Frugal Approaches to Malaria Screening

Polina SOLOVEVA Tyler MARINO
M2 AI, Computer Vision Course

https://github.com/SolovevaPolina/Malaria_Screening

Abstract

This study presents a Frugal framework for automated malaria screening, designed specifically for low-resource hardware. We compared a classical computer vision pipeline against a lightweight Deep Learning benchmark (ShuffleNet). The classical approach evaluated Marker-controlled Watershed and Circle Hough Transform for segmentation. For the classification stage, we trained different classical ML models using optimized statistical color and texture features. Our analysis revealed that while Watershed excels at shape extraction, the Hough Transform provides a more robust total cell count, which is critical for accurate diagnosis. Consequently, the Hough-SVM pipeline achieved a Mean Parasitemia Error of 2.4%, proving its clinical effectiveness despite segmentation noise. Crucially, this classical system required significantly less memory and processing time than deep learning architectures, validating its suitability for deployment in field laboratories where computational power is limited..

1. Introduction

Malaria remains a critical global health challenge, particularly in low-resource regions. According to the World Health Organization [1], early and accurate diagnosis is essential for effective treatment. The gold standard for diagnosis is the visual examination of thin blood smears under a microscope. Clinicians must identify infected Red Blood Cells (RBCs) and calculate the parasitemia rate (the percentage of infected cells relative to the total number of cells). This manual process is time-consuming, tedious, and prone to human error, especially when analyzing thousands of cells per patient.

To automate this task, researchers have recently applied Deep Learning. For instance, Kassim et al. [2] proposed a state-of-the-art architecture combining U-Net and Faster R-CNN. While highly accurate, such models require significant computational power, which is often unavailable in remote or low-resource medical facilities.

Prior to the Deep Learning era, classical computer vision methods offered computationally efficient solutions for segmentation. Tek et al. [3] demonstrated that the Green channel of RGB images provides the optimal contrast for separating RBCs from the background. To handle the problem of overlapping cells, Das et al. [4] successfully applied the Watershed algorithm, while Di Ruberto et al. [5] utilized morphological operations and granulometry. Although these methods are sometimes less robust to noise than neural networks, they consume significantly less memory and processing power.

Following segmentation classification approaches are necessary to determine first if a blood cell is infected with malaria thus allowing for a classification of parasitemia (ratio of blood cells infected with malaria). Training methods using classical machine learning approaches such as Logistic Regression(LR) [7], Random Forest(RF) [8], and Support Vector Machines(SVM) [9] are lightweight and quite successful with feature engineering. More modern deep learning architectures are developed for lightweight edge tasks such as the shufflenet_v2_x1_0 model [10] which come pre-trained of the shelf with weights from ImageNet1k_v1 for example. The convolutional nature of this deep learning allows one to avoid extracting feature values and lets the model learn directly from the images input.

In this work, we propose a frugal automated pipeline for parasitemia estimation. We revisit classical segmentation techniques and evaluate them against the ground truth from the Kassim et al. dataset [2]. Our goal is to demonstrate that optimized classical methods can achieve a necessary balance between accuracy and efficiency, making them suitable for deployment on mobile devices and low-power laptops in developing countries.

2. Dataset and Preprocessing

2.1. Dataset Specification

This study utilizes the publicly available NIH-NLM-ThinBloodSmearsPf dataset described by Kassim et al. [2]. The data was collected at the Medical College Hospital in Bangladesh. It consists of Giemsa-stained thin blood smear images obtained from 193 patients, including both infected individuals and healthy controls. A smartphone camera attached to a light microscope eyepiece captured the images. The image resolution is 5312 x 2988 pixels.

2.2. Ground Truth Annotations

The dataset is divided into two subsets. The "Point Set" includes images from 160 patients with cells marked by

single coordinates. The "Polygon Set" covers 33 patients where cells are outlined with polygonal boundaries. Both subsets contain labels for Uninfected and Parasitized RBC. In this study, we exclusively utilized the Polygon Set boundary annotations for the evaluation of segmentation accuracy, and labels for the classification accuracy.

