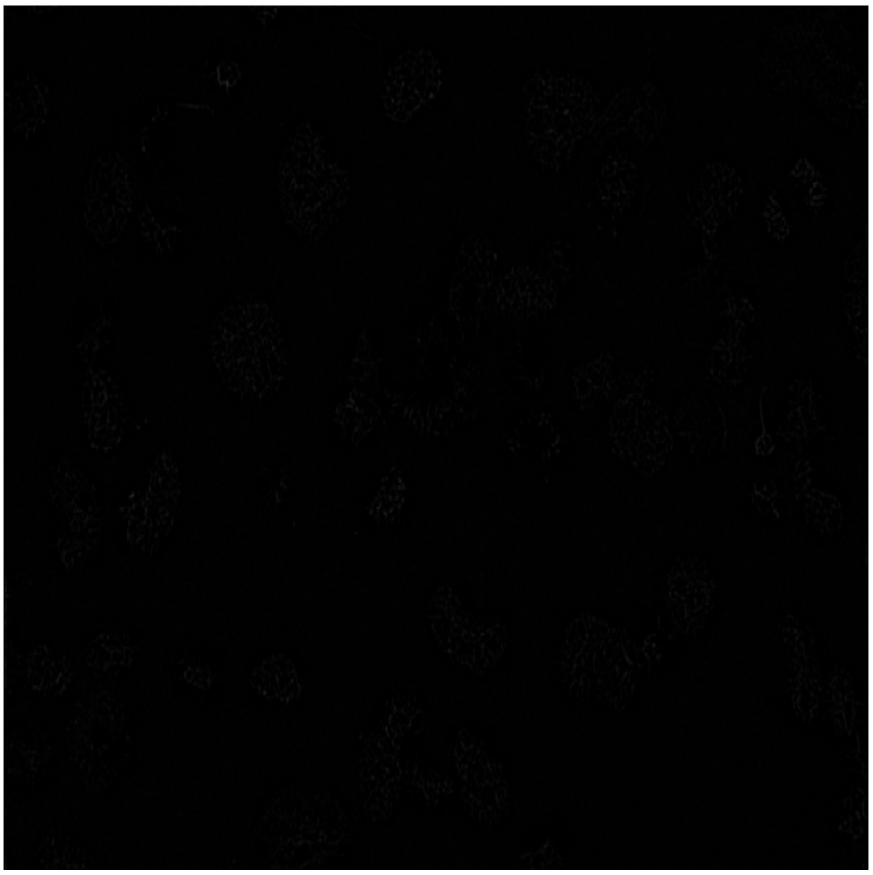


FIGURE 2.11: Image with descending PDF

since such methods are occasionally used as an initial guess for further threshold findings and are not dependable themselves. Finally some methods like *Moments* auto threshold which tries to preserve original image moments in the thresholded result usually gives result images with overestimated foreground pixels because it tries to preserve very weak parts of the original image and also noise. Therefore, global auto thresholds are not usually a proper choice to deal with thresholding images which have descending PDF.

On the other hand, for almost all local thresholding methods like *Mean*, *Median*, *Niblack* and *Sauvola* the basic assumption is that the window size should be large enough to consider both foreground and background regions. In other words, the distribution of the pixel intensities inside the window should not be so unbalanced. However, for images which have decreasing PDF's often this does not occur for foreground candidate pixels. These pixels are usually located between lots of background pixels and so having balanced windows with enough portion of foreground and background pixels is usually not possible. The worst case is when the descending slope of the PDF is so high, that the number of foreground pixels are much lower than background pixels and locating them in different window sizes will not help such algorithms to do the thresholding task properly.


FIGURE 2.12: Histogram of Fig. 2.11

So an ideal Binary Thresholding Method for such kind of images should be able to separate foreground and background pixels in such a way that:

1. Portions of the image which has acceptable density of candidates for foreground pixels be foreground in the result thresholded image.
2. Noisy and weaker candidates of foreground pixels be background in the result thresholded image.

To satisfy the above conditions, local information about neighborhood of each pixel is also necessary to judge which portion of data could be considered either as foreground or as background. So the intended threshold could also be considered as a local auto threshold method.

The proposed algorithm to execute such thresholding is explained below in the following steps:

1. Calculate the Global Mean Intensity($m$) and Global Standard Deviation($stDev$) of the original image.

2. Pixels of the original image which have grayscale intensities equal or below than $m + stDev$ are set to background in result image and are also labeled in the original image as $BG$(Background).

3. For Pixels of the original image which have grayscale intensities greater than $m + stDev$:

   - If such pixels in their 8-connected neighborhood (or any other chosen neighborhood connectivity) have at least $n$ BG pixels, then set them to background.
   - Otherwise set them to foreground.

To be more specific, we can say that the algorithm applies two filters to the original image in order to separate background pixels from foreground. The first filter detects pixels which have intensities near to the mean($0 \leq pixelIntensity \leq m + stDev$). Because of the descending nature of PDF, these pixels are located in the slice of the histogram which has the greatest density of the points. Then using second filter, we also extract the pixels which are neighbor to n background detected pixels from the first filter and set them to background. This group of pixels could be assumed to be a noisier and weaker part of the foreground which are false foreground pixels and so should be rejected. We can conclude that using first and second filters, satisfies the first and second goal of ideal thresholding method, respectively.

What should be pinpointed is the fact that we can use different neighborhood connectivity methods such as 4-connectivity, 6-connectivity or 8-connectivity. Once 8-connectivity is more accurate and also gives the algorithm more comprehensive information about neighborhood pixels, then, the choice of $n$ is important. With increasing $n$ value, we actually decrease the pressure of the second filter on candidate foreground pixels and as a result, we will have more foreground pixels. Also the choice of $n$ value depends on what type of neighborhood connectivity we choose. The result of an application of the proposed thresholding algorithm to a sample image is given in Fig. 2.13.

### 2.3.2 Gradient Based Threshold

Sudden changes in intensity or color of pixels in a given direction is a very important feature of images, usually called *edges*. Using *Edge Detection* algorithms, one can easily extract these portions. We can claim that output images of edge detectors contain edge pixels with different strength based on the magnitude of intensity or color change in images. Therefore, a *Weak Edge* pixel can be defined as a pixel with an insignificant change and similarly, *Strong Edge* pixels are those pixels with considerable changes. Now, if we want to decide which part of the result image contains relatively strong edges

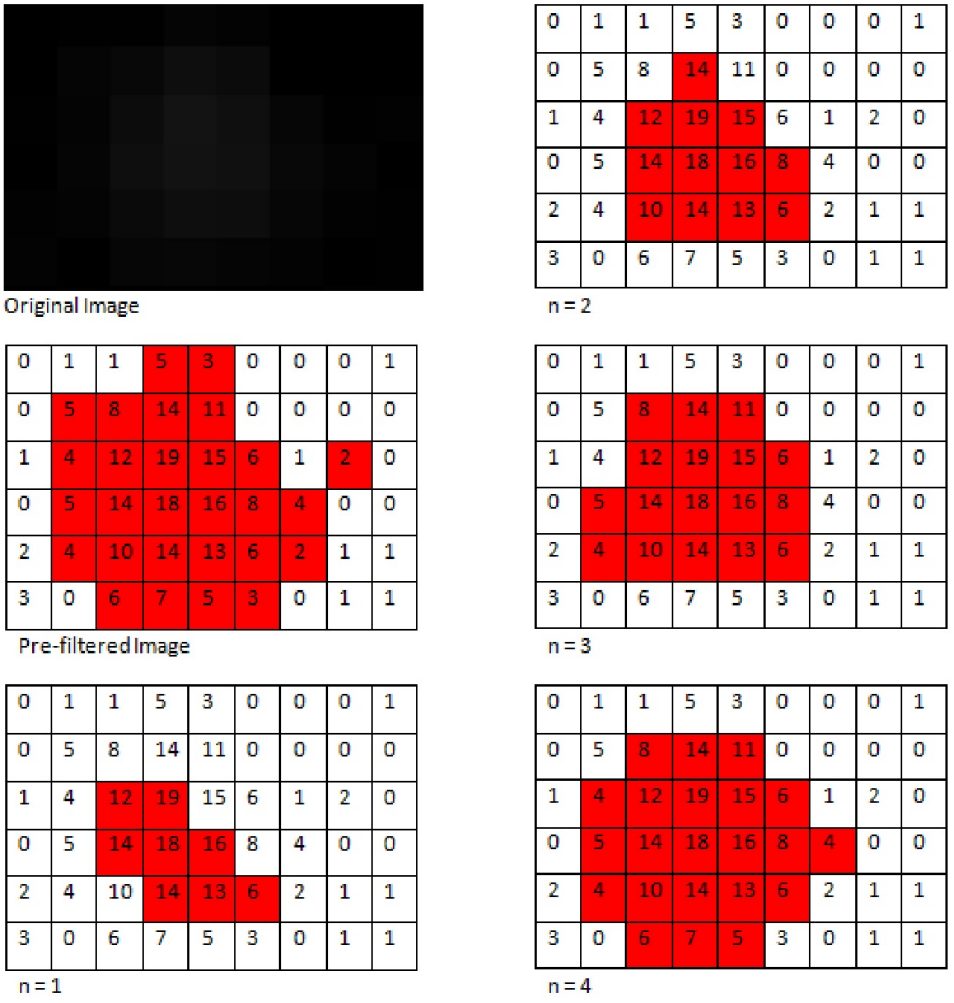



FIGURE 2.13: Application of a proposed Binary Thresholding method on a sample image. Neighborhood Connectivity is assumed to be 8-connectivity and the different values of n $n$ are tested

and which part contains weaker edges, noises or miscalculation of edge detector, then the problem must be analyzed in terms of edge strength detection. The main goal is to find a binary threshold to discriminate between detected edges. In other words, using detected edges information, the algorithm should be able to separate strong true edges from weak false edges using a binarization schema. This algorithm is extremely useful when we deal with images which have many potential edges(or simply intensity changes) with different strengths and one needs to judge which edges are valid and which are not. Then using such method, a binary image will be obtained with intended edges' data in it and post processing filters could be applied to it for further purposes.

According to the above explanations, the input for the algorithm is the image which contains intensity changes at each pixel(or simply edges). The ideal case is when input image contains an approximation of gradient magnitude of the image intensity at each

point giving us the magnitude of the largest possible increase from light to dark values. The image therefore shows how abruptly or smoothly the image changes at that point and therefore how likely it is that that part of the image represents an edge. For this purpose, depending on the nature of the application and features of input image sets, we can use operators like *Sobel*, *Prewitt*, *Roberts* and *Marr Hildreth* already explained in the Edge Detection sectionA.5.1. These different operators also have different sensitivities to random noise in images and should be considered before using them in the algorithm.

Once the resulting input image is obtained from the adequate edge detector operator, the algorithm starts it process. It should discriminate the pixels which represent strong edges from those which represent weak ones. To accomplish this, it does two things: first, it sets an *Intensity Measure* which is an initial estimation of qualified edge pixel intensity. After comparing this value to the intensity of each individual pixel, we can conclude if each pixel is a strong edge or weak one itself. Second, since intensity measure is just an initial estimation of binary threshold, the algorithm uses some local information of each pixel to guarantee maximum or near to maximum candidates of strong edge pixels and these are detected by the binarization process. For this purpose, if we find a pixel with an intensity above intensity measure in some neighborhood of each current pixel, then the pixel is also considered to be strong edge pixel.

So briefly explaining, the algorithm uses a first estimation of binary threshold as intensity measure. It then goes through the image, pixel by pixel and if the pixel's intensity itself is above the intensity measure or if the pixel has some neighbors in a defined neighborhood with intensity above intensity measure, then the algorithm marks the pixel as foreground, otherwise it is marked as background.

Finally, there are some notes and guidelines which should be considered thoroughly to have good results throughout the process:

- The adequate choice of the intensity measure is highly crucial for the algorithm. Actually, this measure is an initial guess for thresholding edge pixels in the image and using it, weak edge pixels will be filtered out. So it should not be so unrepresentative of the discrimination point. Experiments showed that Global Mean Intensity value is occasionally a good choice for this parameter since usually strong parts of the edges have intensities above this value, weaker parts have some neighbours above this value and weakest parts along with its neighbors totally lay under this value. So it is near to optimal choice to separate weak and strong parts of the edges.

- The size and type of the neighborhood used in the final step of the algorithm is also important to detect the pixels which are not filtered out by intensity measure

itself. Usually the 8-connected neighborhood is good enough for the purpose and bigger neighborhoods will overestimate edge regions; however different sizes and shapes could also be used depending of the application's needs.

So the default parameter set of the algorithm is as shown below and could be altered to fulfill application's requirements upon necessity:

**Intensity Measure**: Global Mean Intensity Value

**Neighborhood Type**: 8-connected pixel neighborhood

Finally we should notice that the proposed algorithm could be categorized under Binary Threshold methods and is suitable to threshold images with as many edges with different levels of strength they might have. Therefore the result of the method is just the location of the so called strong edges in the images and this could be used by other filters and methods as an input for further image processing tasks.

# Chapter 3

# Experimental Setup and Results

This chapter is dedicated to explain the details of the proposed algorithm for our problem. The algorithm first deals with Cells(section 3.1) then with Parasites(section 3.2) and Cytoplasms(section 3.3) and finally the observed results are converted to analytic data to obtain Infection Ratio of cells and judge effectiveness of drug candidates(section 3.4). The Chapter ends with gathering all the components of the algorithm in one pipeline in Section 3.5. For helping the reader better understanding flow of the algorithm, the algorithm applied for input image set of Fig. 3.1 and Fig. 3.2 and in each step results are shown.

## 3.1 Cell Pipeline

The objective of this section is to extract cells' portions and keep track of their information. The cells are dark areas in DAPI images which become bright after inversion (see Fig. 3.3). There are several difficulties to segment cells' areas. Firstly, illumination is not uniform in cells' portions and so simple thresholding methods will lead to several fragments of the cells. Secondly, there are so many overlapping cells (see Fig. 3.4) which if not treated well during the segmentation process, the algorithm will not be able to distinguish them. Finally, parasite pipeline is highly dependent on extraction of cells' areas as accurate as possible, so edge preservation is crucial during the pipeline. The algorithm which address all these challenges is explained in detail thorughout this section.

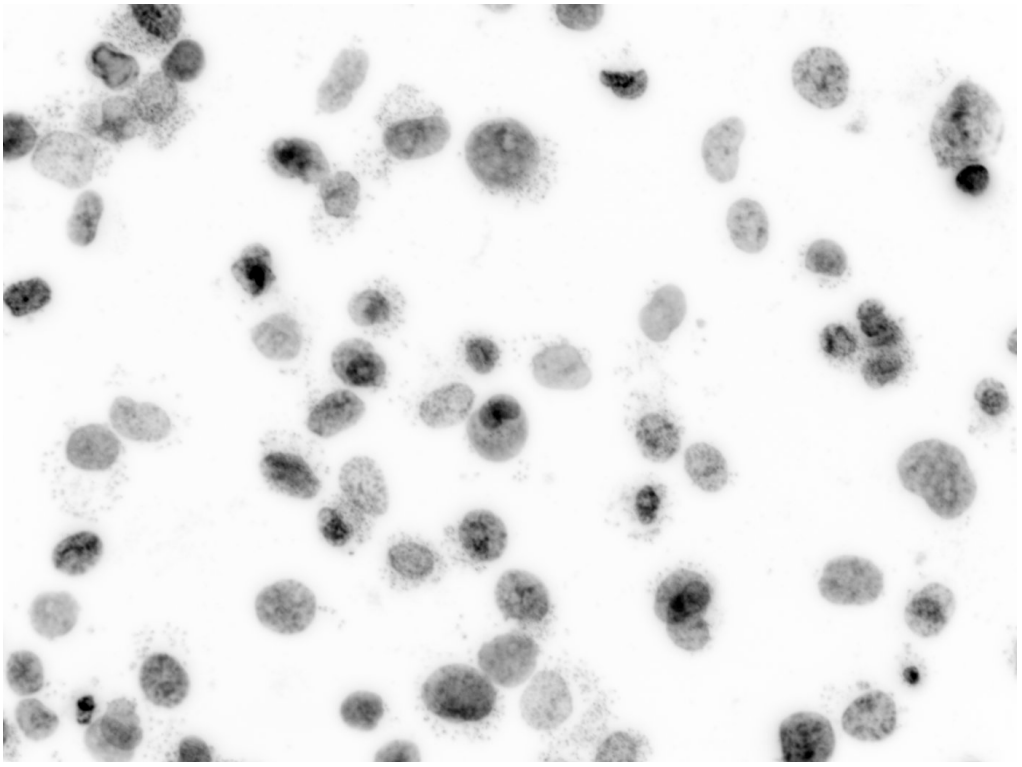


FIGURE 3.1: DAPI test image

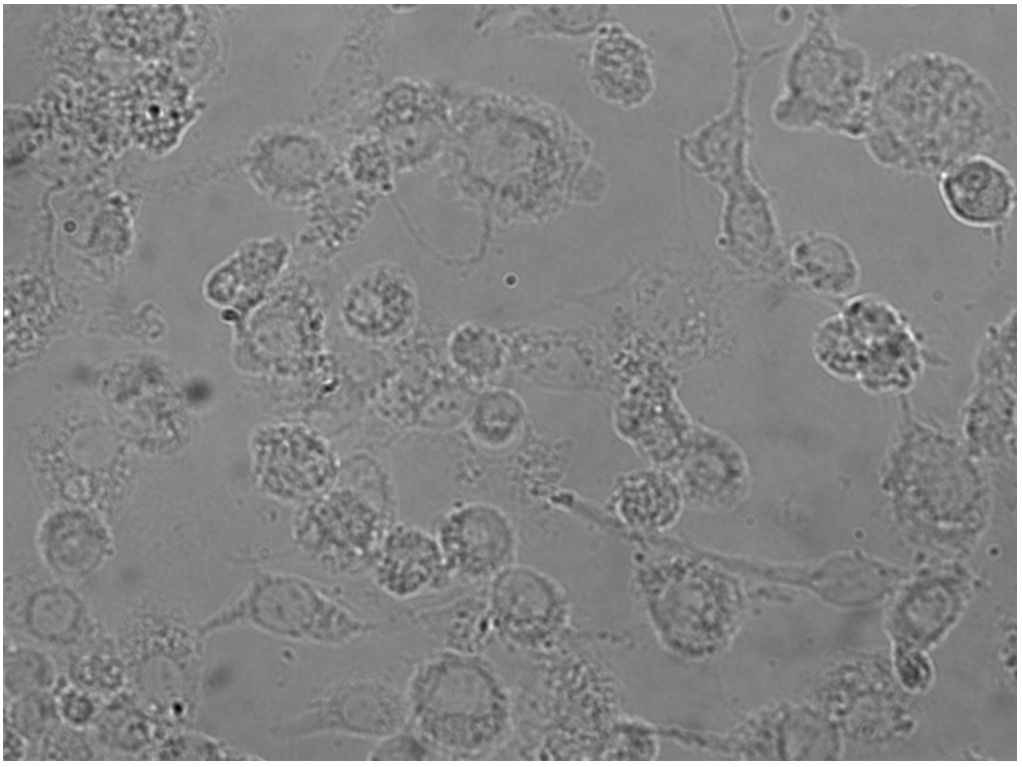


FIGURE 3.2: Phase Contrast test image

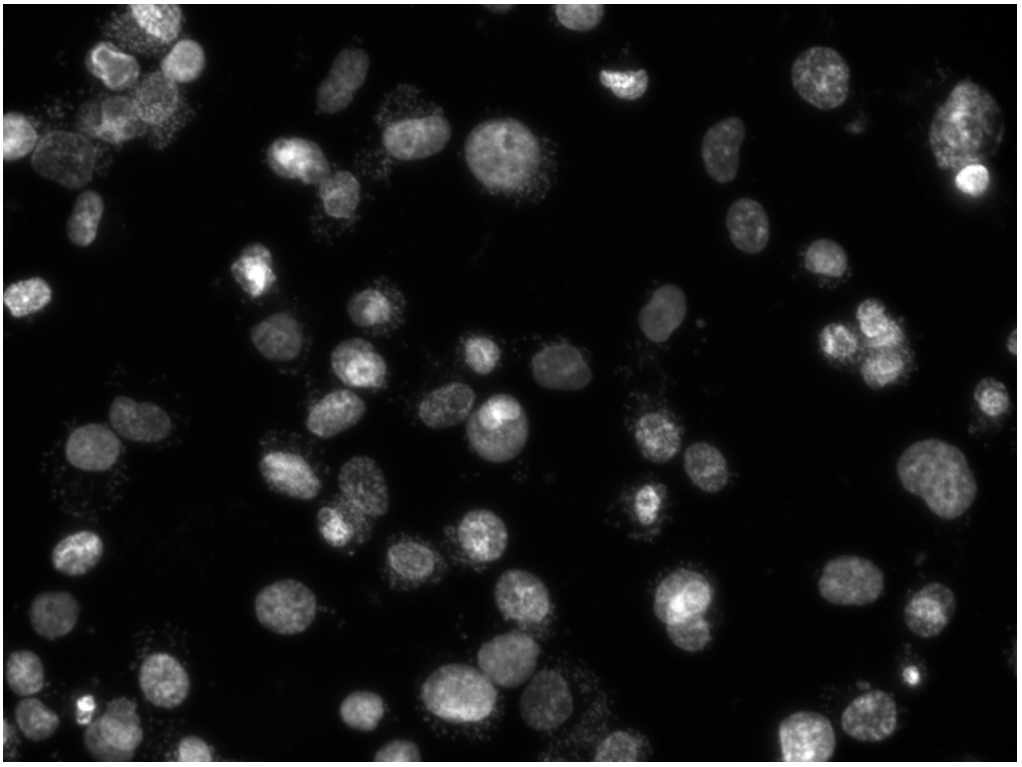


FIGURE 3.3: DAPI inverted input Image

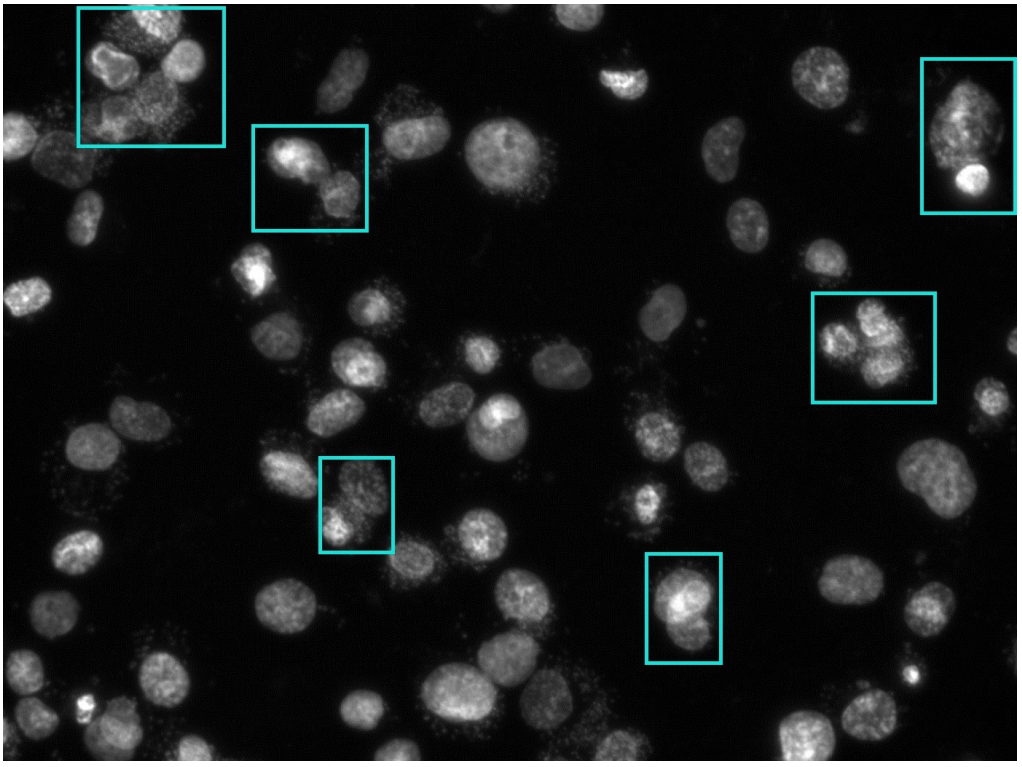


FIGURE 3.4: Overlapping Cells in input image

### 3.1.1 Smoothing

Having DAPI input images at hand, the first step is smoothing. During this step, possible random noise which could affect the final cell extraction task will be removed. The only smoothing filter used in this initial step is *Median Filter*(see section A.2.4.1 ) with relatively small kernel size. First of all, the noise model of DAPI images is assumed to be *Impulsive* therefore, Median Filter is a good choice for denoising such images[42]. Furthermore, edges (or borders) of the cells should be preserved mainly for two reasons:

1. In the following steps this data will be used to segment cells' areas and the more accurate it is the more dependable segmentation results will be.
2. Parasite extraction, as we will see in section 3.2 is highly dependent on extracting these borders with maximum accuracy.

In other words, although using more filters will help us to have smoother images for steps that follow, edge data may be interrupted which is undesirable. The result of applying median filter with kernel size 5 is represented in Fig. 3.5.

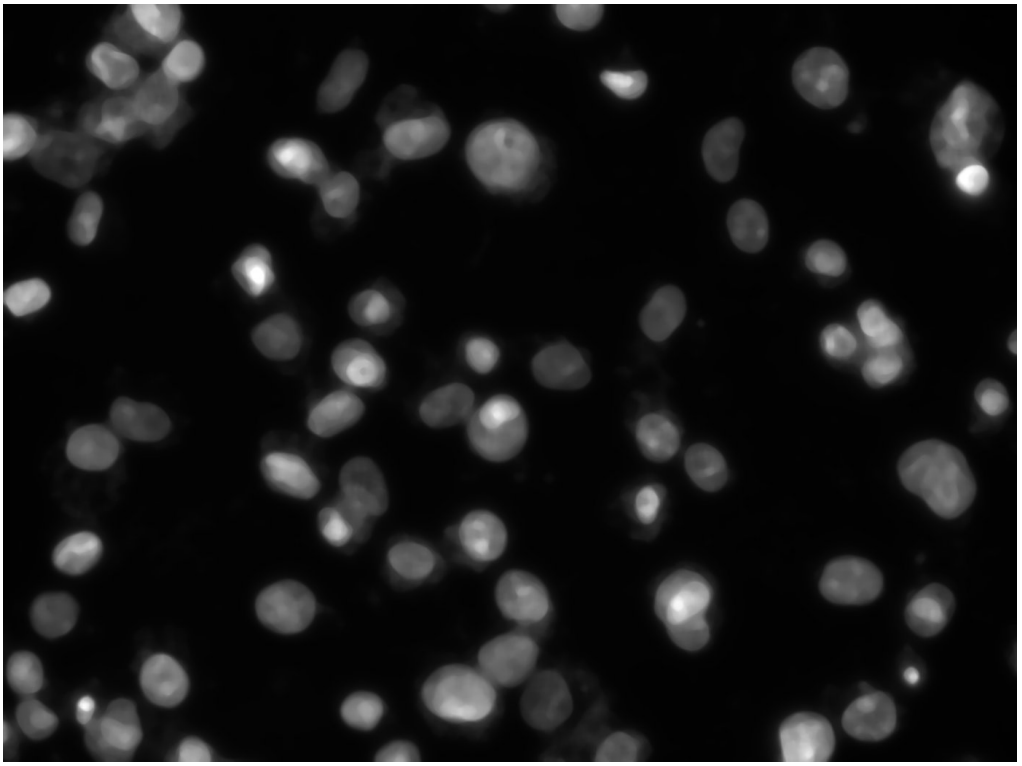


FIGURE 3.5: Cell Pipeline smoothing using Median Filter

### 3.1.2 Thresholding

The smoothed image still suffers from inhomogeneity of illumination; so using global thresholds will lead to mis-detection of cells' areas[42]. Instead, *Sauvola Mean Local Adaptive* threshold section 2.2.5 is used in this step and it considers local information of the image such as illumination variations in a good fashion. The only consideration is that the local window size of the operator should be one that includes both background and foreground pixels. For this purpose, first we find the global standard deviation of intensity values across the image. Considering the fact that bright pixels in DAPI images has very similar intensity values and at the same time very diverse ones from cell portions intensities, then we can conclude that intensity variation between bright points and cell portions will lead to have considerable standard deviation. Therefore, as a model, we suppose that value 1.5 times of global standard deviation will be a good representative for kernel window size of our thresholding method. With choosing this amount, we are almost sure that when algorithm tries to segment cell portions, it will be feed with enough pixels of both foreground and background. Window size in illustrative Fig 3.6 is 67.

