YAŞAR UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING
COMP 4360 IMAGE PROCESSING

Assignment 1: Object Counting Report

Doğa Pirci

Contents

1. Introduction 3

2. Methodology 3

3. Approach 4

3.1. Gaussian Blur 4

3.2 Contrast Stretching 4

3.3 Gamma Correction 4

3.4 Thresholding (Otsu + Logical NOT) 4

3.5 Morphological Opening (7×7) 5

3.6 Flood Fill 5

3.7 Large Opening (20×20) 6

4. Results 6

5. Discussion 7

6. References 8

1. Introduction

This project presents a workflow for cell detection and counting in microscopy images. The method uses image preprocessing, thresholding, and morphological operations to prepare the data. Connected components analysis is then applied to identify and count individual cells. The aim is to create a clear and reproducible process that reduces noise and improves accuracy.

2. Methodology

The image processing workflow was designed to prepare the cells.png image for accurate cell detection and counting by improving contrast, reducing noise, and isolating cells from the background. First, **contrast stretching** was applied to expand the intensity range of the image so that it spans the full dynamic range (0–255), enhancing visibility of cell structures. Next, **gamma correction** with a value of $\gamma = 1/2.2$was used to adjust brightness and improve separation between cells and background. To reduce small-scale noise and smooth edges, a **Gaussian blur** with a 7×7kernel was applied, which facilitates more stable thresholding. Then, **Otsu's thresholding** was performed to create a binary mask, followed by a logical NOT operation to invert the mask so that cells appear white against a black background. To remove small noise and debris, **morphological opening** with a 7×7rectangular kernel was used, ensuring cleaner segmentation. Internal gaps within cells were filled using a **flood-fill operation**, guaranteeing that each cell is represented as a solid region. Finally, a **large morphological opening** with a 20×20kernel was applied to eliminate larger unwanted particles such as dust or non-cell objects, balancing noise removal with preservation of cell shapes. This step-by-step approach ensures that the processed image is optimized for reliable cell counting.

Why the Code Was Split into Three Files

a) **image_process.py:** Contains all preprocessing and morphological functions. This separation makes the workflow modular and easy to adjust parameters.

b) **visualize.py:** Responsible for visualization and counting. It applies connected components analysis, draws bounding boxes, and labels cells. Keeping visualization separate ensures clarity and reusability.

c) **main.py:** Controls the pipeline. It calls functions from both other files, logs results, and organizes the step-by-step execution. This separation makes the project easier to maintain and reproduce.

3. Approach

3.1. Gaussian Blur

A Gaussian filter with a kernel size of (7×7) was applied to reduce high-frequency noise. As shown in Figure 1 this helps smooth the image and prevents small particles from being misclassified as cells during thresholding.

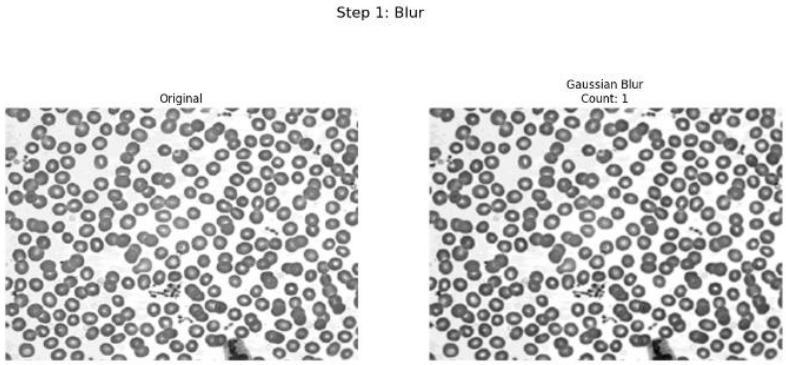

Figure 1: Original +Gaussian Blur

3.2 Contrast Stretching

Contrast stretching expands the intensity range of the image. This improves visibility of cell boundaries, especially in low-contrast regions. It ensures that cells stand out more clearly against the background.

3.3 Gamma Correction

Gamma correction adjusts the brightness non-linearly. A gamma value of 1/2.2 was chosen to brighten mid-tones without overexposing bright regions as shown in Figure 2. This improves separation between cells and background before thresholding.