2.3. Segmentation Preprocessing Pipeline

Raw microscopy images often contain noise and uneven illumination. Therefore, we applied a standardized preprocessing pipeline. First, we extracted the Green channel from the original RGB images. Previous studies [3] demonstrated that the Green channel offers the highest contrast between erythrocytes and the background. Second, we applied Contrast Limited Adaptive Histogram Equalization (CLAHE). This method corrects lighting variations across the circular region of interest and enhances local details without amplifying noise. This prepared image served as the input for our segmentation experiments.

3. Methodology: Segmentation

3.1. Color Space Selection

To select the best input for segmentation, we compared RGB, HSV, and Lab color spaces. The Green channel of RGB was selected because it provides the highest contrast for cell boundaries. Additionally, we chose to test the Hue channel from HSV because it represents color information independent of brightness. To clean the Hue channel, we created a binary mask from the Value channel to remove flares from the dark background. We then used a bitwise operation to isolate the Hue of the cells. Both the Green and Masked Hue channels were kept as candidates for further testing of segmentation methods (Figure 1).

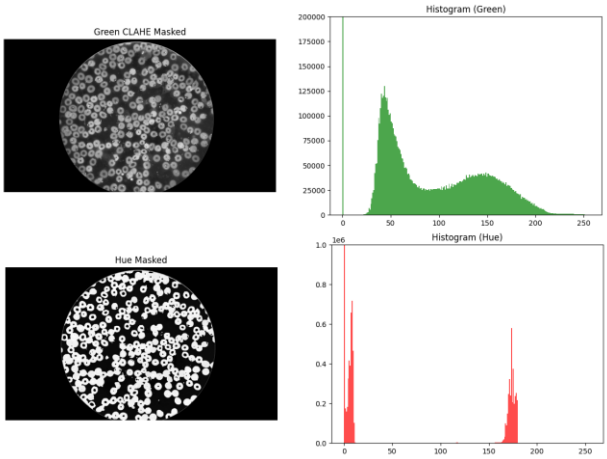

Figure 1. Green and Hue channels example distribution

3.2. Otsu Binarization

We started with Otsu's Binarization as a baseline for global thresholding. This method automatically separates cells from the background by calculating an optimal intensity threshold. We performed a grid search to optimize its performance using three criteria: the smoothing level to reduce camera noise, the cleaning intensity using morphological opening to remove small artifacts, and the minimum artifact size set to filter out debris. The results showed that while Otsu's method effectively finds cell regions, it cannot separate cells that touch each other, which leads to large connected clusters.

3.3. Circle Hough Transform

The Circle Hough Transform was used to detect cells based on their circular geometry (Appendix Figure 3). We tuned five key parameters through a grid search to improve accuracy. We adjusted the edge smoothness to improve boundary detection and set the cell spacing to a minimum of 25-35 pixels between centers to prevent double counting. We also balanced the detection sensitivity to ensure the algorithm catches faint cell boundaries without including background noise. Finally, the cell size limits were constrained to match the actual physical radius of erythrocytes in the microscopy images. To determine the order of parameters, we manually measured the cell diameter, approximately 140-160 pixels. This approach was the most effective method for splitting complex erythrocyte clusters.

3.4. Watershed Optimization

The Watershed algorithm was implemented to separate overlapping cells by treating the image as a topographic surface (Appendix Figure 4). We performed a grid search to optimize the distance mask size for smoother distance calculation and the distance threshold ratio to accurately identify the center markers of each cell. The algorithm calculates the Euclidean Distance Transform to find these markers and then floods the area to find precise boundaries where cells touch. After the initial segmentation, we applied a hole-filling morphological operation to the binary mask. This step is critical because erythrocytes often appear as ring shapes, and filling the centers ensures that no internal cell information is lost before the classification stage.