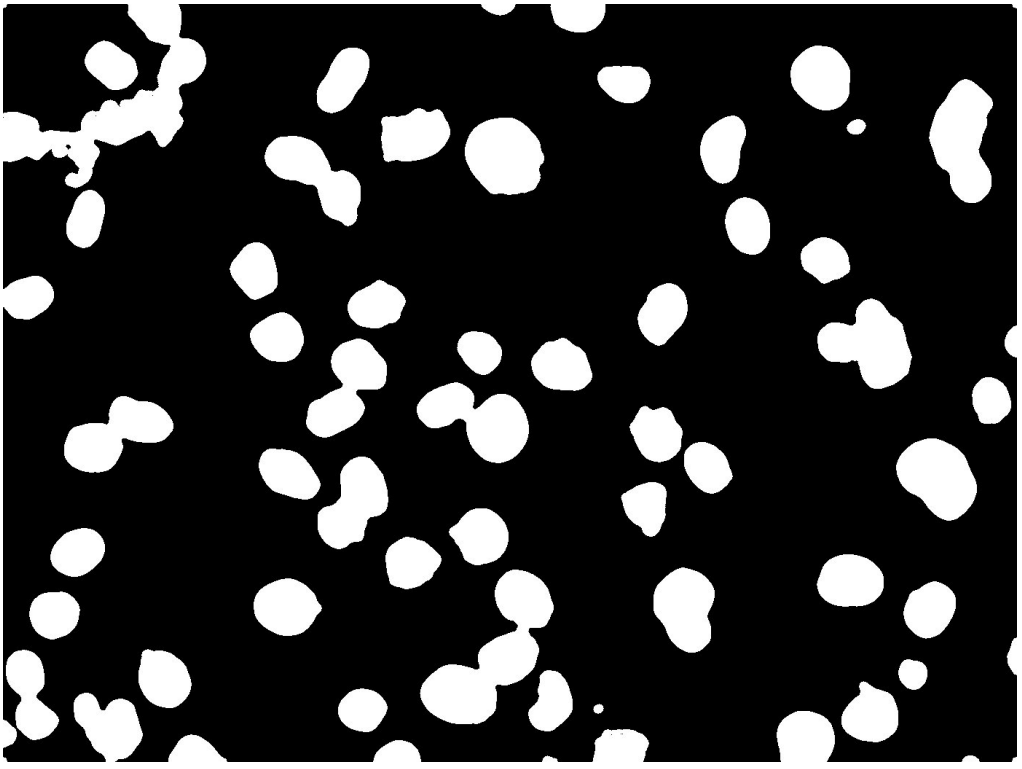


FIGURE 3.6: Cell Pipeline Thresholding using Mean Adaptive Threshold

### 3.1.3 Correction

Thresholded image sometimes include some minor errors which should be corrected before quantifying cells. There are two corrections in this step:

1. **Filling the Holes using Fill Holes Algorithm**: there are some holes between detected foreground (object) pixels(see Fig. 3.6). These holes are mainly the pixels with high unrepresentative intensity values from their neighborhood and these cannot be well thresholded using local information. Gaps can be filled using *fill holes* algorithm of section A.6.1.

2. **Eliminate Mis-detected Portions using Morphological Closing**: when dealing with high concentrations of parasites surrounding some cells or cell scraps, the local Mean Adaptive threshold will consider them as foreground pixels. Such portions are eliminated effectively using Morphological Closing of section A.4.5(see Fig. 3.8).

The results of corrections are shown in Fig. 3.7 and Fig. 3.9. Structuring Element size is 15 in the images.

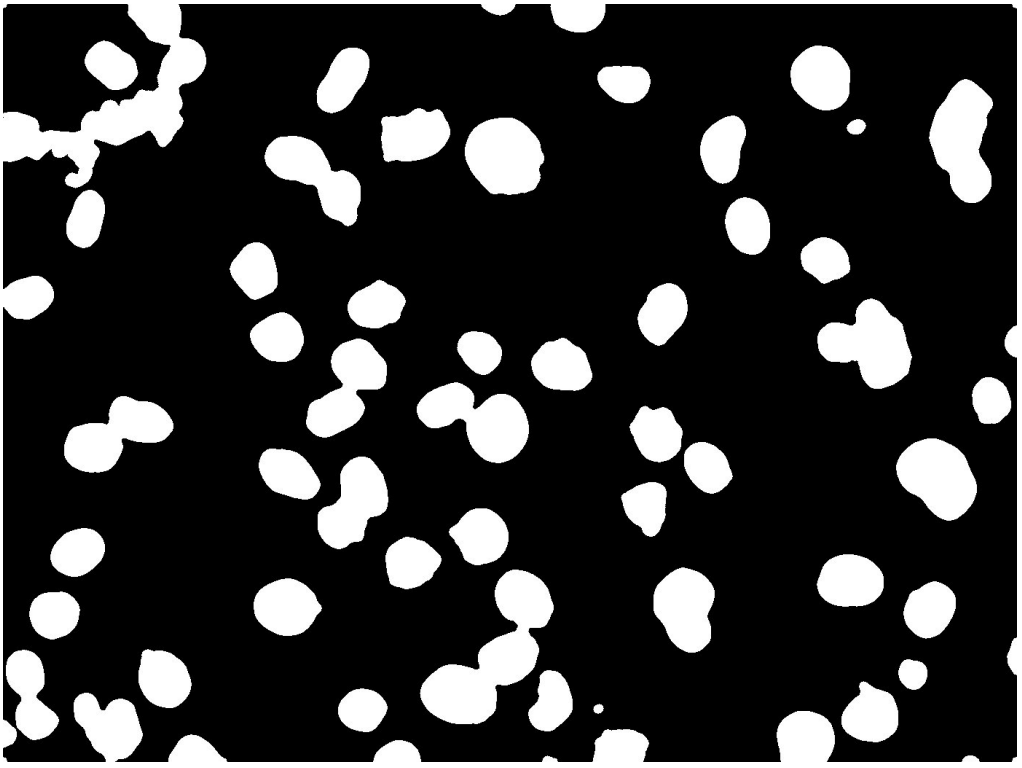


FIGURE 3.7: Cell Pipeline first Correction using Fill Holes Algorithm

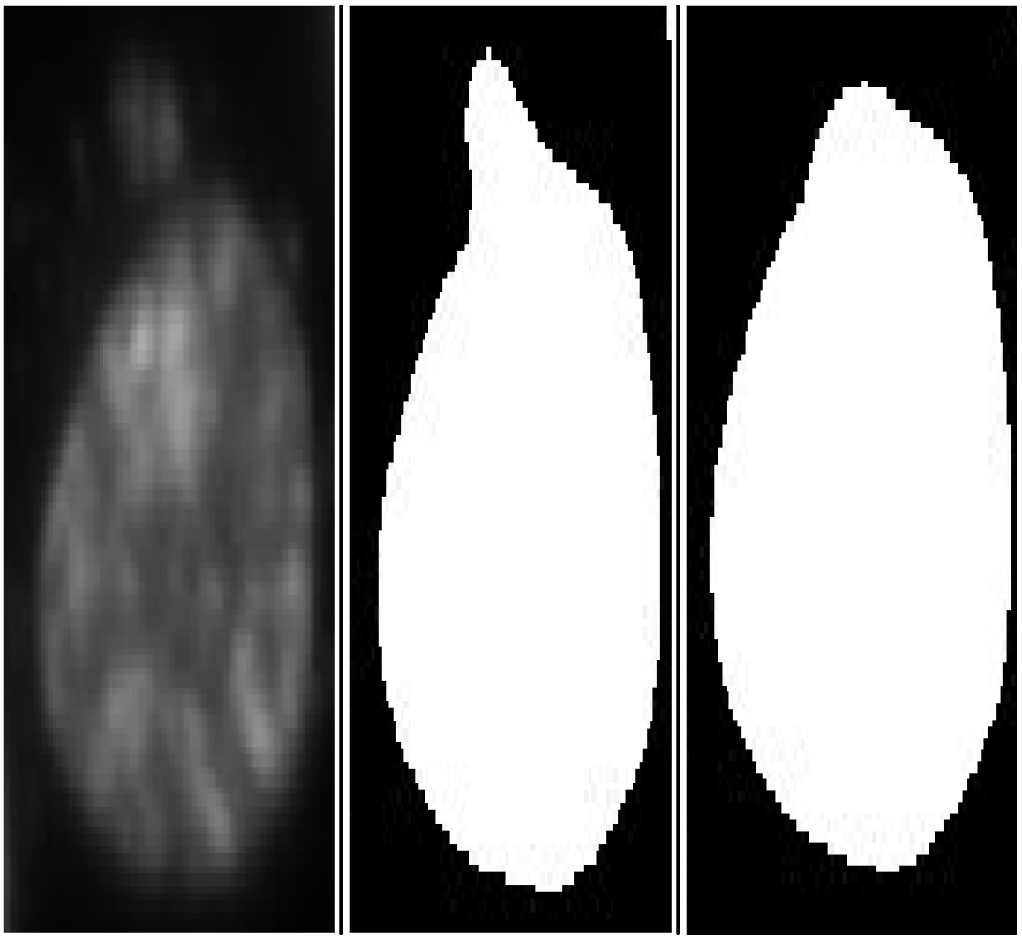


FIGURE 3.8: Cell scrap elimination using Morphological Closing

### 3.1.4 Segmentation and Quantification

Till now, the algorithm successfully detects portions of the cells. However, the only problem is overlapping cells. You can see examples of such cells in Fig. 3.4. Actually distinguishing such cells is usually impossible using Intensity based thresholds. So this step has two goals:

1. Refine segmentation results by detecting overlapped cells
2. Quantify detected cells

For this purpose, first we apply *Euclidean Distance Transform*(see section 2.2.8) to the result of the last step and then we use state-of-the-art segmentation and quantification method of Section 2.2.6. The result will be the binary image with overlapped cells detected and centers of cells marked(see Fig. 3.10 and Fig. 3.11).

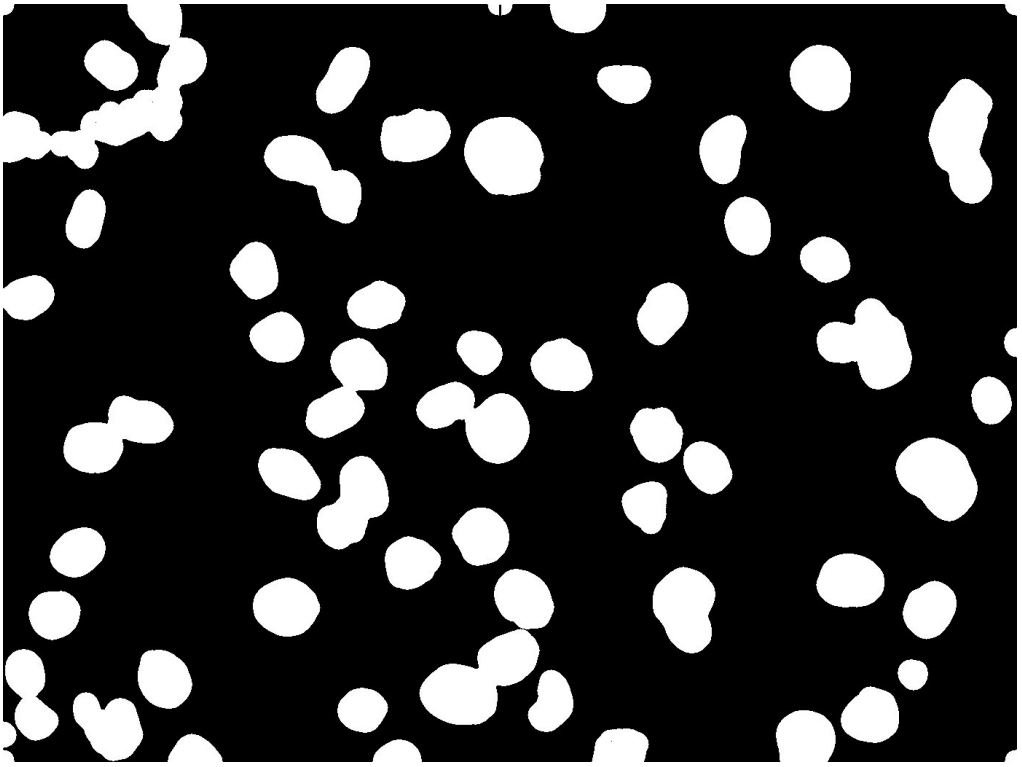


FIGURE 3.9: Cell Pipeline second correction using Morphological Closing

FIGURE 3.10: Euclidean Distance Transform of Thresholded image in Cell Pipeline

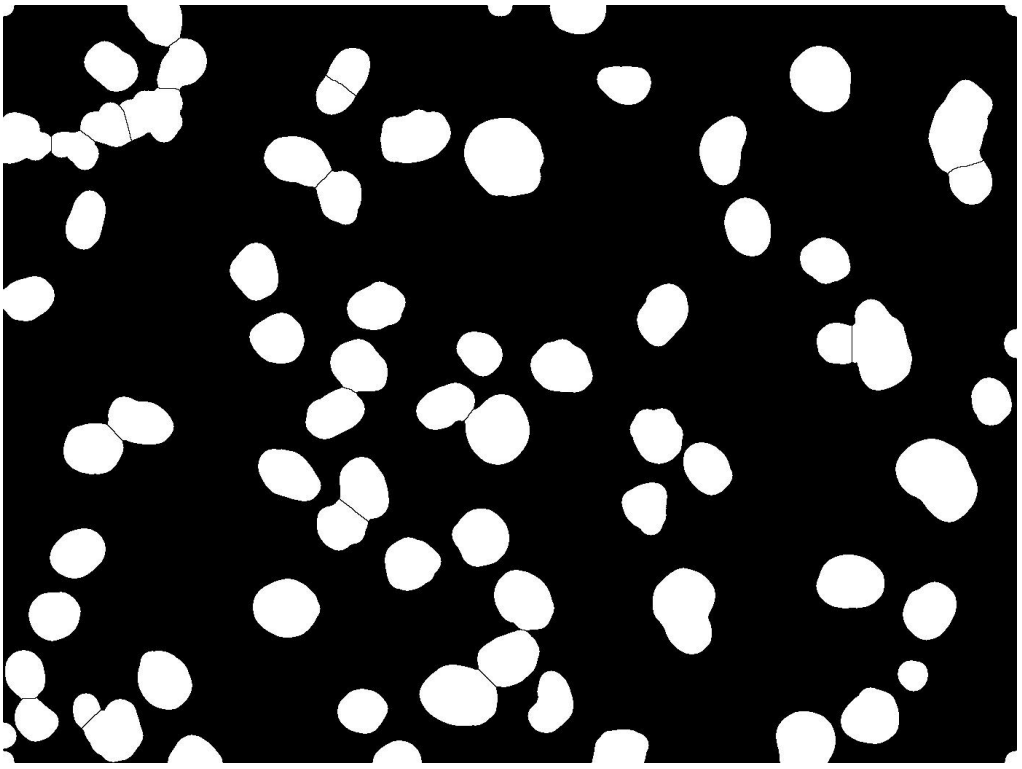


FIGURE 3.11: Cell Pipeline Segmentation and Quantification Using Morphological Watersheds

## 3.2 Parasite Pipeline

In this section, the algorithm should extract and label parasites from DAPI images. Parasites are very small regions not always visible to the human eye. Thresholding them accurately is considered to be a very challenging task because of their small size, different orientations, intensity variations and also their similarity to random noise. An algorithm to extract parasites is proposed.

### 3.2.1 Smoothing

In DAPI images, we deal with two types of small particles. The first group is simply *random noise* which has many origins, namely poor photography. The second group is parasites. They are often occupying very small areas and many of them are not so clear. In this step, we use *Unsharp Mask* filter to highlight such details(see section 2.2.3). This filter helps us to sharp these small particles' edges and let the algorithm detect them easily. However, in the step that follows, the algorithm should be able to distinguish real parasites from possible random noise in images. You can see the results of applying UM filter to input DAPI image in Fig. 3.12.

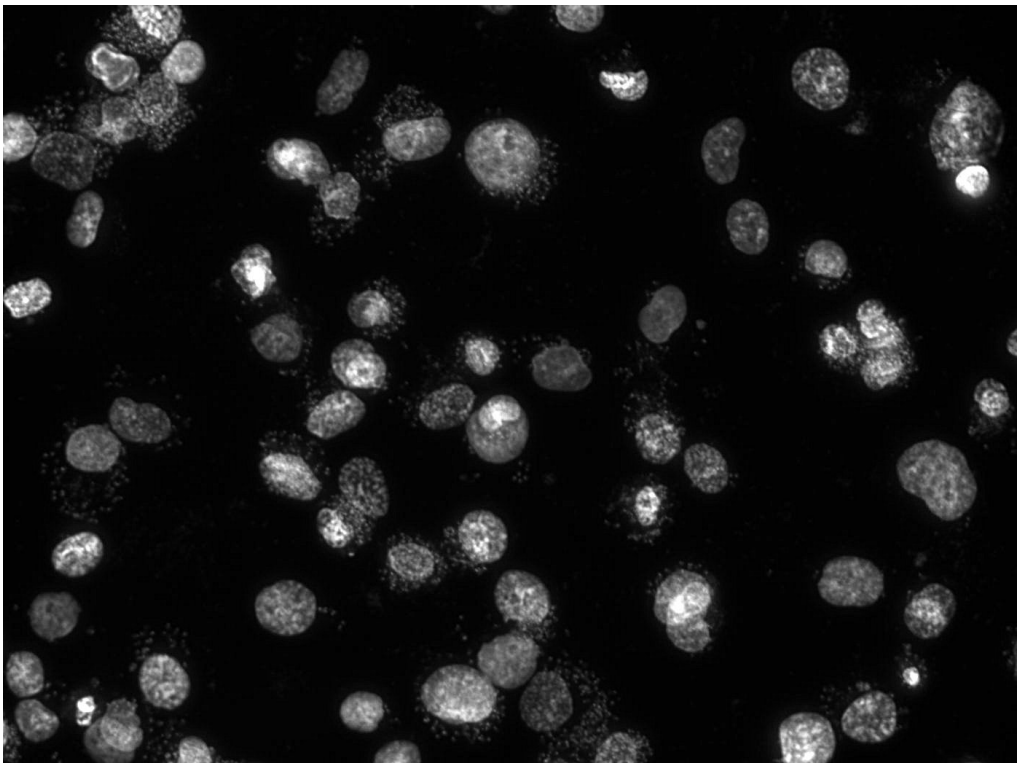


FIGURE 3.12: Parasite Pipeline Smoothing using Unsharp Mask filter

### 3.2.2 Thresholding

This step aims to threshold DAPI images in such a way that parasites become foreground and other pixels become background. Meanwhile it should be able to discriminate between real parasites and random noise. In order to do so, the algorithm runs the following steps:

1. *Black Top Hat Transform* (see section A.4.6) with small structuring element size is used to extract tiny details. The result will be a dark image with small portions of brighter points(see Fig. 3.13).

2. Result image will be subtracted(see section A.1) from the result of cell pipeline image. Here, small detected regions which belong to cells' inner areas are dropped out and remaining portions are kept(see Fig. 3.14). The result will be an image in which cell areas are black(intensity = 0) and other areas remain unchanged in terms of intensity.

3. *Threshold for decreasing PDF*'s, already explained in section 2.3.1, is applied. This threshold has quite good performance in distinguishing random noise from real parasites since it tries to find foreground pixels with acceptable density in the

neighborhood which is in harmony with parasite candidates features. The result will then be a binary image with detected parasites as foreground (see Fig. 3.15).

FIGURE 3.13: Parasite Pipeline Thresholding step1: Black Top Hat Transform

### 3.2.3 Feature Extraction and Quantification

Now that we have thresholded image with parasite candidates in it, we should extract final valid parasites and keep track of their information. For this purpose, first we apply *Connected Component Labeling*(see section 2.2.7) to extract connected components of thresholded image. Then we filter out those components whose size is not in the range of *parasite minimum* and *parasite maximum size*. The remaining components are finally the detected parasites(see Fig. 3.16).

## 3.3 Cytoplasm Pipeline

Cytoplasm regions in Phase Contrast(or DIC) images are the trace borders around cell stains. The complex task of segmenting the cytoplasm regions is due to their chaosed tracing nature. They usually have different patterns of illumination and even for some cells it is very unclear and invisible. Furthermore, the weak and fragmented parts of

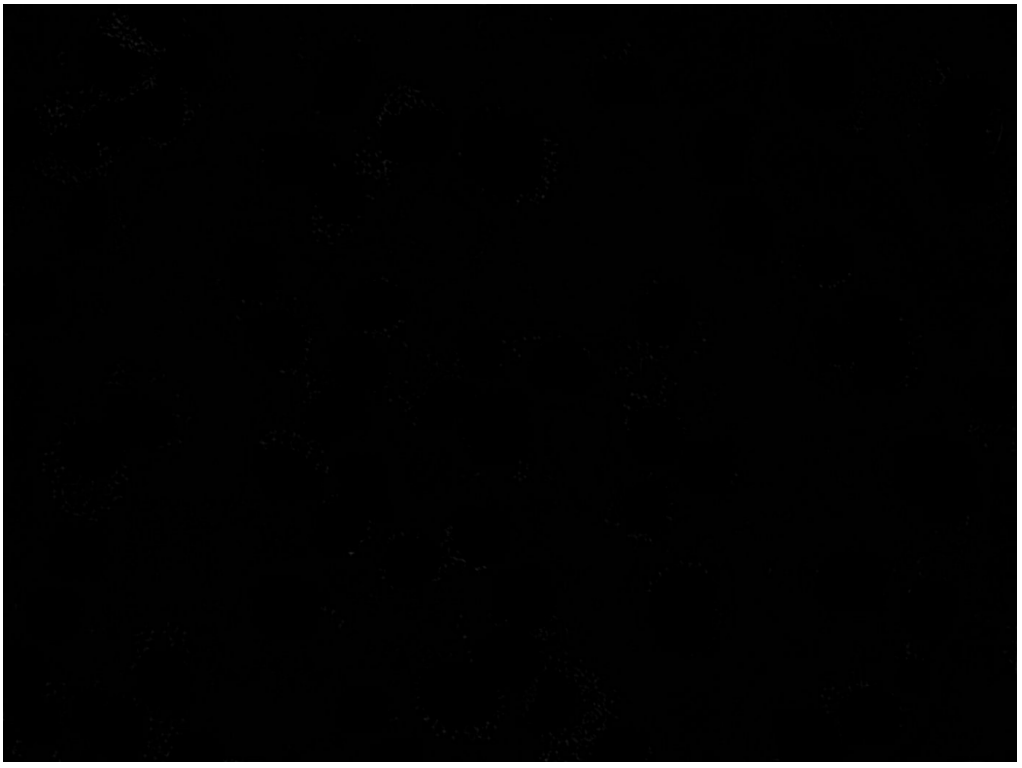


FIGURE 3.14: Parasite Pipeline Thresholding step2: subtraction of BTH transform from cell thresholded image

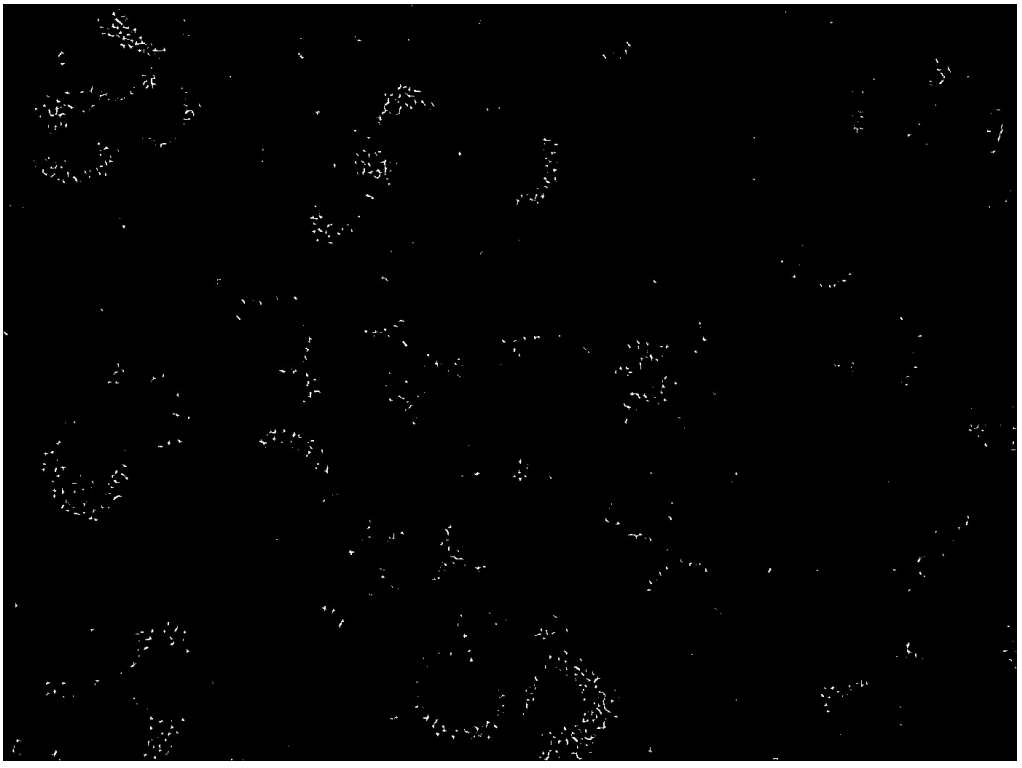


FIGURE 3.15: Parasite Pipeline Thresholding step3: Threshold for Decreasing PDF's

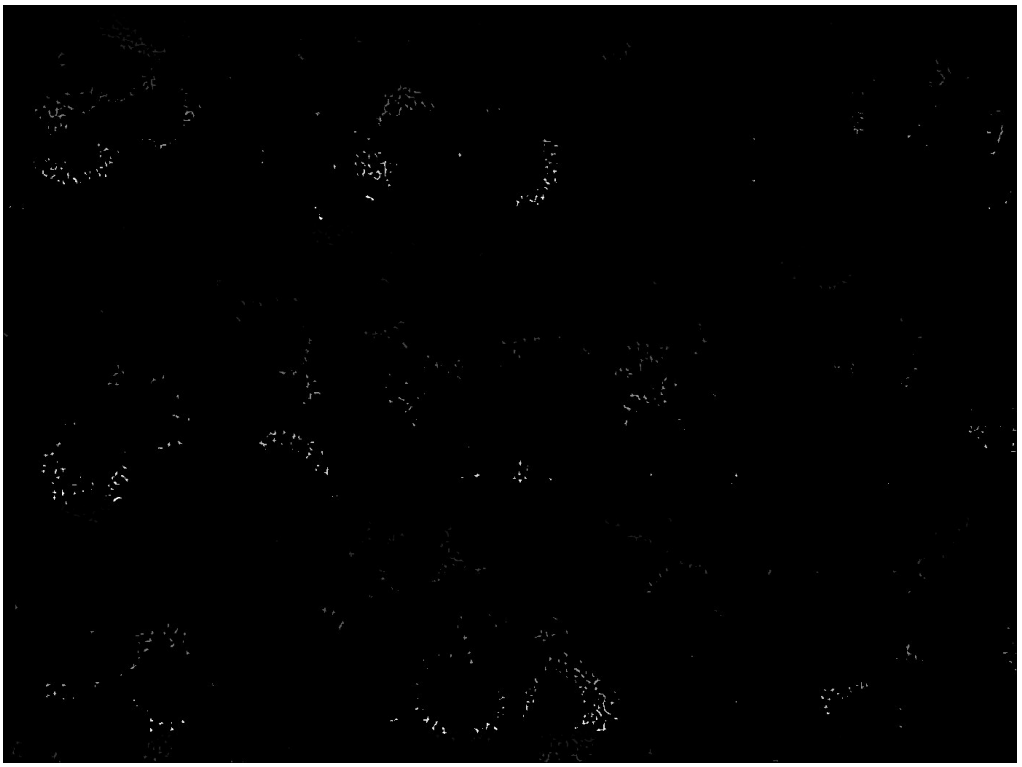


FIGURE 3.16: Parasite Quantification using Connection Component Labeling

them are very similar to noise and other extra by products which will usually not survive during smoothing phases. Now we explain the algorithm for extracting such regions:

### 3.3.1 Smoothing

In Phase Contrast(or DIC) images, cytoplasm regions have noticeable illumination variation. Furthermore, according to histogram shape of such images (see Fig. 3.17), we can assume that *Gaussian Noise* can be fitted to model degradation function efficiently. For these reasons, we use *Dynamic Mean* Filter of section 2.2.1 to remove such noises effectively while preserving edges and smooth cytoplasm regions to a degree. The result of such operation is shown in Fig. 3.18.