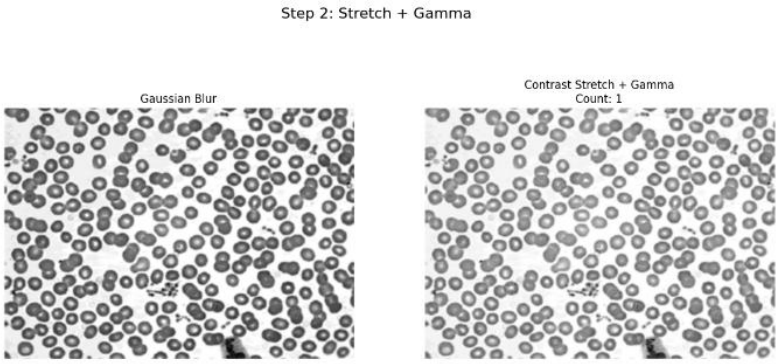

Figure 2: Stretch +Gama

3.4 Thresholding (Otsu + Logical NOT)

Otsu's method automatically selects a threshold value to binarize the image. Logical NOT is applied so that cells appear white and background black, which is required for later morphological operations.

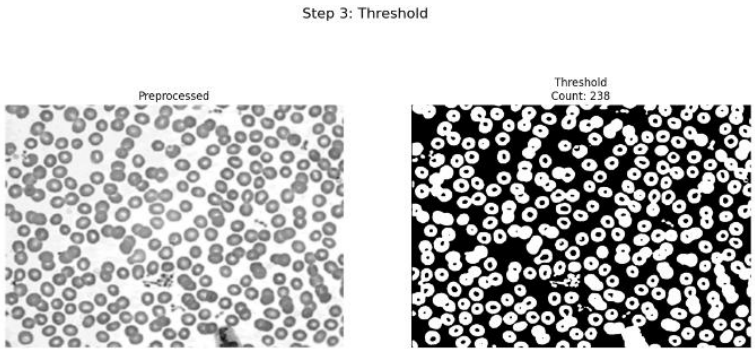

Figure 3: After Threshold output with Preprocess

3.5 Morphological Opening (7×7)

This step removes small noise and debris using a rectangular kernel. It preserves cell shapes while eliminating isolated white pixels that could be miscounted. As shown in Figure 4 this helped to remove the dust and non-cell elements in the picture.Although it did not eliminate all the particles it can be seen that some which affects the count critically.

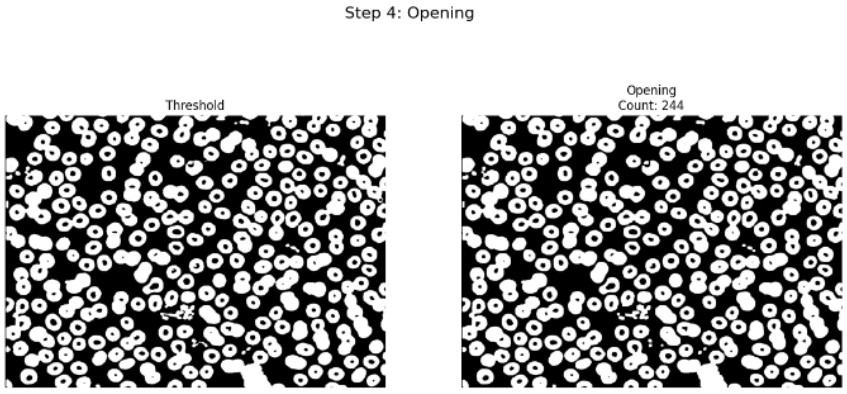

Figure 4: Threshold +Opening

3.6 Flood Fill

Flood fill fills internal holes within detected cell regions. This ensures that each cell is a solid object, improving the accuracy of connected components analysis.