3.5. Cell Extraction for Classification

For the final classification stage, we extracted individual cells using two different strategies: the Watershed method utilized the binary masks to black out the background and crop by bounding boxes, while the Hough method used square crops with the original background to preserve the

natural morphology around the circular detections. Labels were assigned by checking if ground truth points or polygon centers fell within the segmented areas; if a cluster was not fully separated, it was labeled as parasitized if at least one parasite point was detected inside it.

4. Methodology: Classification

In the effort to test a selection of models across both accuracy and efficiency (time and computation) we ran a series of baselines that allowed for reasonable comparisons of Logistic Regression (LR), Random Forest (RF), Support Vector Machines (SVM), and fine-tuned convolutional Deep Neural Networks (Shuffle net, SN). In the training of the LR, RF, and SN we used a grid search protocol to find the most effective model.

To input into the LR, RF, and SVM models, feature vectors were designed by a comparison of M-means clustering class specific statistics. The feature vector that was chosen was statistics surrounding a 3-means clustering of the image. This included 16 features (clustering values, pixel counts of each cluster, variance between clusters and variance within each cluster). As most malaria infected cells had a portion of the cell that was a more vibrant purple, the k-means clustering would be able to identify this pattern and then the two models would be able to find the differentiating values within the feature vector. It was understood that this was computationally expensive as a feature vector, but it promoted transparency in the learning of these models.

As there were not so many features, we conducted different training runs ranging from [10, 30, 50, 70, 90] percent of the data were performed tested on 5% of the total set of data to determine if overfitting was to happen.

For the training of Logistic Regression, we ran a grid search testing regularization strength ('clf__C') [0.01, 0.1, 1, 10, 100], number of iterations ('clf__max_iter') [500, 1000], loss function ('clf__penalty') [‘l1’, ‘l2’, ‘elasticnet’], optimization algorithms ('clf__solver') ['lbfgs', 'liblinear', 'saga'].

For the training of the Random Forest models we ran a grid search test across number of trees (‘n_estimators') [100, 300], the maximum depth of the trees ('max_depth') [10, 20], the maximum features subsampled that each tree intakes ('max_features') ['sqrt', 'log2'], the minimum samples to split ('min_samples_split') [2, 5, 10], the samples for each leaf node ('min_samples_leaf') [1, 2, 4], and finally if bootstrapping is to be carried out ('bootstrap') [true, false]. These were trained on a 5-fold cross validation schema using 50% training and 50% testing.

Fine tuning of the pretrained shufflenet_v2_x1_0 model with weights from ImageNet1k_v1 consisted of freezing of all weights except the final fully connected layer (1024, 2) as this task was a simple binary classification task so the learned embeddings of the image was not necessary to adapt. We trained this model on 30% of the raw data for timeliness and performed a grid search over epochs (2, 3, 5), learning rate (0.01, 0.001, 0.0001), and batch size (16, 32, 64) to find an optimal model.

Then taking our results out of domain from the Kaggle input data into the segmented data using ground truth segmentation, we test the transferability of our models.

5. Experiments and Results

5.1. Segmentation Efficiency and Accuracy

5.1.1 Experimental Protocol

To simulate real-world clinical conditions, we assumed that Ground Truth masks are not available during the diagnostic process. Therefore, we did not optimize parameters for each individual test image. Instead, parameters for Otsu, Hough Transform, and Watershed algorithms were tuned on a single representative image based on its geometric and color features. These parameters were then frozen and applied to the test set. This approach allows for a fair evaluation of the algorithms' generalization capability across different patients and lighting conditions.

5.1.2 Qualitative and Quantitative Analysis

Initial experiments revealed a critical issue with standard global thresholding. The large background area caused standard Otsu to calculate an incorrect threshold, leading to massive flooding where the background was detected as cells. We solved this by implementing a two-step thresholding strategy. We calculated the threshold using only the valid foreground pixels and then applied this value to the entire image. This technique successfully stabilized the segmentation without complex post-processing.