### 3.3.2 Thresholding

The result of this step should be a binary image in which cytoplasm region pixels become foreground and remaining pixels become background. The *Gradient Based threshold* of section 2.3.2 is used for this purpose. This threshold gives us good results since it is useful when one deals with images with lots of intensity changes( or equivalently saying, edges ) and Phase Contrast images(DIC) have this characteristic. The input image for

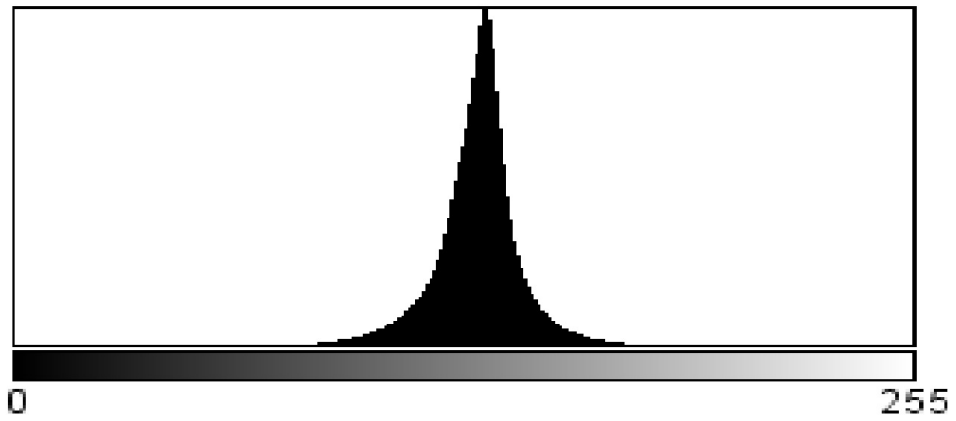



FIGURE 3.17: Histogram of the sample Phase Contrast Image

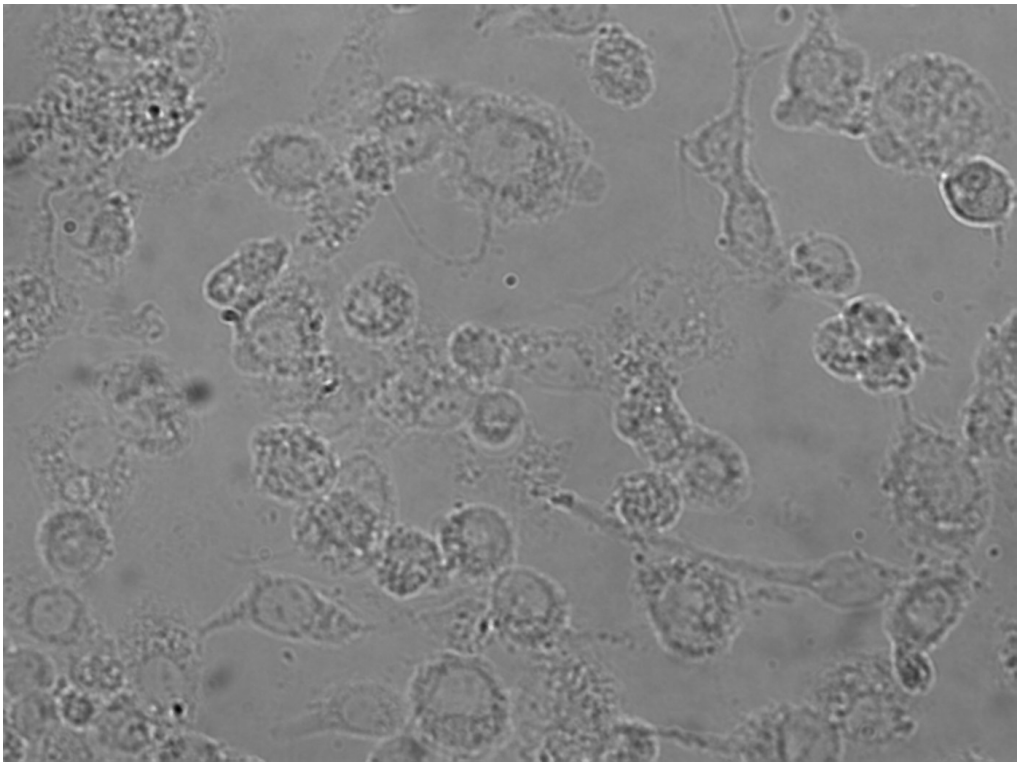


FIGURE 3.18: Cytoplasm Pipeline Smoothing using Dynamic Mean filter

this threshold is the image containing candidate edges and the experiments show that *Sobel Edge Detector*(see section A.5.1.1) which is a gradient based edge detector, not that much sensitive to the noise, is the best choice. So the thresholding process could be summarized as applying Sobel Edge Detector followed by using the Gradient Based threshold. You can see thresholding process in Fig. 3.19 and Fig. 3.20.

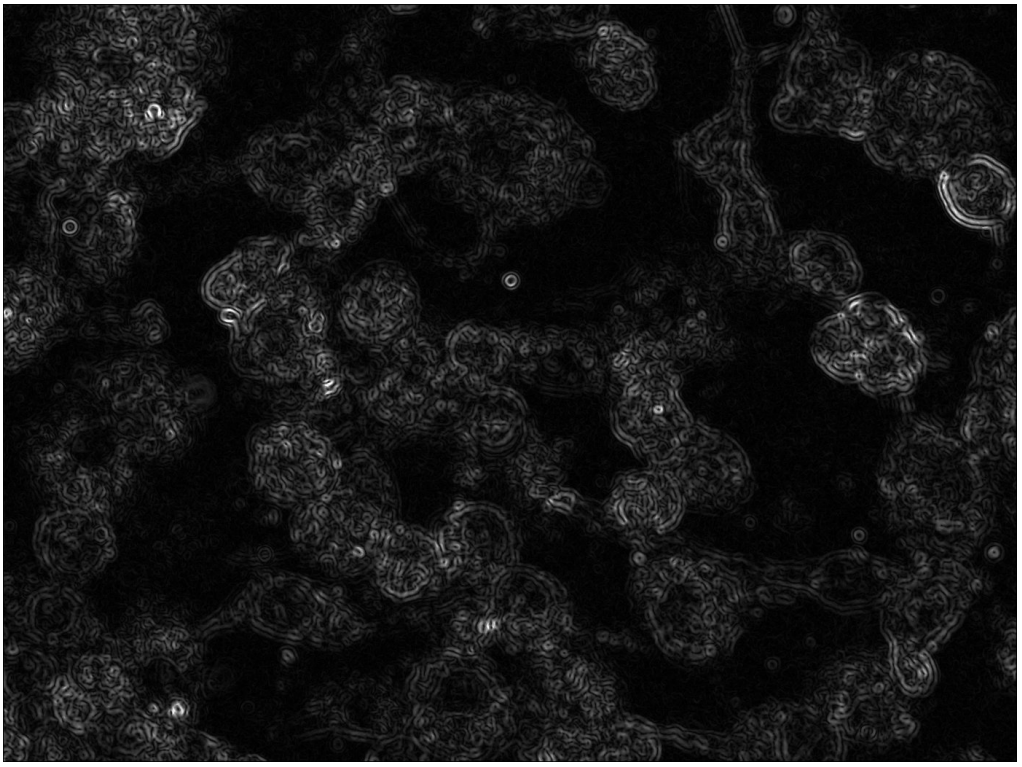


FIGURE 3.19: Edge Detection of smoothed Phase Contrast image using Sobel Edge Detector

### 3.3.3 Correction

The image result of the last step contains several non smoothed fragments of the cytoplasm regions. In fact, Gradient Based threshold extracts cytoplasm regions albeit not completely due to high level of illumination variation. Therefore, the extracted regions could be used to find final estimation of cytoplasm parts which are both convex and smooth. For this purpose, we apply two operations consecutively:

1. **Morphological Closing**(see section A.4.5): to merge and fill the structures of detected cytoplasms(see Fig. 3.21).
2. **Median Filter with relatively big kernel size**(see section 2.2.2): To smooth the Morphological Closing result, fill small holes and discard small detected portions(see Fig. 3.22).

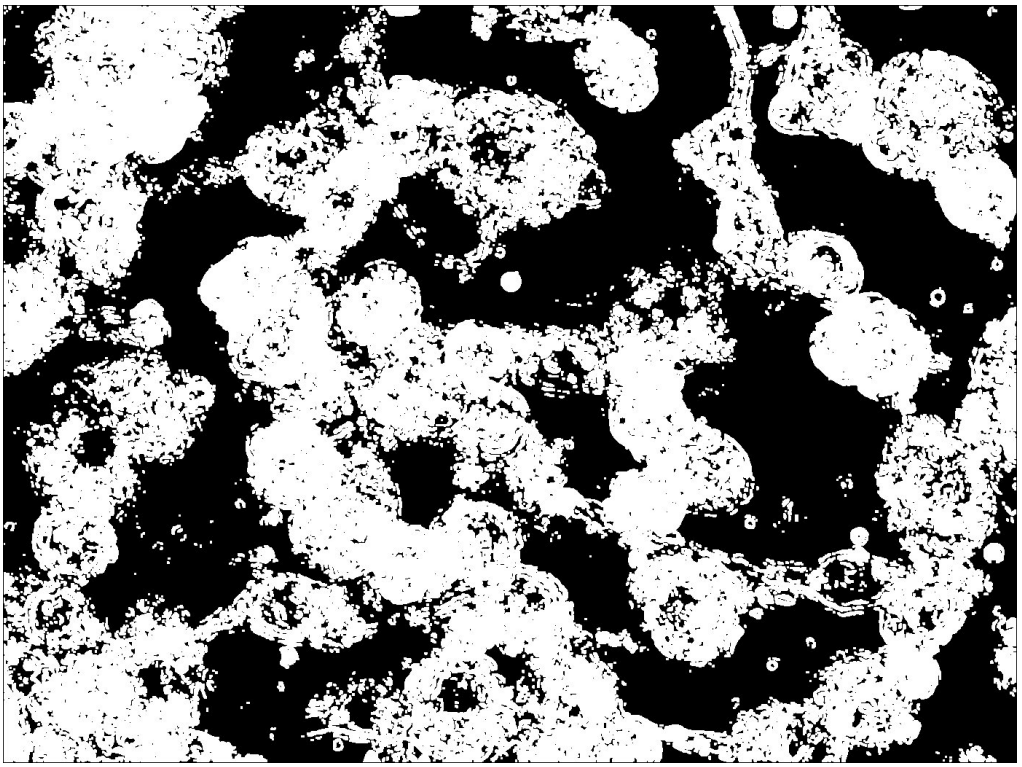


FIGURE 3.20: Cytoplasm Pipeline Thresholding using Gradient Based Threshold

## 3.4 Analytic section

Having cells, parasites and cytoplasms extracted information, the analytic section's objective is to find the *Level of Infection of Cells* in images. This parameter can generally be calculated in different ways: some are just counting the mean number of parasites per cell. However, in our experiments we use another parameter which is called the *Parasitic Index*. This parameter is notably important in drug studies to be sure that the drug is killing parasites inside the cell (by reducing the number of parasites intracellulrly), rather than a lack of an efficient infection. Parasitic Index is defined as:

$$PI = \%ofinfectedcells \times (\frac{numberofparasites}{numberofcells}) \tag{3.1}$$

To calculate PI, we need to extract this set of information from our results:

- Number of Cells
- Number of Parasites
- Number of Infected Cells

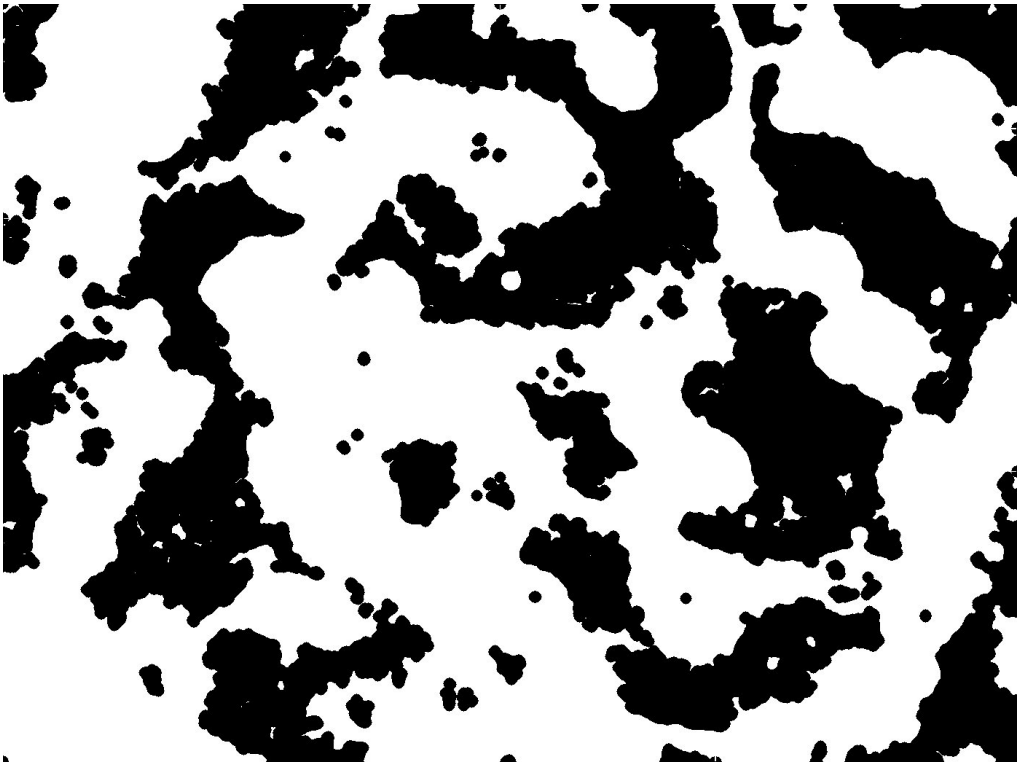


FIGURE 3.21: Cytoplasm Pipeline first Correction using Morphological Closing

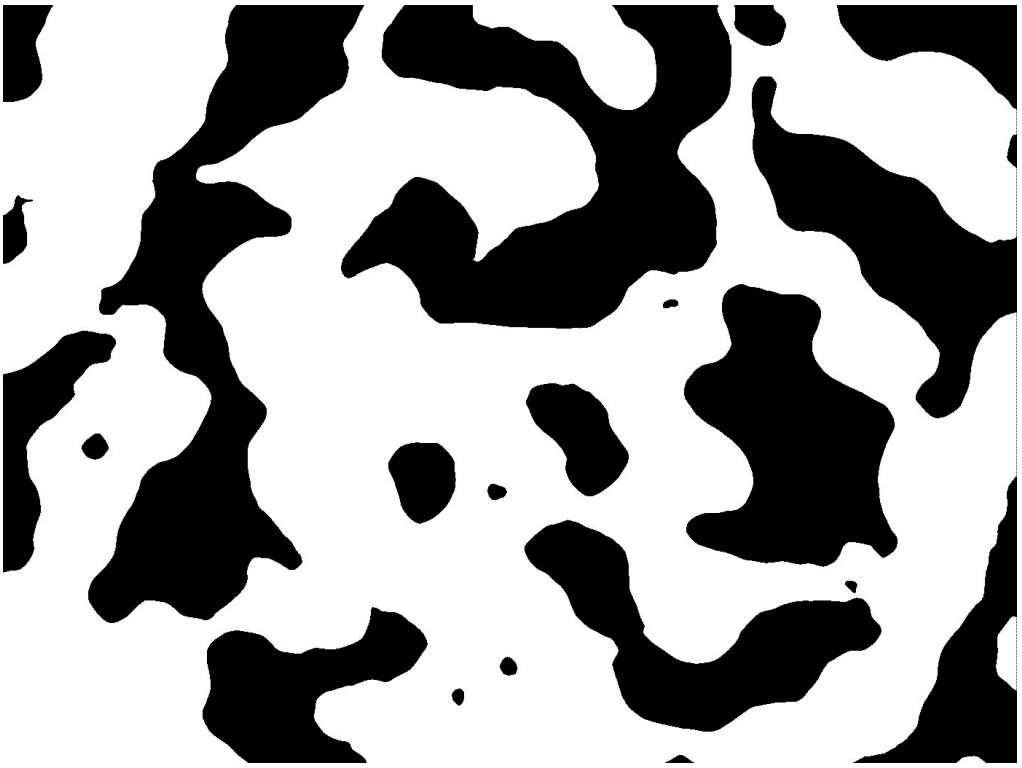


FIGURE 3.22: Cytoplasm Pipeline second Correction using Median Filter with big kernel size

- % of Infected Cells
- Mean Number of Parasites per Cell

Number of Cells and Parasites are extracted from Cell Pipeline and Parasite Pipeline results, respectively. However, in order to extract remaining parameters, we need to identify *Infected Cells* from the results. Infected Cells by definition are the cells which have at least one *Intracellular parasite* around them. Therefore the task is to assign parasites to cells and detect the cells which have parasites located in cytoplasm detected regions. To accomplish this objective, we have the following steps:

- **Assign Parasites to Cells**: For each parasite, the related cell is identified as the cell which has minimum Euclidean Distance from that parasite.
- **Identifying Intra Cellular and Extra Cellular Parasites**: according to the information obtained from Cytoplasm pipeline, parasites are grouped to Intracellular which means they are covered in cytoplasm region and Extracellular which means they are out of cytoplasm regions.
- **Identifying Infected and Non Infected Cells**: the cells which have at least one intracellular parasite are considered infected and the other cells non infected cells.

Fig. 3.23 and Fig. 3.24 show result images, one with cells and parasites marked and the other with parasites assigned numerically to related cells. Also in Table. 3.1 and Table. 3.2 you can see the analytic reports for our test image set.

TABLE 3.1: General Analytic report for the test image set

| Image Set Name | Test |
|---|---|
| Number of Cells | 71 |
| Number of Infected Cells | 67 |
| Number of Parasites | 729 |
| Number of Intracellular Parasites | 718 |
| % of Infected Cells | 94.37 |
| Mean number of Parasites per Cell | 10.72 |
| Parasitic Index(IP) | 1011.65 |

TABLE 3.2: Cells and Parasites report file for the test image set

| **Cell No.** | **#Intracellular Parasites** | **#ExtraCellular Parasites** | **Infected** |
|---|---|---|---|
| 0 | 6 | 0 | YES |

*Continued on next page*

Table 3.2 – *Continued from previous page*

| **Cell No.** | **#Intracellular Parasites** | **#ExtraCellular Parasites** | **Infected** |
|---|---|---|---|
| 1 | 1 | 0 | YES |
| 2 | 18 | 0 | YES |
| 3 | 5 | 0 | YES |
| 4 | 32 | 0 | YES |
| 5 | 7 | 2 | YES |
| 6 | 6 | 0 | YES |
| 7 | 3 | 0 | YES |
| 8 | 2 | 1 | YES |
| 9 | 12 | 0 | YES |
| 10 | 12 | 0 | YES |
| 11 | 35 | 0 | YES |
| 12 | 12 | 0 | YES |
| 13 | 1 | 1 | YES |
| 14 | 32 | 0 | YES |
| 15 | 25 | 0 | YES |
| 16 | 2 | 0 | YES |
| 17 | 2 | 0 | YES |
| 18 | 31 | 0 | YES |
| 19 | 2 | 0 | YES |
| 20 | 0 | 0 | NO |
| 21 | 7 | 0 | YES |
| 22 | 4 | 0 | YES |
| 23 | 1 | 0 | YES |
| 24 | 11 | 1 | YES |
| 25 | 16 | 0 | YES |
| 26 | 10 | 0 | YES |
| 27 | 0 | 0 | NO |
| 28 | 6 | 0 | YES |
| 29 | 3 | 0 | YES |
| 30 | 10 | 0 | YES |
| 31 | 4 | 0 | YES |
| 32 | 7 | 0 | YES |
| 33 | 13 | 0 | YES |
| 34 | 9 | 0 | YES |
| 35 | 10 | 0 | YES |
| 36 | 16 | 0 | YES |

*Continued on next page*

Table 3.2 – *Continued from previous page*

| **Cell No.** | **#Intracellular Parasites** | **#ExtraCellular Parasites** | **Infected** |
|---|---|---|---|
| 37 | 9 | 0 | YES |
| 38 | 15 | 1 | YES |
| 39 | 31 | 0 | YES |
| 40 | 1 | 0 | YES |
| 41 | 4 | 0 | YES |
| 42 | 14 | 0 | YES |
| 43 | 0 | 0 | NO |
| 44 | 10 | 0 | YES |
| 45 | 5 | 0 | YES |
| 46 | 11 | 0 | YES |
| 47 | 11 | 0 | YES |
| 48 | 12 | 0 | YES |
| 49 | 3 | 0 | YES |
| 50 | 10 | 0 | YES |
| 51 | 9 | 0 | YES |
| 52 | 4 | 0 | YES |
| 53 | 7 | 1 | YES |
| 54 | 2 | 0 | YES |
| 55 | 12 | 0 | YES |
| 56 | 2 | 0 | YES |
| 57 | 6 | 0 | YES |
| 58 | 18 | 1 | YES |
| 59 | 22 | 0 | YES |
| 60 | 50 | 0 | YES |
| 61 | 5 | 0 | YES |
| 62 | 1 | 3 | YES |
| 63 | 9 | 0 | YES |
| 64 | 3 | 0 | YES |
| 65 | 8 | 0 | YES |
| 66 | 4 | 0 | YES |
| 67 | 4 | 0 | YES |
| 68 | 14 | 0 | YES |
| 69 | 29 | 0 | YES |
| 70 | 0 | 0 | NO |

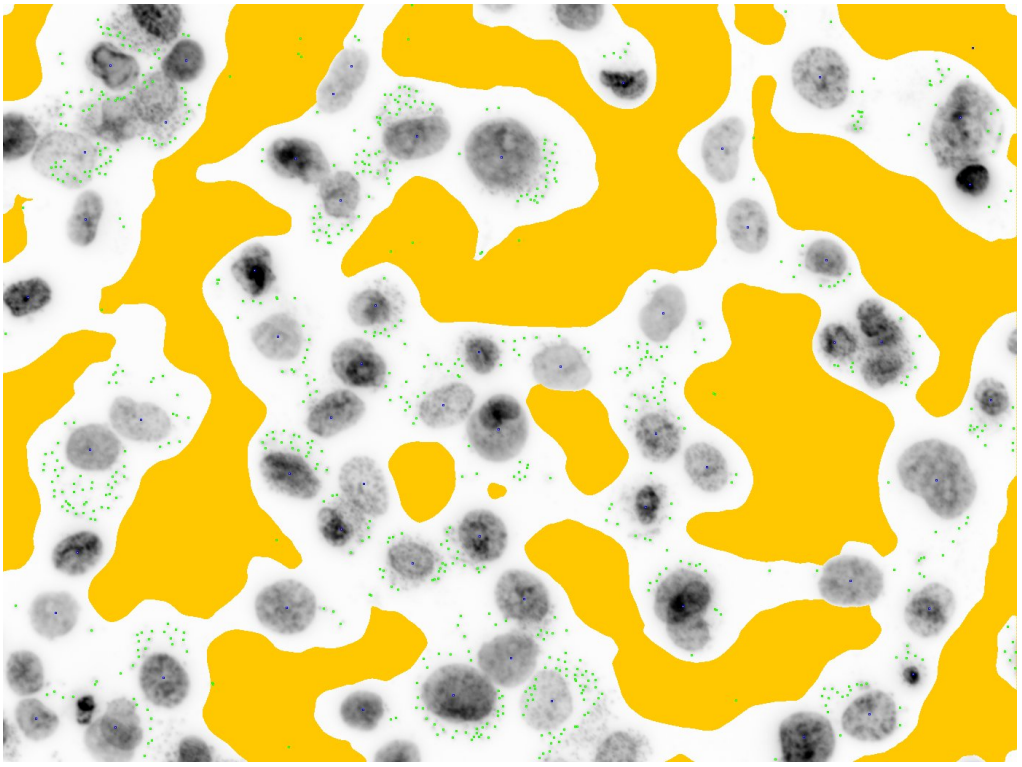


FIGURE 3.23: Result Image with marked parasites and cells

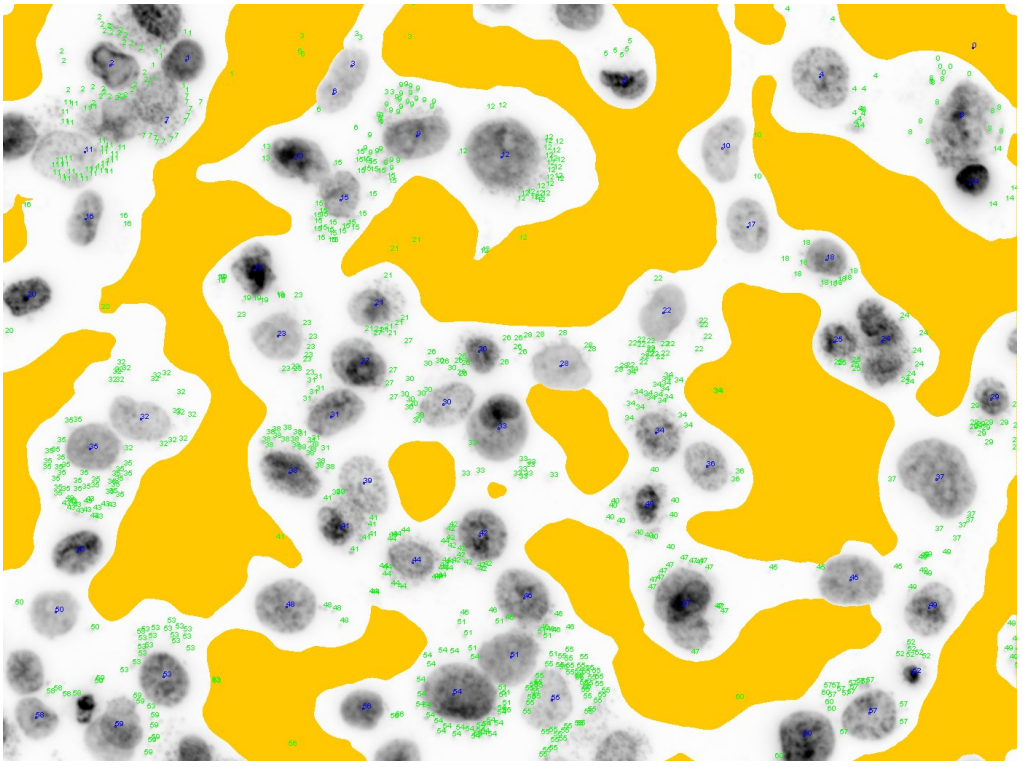


FIGURE 3.24: Result image with parasites assigned numerically to related cells

## 3.5 Putting it all together: The Algorithm

At this point we've already seen all of the individual pieces that compose our proposed algorithm, so the only thing missing is to see exactly how they would work together. Fig. 3.25 shows in a flowchart format the basic structure of the algorithm that is built for a graphical representation.

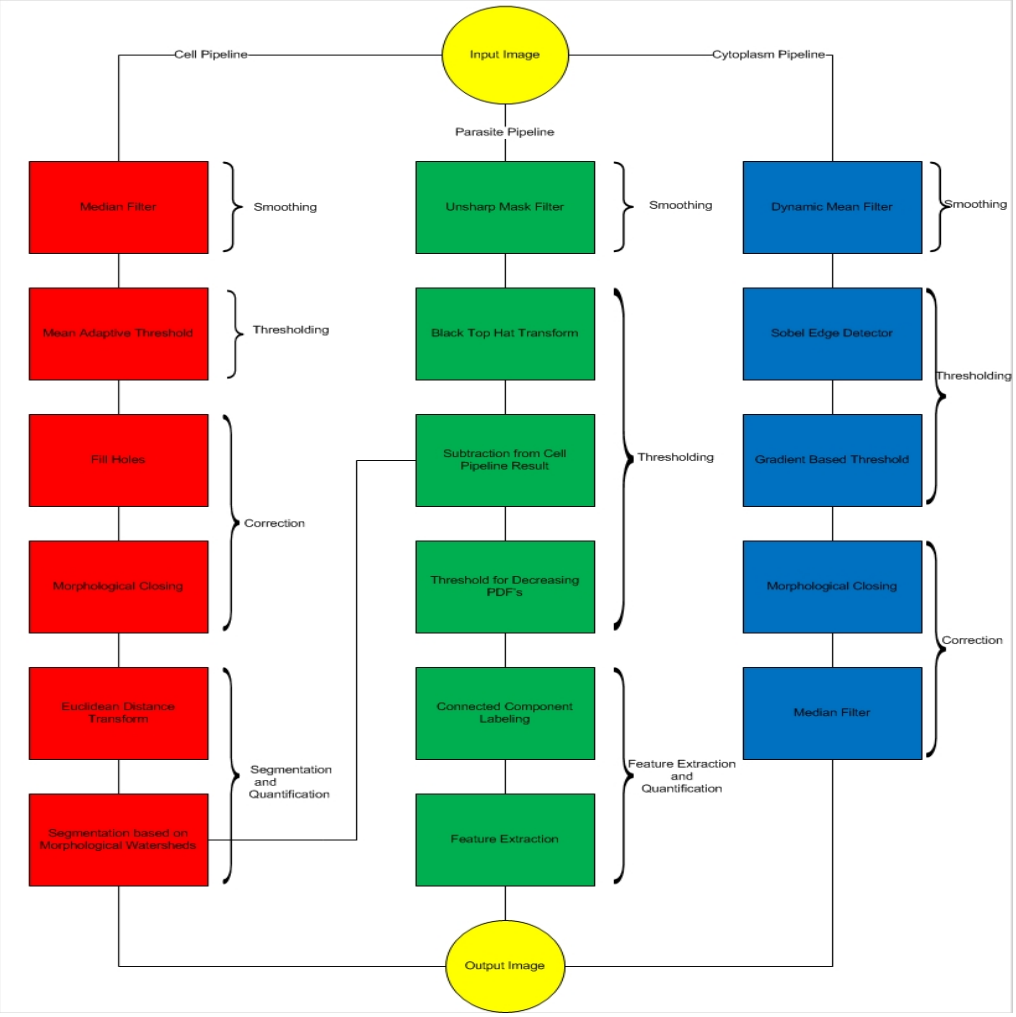



FIGURE 3.25: Flowchart of the proposed solution

# Chapter 4

# Evaluation and Comparison

This chapter tries to analyze the results of the proposed algorithm and validate it comparing the results with manual and automatic existing solutions. In general, in order to validate the results of our problem, we should know the *Ground Truth*, namely the true size, shape or other spatial features of the *components of the interest*. The best way to achieve such truth is to have an human expert observer generate manual outlines and then compare those outlines with computer generated results. Furthermore, comparing results of our method with pre existing computational methods will highlight strengths and weaknesses of our proposed solution[47]. Therefore, this chapter is organized as follows. Section 4.1 is dedicated to validating the results of the proposed solution computationally by comparing our results with the existing academic and commercial solution packages. This comparison is in terms of accuracy and complexity. Then Section 4.2 presents biological validation of results using Ground Truth provided by experts in Biology.

## 4.1 Computational Evaluation

There are no dedicated packages doing exactly the same task of finding Infection Ratio of cells using Recognition of cells, parasites and cytoplasm areas and few existing packages which do some parts of the algorithm are not available for public uses and comparison purposes. To validate results of our solution we chose to compare the results of segmentation for each pipeline with the most common and state of the art segmentation methods in literature.