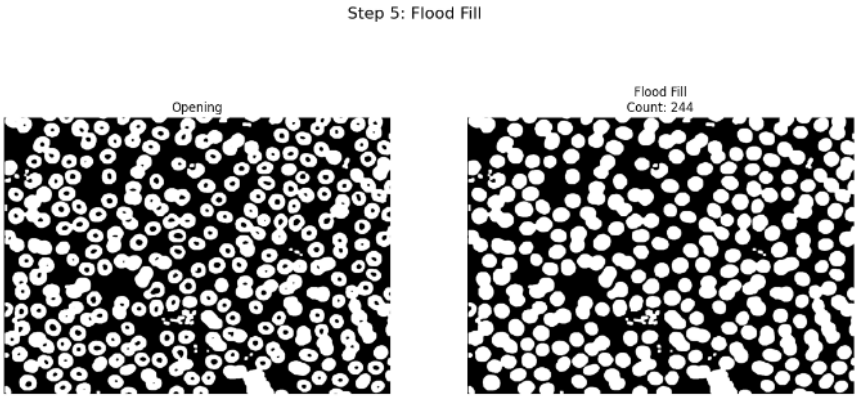

Figure 5: Opening + Flood Fill

3.7 Large Opening (20×20)

A larger kernel is used to remove bigger unwanted particles such as dust or artifacts. This step is sensitive: if the kernel is too large, small cells may be removed. The chosen size balances noise removal and cell preservation.

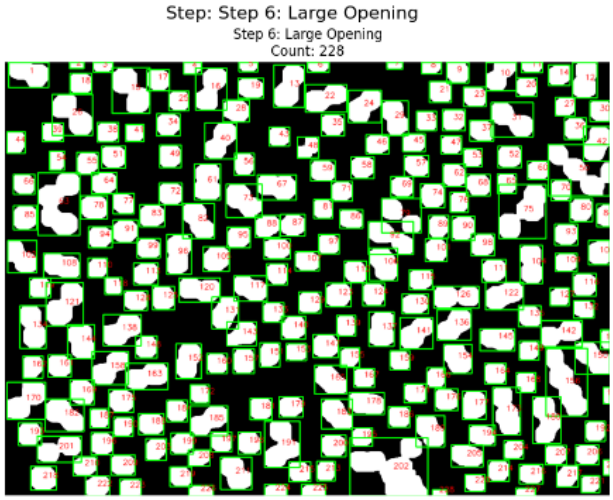

Figure 6: Final Output

4. Results

```
--- Cell Count Pipeline Started ---

Processes implemented                                    Cell Count
-------------------------------------------------------------------
Gaussian Blur                                                 1
Contrast Stretch + Gamma                                      1
Threshold                                                   238
Opening                                                     244
Flood Fill                                                  244
Large Opening                                               228

--- Pipeline Finished ---
```

The initial preprocessing steps, **Gaussian Blur** and **Contrast Stretch + Gamma**, resulted in a cell count of only 1, which is expected since these operations enhance image quality without segmentation. After **Otsu's Thresholding**, the count increased to 238 as cells were separated from the background, though some noise was also detected. Applying **Morphological Opening (7×7)** slightly raised the count to 244 because small noise was removed and some touching cells were split. Following **Flood Fill**, the count remained 244, as this step only filled internal holes without changing object numbers. Finally, **Large Opening (20×20)** reduced the count to 228 by removing large unwanted particles and merging small fragments, which decreased false positives while preserving actual cell shapes. These changes are consistent with the intended effects of each processing step.

5. Discussion

The proposed pipeline successfully enhanced image contrast, reduced noise, and produced a clean binary mask for cell counting. Most steps worked as intended, but several challenges and parameter sensitivities were observed. One limitation was the difficulty in removing large non-cell particles without affecting actual cells. When the morphological opening kernel size exceeded 7×7, some cells were eroded or completely removed, leading to a noticeable drop in cell count. To address this, a smaller opening was applied before flood fill, followed by a larger opening (20×20) after hole filling to remove remaining large artifacts while preserving cell integrity.Gamma correction also showed strong sensitivity. Tests with γvalues between 0.4 and 2.2 revealed that very low values (e.g., 0.4) produced poor separation and resulted in undercounting (around 225 cells), while values slightly above 1 (e.g., 1.2–1.5) gave more accurate results. However, beyond 1.7, performance degraded again, indicating an optimal range near 1.2–1.5 for this dataset.Another challenge was separating adjacent or overlapping cells. The current pipeline often counted clusters as single cells, reducing accuracy. Advanced segmentation techniques such as **watershed** could improve separation in future work. Histogram equalization was also tested but produced inconsistent results (counts between 110 and 219), likely due to its incompatibility with the chosen preprocessing steps.Overall, the pipeline achieved reliable cell detection under tuned parameters, but its performance is sensitive to kernel sizes and gamma values. Future improvements could include adaptive morphological operations and advanced segmentation methods to handle overlapping cells more effectively.

6. References

https://colab.research.google.com/drive/1eS4952_xnsUxy9tm0MpUvQJKgmmn6v2?usp=sharing#scrollTo=GYx5vNexLcoc