Table 1. Best performance of segmentation methods

Method	IoU	F1	Count Diff, %	Time, sec
Watershed_Green (Raw)	0,727	0,912	33,1	0,802
Watershed_Green (+ Refine)	0,763	0,909	33,9	0,849
Watershed_Hue (+ Refine)	0,761	0,875	18,8	0,733
Watershed_Hue (Raw)	0,728	0,870	16,4	0,697
Hough_Green (Raw)	0,765	0,828	4,3	0,585
Otsu_Green (+ Refine)	0,813	0,787	32,7	2,495
Otsu_Green (Raw)	0,789	0,786	32,7	2,462
Otsu_Hue (+ Refine)	0,790	0,739	19,6	2,178
Otsu_Hue (Raw)	0,773	0,739	20,0	2,142

We valuated segmentation quality on five patients with varying smear characteristics. As shown in Appendix Figure 2, Global Otsu method failed to separate overlapping cells, merging clusters into single objects. The Circle Hough Transform detected round cells well but missed deformed ones. The Marker-controlled Watershed algorithm successfully separated touching cells.

Quantitatively, Marker-controlled Watershed (using the Green channel) achieved the highest shape accuracy with an F1-Score of 0.91 and IoU of 0.76. However, it demonstrated a tendency towards over-segmentation. As shown in Table 1, it detected 33.9% more cells than the Ground Truth. This indicates that while it captures cell shapes accurately, it sometimes splits single cells or detects noise. In contrast, the Circle Hough Transform was the most accurate in counting, with a difference of only +4.3%.

Table 2. Evaluation of the best methods on the whole set

Method	IoU	F1	Count_Diff	Time_sec
Watershed_Green (+Refine)	0,791	0,920	19,79	1,477
Hough_Green (Raw)	0,763	0,748	-42,24	0,978

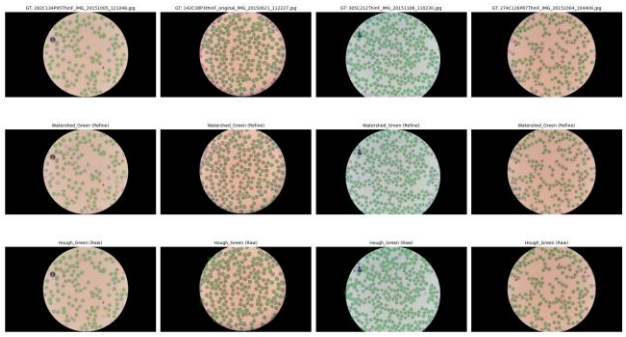

Figure 2. Evaluation of the best methods, ramdom exaples

5.1.3 Computational Efficiency

We analyzed the computational resources required for each method to assess their suitability for low-resource hardware. This aspect is critical for deployment on standard laptops used in field laboratories (Table 3).

Table 3. Computational Efficiency

Method	MaxTime (sec)	AvgTime (sec)	AvgPeakRAM (MB)
Watershed	47119	46327	403.07
Hough	46388	46082	168.77

The Circle Hough Transform proved to be highly memory-efficient. It required only 168.8 MB of peak RAM. In contrast, the Marker-controlled Watershed algorithm was more resource-intensive, consuming 403.1 MB due to complex matrix operations. In terms of speed, both methods performed comparably, processing images in approximately 1.0 to 1.1 seconds. Global Otsu was significantly slower (approx. 2.4 seconds) in our implementation. These results confirm that Hough is the most lightweight solution for extremely constrained devices.

5.2. Classification on Ground Truth

The Kaggle dataset delivered good results for our model when we stayed within the domain of the dataset, however when we stepped beyond into other segmentation methods including our own segmentation attempts, the transferability of the models was not profound. The results of the modeling efforts primarily concern the model's ability to perform on the Kaggle Dataset, while a small portion is dedicated to the transferability.