Generally if we consider Image Processing techniques as a pipeline which has five steps, namely Image Acquisition, Image Pre-Processing, Image Segmentation, Image Post-Processing and Image Analyzing, then the Image Segmentation step is an appropriate

step to measure validity and accuracy of the algorithm since the results of the Segmentation step are usually simplified and more meaningful images in which features of components of the interest such as size, shape and other spatial features can be visualized and analyzed easily[48]. Furthermore, most of Image Processing techniques rely heavily on the quality of segmentation operators during their task. However, this assumption does not mean that segmentation step individually can do all the tasks; it means the results of this step should be rich enough to let following steps produce final results effectively.

Therefore, in this section, in order to validate the proposed algorithm outcomes, we compare results of our segmentation methods for cell, parasite and cytoplasm pipelines to some well known state-of-the-art segmentation methods in literature. To highlight effectiveness of the proposed solution and also give a comprehensive framework for the cross checking of the results, first *Intensity-Based Thresholds(Global and Local)* are chosen for comparison purposes and then some other state of the art Thresholding methods(which could not strictly be categorized as Intensity Based Thresholds) are compared with our method's results.

Finally, it should be noticed that although it is pretty observable for the reader to see major differences between methods' segmentation results through figures, for better clarification, we also use some qualitative measures to investigate different methods performance. 10 test images were chosen to be the basis for assigning such parameters for each method. Assigning values to these measures are done firstly with help of well-known biological patterns for under judge components and secondly with comparing method's outcomes with ground truth results generated by biologists for the under study images. Judgements will then be done with assigning Very Weak to Very Strong labels to each parameter and for each method.

**Qualitative Measures for Cell Pipeline:**

- **Overlapped Cells:** Is the method able to seperate existing Overlapped Cells?
- **Cell Scraps:** Are Cell Scraps eliminated and ignored by the method?
- **Edge Preservation:** Is the method successfull in preserving Cells' Edges' data?
- **Noise Filtering:** Are small random noise which could not be denoised and smoothed by the previous steps, filtered out by the method?
- **Connectivity:** Do method's detected foreground pixels(cell candidates) have acceptable Connectivity level?
- **Smoothness:** Are method's results Smooth enough?

**Qualitative Measures for Parasite Pipeline:**

- **Edge Preservation:** Is the method successfull in preserving Parasites' Edges' data?
- **Noise/Parasite Discrimination:** To what degree random noise and real parasites are distinguished in the method's results?

**Qualitative Measures for Cytoplasm Pipeline:**

- **Edge Preservation:** Is the method successfull in preserving Cytoplasms' Edges' data?
- **Noise Filtering:** Are small random noise which could not be denoised and smoothed by the previous steps, filtered out by the method?
- **Connectivity:** Do method's detected foreground pixels(cytoplasm trace regions) have acceptable Connectivity level?
- **Smoothness:** Are method's results Smooth enough?

### 4.1.1 Comparing Results with Global Intensity-based Thresholds Results

In this section, results of our method segmentation for cell, parasite and cytoplasm are compared to seven commonly used Global Intensity-Based thresholds, namely *Huang*[49], *Isodata*[50], *Li*[51], *MaxEntropy*[52], *MinError*[53], *Otsu*[54] and *Percentile*[55] and the results for illustrative image are represented in Fig. 4.1, Fig. 4.2 and Fig. 4.3. Furthermore, more specific comparisons based on defined measures, can be observed in Tables.

We can observe that our proposed method's results excels almost all other segmentation methods' results of this category. This is due to the fact that components of the interest (cells, parasites and cytoplasms) are not totally separated and they are not always brighter(or darker) than the background and consequently, Global Intensity-based Thresholds give poor segmentation results. In fact, global thresholding methods do not consider intensity variation across the images and they just propose a single threshold value for whole image pixels. Furthermore, some factors such as existence of saturated cell portions, tiny size of parasites and diverse illumination pattern of cytoplasm traces affect efficiency of such methods by violating the fact that distribution of intensities across images should be uniform.

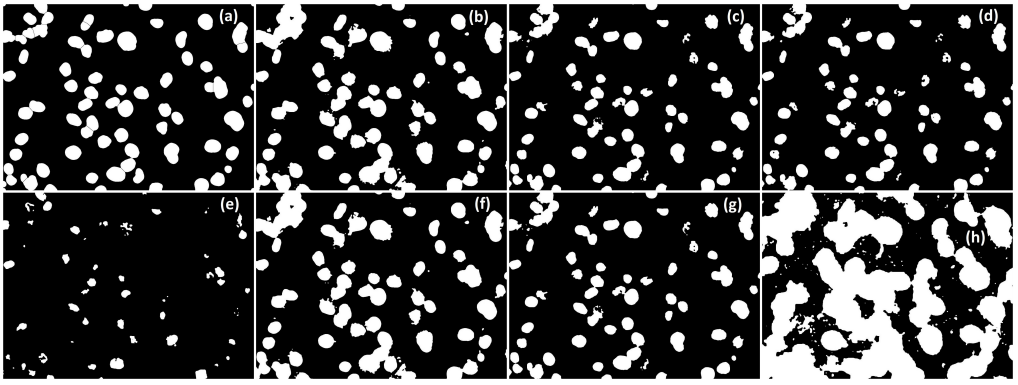



FIGURE 4.1: Comparison of the cell segmentation quality on the test image (a)between our algorithm (b) and a number of commonly used Global Intensity-Based Thresholds: Huang's (c) IsoData (d) Li (e) MaxEntropy (f) MinError (g)Otsu (h)Percentile

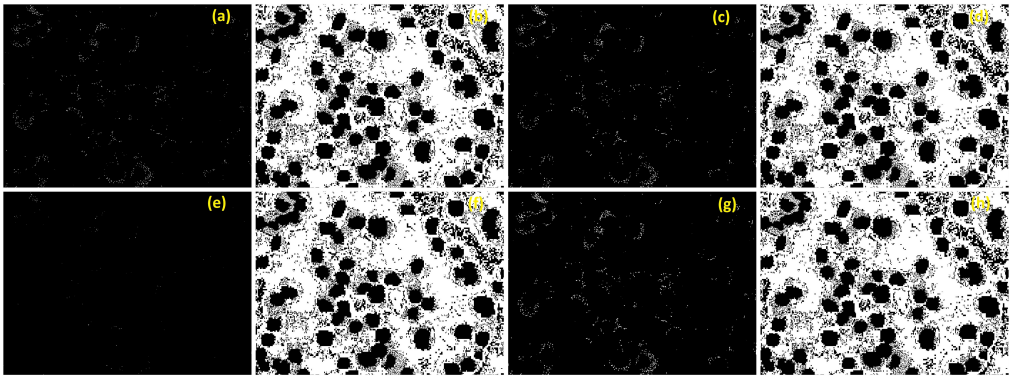



FIGURE 4.2: Comparison of the parasite segmentation quality on the test image (a)between our algorithm (b) and a number of commonly used Global Intensity-Based Thresholds: Huang's (c) IsoData (d) Li (e) MaxEntropy (f) MinError (g)Otsu (h)Percentile

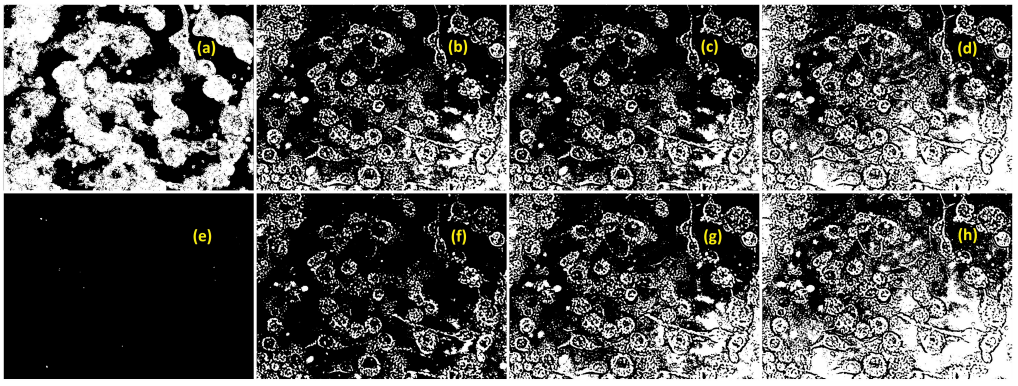



FIGURE 4.3: Comparison of the cytoplasm segmentation quality on the test image (a)between our algorithm (b) and a number of commonly used Global Intensity-Based Thresholds: Huang's (c) IsoData (d) Li (e) MaxEntropy (f) MinError (g)Otsu (h)Percentile

TABLE 4.1: Comparison between global intensity based segmentation methods for cell pipeline in terms of defined measures

| **Method** | **Overlapped Cells** | **Cell Scraps** | **Edge Preservation** | **Noise Filtering** | **Connectivity** | **Smoothness** |
|---|---|---|---|---|---|---|
| Our Method | Very High | Very High | Very High | Very High | Very High | Very High |
| Huang | Very Low | Very Low | High | High | High | Medium |
| IsoData | Low | Very Low | Medium | High | High | High |
| LI | Medium | Low | Medium | High | High | Medium |
| MaxEntropy | Very Low | Very Low | Very Low | Very Low | Very Low | Very Low |
| MinError | Very Low | Very Low | High | High | High | Medium |
| Otsu | Low | Low | Medium | High | High | Medium |
| Percentile | Very Low | Very Low | Very Low | Very Low | Low | Very Low |

TABLE 4.2: Comparison between global intensity based segmentation methods for parasite pipeline in terms of defined measures

| **Method** | **Edge Preservation** | **Noise/Parasite Discrimination** |
|---|---|---|
| Our Method | Very High | Very High |
| Huang | Very Low | Very Low |
| IsoData | High | Medium |
| LI | Very Low | Very Low |
| MaxEntropy | Very Low | Low |
| MinError | Very Low | Very Low |
| Otsu | Medium | High |
| Percentile | Very Low | Very Low |

### 4.1.2 Comparing Results with Local Intensity-based Thresholds Results

In this section, results of our method segmentation for cell, parasite and cytoplasm is compared to five commonly used Local Adaptive Intensity-Based thresholds, namely Bernsen[56], Mean[42], Median[42], Midgrey[42] and Niblack[57]. For having more fair and comprehensive comparison, window size of all of these methods are chosen in such a way that both background and foreground pixels exist in local windows. The results for illustrative image are represented in Fig. 4.4, Fig. 4.5 and Fig. 4.6. Furthermore, more specific comparisons based on defined measures, can be observed in Tables.

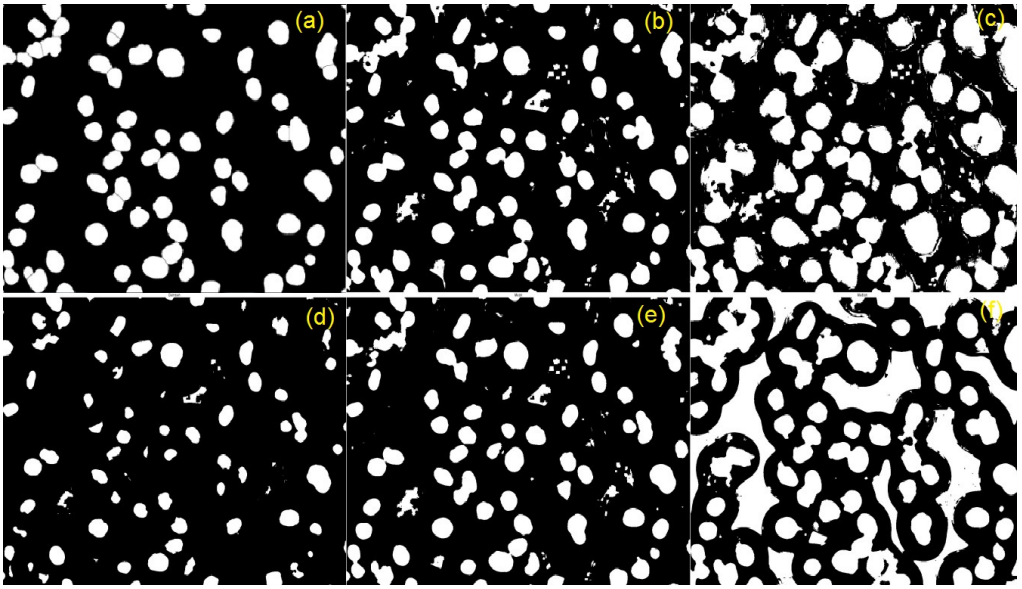



FIGURE 4.4: Comparison of the cell segmentation quality on the test image (a)between our algorithm (b) and a number of commonly used Local Adaptive Intensity-Based Thresholds: Bernsen (c)Mean (d)Median (e)Midgrey (f)Niblack

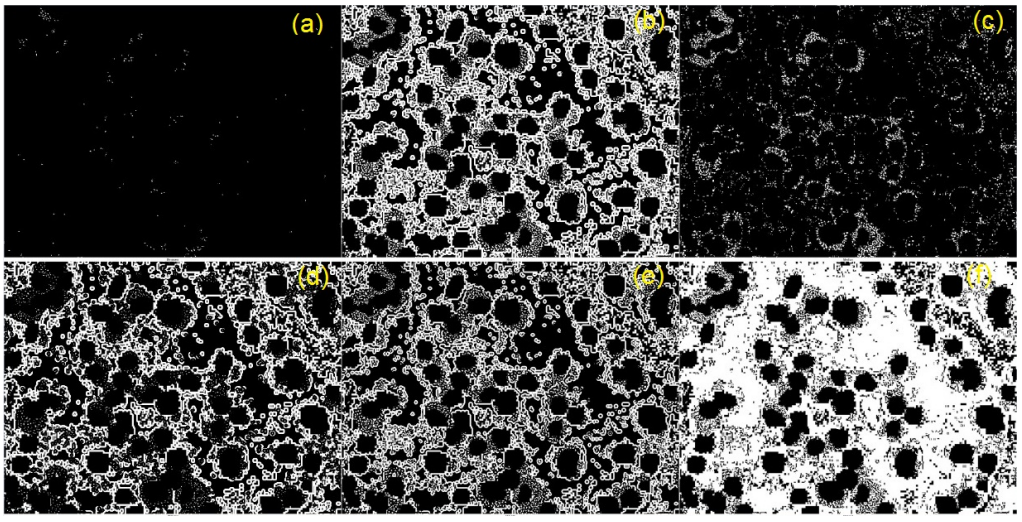



FIGURE 4.5: Comparison of the parasite segmentation quality on the test image (a)between our algorithm (b) and a number of commonly used Local Adaptive Intensity-Based Thresholds: Bernsen (c)Mean (d)Median (e)Midgrey (f)Niblack

TABLE 4.3: Comparison between global intensity based segmentation methods for cytoplasm pipeline in terms of defined measures

| **Method** | **Edge Preservation** | **Noise Filtering** | **Connectivity** | **Smoothness** |
|---|---|---|---|---|
| Our Method | Very High | Very High | Very High | Very High |
| Medium Huang | Low | Medium | Low | Very Low |
| IsoData | Low | Medium | Low | Very Low |
| LI | Low | Low | Medium | Very Low |
| MaxEntropy | Very Low | Very Low | Low | Very Low |
| MinError | Low | High | Low | Very Low |
| Otsu | Low | Medium | Low | Very Low |
| Percentile | Very Low | Low | Medium | Very Low |

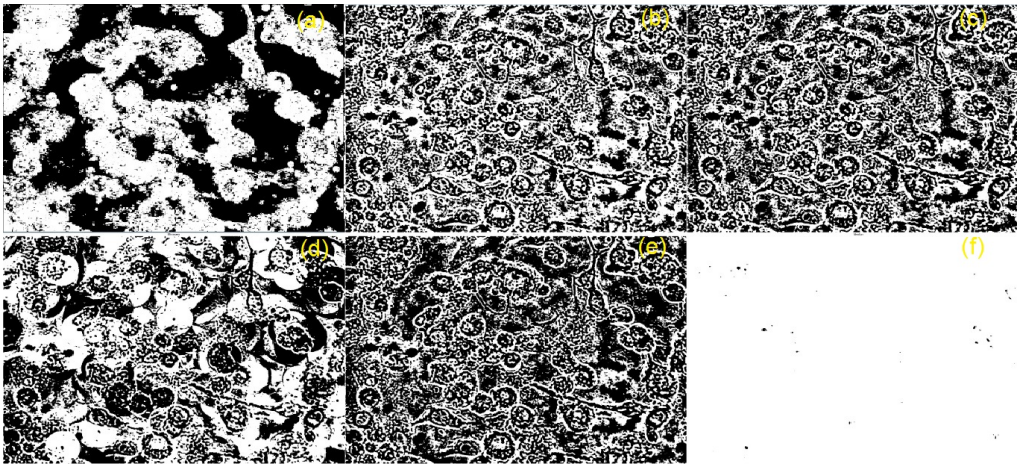



FIGURE 4.6: Comparison of the cytoplasm segmentation quality on the test image (a)between our algorithm (b) and a number of commonly used Local Adaptive Intensity-Based Thresholds: Bernsen (c)Mean (d)Median (e)Midgrey (f)Niblack

As it is obvious in the results, segmentation results of common approaches for parasite and cytoplasm pipelines are totally poor and inaccurate and cell segmentation using such methods has some problems such as non smoothed regions, noise existence and underestimating/overestimating cells regions. In fact, what the results of local and global thresholds actually show is that using intensity information of window elements itself is not all that relevant when trying to achieve good quality segmented results and for solving our problem, one should consider other local and global information of pixels as well.

TABLE 4.4: Comparison between local intensity based segmentation methods for cell pipeline in terms of defined measures

| **Method** | **Overlapped Cells** | **Cell Scraps** | **Edge Preservation** | **Noise Filtering** | **Connectivity** | **Smoothness** |
|---|---|---|---|---|---|---|
| Our Method | Very High | Very High | Very High | Very High | Very High | Very High |
| Bernsen | Low | Very Low | Low | Low | Medium | High |
| Mean | Very Low | Very Low | Very Low | High | High | Low |
| Median | Very Low | Very Low | Very Low | Very Low | High | High |
| Midgrey | Very Low | Very Low | Low | Low | High | High |
| Niblack | Very Low | Very Low | Very Low | Very Low | High | Low |

TABLE 4.5: Comparison between local intensity based segmentation methods for parasite pipeline in terms of defined measures

| **Method** | **Edge Preservation** | **Noise/Parasite Discrimination** |
|---|---|---|
| Our Method | Very High | Very High |
| Bernsen | Very Low | Very Low |
| Mean | Medium | Very Low |
| Median | Low | Very Low |
| Midgrey | Very Low | Very Low |
| Niblack | Very Low | Very Low |

### 4.1.3 Comparing Results with non Intensity Based Thresholding methods Results

In this section, results of our method Thresholding are compared to three state of the art segmentation methods. The first method is *k-means Clustering*[58] which performs pixel-based segmentation by assigning each pixel to one of the clusters. The second method is called segmentation based on *Level Sets*[59] which is based on using *Partial Differential Equations(PDE's)* and tries to find object boundaries by progressive evaluation of the differences between neighboring pixels. The last method is *Trainable Segmentation*[60]. Here, the user draws some examples of regions of interest and then a classifier is trained by the examples and segments the rest of the image. None of these methods could be exactly categorized as Intensity-Based Thresholding methods; instead they use some other features of pixels, heuristics and also one of them involves user interaction for better segmentation. So it helps us to draw better conclusion about effectiveness level of our

TABLE 4.6: Comparison between local intensity based segmentation methods for cytoplasm pipeline in terms of defined measures

| **Method** | **Edge Preservation** | **Noise Filtering** | **Connectivity** | **Smoothness** |
|---|---|---|---|---|
| Our Method | Very High | Very High | Very High | Very High |
| Bernsen | Low | Very Low | Medium | Low |
| Mean | Medium | Low | Low | Low |
| Median | High | Medium | High | Medium |
| Midgrey | Medium | Low | Low | Very Low |
| Niblack | Very Low | Very Low | Very Low | Very Low |

method. The comparison results for illustrative image are represented in Fig. 4.7, Fig. 4.8 and Fig. 4.9. Furthermore, more specific comparisons based on defined measures, can be observed in Tables.

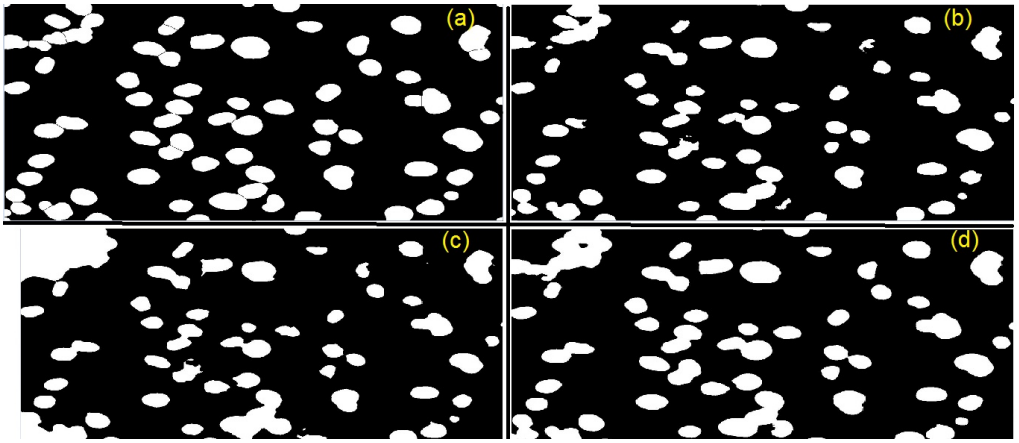



FIGURE 4.7: Comparison of the cell segmentation quality on the test image (a)between our algorithm (b) and a number of state of the art non Intensity-Based Thresholds: k-means clustering (c)Level Sets (d)Trainable

While Trainable method's results are competable with our method's one, it needs user interaction. Also, some prior knowledge about component under study would be needed in user side to use this method efficiently. k-means clustering method also suffers from illumination variation effects affecting some parts of cells', parasites' and cytoplasm pipeline failing to give good segmentation fragments. Thresholding based on Level Sets is very slow and highly dependent on the input parameter set. If train data is not chosen well, then the result will be highly unwelcome. As a result, we can conclude that our proposed method's performance also excels segmentation results of this last category.

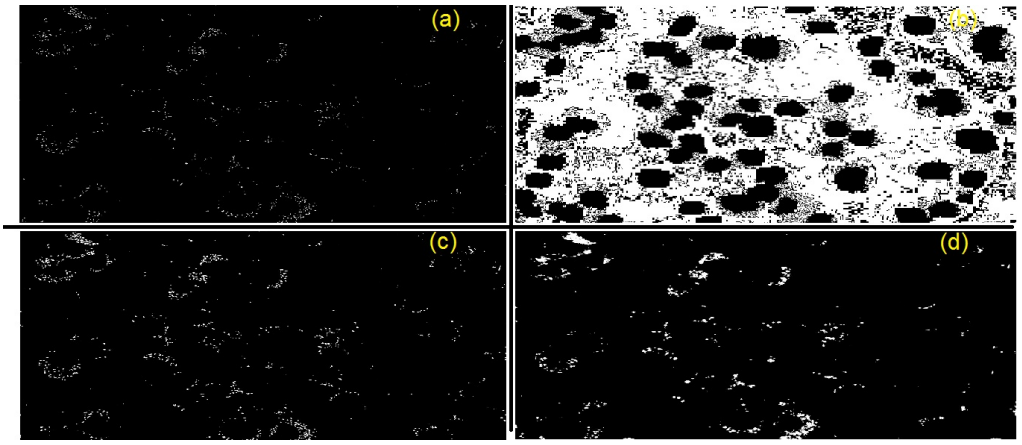



FIGURE 4.8: Comparison of the parasite segmentation quality on the test image (a)between our algorithm (b) and a number of state of the art non Intensity-Based Thresholds: k-means clustering (c)Level Sets (d)Trainable

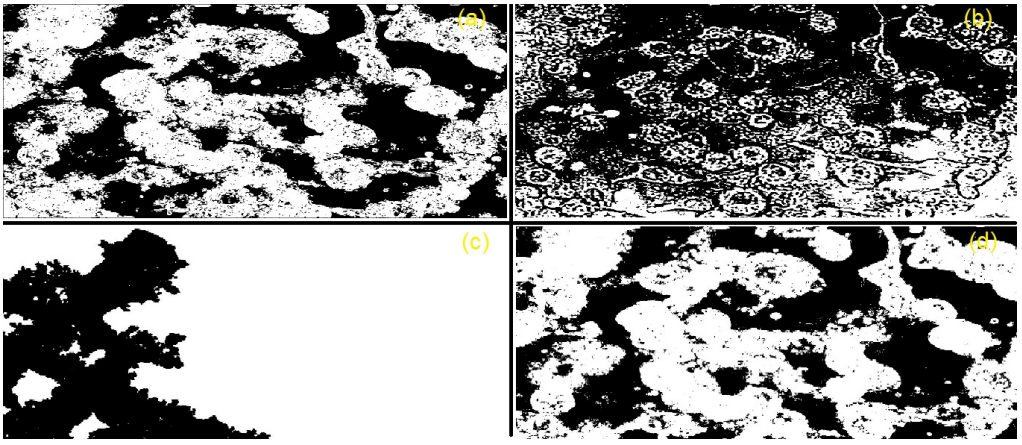



FIGURE 4.9: Comparison of the cytoplasm segmentation quality on the test image (a)between our algorithm (b) and a number of state of the art non Intensity-Based Thresholds: k-means clustering (c)Level Sets (d)Trainable

## 4.2 Biological Evaluation

### 4.2.1 Software validation versus Manual Giemsa Counting: susceptibility of intramacrophagic L. infantum towards glucantime

In order to validate the accuracy of software dealing with intracellular Leishmania parasites, two experiments were made. In the experiments, *THP1 infected cells* were subjected to different concentration of *glucantime*® drug candidate, the mainstay treatment for leishmaniasis disease. For this purpose, *Infection*, *Drug Incubation* and *Cell Fixation* steps were done first. Then, one series was stained using *GIEMSA* for manual counting and the other one with the *fluorescent DNA marker DAPI*. To calculate the parasitic burden(parasitic index) manually, for each drug concentration, 300 cells(100 for each

TABLE 4.7: Comparison between non intensity based segmentation methods for cell pipeline in terms of defined measures

| **Method** | **Overlapped Cells** | **Cell Scraps** | **Edge Preservation** | **Noise Filtering** | **Connectivity** | **Smoothness** |
|---|---|---|---|---|---|---|
| Our Method | Very High | Very High | Very High | Very High | Very High | Very High |
| k-means clustering | Low | Very High | Medium | Medium | High | High |
| Level Set | Low | Medium | Medium | Medium | High | Medium |
| Trainable | Very Low | Medium | High | Very High | Very High | High |

TABLE 4.8: Comparison between non intensity based segmentation methods for parasite pipeline in terms of defined measures

| **Method** | **Edge Preservation** | **Noise/Parasite Discrimination** |
|---|---|---|
| Our Method | Very High | Very High |
| k-mean clustering | Very Low | Very Low |
| Level Set | Medium | Very High |
| Trainable | Low | High |

TABLE 4.9: Comparison between non intensity based segmentation methods for cytoplasm pipeline in terms of defined measures

| **Method** | **Edge Preservation** | **Noise Filtering** | **Connectivity** | **Smoothness** |
|---|---|---|---|---|
| Our Method | Very High | Very High | Very High | Very High |
| k-means clustering | Low | Low | Medium | Medium |
| Level Set | Very Low | Very Low | Very Low | Very Low |
| Trainable | Very High | Very High | Very High | Very High |

triplicate) were counted under light microscope. Simultaneously, DAPI and phase contrast image pairs were acquired using fluorescent microscopes at magnification x40. 32 images were taken containing approximately 300 cells. The images obtained were then processed using developed software and Parasitic Index values were calculated for each cell to be then compared with manual counting results. Results presented in Fig. 4.10 show that the number of detected parasites(and parasitic index) in software is drastically higher in comparison with manual results. However, this observation was expected considering the fact that in manual counting processes, the human observer often tries to detect and mark most visible parasites. Furthermore, high concentration of parasites around some cells complicates the identification of individual parasites. However, after normalizing the data using percent reduction of the total parasitic burden compared to

the non-treated infected control(%PI), results of both screening methods show very high correlation together.

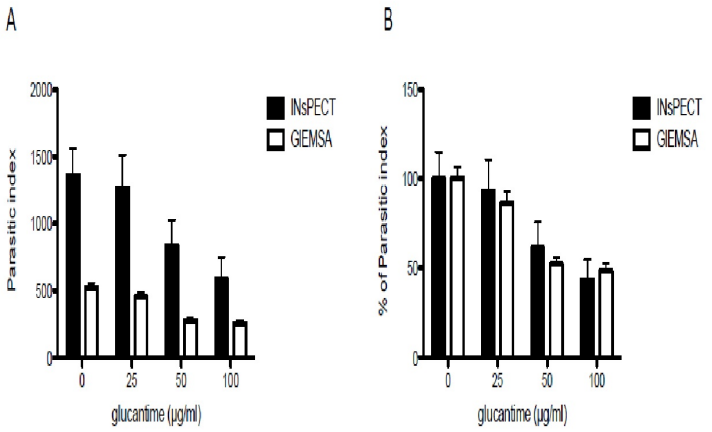



FIGURE 4.10: Results of Software validation versus Manual Giemsa Counting: a) Number of detected parasites in software is higher than manual counting due to human-laziness and high concentration of parasites around some cells b) normalized counting results of both methods show high correlation together

### 4.2.2 Software validation towards intracellular Toxoplasma gondii parasite

In order to validate software robustness against unrelated intracellular parasite, analyses on HFF fibroblasts infected by Toxoplasma gondii protozoan parasites were performed. These components are labelled with Hoechst fluorescent DNA stain. Input image acquisition is accomplished using HCS system which is designed specifically for fully automated image acquisition and analyze tasks. To have a measure for comparison purposes, parasites were labeled with specific antibody coupled to FITC fluorescent conjugate. Toxoplasma parasites are different from that of Leishmania infantum amastigotes in terms of size and distribution, and they are mostly concentrated in small aggregates inside cell cytoplasm.

Analyzes of some representative images were performed with software and then comparison of visual output annotated images to FITC positive labeled parasites confirms a good correspondence between the two methods(see Fig. 4.11).

This validation test qualifies the software in the sense that it could work accurately for the input image sets of different laboratories as long as input images' quality and acquisition methods follow some standards. Furthermore it proves that under different circumstances, software works quite automatically without need of user interaction and generates reasonable results.

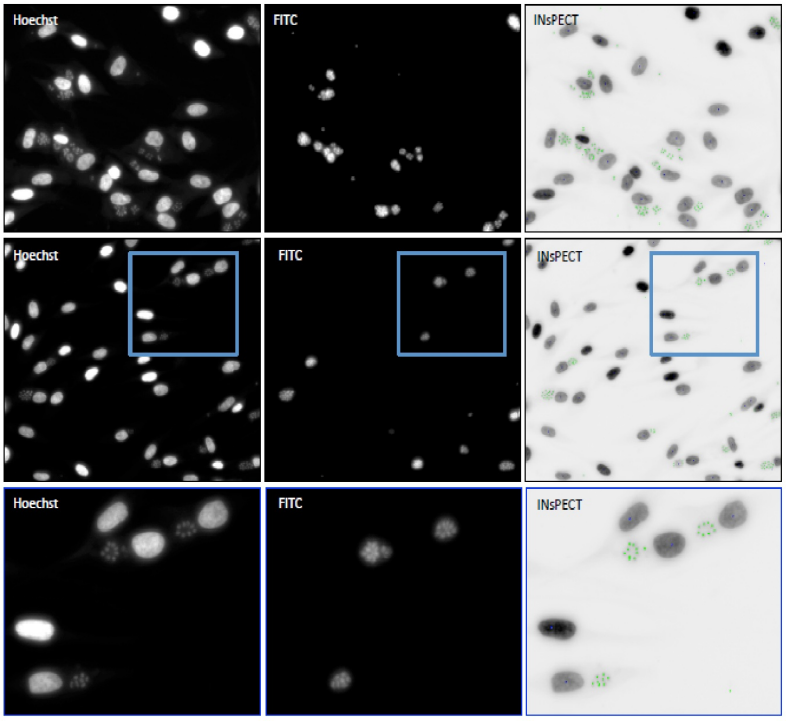



FIGURE 4.11: Software validation towards intracellular Toxoplasma gondii parasite

# Chapter 5

# Conclusions and Further Research

## 5.1 Thesis Summary and Concluding Remarks

Manual monitoring of components of the interest in Leishmaniasis image sets is subjected to several human made errors. The major objective of this project is the development of an automatic dependable framework to investigate Infection Ratio of Leishmania parasites in host cells. Consequently, leishmaniasis drug discovery pipeline gains a dramatic speed up in finding drug candidates for the disease.

Cell Segmentation is the process in which Cell portions are extracted from input DAPI images. This process has received a lot of attention between researchers in last decades; however it seemed to be a non trivial task for automated image segmentation packages. Illumination inhomogeneity across the cells' contours, the existence of intensity saturated and overlapped cells and variety of shape and orientation of cell portions are between the most challenging issues which make cell segmentation a complex task. Furthermore, different image acquisition techniques and tools(stains, microscopes, imaging wavelengths, etc) in laboratories often lead researchers to use a variety of approaches towards solving the problem. *Intensity Thresholding* methods try to segment cells assuming their intensities are significantly different from the background, locally or globally. *Feature Detection* methods, using linear image filters, derives cell portions intensity based features and segment cells based on them. *Morphological Based* methods acquire topological properties of cells in the image by using a combination of its non linear operators such as erosion, dilation, opening and closing. *Region Accumulation* methods identify cell regions starting from initial seed points and iteratively add connected points to previously labeled regions. Finally, *Deformable Model Fitting* methods try to fit some deformable model to the image data and extract cells areas with assigned set of information.

Parasite Segmentation is the process in which small parasites are extracted from input DAPI images. Since parasites occupy very tiny portions of input images, the task of extracting them with existing thresholding methods becomes highly complex because the majority of available methods often work efficiently for the objects of interest which have quite noticeable size and shape and investigable intensity functions. Furthermore, the similarity of parasites to random generated noise which occasionally exist in all images will lead to having further discrimination steps to judge between these byproducts and real parasites. To the best knowledge of authors, till the date of publishing this report, there is no efficient available method dedicated to segmenting parasites(or alternatively tiny particles) in DAPI images. Some researchers use feature based methods such as segmenting parasites with setting minimum and maximum size feature and the others usually reuse thresholding methods such as local and global intensity based thresholding.

Cytoplasm Segmentation using Phase Contrast(or DIC) input images is defined as extracting cytoplasm traces around cells and could be considered as a state-of-the-art task in the sense that previous researches towards accomplishing it is seldom in literature. Cytoplasm areas are visible in Phase Contrast(or DIC) images as traces around cells which often could be distinguished from its surroundings according to high concentration of its representative pixels. However, high gray level intensity variance of pixels in such images makes the separating task of foreground and background a very difficult one to achieve. Some parts of the cytoplasm traces are so unrepresentative of its vicinity and occasionally contours are not so visible and distinguishable from background. Bright halos surrounding cells often misleads edge detector operators to produce right edges. Finally, the existence of some by products such as random generated noise of image acquisition phase in such images makes the thresholding task more complicated. *Active Contour* based methods identify position of the objects of the interest by guiding the active contour using the vector flow of the leading protrusion. *Morphological Watershed* based approaches using markers of objects centroids tries to segment region in Phase Contrast images. *Pattern Recognition* based methods relies on the fact that prior knowledge of spatial features of objects of the interest such as shape will help algorithm to segment such regions.

Considering all above mentioned challenges and considering the high amount of processing load, a final solution was proposed in such a manner that it first generates dependable results with acceptable accuracy and then addresses all the challenges properly. Noise models assigned to both DAPI and Phase Contrast images and most fitted smoothing filters were used to denoise or/and smooth images prior to the migration to segmentation and quantification steps. Illumination variation side effects were minimized using local adaptive thresholds rather than global thresholding methods. Discrimination between random noise and real parasites were effectively handled using proposed thresholding

method, called Threshold for decreasing PDF's. Overlapping cells were also detected and separated in a good fashion using state of the art Watershed Morphological Thresholding method. Gradient-based Threshold were proposed for the first time in this report to accomplish the task of segmentation for Phase Contrast image set. Finally, for each operation involved in the pipeline, the processing time and used memory were minimized using low cost implementations of such operators.

Obtained results were validated both computationally and biologically.

Ever since this project began, no dedicated academic and commercial packages for investigating Infection Ratio of Leishmania parasites using Image Processing techniques were being used. Therefore, the basis for validating results from the perspective of Image Processing accuracy were chosen to be Segmentation results of cell, pipeline and cytoplasm regions. The results of our method then compared to the most common thresholding methods such as Global and Local Adaptive Intensity Based methods, k-means Clustering, Level Sets and Trainable segmentation.

- In cell segmentation, our method is the only method which address ALL challenges, namely Overlapped Cells, Cell Scraps, Edge Preservation, Noise Filtering, Connectivity and Smoothness while the nearest competent methods, k-means clustering and trainable could not deal well with overlapping cells.
- In parasite segmentaion, our proposed thresholding method excelled all other methods. Level Set was the only method which had almost the same results. However, it was very time consuming and needed a high degree of user interaction.
- Cytoplasm Segmentation using our method was proven to be very promising. While most of the tested methods did not perform well when confronted with challenges of Edge Preservation, Noise Filtering, Connectivity and Smoothness, the proposed method handled all of such challenges well and generated acceptable results. The results almost had the same quality as the Trainable Segmentation method, however excelled it in the sense that it does not need prior train data of cytoplasm regions for its processes.

Analysis of the obtained results proved that our method's accuracy excels all other methods and it also executes the task with maximum degree of automacy and minimum interaction with user.

In order to validate the results biologically, *ground truth* generated by manual interference of biologists was used and comparing the results, proved that the work was very

promising. To detect susceptibility of intramacrophagic L. infantum towards glucantime, software validation versus *Manual Giemsa Counting*(gold standard method) was accomplished and high correlation between results was proved. Furthermore, software validation towards intracellular Toxoplasma gondii parasite proves that software is capable of generating good results confronting with different input image sets.

Major findings of this project is published in biological and computer science journals. Developed Framework is also available in public domain for the use of biologists and scientists with the main goal of accelerating the process of Drug Discovery for neglected Leishmaniasis disease.

## 5.2 Recommendations for Future Works

Despite the fact that project outcomes address so many challenges of the problem in hand and builds a comprehensive framework which could be used as a reliable tool for Drug Discovey pipeline of Leishmaniasis disease, there are still some issues remaining for the future work.

- There are some parameters such as spatial kernel size and morphological operators structuring element size associated with components of proposed algorithm pipelines which currently needs to be set up manually. Such parameters are somehow sensitive to input images' quality and therefore need to be reset when new image sets are tested. It would be ideal if in the future, these parameters could be automatically configured.
- Although there is a lot of background research on processing objects with relatively big areas in literature, the research of small particles are seldom. There are very few existing methods to extract small details in literature and almost no efficient method is proposed for segmenting such details. As the result, for cell processing and cytoplasm processing sections of the project, we could deal with different methods and then measure and compare their outcomes to find efficient solution. On the other hand, due to lack of sufficient background enrichment in parasite processing section, we ought to propose our method for extracting and segmenting parasites. The results were shown to be very promising both computationally and biologically; however in the future, upon proposing new methods, our results could be compared to them and be substituted with them in case novel methods excel our results.
- Extracting regions of the interest in medical images using Image Processing techniques are often subjected to different levels of error. Such errors have different

sources and reasons such as lack of prior information about input image set, improper modelings, hiring not suitable operators during pipeline, and so on. To name some specific errors which could occur for our image set in hand, consider misdetection of cells, parasites and cytoplasm regions, inability to discriminate between random noise and real parasites, misdetection of overlapping cells as unified cells and underestimating/overestimating cytoplasm regions around cells. Despite the fact that prior knowledge about images were considered as well, the pipeline components were used properly and modelings were representative of images' features and also considering that results were very near to ground truth, as proved in chapter 4 , still some corrections need to be performed to optimize future results. These corrections could be applied to results by building a learning framework. The framework then interacts with user, obtains feedback as corrections and passes it onto the pipeline to learn.

# Appendix A

# Image Processing: Notation and Terminology

## A.1 What is Digital Image Processing?

An image is defined as a two dimensional function $f(x, y)$ where $x, y$ are the spatial coordinates of picture elements and $f$ is the amplitude of the element located at each point(which is also called *intensity* or *gray level*). According to the definition, picture elements are considered to be the smallest elements which have a particular location in the image and they are usually called *pixels*. When $x$, $y$ and $f$ values are all finite discrete values, we call the image a *digital image*. Processing digital images using digital computers is called *Digital Image Processing*[42].

Usually there is no agreement between authors regarding exact categorization of Image Processing operations. Moreover, it is not often clear where image processing steps finish and other related areas such as image analysis and computer vision steps start. However, a useful categorization is to generally consider three steps during the process[42]:

1. **Low Level step:** includes primitive operations such as noise reduction, contrast enhancement and image sharpening. Both input and output of this step are images.

2. **Mid Level step:** includes tasks such as segmentation, description and classification of individual objects. The input of this step are images and the output are often some attributes like edges, contours or identity of individual objects extracted from input images.

3. **High Level step:** includes high level processing tasks such as image analysis or image recognition and can also be associated with other areas such as computer vision.

For the purpose of this report, we mainly concentrate on operations of Low Level and Mid Level steps. The following section of this chapter explains and discusses corresponding operations of each category in detail. The classes of operations discussed throughout the section include Image Enhancement, Image Restoration, Morphological Processing and Image Segmentation.

## A.2 Image Enhancement

The main objective of enhancement is to process an image in such a way that the resulting image is more suitable than the original one for the purpose of use in a specific application.

Image Enhancement approaches can be divided into two categories[42]: *Spatial Domain* methods which are based on direct manipulation of pixels in an image and *Frequency Domain* methods which relies on modifying the *Fourier Transform* of an image. Some Spatial Methods will be covered in this chapter and Frequency Domain methods are out of the scope of this report.

Spatial Domain processes is defined by the expression

$$g(x, y) = T[f(x, y)] \tag{A.1}$$

Where $f(x, y)$ is the original image, $g(x, y)$ is the result image and $T$ is an operator on $f$, defined over some neighborhood of $f(x, y)$. $T$ also can be applied on a set of input images like arithmetic or logical pixelwise operators. This will be discussed soon.

### A.2.1 Graylevel Transformations

These transformations are among the basic and simplest Image Enhancement techniques. Suppose $r$ and $s$ are the intensity value of pixels before and after applying the transformation. Then:

$$s = T(r) \tag{A.2}$$

$T$ is a transformation mapping a pixel value $r$ to a pixel value $s$.

Frequently used functions of this group are *linear*(negative and identity transformations), *logarithmic*(log and inverse log transformations), and *power-law*(n'th power and n'th transformations).

#### Invert

The *Invert* or *Logical NOT* is an operator which takes binary or grayscale image as input and produces its negative image as output. In other words, applying this operator causes light pixels of the image to become dark and dark pixels to become light. The Invert operator is usually used to make the features in the image appear clearer to a human observer. For example, in medical images, using the Invert operator, tissues, cells or other medical components become more visible to the human eye[36].

Let maximum intensity level of the input image be $M$ and the intensity of the pixel located at coordinates $i, j$ be $P(i, j)$. Then the inverted intensity value of the pixel is given by $Q(i, j)$ and

$$Q(i, j) = M - P(i, j) \tag{A.3}$$

### A.2.2 Histogram Processing

The *Histogram* of a digital image is a discrete function $h(r_k) = n_k$ where $r_k$ is the $k^{th}$ gray level and $n_k$ is the number of pixels in the image having gray level $r_k$ ( $0 <= r_k <= L$ ), $L$ is the maximum possible value for intensity[42].

Since histograms provide many useful image statistics, they are considered to be a basic tool for many spatial domain processing techniques and also other image processing applications such as *Image Compression* and *Image Segmentation*. The most common applications of Histograms in Image Enhancement area are *Histogram Equalization*, *Histogram Matching* and *Local Enhancement* using statistics obtained from Histogram.

### A.2.3 Image Arithmetic

*Image Arithmetic* operations are one of the most simple and basic operations in Image Processing packages and it has a wide range of applications in areas such as *Noise Reduction*, *Motion Detection and Prediction*, *Illumination Correction* and so on. Briefly speaking, these operators take two or more images of the same size and apply some bitwise or logical operators, pixel by pixel to produce a unique result image. Also there are other forms of operators which use an offset instead of second image and use this offset

as bitwise or logical pattern. In Fig. A.1, you can see an example application of these operators for background subtraction. The Right image is a text with bad illumination across the borders. As a result, trying to find a threshold to separate foreground text from background area fails, as shown in the middle image. So we try to find a model for illumination gradient and the result is an image like the left image called *lightfield*. Then using this image, we eliminate variation in background intensity by subtracting it from the original image. The most common Image Arithmetic operators will be explained now.

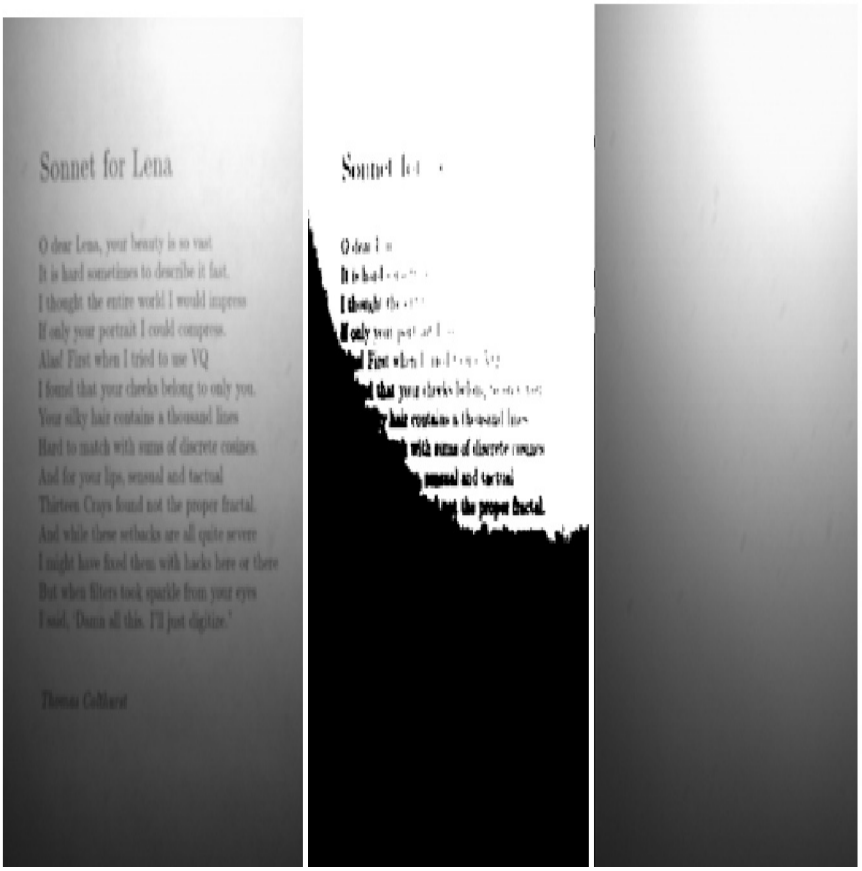



FIGURE A.1: Background Illumination correction using Lightfield image[61]

Lets $P_1(i, j)$ and $P_2(i, j)$ be the pixel values located at coordinates $i$ , $j$ of the first image and second image, $Q(i, j)$ be the value of the pixel in corresponding coordinates of the resulting image for the intended operator. $C$ is the offset pixel value when we use it as the second image. Then basic arithmetic operators are defined.

### Addition

Adding two images:

$$Q(i, j) = P_1(i, j) + P_2(i, j), \tag{A.4}$$

adding a constant offset to one image:

$$Q(i,j) = P_1(i,j) + C. \tag{A.5}$$

### Subtraction

Subtracting two images:

$$Q(i,j) = P_1(i,j) - P_2(i,j), \tag{A.6}$$

subtracting a constant offset from one image:

$$Q(i,j) = P_1(i,j) - C. \tag{A.7}$$

### Multiplication:

Multiplying two images:

$$Q(i,j) = P_1(i,j) \times P_2(i,j), \tag{A.8}$$

multiplying a constant offset to one image:

$$Q(i,j) = P_1(i,j) \times C. \tag{A.9}$$

### Division:

Dividing two images:

$$Q(i,j) = P_1(i,j) \div P_2(i,j), \tag{A.10}$$

Dividing one image by a constant offset such that C $\neq$ 0:

$$Q(i,j) = P_1(i,j) \div C. \tag{A.11}$$

It should be noted that in the implementation of these operators, some checks should be done otherwise results could be unwelcomed and with error. In addition and multiplication operators, the result of the adding or multiplying can exceed maximum intensity value which is usually called *saturation*. Then, based on implementation policies, we can change this value to maximum legal intensity value or we can continue adding or multiplying process from minimum meaningful value again. In the subtraction operator, on the other hand, subtracted intensity result can go below zero. Then the same policies applied to the addition operator, which is rounding to zero or continuing from maximum

intensity value can be implemented here also. Alternatively we can use absolute difference of input images and then negative values will not generate, at all. For the division operator, the divisor should not be zero and also if division is integer division, then we should have policies like rounding down the result value to the smallest integer[31, 36, 42].

The other less common operators are *Blending*, which does linear combinations of images and logical operators *AND*, *NAND*, *OR*, *NOR*, *XOR* and *XNOR* which do pointwise logical corresponded operators for binary images.

### A.2.4 Spatial Filtering

Some Image Processing operators work with the values of the image pixels in the defined neighborhoods. Such neighbourhood is called *subimage*, *filter*, *mask*, *kernel*, *template* or *window*[42]. Spatial filters work by moving the filter mask, pixel by pixel in the image and for each pixel, the result value of the filter is calculated using a predefined relationship. There are mainly two categories of Spatial Filters: *Smoothing Spatial Filters* and *Sharpening Spatial Filters*[42].

#### A.2.4.1 Smoothing Spatial Filters

These filters are usually used for *blurring* and *noise reduction* purposes. Blurring is used in very the first steps of processing to remove small details and also to fill small intensity gaps. Two famous and useful filters of this category, called *Mean* and *Median* Filters are explained.

##### Mean Filter

This linear low pass filter also called *Average filter* is mainly used to smooth images and eliminate noises. It improves noisy images, relaxes local variations and reduces sharpness. Furthermore mean filter is poor in edge preserving and it should be used wisely[30].

In mean filter, we have a sliding window which goes through the image, pixel by pixel and for each pixel, the intensity value is replaced by mean or average value of window pixels intensities. An example is given in Fig. A.2. In the example, the intensity value of the central pixel ( 223 ) is replaced by the average intensity value of neighborhood pixels, which is 211.6. In this example, sliding window size is considered to be 5 and so sliding window is a square of size 5. The time complexity of mean filter is $O(r^2)$, where $r$ is the square radius size of the filter.

##### Median Filter

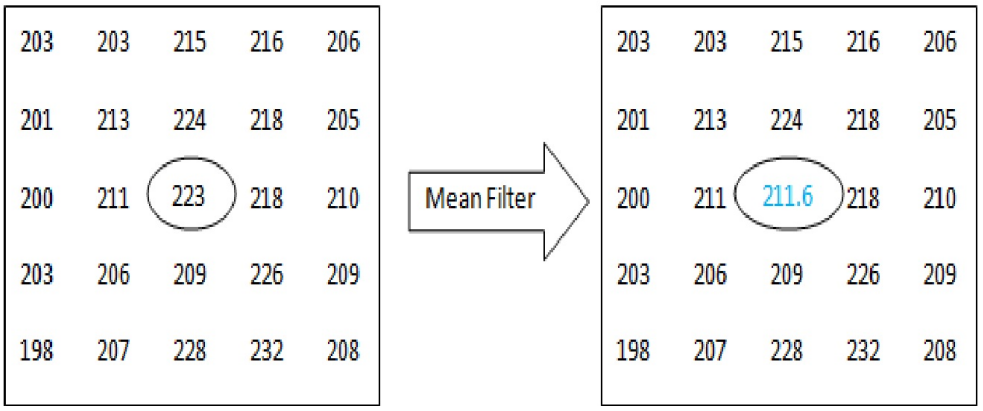



FIGURE A.2: Mean Filtering example

This non-linear filter is mainly used to reduce the random small noises in images. It is particularly useful in images with *impulsive* or *salt and pepper* noise; however, since most of the time it preserves the edges and other useful data while reducing noise, people use this filter as a satisfactory noise removing smoother filter[34].

In median filter, we have a sliding window which goes through the image, pixel by pixel and for each pixel, the intensity value is replaced by median value of window pixels' intensities. An example is given in Fig. A.3. In the example, the intensity value of the central pixel ( 223 ) is replaced by *median intensity value* of neighborhood pixels. In this example, sliding window size is considered to be 5 and so sliding window is a square of size 5. After sorting the intensity values of neighbors, we have values: 198, 200, 201, 203, 203, 203, 205, 206, 206, 207, 208, 209, 209, 210, 211, 213, 215, 216, 218, 218, 220, 223, 224, 226, 228, 232. Then the median value of this set is 209 therefore, the new value of the pixel will be changed to this value, respectively.

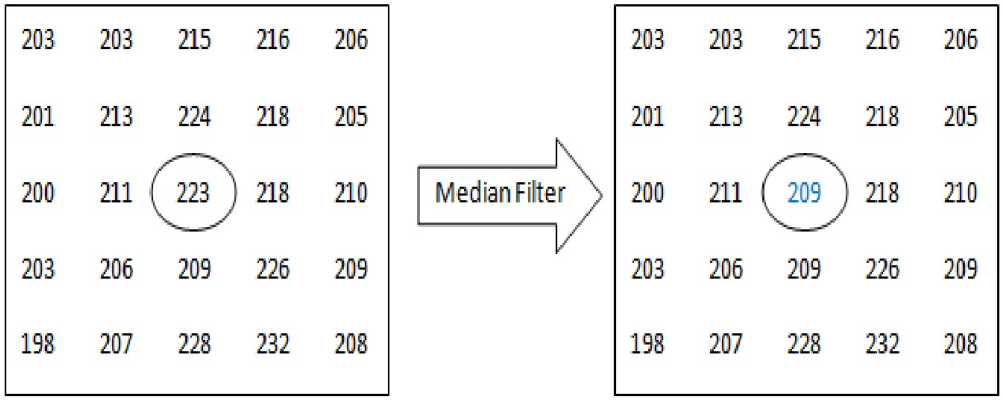



FIGURE A.3: Median Filtering example

In fact, what median filter does is consider each pixel at a time and then decide whether it is representative of its surroundings or not? If it is, the filter just smoothes the value to a degree and if not, then it turns out that the pixel can be considered a noise candidate and should be removed by the filter.

This algorithm's complexity is $O(r^2 Logr)$, where r is the square radius size of the filter and using bucket sort, when number of possible pixel intensity values are constant, this complexity can also drop down to $O(r^2)$ which is fast enough for small kernels.

#### A.2.4.2 Sharpening Spatial Filters

The main goal of sharpening process is to highlight some details which are not so clear in original image because of blurring effects or any other error including image acquisition method errors. Its applications range from electronic printing and medical imaging to industrial inspection and autonomous guidance in military systems[42]. A well known method which has been used for many years in the publishing industry and most recently in Image Processing packages is *Unsharp Masking*.

## A.3 Image Restoration

Like *Image Enhancement*, *Image Restoration* techniques also try to improve quality of images in some sense. The basic assumption before applying such techniques is that the images in hand are degraded images and then using some prior knowledge and models about degradation function, restoration techniques try to improve degraded images and recover the original one. So usually the restoration problem is considered from the point where a degraded digital image is given[42].

Image degradation could be modeled as a linear, position invariant process followed by *additive noise* that is not correlated with image values. Even this assumption is not completely valid, however good enough restored images can be obtained assuming these basic assumption. Fig. A.4 shows the degradation process. Given degraded image $g(x, y)$ and some knowledge about the additive noise $\eta(x, y)$, the objective of restoration process is to obtain and estimate $\hat{f}(x, y)$ of the original image $f(x, y)$. The more we know about degradation function and additive noise, the more likely that estimation is closer to original image. So in the following section Noise Models and some image restoration filters for denoising are discussed.

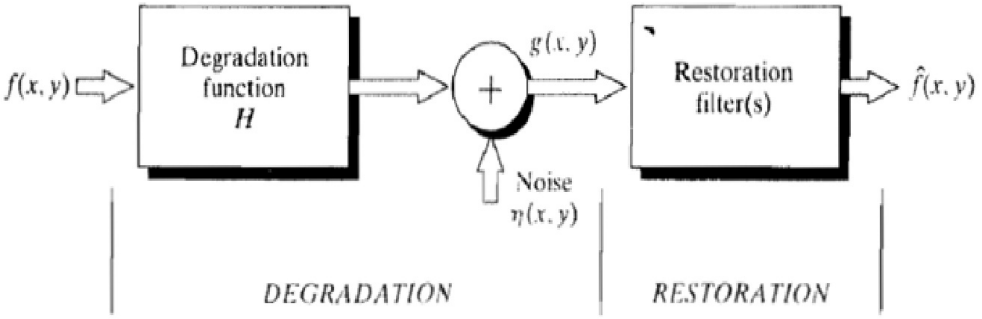



FIGURE A.4: Model of the image degradation/ restoration process[42]

### A.3.1 Noise Models

*Noise* is simply defined as random unwanted variations in illumination of images that have no spatial dependency from image to image. There are different sources which could generate noise in images. For instance, in digital images, different kinds of noise can be generated during image acquisition or transmission phases. Sensors for taking such images could be affected by ambient conditions or interference can be added to the image during the transmission step. Image noise is unwanted by product in image which often should be eliminated or reduced for better post processing. Here we review the most common types of noise in literature and also general denoising policies towards eliminating or reducing effects of each of them.