5.2.1 Performance on Kaggle Data

Selected for comparison were the strongest models for each Linear Regression, Random Forest, Support Vector Machine and Shuffle Net as well as the fastest run time Shuffle Net.

For the classical learning methods, we tested different training sizes as the number of learnable features is set to 16 and the number of available data is ~30,000 so there is a concern of overfitting. With the discovery of a superior performance by both the Logistic Regression and the Random Forest at a training size of 50%, we were able to reduce the training time and avoid overfitting.

The optimal hyperparameter tuning for the logistic regression occurred at {'clf__C': 100, 'clf__max_iter': 500, 'clf__penalty': 'l2', 'clf__solver': 'lbfgs'}

The optimal hyperparameter tuning for the random forest {'bootstrap': False, 'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 300}.

Across all classical models when we conducted the training size tests post optimization of hyper parameters, we saw steady increase in performance from 10-50% of the possible dataset for training before a sharp decline from 50-70%. There was a relatively small recovery in the SVM and RF but a significant recovery of the LR from 70-90% of the possible training data usage (figure 4a).

The Support vector performs the weakest of all the models tested, with a maximum F1 score of .88 at 50% of dataset for training (figure 4a).

We see relatively consistent performance for the Logistic Regression regardless of the training size except for 70%. The peak of performance at 50% training size with an F1 score of 0.91 (figure 4a). The training time of Logistic Regression is the smallest when we compare across models and training size(figure 4b).

Random Forest shows significant gains from initial tests at training sizes of 10% to 30% and a very slight decline to 50% of the dataset before falling at 70% of the data. At 50% data input into the training the F1 score is stronger, 0.94, than that of the strongest fine-tuned Shuffle Net model 0.93. When comparing the two models, the training time of the Random Forest is significantly smaller than that of the Shuffle Net. Random Forest has the best recall score 0.94(figure 4c).

To compare the grid searched optimum with the best Logistic Regression and Random Forest, we train one Shuffle Net model with the same hyper parameters on 50% of the training set. The training time ~5x but the

performance falls short of the optimum of the grid searched model(figure 4).

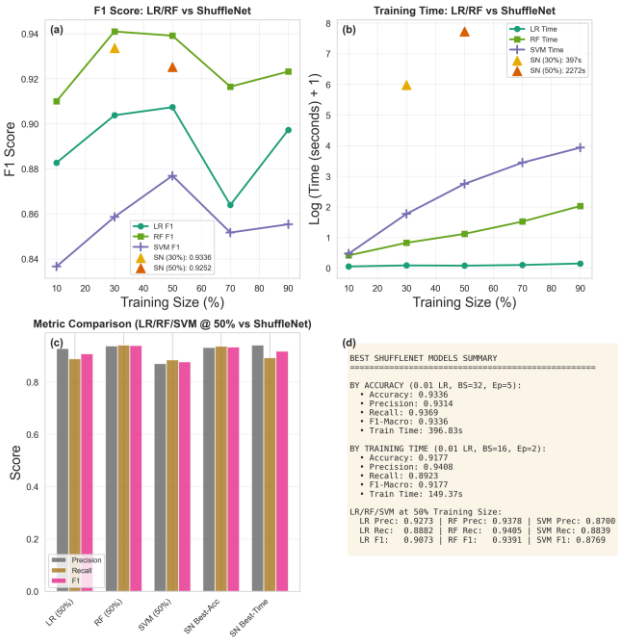

Figure 4. Comparison across models of F1 scores vs training size (a), Training time vs training size (b), (precision, recall, and F1) across the best key models (c) and a final report of scores and training time for each of the best preforming models (d).

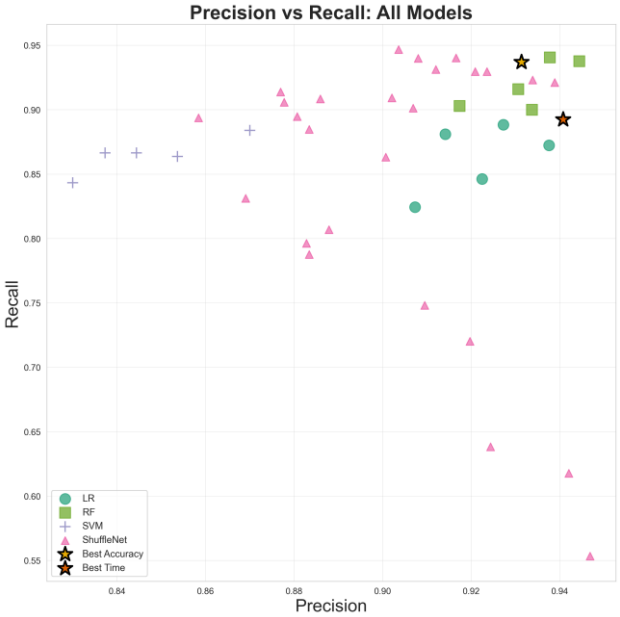

Figure 5. Precision vs Recall of the set of models. The best preforming models for accuracy and speed of the shufflenet are indicated with stars, whereas the best preforming models are not indicated for the LR, RF, of SVM. Refer to the Figure 4(d).

The grid search performed for shuffle net when using 30% of the training data can be understood through Figure 6. The various dimensions: batch size, epochs and learning rate are shown with respect to training time and accuracy. As expected, the training time grows roughly linear with epochs and a possibly linear performance increase with additional epochs. The learning rate is by far the greatest indicator of quality of the model. Learning rate at scales of 1/100 are significantly better than 1/1000 and 1/10000 regardless of the number of epochs and batch size. The accuracy dependent on batch size with the bigger learning rate is consistently maximized with a size of 32, whereas the smaller learning rates perform best with a smaller batch size of 16. The batch size of 32 and 64 take roughly the same training time, but the smaller batch size of 16 takes longer.

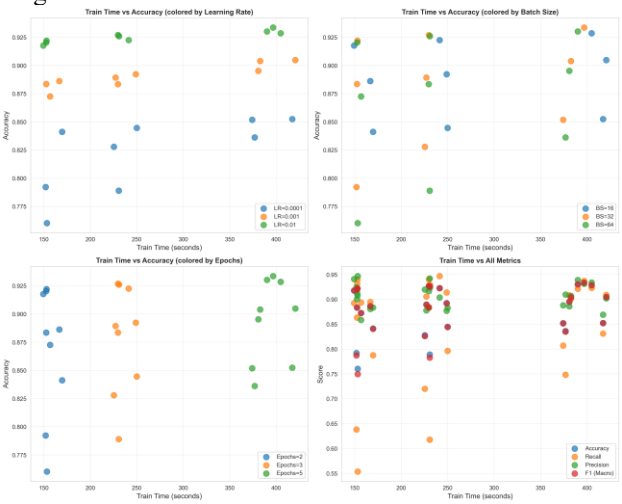

Figure 6. Understanding how each component of the grid search affects the performance of the model at testing time.

The overall performance of models are compared through precision and recall measuring the quality of true predictions and ability of the model to find the true positives respectively. In figure 2, we can see that the best scores for precision are achieved by random forest models, but for recall Some of the Shuffle Net grid search preform the best. The Logistic Regression does not perform well in either recall score or precision.

5.2.2 Out of Domain data Performance

Initial results across the RF, LR, and SN indicate that moving to out of domain data induces significantly more false positives. We notice a relative stability in recall for the LR and RF models, 0.88 and 0.92. A severe reduction precision to values 0.14 for both the random forest and logistic regression models tested on ~10,000 segmented cells using the ground truth segmentation. The Shuffle Net seems to not be able to transfer at all resulting in an accuracy score of 0.36, a near 0 precision score, and < 0.1 Recall score. The F1 score for all models drops below .3.