The most common perspective towards categorizing noise is using their shape of *Probability Density Function(PDF)* or equivalently their Histogram[42]. As a result, according to the different shapes of the histogram, the most typical image noise are:

**Uniform Noise**

can be analytically described by equation A.12 and it means that grayscale intensity value of the noise is distributed evenly across the image. This noise model is often used to generate other types of the noise. Due to its unbiased and neutral nature, it is often used to validate Image Restoration Algorithms' results.

$$p(z) = \begin{cases} \frac{1}{b-a} & \text{for a} \leq \text{z} \leq \text{b} \\ 0 & \text{otherwise} \end{cases} \tag{A.12}$$

**Gaussian(Amplifier) Noise**

is the statistical noise which has Probability Density Function of the Normal(or Gaussian) distribution(See equation A.13). Gaussian noise is often used as additive white noise to yield additive white Gaussian noise.

$$p(z) = \frac{1}{\sqrt{2\pi}\theta} e^{-\frac{z-z^2}{2\theta^2}} \tag{A.13}$$

**Salt and Pepper Noise**

can be analytically described by equation A.14. Salt and Pepper noise is usually generated in images because of malfunctioning pixel elements in the camera sensors, faulty memory locations or timing errors in the digitization process. Moreover, for two possible values of the noise, called $a$ and $b$, possibility of occurrence, if set by user, should be logical (for instance below 0.2); otherwise salt and pepper noise will dominate the image and restoration techniques often cannot restore original image well.

$$p(z) = \begin{cases} P_a & \text{for z = a(pepper)} \\ P_b & \text{for g = b(salt)} \\ 0 & \text{otherwise} \end{cases} \tag{A.14}$$

**Rayleigh Noise**

is the statistical noise which has Probability Density Function of the Rayleigh distribution(see equation A.15 ). Radar Range and Velocity images typically contain noise that can be modeled by the Rayleigh Distribution.

$$p(z) = \begin{cases} \frac{2}{b}(z-a)e^{-\frac{(z-a)^2}{b}} & forz \geq a \\ 0 & forz < a \end{cases} \tag{A.15}$$

**Gamma Distribution Noise**

is the statistical noise which has Probability Density Function of the Gamma distribution(see equation A.16). This noise can be obtained by *lowpass* filtering of laser-based images.

$$p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az} & forz \geq 0 \\ 0 & forz < 0 \end{cases} \tag{A.16}$$

**Exponential Noise**

The PDF of exponential noise is given by equation A.17.

$$p(z) = \begin{cases} ae^{-az} & forz \geq 0 \\ 0 & forz < 0 \end{cases} \tag{A.17}$$

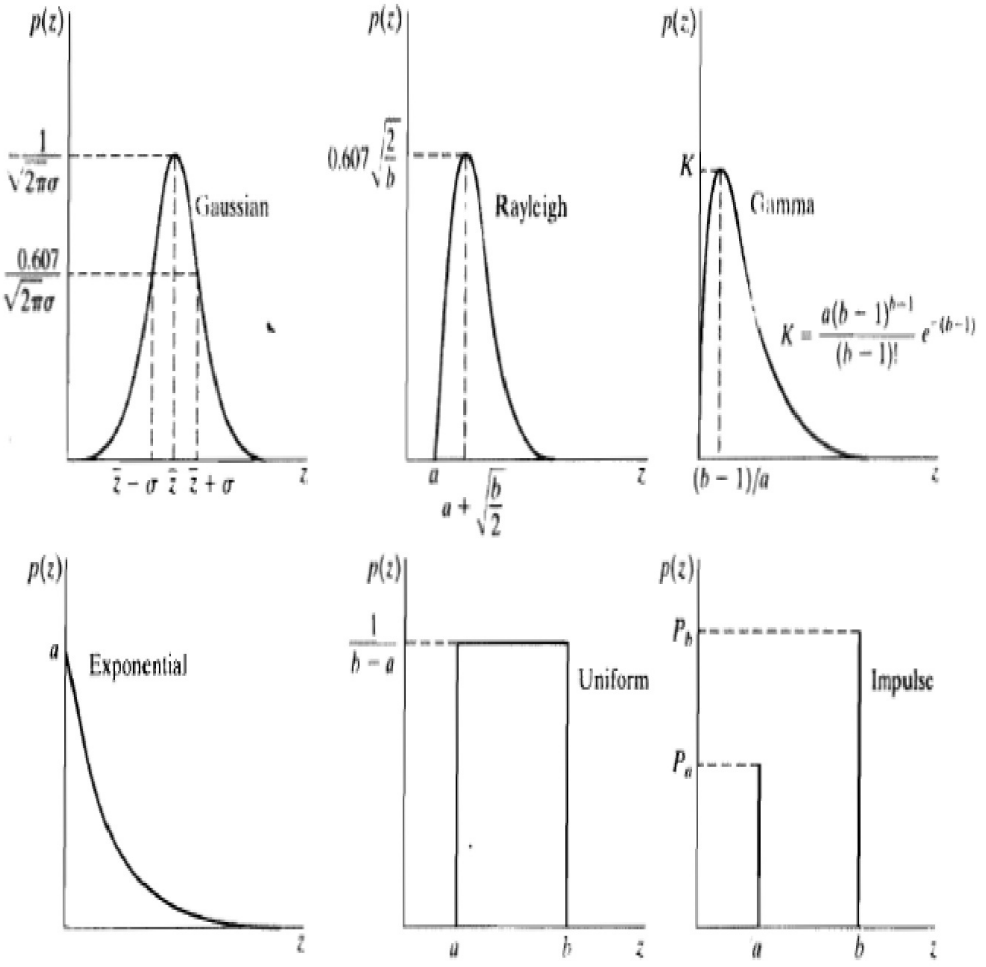



FIGURE A.5: Some Important Probability Density Functions[42]

There are lots of approaches in literature to denoise images with different kinds of noise. However, in general most of these methods use some spatial or frequency filters to undertake the denoising task. Mean Filter blurs the image, smoothens the noise and decreases their visibility in the image. Median filter, on the other hand is particularly good when salt and pepper noise is present, in addition using this filter will not have that much smoothing effect. Max filter is good for pepper noise and Min is good for salt noise and finally, Midpoint filter is good for Gaussian and Uniform noise. Although predicting the model of the noise in images is not usually that much straightforward, fitting a noise model to a set of images will help us to better filter out unwanted details from the images and as a result enhance the performance of further image processing tasks.

## A.4 Morphological Image Processing

*Mathematical Morphology(MM)* or simply Morphology is the set of theories and concepts for the analysis of spatial structures in the images. It is called Morphology because its analysis is mainly in the level of the objects' shape and it is mathematical because the related concepts behind MM are *set theory*, *integral geometry* and *lattice algebra*[62]. MM operators alone cannot solve image processing problems in hand. Nevertheless, there are a wide range of applications for MM. Some are listed below[63]:

- Image Preprocessing Tasks: Noise Filtering, Shape Simplification
- Object Structure Enhancement: Skeletonizing, Thinning, Thickening
- Segmentation
- Quantitative description of objects.

There are two main operators called *Erosion* and *Dilation* which all other MM operations are based on them. The Erosion operator is used to reduce the objects area in the input image and the Dilation operator is responsible for expanding the shapes contained in the input image. Moreover, the basic idea behind the examination of shapes with such operators is to compare structures of the image with a reference shape called *Structuring Element(SE)* locally. So Erosion, Dilation and Structure Element are three main concepts we mainly deal with in MM[39, 63, 64].

Let $X$ be the input image, $E$ the set of translation, $B$ the structuring element and $B(x)$ the translation of $B$ by the vector $x$ in $E$, therefore

$$B_x = \{b + x | b \in B\}, \forall x \in E \tag{A.18}$$

### A.4.1 Structuring Element(SE)

*Structuring Element(SE)* is a set of pixels which creates a shape and is used together with MM operators to analyze features of the images. The size and shape of the SE is highly dependent on the nature of image structures which we want to analyze and also on what one wants to keep or suppress, however, usually it is symmetrical, connected and convex. You can see some examples of SE in Fig. A.6.

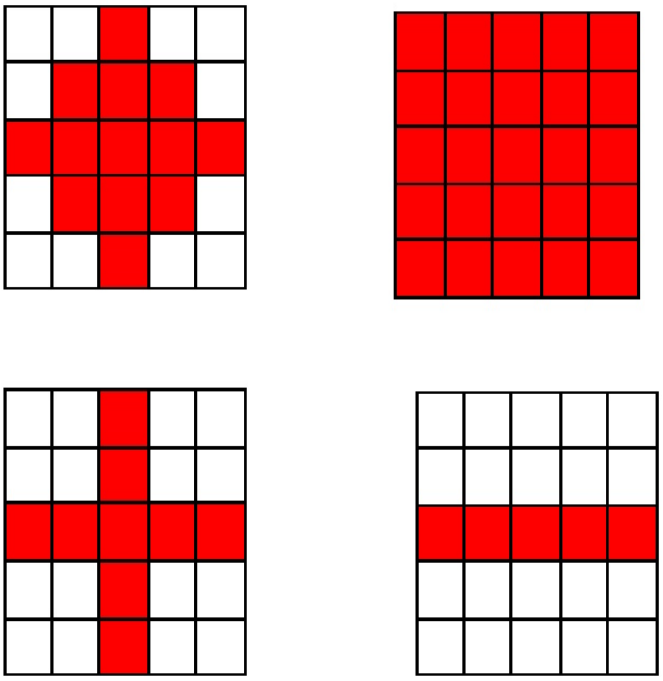


FIGURE A.6: Examples of different Structuring Element shapes

### A.4.2 Erosion

For binary images, erosion is defined as:

$$\epsilon_B(X) = \{x \in E | B_x \subseteq X\} \tag{A.19}$$

If SE has a center , for example if it is a circle or a disk, then erosion of the shape with respect to this SE also means the set of points from $X$, when SE locates on it, are also SE points covered in the $X$.

And for grayscale images, we have:

$$\epsilon_B(f) = \wedge_{b \in B} f_{-b} \tag{A.20}$$

which also could be stated as:

$$[\epsilon_B(f)](x) = min_{b \in B} f(x + b) \tag{A.21}$$

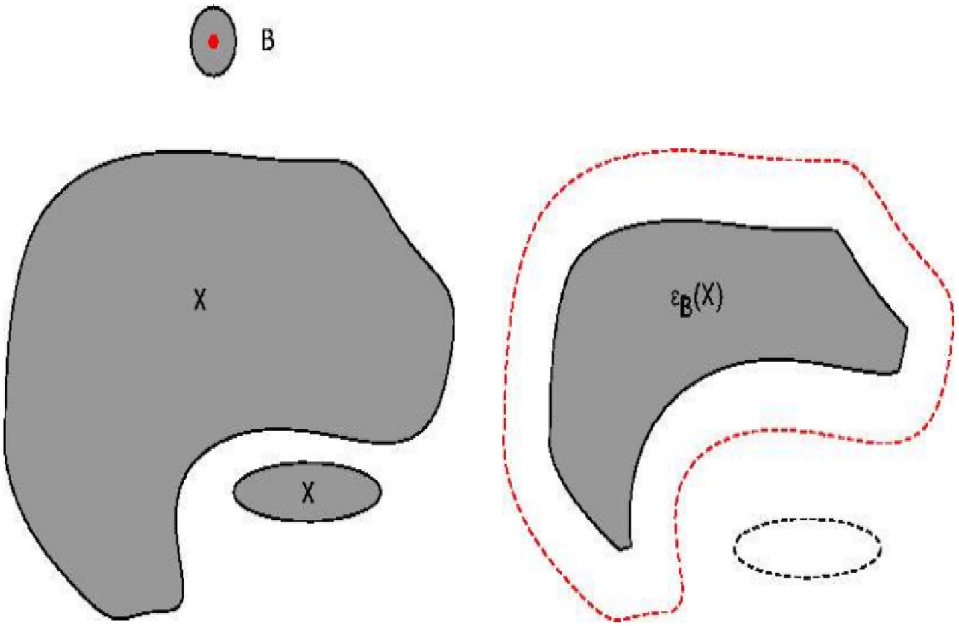



FIGURE A.7: Erosion example for a binary image[39]

### A.4.3 Dilation

For binary images, dilation is defined as:

$$\delta_B(X) = \{x \in E | B^s \cap X \neq \emptyset\} \tag{A.22}$$

Such that

$$B^s = \{x \in E | -x \in B\} \tag{A.23}$$

is the symmetric of $B$.

Again if SE has a center , for example if it is a circle or a disk, then dilation of the shape with respect to this SE also means the set of covered points when SE moves inside the shape.

And for grayscale images, we have:

$$\epsilon_B(f) = \vee_{b \in B} f_{-b} \tag{A.24}$$

which also could be stated as:

$$[\epsilon_B(f)](x) = max_{b \in B} f(x+b) \tag{A.25}$$

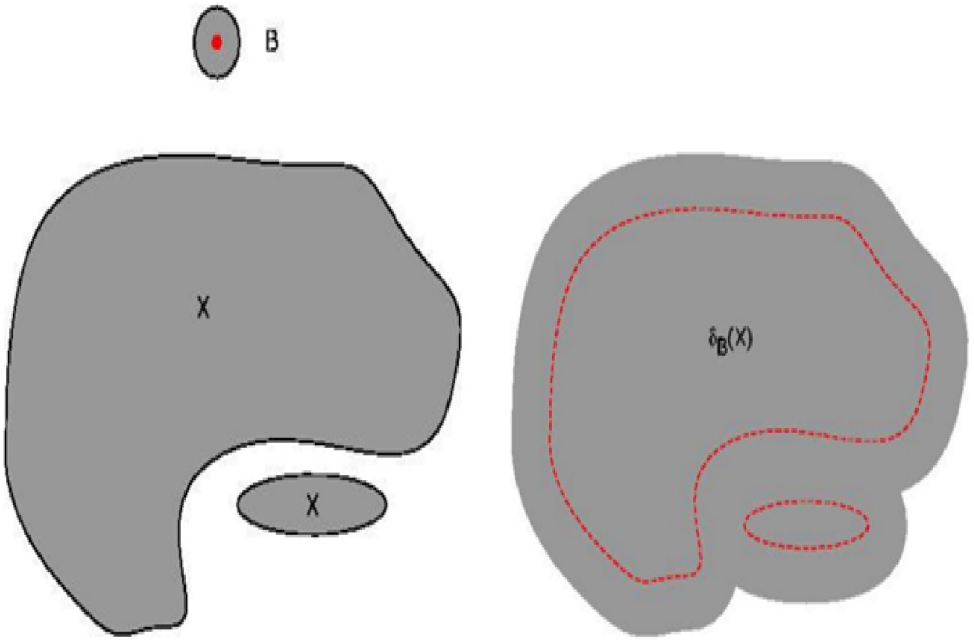



FIGURE A.8: Dilation Example for a binary image[39]

Now that we know the definitions for SE, erosion and dilation, we can define other Morphological operators based on these concepts.

### A.4.4 Morphological Opening

This operator is mainly used to eliminate unnecessary structures such as noises inside the image and it is defined as an erosion followed by a dilation. In fact, the effect of applying this operator to images is that objects smaller than SE will disappear and other objects remain unchanged (see Fig. A.9 ).

### A.4.5 Morphological Closing

This operator is mainly used to merge or fill the structures inside the image and it is defined as a dilation followed by an erosion. This operator actually removes holes smaller than SE and leaves other objects unchanged(see Fig. A.10).

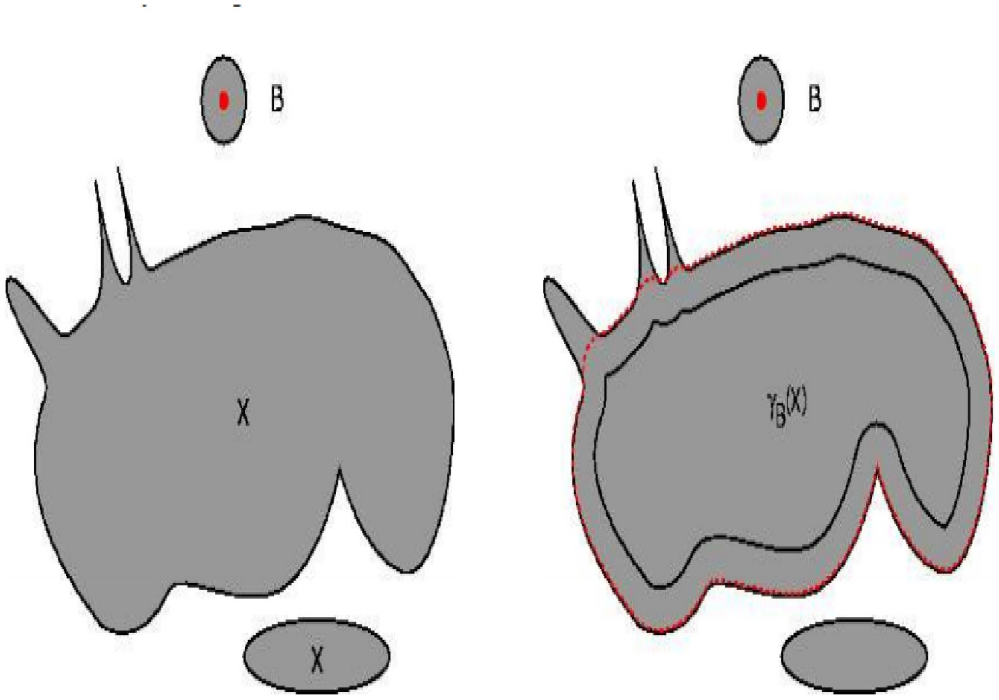



FIGURE A.9: Morphological Opening for a binary image[39]

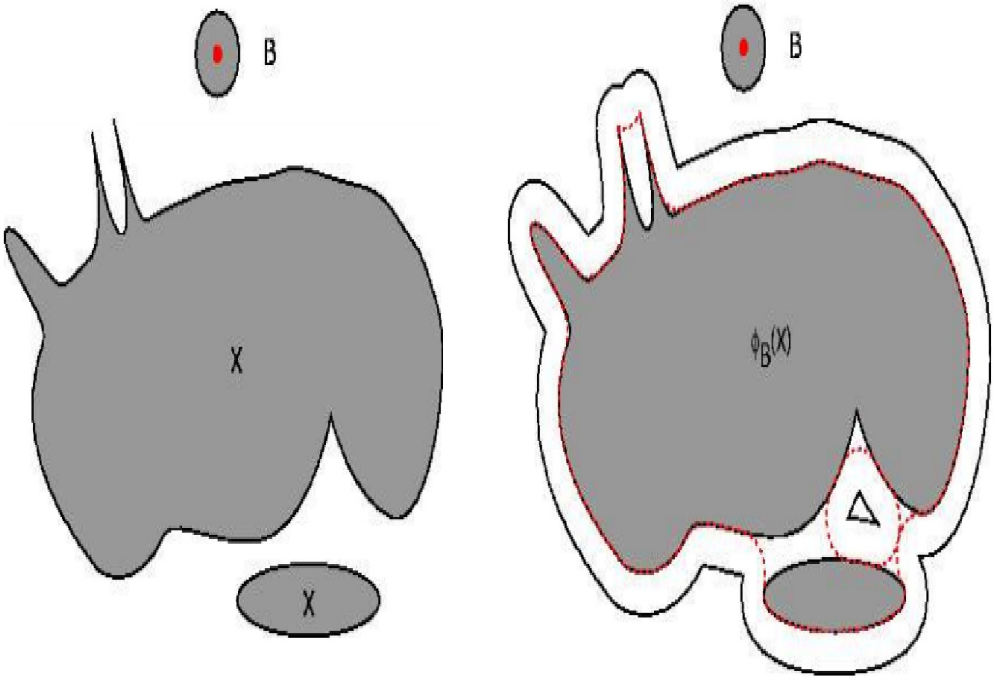



FIGURE A.10: Morphological Closing for a binary image[39]

### A.4.6 Top Hat Transform

Top Hat Transform is an operation which can exploit small structures from the images and is mainly used for feature extraction, background equalization and image enhancement tasks[65]. There are two types of this Transform:

- White Top Hat Transform, which is defined as the difference between original image and its Morphological Opening by some SE. The objects which are brighter than their neighborhood and also smaller than SE are the results of this transform.
- Black Top Hat Transform, which is defined as the difference between Morphological Closing of the original image and the image itself. The objects which are darker than their neighborhood and also smaller than SE are the results of this transform.

### A.4.7 Watershed Transform

Watershed Transform also called WST roots in the Mathematical Morphology area and is considered to be a region-based segmentation approach. The idea behind WST comes from geography.

There are different algorithms for Watershed Transformation[66], each of them have their pros and cons. A factor that is extremely important to evaluate the performance of each of these methods is the attitude of such methods towards the existence of image plateus. A *plateu* is simply the area that its neighbor has the same level of grayscale values. If the WST can handle plateus, then further transformations and smoothings are often not necessary anymore. One of the best algorithms with this feature is the well-known version of WST, called *WST by immersion* which is proposed by Vincent and Soille[67]. The idea behind this method is that it sees the image as a topographical surface and it tries to simulate a flood from this surface, starting from regional minimas. Then *dams* will be built whenever water flooding from different catchment basins meet. The *catchment basins*(or simply *basins*) are the partitions in which the WST breaks the topological surface. The results of the flooding process are some lines(not necessarily complete ones) called *watershed lines* which separate topological surface into different regions. You can find a simple illustration of the idea of watershed by immersion in Fig. A.11.

It should be noted that usually we do not apply WST on the original images; instead we often apply the operator on morphological gradient of the image since using this perspective, watersheds will be produced in points with grayscale value discontinuities which is more desirable in image segmentation[45].

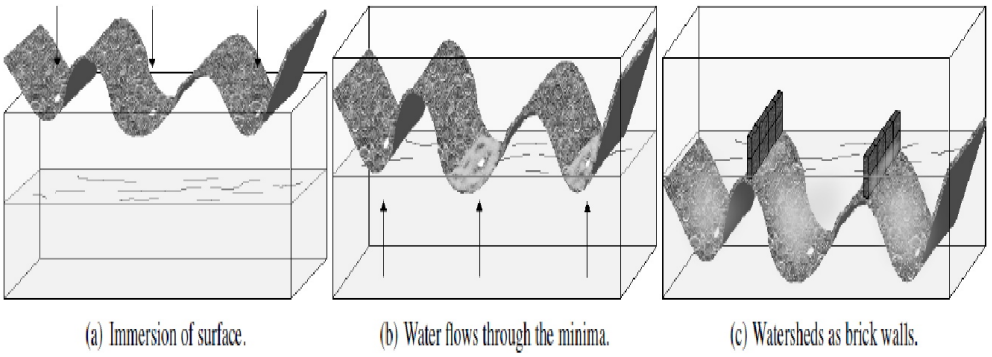



FIGURE A.11: Immersion simulation[45]

## A.5 Image Segmentation

*Segmentation* divides an image into smaller regions or objects of the interest. The *interest region* definition is highly dependent on the image which one tries to detect and segment and it can vary from very small particles of medical images to huge objects of mars planet. Segmentation accuracy occasionally determines the eventual success or failure of subsequent processing steps and so it has an important role in Image Processing problems.

Intensity discontinuities and similarities are the two most important properties of the pixels which are used in almost all Segmentation algorithms[42]. Using discontinuities we can usually partition an image based on abrupt changes in intensity. On the other hand, using similarities we can partition the image into regions that are similar according to a set of predefined criteria. In sections that follow, both of these approaches will be discussed with some level of detail.

### A.5.1 Edge Detection

*Edge Detection* is generally the assigned name for a class of algorithms which aims to detect and extract regions of the images with obvious variations. These changes could be discontinuities in depth, brightness or in surface orientation, changes in objects features or variations in scene illumination and occasionally they occur at foreground objects' boundaries. So this class of algorithms is specifically useful in fields of *feature detection* and *feature extraction*.

Based on the definition of edges, sensitivity of operators in detection process, existence of different kind of noise in images and many other factors, there are lots of Edge Detectors

in literature which are specifically proposed to identify different class of edges with different levels of sensitivity. However, we can generally divide the Edge Detectors into two categories, called *Gradient Based Edge Detectors* and *Laplacian Based Edge Detectors*. The first category, extracts edges using the first derivative of the input image, while the second category detectors use the second derivative[68–70].

#### A.5.1.1 Gradient Based Edge Detectors

Consider a one variable function $f(x)$ and its derivative:

$$f'(x) = \frac{df}{dx} \tag{A.26}$$

We see that for the points with maximum slope, the first derivative also has maximum value. So based on edge definition mentioned above, such points could be considered as edge candidates. Now the task is to translate the first derivative for a two variable function $f(x, y)$ which is:

$$G[f(x,y)] = \begin{bmatrix} Gx \\ Gy \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$|G| = \sqrt{Gx^2 + Gy^2} = |Gx| + |Gy|$$

$$\theta = \tan^{-1}(\frac{Gy}{Gx}) \tag{A.27}$$

Now in order to find the edges we just need to compare the magnitude of the first gradient with a threshold value. Furthermore using $Gx$ or $Gy$ alone will give us directional edges in vertical and horizontal direction. However, to approximate magnitude of the Gradient in $x$ and $y$ direction and then detect edges, we usually use convolution masks. The three most common Gradient based Edge Detectors are explained here. Each of these detectors use different convolution mask to find the image gradients.

**Sobel Edge Detector**

3*3 convolution kernel of this operator for $Gx$ and $Gy$ is:

$Gx =$

| | | |
|---|---|---|
| -1 | 0 | +1 |
| -2 | 0 | +2 |
| -1 | 0 | +1 |

$Gy =$

| | | |
|---|---|---|
| +1 | +2 | +1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

The good thing about this operator is that it is not that much sensitive to noise, however the produced edges are sometimes thick which makes edge localization poor.

**Robert Cross Edge Detector**

It provides an approximation to the gradient using these kernels for $Gx$ and $Gy$:

$Gx =$

| | |
|---|---|
| +1 | 0 |
| 0 | -1 |

$Gy =$

| | |
|---|---|
| 0 | -1 |
| +1 | 0 |

To formulate this, we have:

$$G[f(i,j)] = |Gx| + |Gy| = |f(i,j) - f(i+1,j+1)| + |f(i+1,j) - f(i,j+1)| \quad \text{(A.28)}$$

This operator in comparison with Sobel operator is more sensitive to noise and produces thinner edges. Moreover, it sometimes misses some edges.

**Prewitt Edge Detector**

It is very similar to the Sobel operator and it uses the following kernels for $Gx$ and $Gy$:

$Gx =$

| | | |
|---|---|---|
| -1 | 0 | +1 |
| -1 | 0 | +1 |
| -1 | 0 | +1 |

$Gy =$

| | | |
|---|---|---|
| +1 | +1 | +1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

### A.5.1.2 Laplacian Based Edge Detectors

We saw in gradient based methods that, for instance for a one variable function $f(x)$, its derivative is considered large when its magnitude is larger than a threshold. An equivalent assumption is that when second derivative reaches zero or equally saying, when it has zero crossing, then the first derivative is in its maxima and this is the philosophy behind using Laplacian Based Edge Detectors.

*Laplacian of Gaussian* which is also called *Marr-Hildreth* Edge Detector is a common operator of this category and it has the following steps:

1. Smooth the image using Gaussian Filter to remove high frequency noise components
2. Enhance the edges using Laplacian operator. Zero crossings will denote the location of the edges.
3. Use linear interpolation to extract locations of the edges.

In the above definition steps, Laplacian of the image with pixel intensity values $I(x, y)$ is defined as:

$$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{A.29}$$

And can be approximated using one of these convolution kernels:

$Gx =$

| | | |
|---|---|---|
| +1 | +1 | +1 |
| +1 | -8 | +1 |
| +1 | +1 | +1 |

$Gy =$

| | | |
|---|---|---|
| -1 | +2 | -1 |
| +2 | -4 | +2 |
| -1 | +2 | -1 |

And also for Gaussian Filters, we use the Laplacian of Gaussian function:

$$LoG(x, y) = \frac{1}{\pi\sigma^4}\lceil 1 - \frac{x^2 + y^2}{2\sigma^2}\rceil e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{A.30}$$

Which can be approximated using kernels like this one:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | +1 | +1 | +2 | +2 | +2 | +1 | +1 | 0 |
| +1 | +2 | +4 | +5 | +5 | +5 | +4 | +2 | +1 |
| +1 | +4 | +5 | +3 | 0 | +3 | +5 | +4 | +1 |
| +2 | +5 | +3 | -12 | -24 | -12 | +3 | +5 | +2 |
| +2 | +5 | 0 | -24 | -40 | -24 | 0 | +5 | +2 |
| +2 | +5 | +3 | -12 | -24 | -12 | +3 | +5 | +2 |
| +1 | +4 | +5 | +3 | 0 | +3 | +5 | +4 | +1 |
| +1 | +2 | +4 | +5 | +5 | +5 | +4 | +2 | +1 |
| 0 | +1 | +1 | +2 | +2 | +2 | +1 | +1 | 0 |

As we see there are many ways[71] to perform edge detection. The adequate way to solve the problem in hand depends highly on the nature of it. Edge Thickness, Noise Sensitivity and Edge Connectivity are between the parameters which give us some clues about choosing the most convenient detector for the problem. For instance, you can see the result of four explained operators in Fig. A.12 for better understanding of the difference between detecting mechanism of these operators.

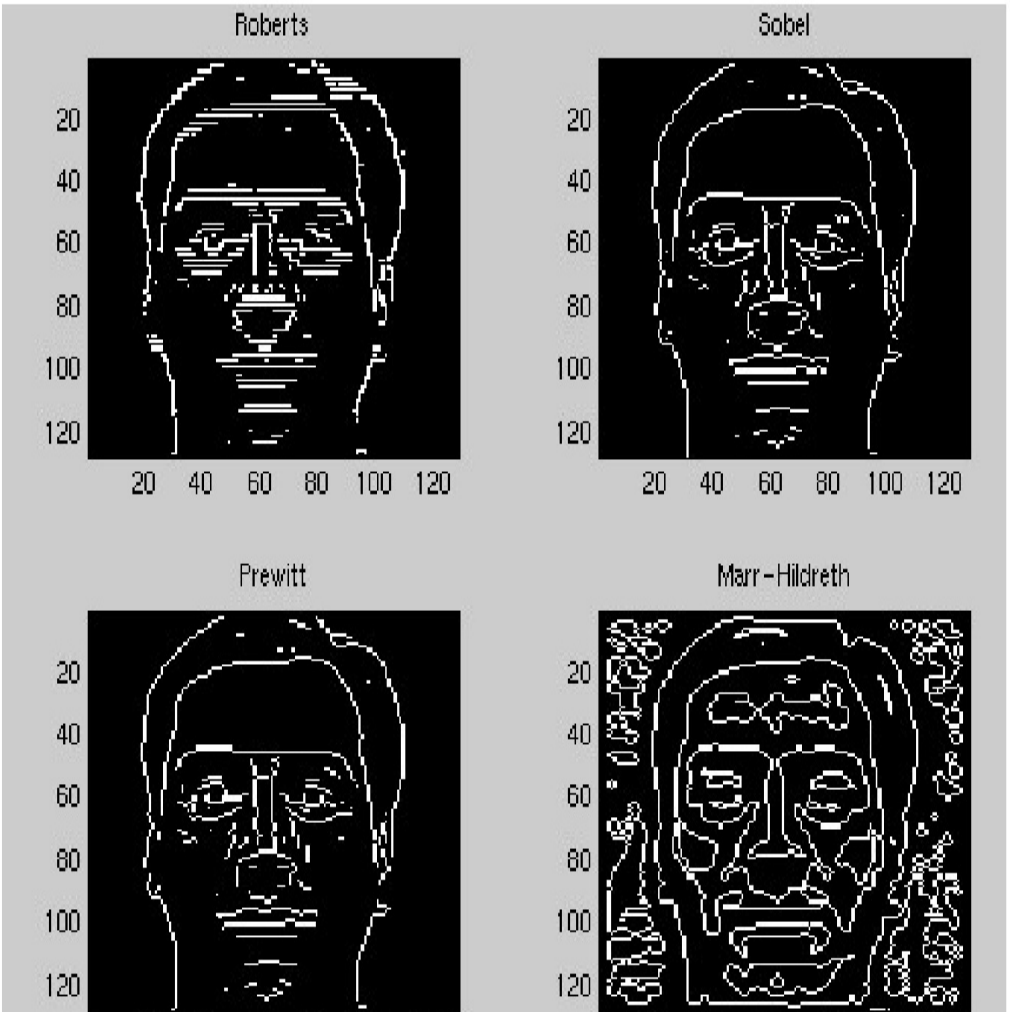



FIGURE A.12: Example of various Edge Detector filters[71]

### A.5.2 Thresholding

Thresholding is the general term for the group of techniques which tries to partition images into regions based on intensity values and/or properties of these values[42]. Because of intuitive properties, simplicity of implementation and computational speed, thresholding plays an important role in applications of Image Segmentation.

### A.5.2.1 Global vs Local Intensity-based Binary Thresholds

The most common type of Thresholding is *Binary Threshold* in which each pixel in the original image becomes either *Foreground(object)* or *Background* in thresholded image. This process is also called *Binarization*. Therefore, *Intensity Binary Thresholding* is defined as the process in which the algorithm decides whether a pixel is considered to be Foreground or Background in thresholded result image, based on intensity information of each pixel. Now, there are two approaches towards selection of threshold value $t$ which is the comparison parameter all pixels are compared to:

FIGURE A.13: Binary Thresholding effect on an example image. The left colored image is converted to bi-level right image during binarization

1. **Global Intensity Binary Thresholding**: the value of t is the same for all the pixels. The methods of this group are particularly useful when the intensity distributions of objects and background pixels are sufficiently distinct and as the result, a single global threshold can be used for the separation of objects from background. If we view the thresholding as a statistical decision theory problem whose goal is to minimize the average error incurred in assigning pixels to two or more classes, then it could be proved that the problem has optimum solution in the sense that the solution maximizes the between class variance. Such solution is called *Otsu Method* and is considered to be the most common Global Intensity Binary Thresholding method.

2. **Local Intensity Binary Thresholding**: the value $t$ is adaptive and may be different per pixel in the image. These kind of thresholding methods are often used when factors such as noise or non uniform illumination exist in the images and

affects performance of global thresholds. There are several ways to handle such problems. *Image Partitioning* is a category of these thresholds where the image is uaually divided into non-overlapping rectangles and then threshold values for each rectangle is found. Using this perspective, non-uniformities in illumination and/or reflectance are usually compensetad. Another category of Local Thresholds is called *Variable Thresholding based on local image properties*. Here, thresholds are computed at each pixel based on one or more specified properties computed in a neighborhood of that pixel. *Moving Averages* methods are also a special case of local thresholding in which moving averages along scan lines of the image is computed.

Finally, concepts of some operators which are used in experimental part of this report and cannot be directly put under one of above classes of operators will be explained and discussed in next section.

## A.6 Other Methods

### A.6.1 Fill Holes

The algorithm to fill the holes in binary images is proposed by Gabriel Landini[72]. The algorithm works as follows.

1. Lets background pixels intensity value be $b$ and foreground pixels intensity value be $f$. Border pixels intensity value are set to $x$.
2. The algorithm goes through the image pixel by pixel and checks 4-connected neighborhood pixel intensities for background pixels . If $x$ value is found in the neighbourhood, then pixel value changed from $b$ to $x$. otherwise it remains unchanged.
3. $b$ values which remain unchanged after step 2, are identified as hole pixels. The algorithm changes these pixel values to $f$ and also alter $x$ values to $b$ again.

### A.6.2 Connected Component Labeling

Labeling of Connected Components is one of the most popular and common operators for identifying and recognizing the shape of the objects in binary and also grayscale images. This operator scans the input image ( which is usually the result binary image of the thresholding step), pixel by pixel and then based on pixels connectivity tries to

group them. The result, will then be a set containing components(or regions). Each component(region)'s pixel set, shares the same intensity and are connected to each other somehow.

It should also be noticed that following concepts are used in almost all of these algorithms:

**Pixel Connectivity**: is the way in which a pixel in the image relates to its neighbors. There are mainly two types of pixel connectivity for 2-dimensional images:

**4-connected**: the pixels located at north, south, west and east part of each pixel are called its 4-connected pixels.

**8-connected**: the pixels located at north west, north, north east, south west, south, south east, west and east part of each pixel is called its 8-connected pixels.

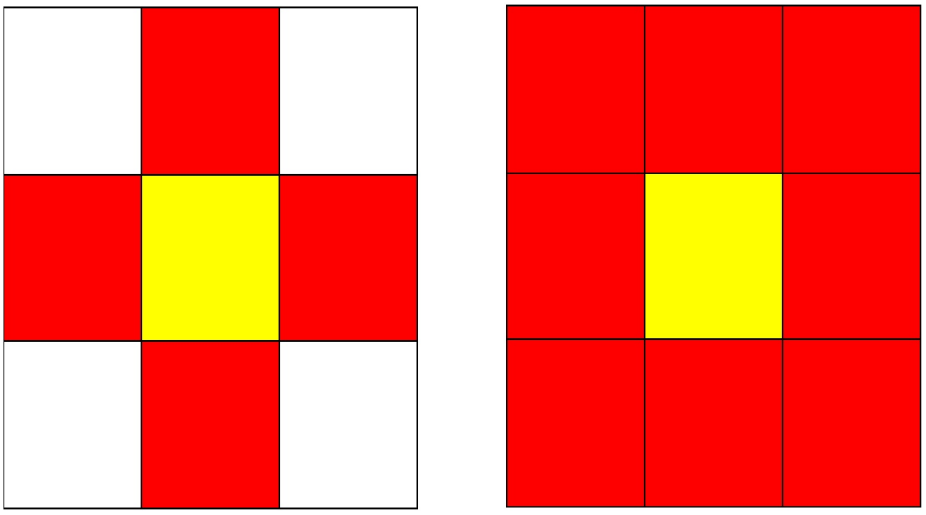


FIGURE A.14: Pixels with 4 and 8 connectivity

**Pixel Label**: is a numeric label assigned to each pixel. This label will then be used to find components and equivalency sets.

**Eqivalency**: is defined on label set. Two labels are in equivalency set with each other if for at least one pixel, they are in pixel connectivity(4-connectivity or 8-connectvity) to each other.

### A.6.3 Distance Transforms

*Distance Transform* is a fundamental geometrical operator which tries to find minimum distance between each pixel inside the image from some region of the interest. In other

words, the main problem here, is computing minimum distances between each point of the plane and a given subset of it. This operator has many applications in *Computer Vision, Shape Analysis, Pattern Recognition, Computational Geometry, Robot Navigation, Image Registration, Image Enhancement, Image Segmentation, Medical Image Analysis* and many other areas. The output image of this operator is a grayscale image and it is very similar to the input image. The only difference is that foreground pixels have a range of grayscale intensities to show their minimum distance from the regions of the interest[36, 73].

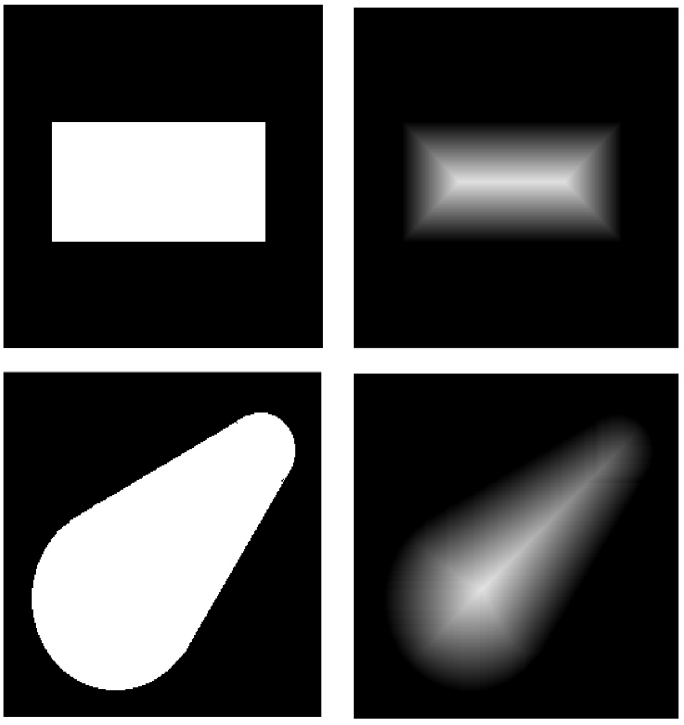


FIGURE A.15: Effect of applying Distance Transform on two simple binary images[74]

According to the definition, there are two main concepts which are central for Distance Transforms. The first one is the *Region of the Interest* and the other is *Distance Metric*.

Region of Interest of Distance Transforms are usually defined as regions with *obstacle pixels*. For a binary image, for instance, the Region of Interest for foreground pixels is background area and vice versa.

Distance Metric is a tool to measure the distance between the pixels in the image. There are different definitions for *pixel distance* in literature and as a result, different Distance Metrics exist. The three most common Distance Metrics are explained here:

Assume we have pixels $P_1$ and $P_2$ at cooradinates $x_1, y_1$ and $x_2, y_2$, then we have:

### Euclidean Distance

$$D_{Euclid} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{A.31}$$

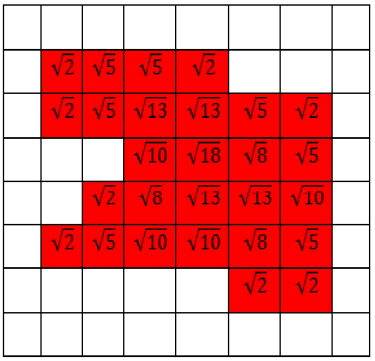



FIGURE A.16: Euclidean Distance Transform for a sample input image

### City Block(Manhattan) Distance

$$D_{City} = |x_2 - x_1| + |y_2 - y_1| \tag{A.32}$$

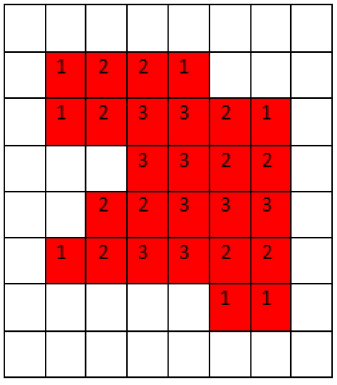



FIGURE A.17: City Block Distance Transform for a sample input image

### Chessboard Distance

$$D_{Chess} = max(|x_2 - x_1|, |y_2 - y_1|) \tag{A.33}$$

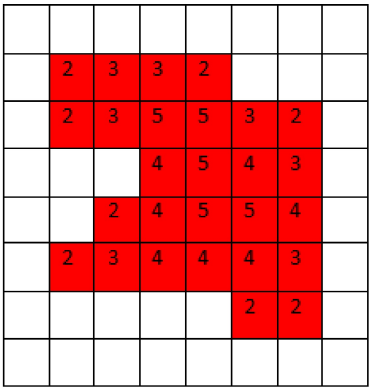



FIGURE A.18: Chessboard Distance Transform for a sample input image

The difference between these three types of the Distance Metrics is their viewpoint about pixel grid. Euclidean Distance sees two pixels like two lines and then calculates their distance with well known line distance formula. City Block distance assumes that when going from one pixel to another, you can just go vertically and horizontally while Chessboard Distance also allows diagonal movements.

Generally speaking, Euclidean Distance metric is usually more accurate than other types and it is the adequate model to numerous geometrical facts of the human scale world[75]. However, City Block and Chessboard Distance Metrics are usually faster to compute and easier to manipulate.

Now we can redefine the Distance Transform using meaning of Region of Interest and Distance Metric in a more comprehensive way: Distance Transform is an operator which labels each pixel of the image with the distance to its nearest obstacle pixel.

# Appendix B

# Developed Software

This guide describes how to install and get started using *INsPECT* software to analyze macrophage infection ratio of Leishmaniasis parasites.

The following topics are covered:

- Software Introduction
- Requirements
- Input Images' Format and Protocols
- Software Setup and Features

## B.1 Introduction

*INsPECT* is a public domain Java-based Image Processing and analysis framework developed by *Ehsan Yazdanparast*. It runs as a Java application, on any computer with Java 1.6 or later installed on it. Application is available for *Windows*, *Mac OS X* and *Linux*.

The main functionality of application is to process *DAPI* and *Phase Contrast/DIC* image sets(or *DAPI* image sets individually) of *Leishmaniasis* infected cells acquired in laboratory conditions and to generate visual and text results. Furthermore basic and common image processing operators are embedded in the software as an image viewer package.

*INsPECT* is being developed on *Windows* platform and all parts of the software are coded by the author using *Java* programming language.

## B.2 Requirements

Before running the software, you make sure that your device has the minimum requirements for running this software.

TABLE B.1: minimum requirements for running the INsPECT software

| Resource | Value |
|---|---|
| Memory(RAM) | About 1GB |
| Hard Disk space | having high quality images and running in Full-Option mode: maximum 30MB for each input image |
| Processor | any x86 or x64 |
| Operating System | Windows, Mac OS, Linux |
| User Permissions | Any user can run INsPECT |

**Notice:** Due to high load of processes and to prevent software crash, it is highly recommended to run the program in devices with more RAM space.

- Try to close or idle ongoing processes while running the software to have timely results.
- In whatever platform, if you have problem with opening the Jar file or if the system could not assign enough memory for the program heap space, you can also try to run the program from the command line terminal. When you open the terminal, you should first go to the directory in which Jar file exists and then type this command:

  **Java –Xmx<maxHeapSpace>m –Xms<minHeapSpace>m -jar <jarfilename>.jar**

  For instance, the following command:

  **Java – Xmx2000m –Xms128m –jar INsPECT.jar**

  Will run the program considering that the name of jar file is INsPECT.jar and minimum and maximum dedicated heap space to the software will be 128 megabytes and 2 giga bytes, respectively.

## B.3 Input Images' Format and Protocols

To feed the software with input, you need to put *DAPI* and *Light Microscopic Images(DIC or Phase Contrast)* or alternatively just *DAPI* images in *Input Folder*. To run the software properly, you should follow these instructions before putting images in *Input Folder*:

- Images should be in **8-bits** depth. You can simply do this using *ImageJ* software under the menu **image -> type -> 8-bits**
- *DAPI* images should be in **bright background**. If not, try **edit -> invert** on *ImageJ*.
- In case, processes take place for pair images, each set of *DAPI* and *Light Microscopic Images(DIC or Phase Contrast)* should be placed exactly after each other. Furthermore, for each pair, *DAPI* image should be placed before corresponding *DIC or Phase Contrast* image(see Fig. B.1).

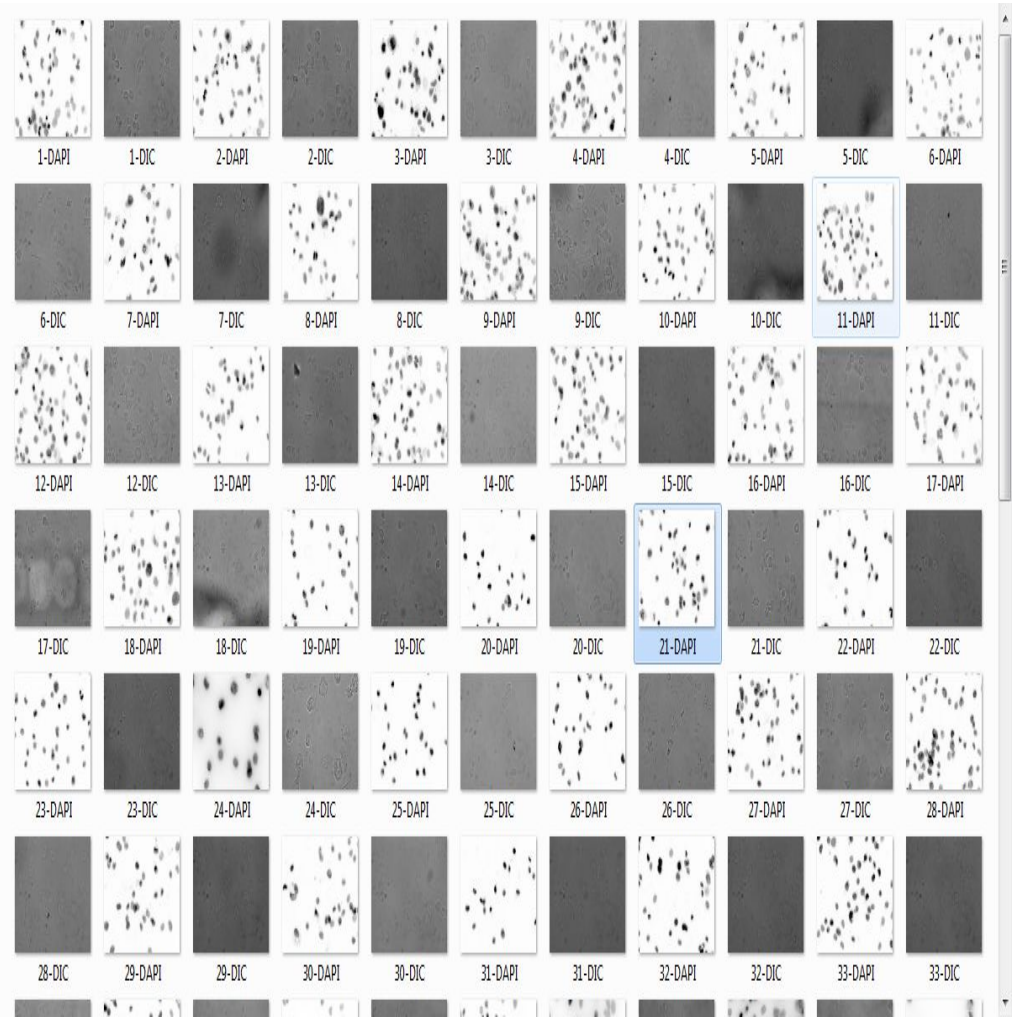



FIGURE B.1: Right order of putting images in Input Folder

## B.4 Software's General Flow

When you run the software, the *Main Panel* will be opened where you can access to options with both menu items and toolbar icons(see Fig. B.2).


FIGURE B.2: Main Panel of INsPECT

First of all, an *Image Viewer* with basic Image Manipulation operators are embedded in the software. These functionalities could be accessed through the main frame of the software. Options are listed below:

- Open and Close images
- Zoom In/Zoom Out open image
- Invert open Image

To process input images and generate results and reports, two scenarios are embedded in the software(see Fig. B.3):

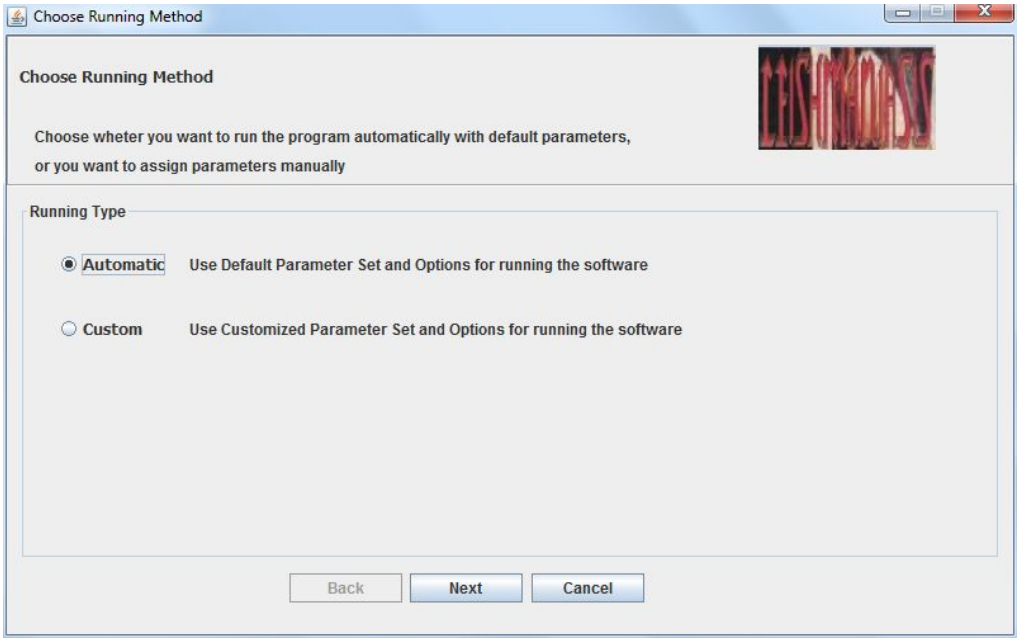



FIGURE B.3: Running Method Selection Panel: Users can choose Custom and Automatic methods to run the software

1. **Automatic** running method in which pipeline parameters are assigned to input images automatically. In fact, according to experimental and analytical results, default values for each parameter are set in this method.

2. **Custom** running method which allows the user to specify his/her intended parameters, step by step. If this option is chosen by the user, cell, parasite and cytoplasm pipeline parameters are taken in three steps from the user and then program will go to *Ready to Run* state.

Users are recommended to first use the software in *Automatic* mode, to test it and if the results are not satisfactory enough for them, they can then switch to *Custom* mode and feed the software with their parameters. *Automatic* mode's results could also be a very promising way for users to gain ideas regarding parameter ranges.

Using both scenarios, users also should specify some *Running Parameters* through a panel(see Fig. B.4 ) as follows:

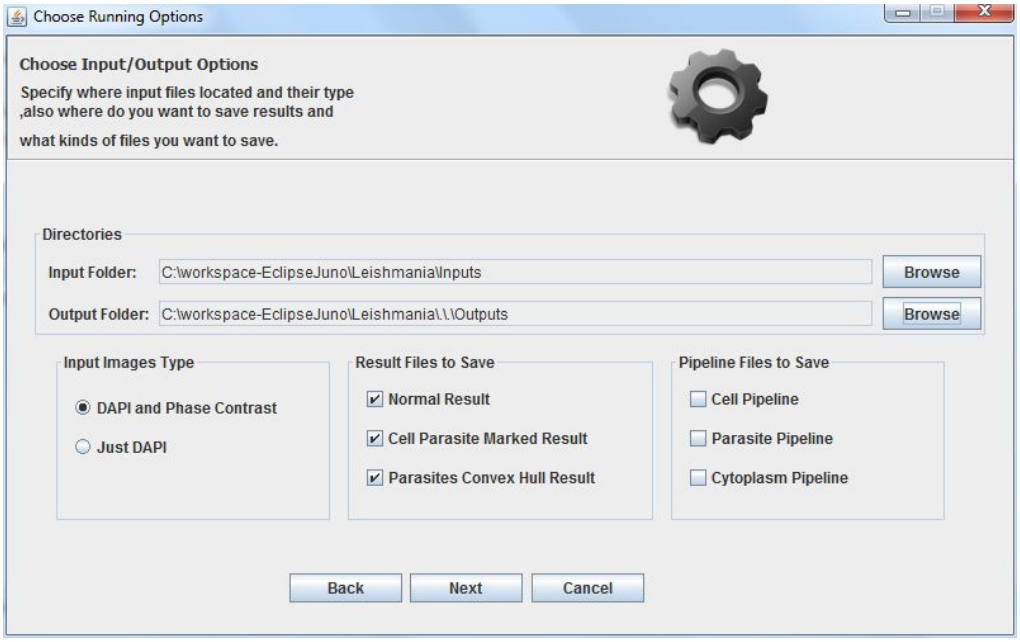



FIGURE B.4: Running Options Panel: Users should specify Running Options for both Automatic and Custom scenarios

1. **Directories**: *Input Folder* which contains input image set and *Output Folder* to save the results and reports. Notice that all results, containing images and reports will be saved under *Output Folder*.

2. **Input Images' Type**: the software comes with two options for processing input image set. If you choose *DAPI and Phase Contrast* option, then you should

put pairs of such images in *Input Folder*. Alternatively, with choosing *Just DAPI* option, software will ignore Cytoplasm Regions related processing from *Light Microscopy* images and concentrate on processing input *DAPI* images.

3. **Results Files to Save**: *Normal Result* is the processed image with markers and regions' boundaries. *Cell Parasite Marked Result* is similar to *Normal Result*, except that cells and parasites are labeled using numbers in such images. Finally, *Parasites' Convex Hull Result* images show convex hull of related parasites for each cell. If checked, any of those results will be saved under *Output Folder* and in relevant sub directory for related input image(s).

4. **Pipeline Files to Save**: each checkbox, namely *Cell Pipeline*, *Parasite Pipeline* and *Cytoplasm Pipeline*, if checked, then related output images for that pipeline will be saved under *Output Folder* and in relevant sub directory for related input image(s). Notice that, if input images' type is chosen to be *Just DAPI* , then *Cytoplasm Pipeline* checkbox will be disabled since we do not have Phase *Contrast* or *DIC* images, anymore.

If you use the software in *Automatic* mode, after choosing above running options, the program is ready to run for you. And you can run the software in the next panel by clicking *Run* button(see Fig.B.5 and Fig. B.6). However, if you choose *Custom* mode, then you need to specify parameters manually through three upcoming panels, called *Cell Pipeline Running Parameters*, *Parasite Pipeline Running Parameters* and *Cytoplasm Pipeline Running Parameters*.

Before explaining in detail parameter setting of these three panels, one important note should be mentioned. *INsPECT* software's target users community is defined to be the biologists who work on *Leishmaniasis* disease. It goes without any saying that such communities probably do not have enough *Image Processing* background. Therefore, parameter adjustment for them could be complicated without having any sense about the concepts behind such values. As a result, for all needed parameters, more meaningful and touchable synonyms are defined and also numeric values are substituted with labels such as *Low*, *Medium* and *High*. Now we discuss the best way to set needed parameters for each of components of interest in input images(cells, parasites and cytoplasms).

## B.5 Cell Pipeline Parameters Setting

In *Cell Pipeline Running Parameters* panel, you can adjust parameters to detect cell portions in *DAPI* input images(see Fig. B.7). It includes two subpanels, called *Parameters* and *Options*.

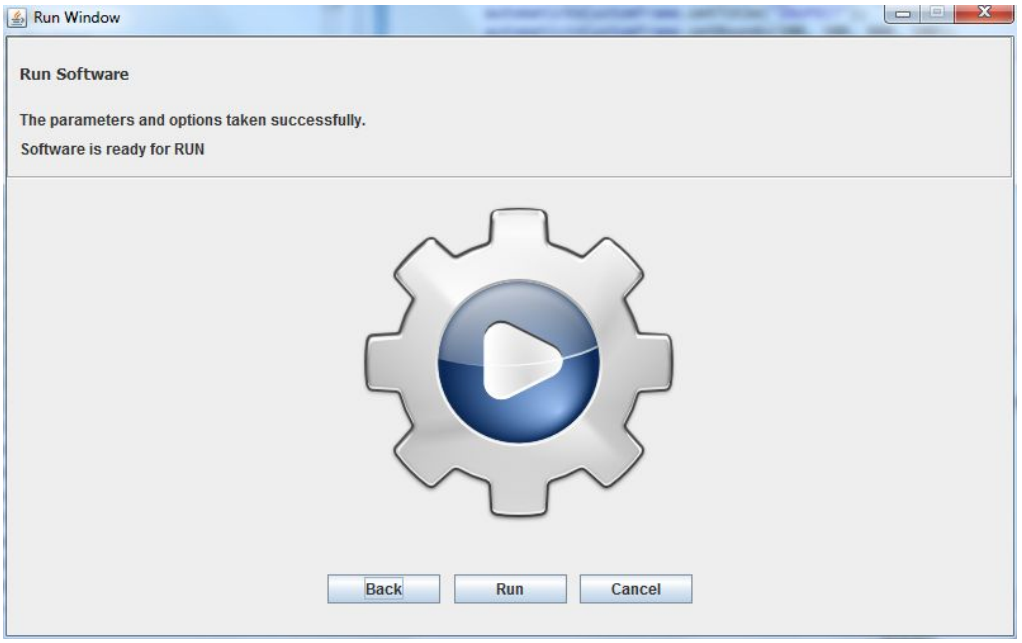



FIGURE B.5: Ready to Run Panel: After taking parameters and options(manually or automatically), by clicking on Run button, software starts to process the input images

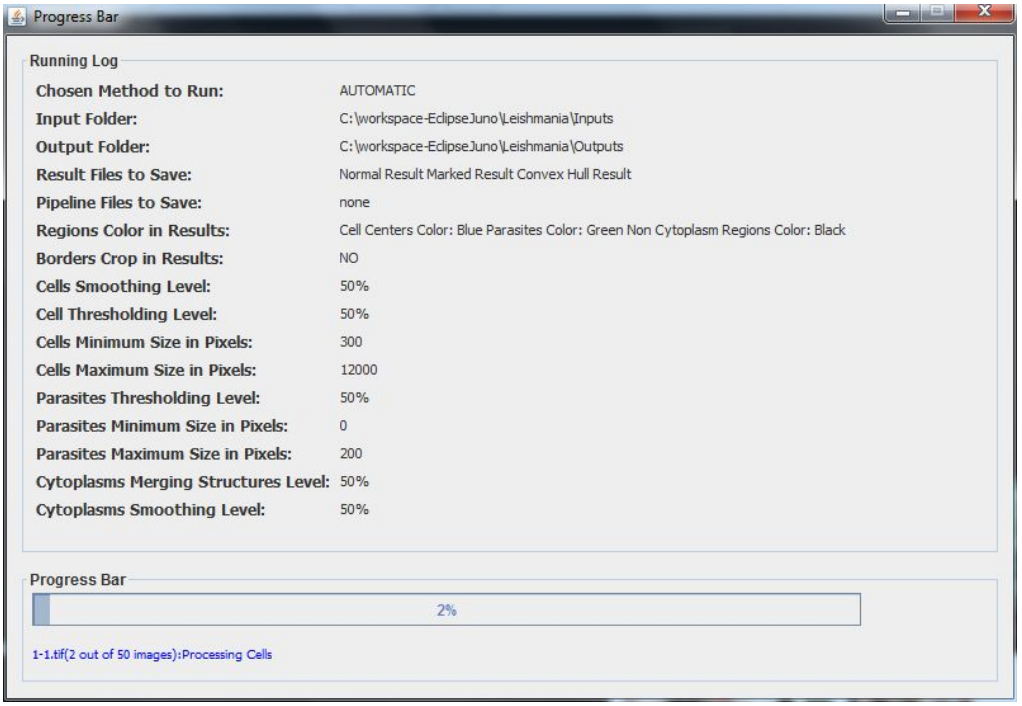



FIGURE B.6: Runtime Log Panel: During Process time, users can trace flow of running software in this panel

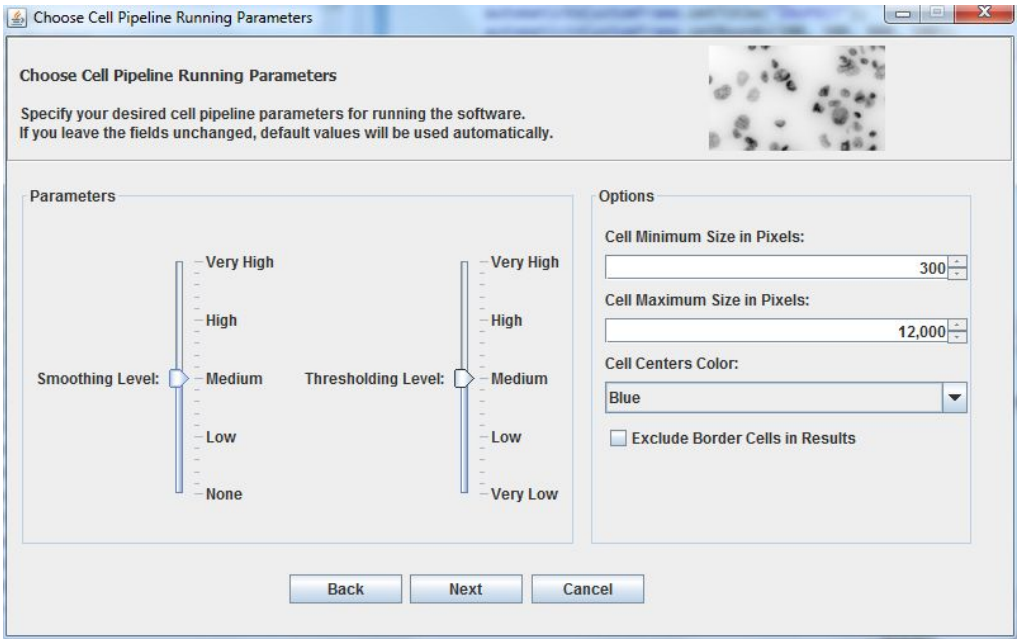



FIGURE B.7: Cell Pipeline Running Parameters Panel

### B.5.1 Parameters subpanel

the parameters associated with *Cell Pipeline* could be customized here.

- **Smoothing Level**: in terms of *Image Processing* operators it is equivalent to *Median Kernel Radius*. It is the main parameter to tell the algorithm how much it should smooth and denoise *DAPI* input images. The default value is set to numeric value 15(which is labeled as *Medium*) and is logical when the quality of input *DAPI* image is reasonable and does not contain much noise. However, if quality of image is even more or equivalently, if you feel that cells have very clear boundaries and noise does not exist across the image, *Smoothing Level* could be decreased. On the other hand, if cells are hard to see and detect and if noise is everywhere in the image, you can increase this value. *None* label is equivalent to 0 numeric value and *Very High* label is equivalent with numeric value 35.

**Notice**: if you have bad quality *DAPI* images and then use *High Smoothing Level*s, it is likely that algorithm overestimates the volume of the cells. For such images, having *Low Smoothing Level*s will also lead to misdetection of cell portions because of the existence of noise and other by products.

- **Thresholding Level**: in terms of *Image Processing* operators it is equivalent to *Mean Adaptive Threshold Kernel Radius*. This parameter is used locally in

subimages to make the binarizion of cell portions and background area. The default value is set to 70(which is labeled as *Medium*) and is logical when the density of cell portions in images is not too high and distribution of cells are not very unbiased across the image. In fact, to have better usage of such threshold, we need to have subimages with relatively considerable amount of both foreground(cell portions) and background areas. Therefore, if you encounter with some *DAPI* images which have very dense cell distribution(majority of dark points), you can increase *Thresholding Level* to higher degrees. On the other hand, if in input *DAPI* images, cells are very small portions with normal distribution(majority of bright points), you should decrease *Thresholding Level* to overcome unbiasness of foreground and background pixels. *Very Low* label is equivalent to 1 numeric value and *Very High* label is equivalent with numeric value 140.

### B.5.2 Options subpanel

the options associated with *Cell Pipeline* could be customized here.

- **Cell Minimum(Maximum) Size in Pixels**: these two parameters help algorithm to filter out those detected cell portions in final result which do not have sizes between *Minimum* and *Maximum*. The default values for minimum and maximum size is set to 300 and 12000 pixels, respectively.

**Notice**: Sometimes high density of parasites exist around some cells and denoising them would lead to miss some valuable border data of cells in final results. However, such kind of misdetected portions could be ignored by setting *Cell Minimum Size in Pixels* properly. Experiments with available image sets in hand showed that size 300 is a suitable choice to filter out those regions while preserving all other valid spots.

- **Cell Centers Color**: you can choose in which color you want to see the cells' centers' markers in final output.
- **Exclude Border Cells in Results**: there might be some cell scraps across border of some *DAPI* images. Such portions are incomplete cells which could not be totally placed inside the images. Checking this option will allow the algorithm to ignore such portions in final visualized results and also in calculations. It should also be mentioned that eliminating such cells will lead to eliminating their relevant parasites, too.

## B.6 Parasite Pipeline Parameters Setting

In *Parasite Pipeline Running Parameters* panel, you can adjust parameters for extracting parasites in *DAPI* input images(see Fig. B.8). It includes two subpanels, called *Parameters* and *Options*.

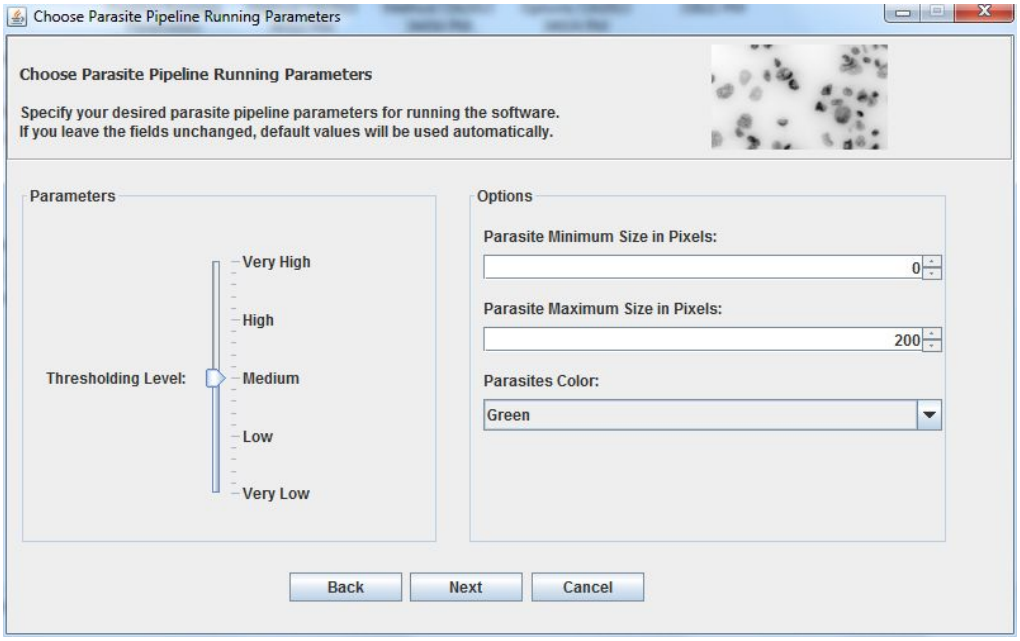



FIGURE B.8: Parasite Pipeline Running Parameters Panel

### B.6.1 Parameters subpanel

the parameters associated with *Parasite Pipeline* could be customized here.

- **Thresholding Level**: in *Image Processing* terms, this parameter is binded to *Structuring Element Size* of *Black Top Hat Transform* operator. The size of the *Structuring Element* is highly dependent on nature of image structures which we want to analyze and also on what one wants to keep or suppress. Here, the components of the interest for suppressing are parasites. Therefore, increasing or decreasing the value of *Thresholding Level* will enable algorithm to suppress those components more or less. The default value is set to 2(which is labeled as *Medium*) and works quite fine for all *DAPI* images with standard *Average Parasite Size*. By standard *Average Parasite Size*, we mean those parasites which occupy very tiny portions of the images and are much smaller than cells. However, for any reason, if parasites are bigger(or smaller) than normal, then you could increase(or decrease) *Thresholding Level*. In this case, *Structuring Element* will be bigger(or smaller)

and as a result, it will suppress bigger(or smaller) candidate areas for parasites. Finally, *Very Low* label is equivalent to 1 numeric value and *Very High* label is equivalent with numeric value 4.

**Notice**: Decreasing *Thresholding Level* is not recommended since using this option, often leads to overdetection of parasites. However, if users are aware of the fact that in reality, such density of parasites exist around cell portions, they can use this option too.

### B.6.2 Options subpanel

the options associated with *Parasite Pipeline* could be customized here.

- **Parasite Minimum(Maximum) Size in Pixels**: these two parameters help the algorithm to filter out those detected parasites which do not meet requirements of size range. The default size values for *Minimum* and *Maximum* is set to 0 and 100 pixels, respectively.

**Notice**: in some images, there may exist detected parasites with very small sizes, namely 1 to 10 pixels. It is recommended for users not to alter minimum size for parasites unless they are confident that such tiny details are not parasites and could be any other by products.

- **Parasites Color**: you can choose in which color you want to see the parasites markers in the final output.

## B.7 Cytoplasm Pipeline Parameters Setting

In *Cytoplasm Pipeline Running Parameters* panel, you can adjust parameters to extract cytoplasm traces from *Phase Contrast* or *DIC* input images(see Fig. B.9). It includes two subpanels, called *Parameters* and *Options*.

**Notice**: If you have already chosen the option *Just DAPI* to run the software, then this panel will not be shown to you since processing *Phase Contrast* or *DIC* input images are no longer the concern.

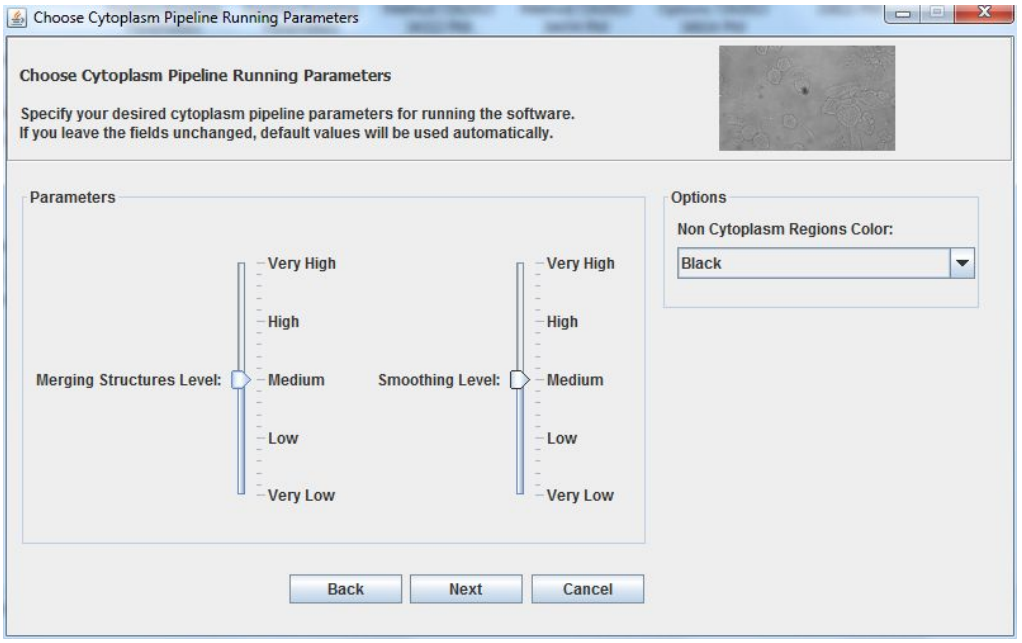



FIGURE B.9: Cytoplasm Pipeline Running Parameters Panel

### B.7.1 Parameters subpanel

The parameters associated with *Cytoplasm Pipeline* could be customized here.

- **Merging Structures Level**: in terms of *Image Processing* operators, this parameter is equivalent to *Structuring Element Size* of *Morphological Closing* operator. This operator is used to merge and fill the structures of detected cytoplasms after smoothing and thresholding. The default value is set to 7(which is labeled as *Medium*) and is acceptable when *Phase Contrast* or *DIC* input image is not suffering from very high illumination variance(so the thresholded cytoplasm traces need to be merged more). However, if the nature of such images is such that cytoplasm traces cannot be visualized(with human eye) and detected in a straightforward manner, then increasing *Merging Structures Level* value could help to at least find merged portions for cytoplasms. On the other hand, if quality of *DIC* or *Phase Contrast* images are good and traces are visible and easily detectable, then decreasing *Merging Structures Level* value helps the algorithm to find more exact traces around cells. *Very Low* label is equivalent to 1 numeric value and *Very High* label is equivalent with a numeric value of 14.

**Notice**: The nature of *DIC* or *Phase Contrast* input images are in such a way that contain a high illumination variance. This factor affects *Image Processing* operators, including *Thresholding* methods, significantly and makes extracting cytoplasm traces'

complicated. To compensate such effect, we use *Merging Structures Level* parameter to have at least more smooth and connected regions. Therefore, it should be noticed that altering this value could sacrifice exact detection of some cytoplasm regions.

- **Smoothing Level**: in terms of *Image Processing* operators, this parameter is equivalent to *Median Filter Kernel Radius* and it is used to smooth the *Morphological Closing* result, fill small holes and discard small detected portions. The default value is set to 50(which is labeled as *Medium*). This parameter also tries to compensate the effects of bad quality of cytoplasm traces and its task is more or less similar to *Merging Structures Level* parameter. Therefore, the explanations for changing its value is similar to those mentioned for *Merging Structures Level* parameter. *Very Low* label is equivalent to 1 numeric value and *Very High* label is equivalent with a numeric value 50.

### B.7.2 Options subpanel

the options associated with *Cytoplasm Pipeline* could be customized here.

- **Non Cytoplasm Regions Color**: you can choose in which color you want to see the non cytoplasm regions in the final output. The regions encapsulated inside cytoplasm regions will be shown in original image in results.

## B.8 Outputs

Upon completion of the running of the software, visual and text results will be saved under *Output Folder* directory. Visual outputs are already explained in section B.4. In each run, software also generates two report files, called *report* and *cellParasitesReport*.

The *report* file contains general outcomes of processing each pair(or individual). *FileName*, T*otal Number of Cells,Number of Infected Cells*, *Total Number of Parasites*, *Total Number of Intra-Cellular Parasites*, *Percentage of Infected Cells*, *The Mean Number of Parasites per Cell* and *Parasitic Index(Percentage of Infected Cells * Mean Number of Parasites Per Cell)* are the parameters recorded in this file for each image pair(or individual).

The *cellParasitesReport* keeps the record of each cell's features in image pairs(or individuals). You can find these fields in this report: *FileName*, *Cell Number*, *Cell Area*, *Cell*

*Volume*, *Total Number of Parasites*, *Number of Intra Cellular Parasites* and *Number Of Extra Cellular Parasites*.

**Notice**: *cellParasitesReport* could be helpful when one wants to investigate individual cells. On the other hand, *report* file is useful for generic analysis.

# Bibliography


[1] Perreault S and Hébert P. Median filtering in constant time. *IEEE Trans Image Process*, 16(9), 2007.

[2] Ajay Narayanan. Fast binary dilation/erosion algorithm using kernel subdivision. *Computer Vision – ACCV*, 3852, 2006.

[3] World Health Organization. Leishmaniasis. URL http://www.who.int/leishmaniasis/en/.

[4] National Institute of Allergy and Infectious Diseases. Leishmaniasis life cycle, September 2008. URL http://www.niaid.nih.gov/topics/leishmaniasis/pages/lifecycle.aspx.

[5] William D. James, Timothy G. Berger, and Dirk M. Elston. *Andrews' Diseases of the Skin Clinical Dermatology*, pages 422–428. W B Saunders Co, 2005. ISBN 0-7216-2921-0.

[6] Desjeux P. The increase of risk factors for leishmaniasis worldwide. *Transactions of the Royal Society of Tropical Medicine and Hygiene*, 95(3):239–243, 2001.

[7] de Oliveira Cl, Nascimento IP, Barral A, Soto M, and Barral-Netto M. Challenges and perspectives in vaccination against leishmaniasis. *Parasitol Int.*, 58(4):319–324, 2009.

[8] Alvar J, Croft S, and Olliaro P. Chemoterapy in the treatment and control of leishmaniasis. *Adv Parasitol*, 61:223–274, 2006.

[9] Chappius F, Sundar S, Hailu A, Ghalib H, and Rijal S. Visceral leishmaniasis:what are the needs for diagnosis, treatment and control? *Nat Rev Microbiol*, 5:873–882, 2007.

[10] Croft SL, Sundar S, and Fairlamb AH. Drug resistance in leishmaniasis. *Clin Microbiol Rev*, 19:111–126, 2006.

[11] U.S. Food and Drug Administration Section 210. URL http://www.fda.gov/RegulatoryInformation/Legislation/FederalFoodDrugandCosmeticActFDCAct/default.htm.

[12] Directive 2004/27/ec of the european parliament and of the council of 31 march 2004 amending directive 2001/83/ec on the community code relating to medicinal products for human use., March 2004. URL http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32004L0027:EN:HTML.

[13] Anthony J Swinney DC. How were new medicines discovered? *Nat Rev Drug Discov*, 10(7):507–519, 2011.

[14] Steven A. Haney. *High Content Screening: Science, Techniques and Applications*. New York: Wiley-interscience, 2008. ISBN 0-470-03999-X.

[15] Kenneth A. Giuliano D. Lansing Taylor, Jeffrey R. Haskins. *High content screening: a powerful approach to systems cell biology and drug discovery*. Humana Press, 2010. ISBN 1-61737-746-5.

[16] Siqueiro-Neto JL, Song O-R, Oh H, Sohn J-H, and Yang G et al. Antileishmanial high-throughput drug screening reveals drug candidates with new scaffolds. *PLoS Negl Trop Dis*, 4(5), 2010.

[17] Siqueiro-Neto JL, Moon S, Jang J, Yang G, and Lee C et al. An image-based high-content screening assay for compounds targeting intracellular leishmania donovani amastigotes in human macrophages. *PLoS Negl Trop Dis*, 6(6), 2012.

[18] Aulner N, Danckaert A, Rouault-Hardoin E, Desrivot J, and Helynck O et al. High content analysis of primary macrophages hosting proliferating leishmania amastigotes: Application to anti-leishmanial drug discovery. *PLoS Negl Trop Dis*, 7(4), 2013.

[19] Eisenstein M. Quality control. *Nature*, 442(7106):1067–1070, 2006.

[20] Robert Dunkle. Role of image informatics in accelerating drug discovery and development. Technical report, Drug Discovery World, 2003.

[21] Rester U. From virtuality to reality - virtual screening in lead discovery and lead optimization: A medicinal chemistry perspective. *Curr Opin Drug Discov Devel*, 11 (4):559–568, 2008.

[22] Rollinger JM, Stuppner H, and Langer T. Virtual screening for the discovery of bioactive natural products. *Prog Drug Res. Progress in Drug Research*, 65(211): 213–249, 2008.

[23] Kapuscinski J. Dapi: a dna-specific fluorescent probe. *Biotech Histochem*, 70(5): 220–233, 1995.

[24] Richard Fitzgerald. Phase-sensitive x-ray imaging. *Physics Today*, 53(7):23, 2000.

[25] Murphy D. *Differential interference contrast (DIC) microscopy and modulation contrast microscopy, in Fundamentals of Light Microscopy and Digital Imaging*, pages 153–168. New York: Wiley-Liss, 2001.

[26] Hongmei Zhu. Medical image processing overview. Technical report, University of Calgary, 2003.

[27] Ingrid Scholl, Til Aach, Thomas M. Deserno, and Torsten Kuhlen. Challenges of medical image processing. *Computer Science - Research and Development*, 26(1-2): 23, 2011.

[28] C. S. Garbe and B. Ommer. Parameter estimation in image processing and computer vision. *Mathematical and Computational Sciences*, 4, 2013.

[29] GE Healthcare Life Sciences. In cell analyzer. URL http://www.gelifesciences.com/webapp/wcs/stores/servlet/catalog/pt/GELifeSciences-PT/brands/in-cell-analyzer.

[30] P. Patidar, M. Gupta, S. Srivastava, and A. K. Nagawat. Image de-noising by various filters for different noise. *Computer Applications*, 9(4), 2010.

[31] E. Davies. *Machine Vision: Theory, Algorithms and Practicalities*. Academic Press, 1990. ISBN 978-0123869081.

[32] R. D. Boyle and R. C. Thomas. *Computer Vision: A First Course*. Alfred Waller Ltd, 1988. ISBN 978-0632015771.

[33] V.R.Vijaykumar, P.T.Vanathi, and P.Kanagasabapathy. Fast and efficient algorithm to remove gaussian noise in digital images. *IAENG International Journal of Computer Science*, 37(1), 2010.

[34] National Programme on Technology Enhanced Learning(NPTEL). Digital image processing-noise smoothing. URL http://www.nptel.iitm.ac.in/courses/Webcourse-contents/IIT-KANPUR/Digi_Img_Pro/chapter_8/8_16.html.

[35] Thomas S. Huang, George J. Yang, and Gregory Y. Tang. A fast two dimensional median filtering algorithm. *IEEE Trans on Acoustics, Speech and Signal Processing*, 27(1), 1979.

[36] A. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, 1989. ISBN 978-0133361650.

[37] Wayne Fulton. A few scanning tips. URL http://www.scantips.com/simple6.html.

[38] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. Unsharp filter, 2003. URL http://homepages.inf.ed.ac.uk/rbf/HIPR2/unsharp.htm.

[39] Ole Christian Eidheim. Introduction to mathematical morphology. URL http://www.idi.ntnu.no/emner/tdt4265/lectures/lecture3b.pdf.

[40] Faisal Shafait, Daniel Keysers, and Thomas M. Breuel. Efficient implementation of local adaptive thresholding techniques using integral images. *Document Recognition and Retrieval XV*, 6815, 2008.

[41] J. Sauvola and M. PietikaKinen. Adaptive document image binarization. *Pattern Recognition*, 33, 2000.

[42] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Pub, 1992. ISBN 978-0201508031.

[43] Olson AD and Miller MJ. Elsie 4: quantitative computer analysis of sets of two-dimensional gel electrophoretograms. *Anal. Biochem.*, 169:49–70, 1988.

[44] Appel RD, Vargas JR, Palagi PM, Walther D, and Hochstrasser DF. Melanie ii–a third-generation software package for analysis of two-dimensional electrophoresis images: Ii. algorithms. *Electrophoresis*, 18(15), 1997.

[45] Hamid R. Shahbazkia, Antonio dos Anjos, Anders Laurell Blom Møller, Bjarne K. Ersbøll, and Christine Finnie. New approach for segmentation and quantification of two-dimensional gel electrophoresis images. *Bioinformatics Advance Access*, 27(3), 2011.

[46] Lifeng He, Yuyan Chao, and Kenji Suzuki. A run-based two-scan labeling algorithm. *IEEE TRANSACTIONS ON IMAGE PROCESSING*, 17(5), 2008.

[47] Erik Meijering. Cell segmentation: 50 years down the road. *IEEE Signal Processing Magazine*, 29(5), 2012.

[48] R. Adollah1, M.Y. Mashor, N.F. Mohd Nasir, H. Rosline, H. Mahsin, and H. Adilah. Blood cell image segmentation: A review. *Biomed*, 21, 2008.

[49] L-K Huang and Wang M-J J. Image thresholding by minimizing the measure of fuzziness. *Pattern Recognition*, 28(1), 1995.

[50] Ridler TW and Calvard S. Picture thresholding using an iterative selection method. *IEEE Transactions on Systems, Man and Cybernetics*, 8, 1978.

[51] Li CH and Tam PKS. An iterative algorithm for minimum cross entropy thresholding. *Pattern Recognition Letters*, 18(8), 1998.

[52] Kapur JN, Sahoo PK, and Wong ACK. A new method for gray-level picture thresholding using the entropy of the histogram. *Graphical Models and Image Processing*, 29(3), 1985.

[53] Kittler J and Illingworth J. Minimum error thresholding. *Pattern Recognition*, 19, 1986.

[54] Otsu N. A threshold selection method from gray-level histograms. *IEEE Trans. Sys.*, 9, 1979.

[55] Doyle W. Operation useful for similarity-invariant pattern recognition. *Journal of the Association for Computing Machinery*, 9, 1962.

[56] Bernsen J. Dynamic thresholding of grey-level images. *Proc. of the 8th Int. Conf. on Pattern Recognition*, 1986.

[57] Sauvola J and Pietaksinen M. Adaptive document image binarization. *Pattern Recognition*, 33(2), 2000.

[58] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988. ISBN 9780130222787.

[59] Oleh Dzyubachyk, Wiggert A. van Cappellen, Jeroen Essers, Wiro J. Niessen, and Erik Meijering. Advanced level-set-based cell tracking in time-lapse fluorescence microscopy. *IEEE TRANSACTIONS ON MEDICAL IMAGING*, 29(3), 2010.

[60] Verena Kaynig, Ignacio Arganda-Carreras, and Albert Cardona. Trainable segmentation plugin for fiji. URL http://fiji.sc/Trainable_Segmentation_Plugin.

[61] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. Pixel subtraction, 2003. URL http://homepages.inf.ed.ac.uk/rbf/HIPR2/pixsub.htm.

[62] Pierre Soille. *Morphological Image Analysis: Principles and Applications*. Springer-Verlag New York-Inc. Secaucus, 2003. ISBN 3540429883.

[63] Lasko Laskov. Image processing: mathematical morphology. URL http://www.nbu.bg/PUBLIC/IMAGES/File/departamenti/informatika/6.pdf.

[64] Yuliya Tarabalka and Jón Atli Benediktsson. Mathematical morphology with emphasis on analysis of hyperspectral images and remote sensing applications. URL https://notendur.hi.is/yut2/files/UL_01.pdf.

[65] Edward R. Dougherty. *Introduction to Morphological Image Processing*. SPIE-International Society for Optical Engine, 1992. ISBN 978-0819408457.

[66] Jos B.T.M. Roerdink and Arnold Meijster. The watershed transform: De nitions, algorithms and parallelization strategies. *Fundamenta Informaticae*, 41, 2001.

[67] Vincent L. and Soille P. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Trans. Patt. Anal. Mach. Intell.*, 13(6), 1991.

[68] Sarbjeet Singh. Edge detection. URL http://www.slideshare.net/contactsarbjeet/image-processing-edge-detection.

[69] University Cape Town. Basic image processing: edge detection. URL http://www.dip.ee.uct.ac.za/~nicolls/lectures/eee401f/03_edgedet_notes_2up.pdf.

[70] Raman Maini and Himanshu Aggarwal. Study and comparison of various image edge detection techniques. *International journal of image processing*, 3(1), 2011.

[71] Rice University. Digital image processing-other methods of edge detection. URL http://www.owlnet.rice.edu/~elec539/Projects97/morphjrks/moredge.html.

[72] Gabriel Landini. Clever binary fill algorithm. URL http://imagej.nih.gov/ij/source/ij/plugin/filter/Binary.java.

[73] A. Rosenfeld and J. Pfaltz. Distance functions in digital pictures. *Pattern Recognition*, 1:33–61, 1968.

[74] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. Distance transform, 2003. URL http://homepages.inf.ed.ac.uk/rbf/HIPR2/distance.htm.

[75] R. Fabbri, L. DA F. Costa, J. C. Torelli, and O. M. Bruno. 2d euclidean distance transform algorithms: A comparative survey. *ACM Computing Surveys*, 40(1), 2008.