5.3. System Performance on Segmented Data

5.3.1 Data Preparation and Feature Optimization

To evaluate the full pipeline, we generated a dataset of individual cell images based on the masks produced by the Watershed and Hough algorithms (Appendix Figure 7). To ensure rigorous evaluation, the dataset was split at the patient level. This prevents data leakage, ensuring that cells from the same patient never appear in both training and test sets. The final partition consisted of 20 patients for Training, 5 for Validation, and 8 for Testing (Figure 7). The split was also stratified by parasitemia rate to guarantee that high-infection cases were balanced across all subsets.

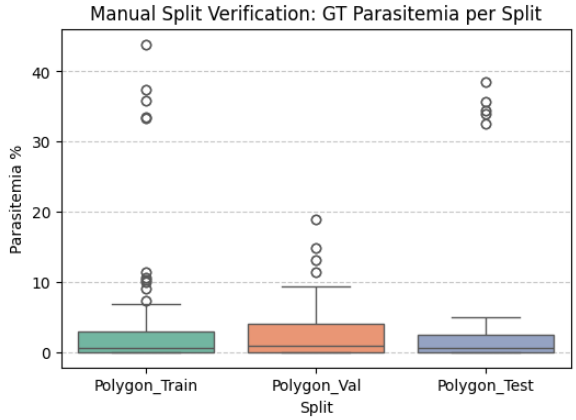

Figure 7. Train/test/val stratification

A critical adjustment was made in the feature extraction stage compared to the preliminary Ground Truth experiments. While the initial tests utilized K-Means clustering, we identified this approach as too computationally expensive. Therefore, to align with the frugal objective, we replaced K-Means with fast statistical features. We computed the Mean and Standard Deviation in RGB and HSV color spaces to capture general color properties. To specifically detect the small, highly saturated spots characteristic of parasites, we calculated the 90th, 95th, and 99th percentiles of the Saturation and Value channels. Additionally, normalized Color Histograms (32 bins) for Hue and Saturation were used to represent the color distribution. These features are computed in constant time, significantly reducing the processing load.

5.3.2 Classification Results

We evaluated the classification performance using the Logistic Regression, Random Forest and SVM models, which showed the best stability in previous tests. Appendix Table 1 presents the drop in performance when moving from ideal Ground Truth masks to automated segmentation. On the Ground Truth baseline, the SVM model achieved an F1-Score of 0.87. On the Watershed-segmented data, the F1-Score decreased to 0.64, and Recall dropped to 0.49. However, this method maintained a high Precision of 93%, meaning that positive detections are highly reliable, though some infected cells are missed. The Hough-segmented data showed the lowest classification performance with an F1-Score of 0.56 and Recall of 0.44. The lower precision indicates that the Hough transform captures more background artifacts, which confuses the classifier (Figures 8,9, Appendix Figures 5,6).

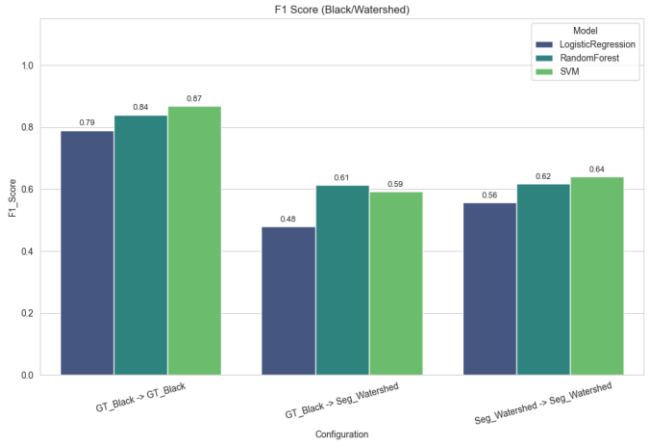

Figure 8. Comparison of various classical segmentation models and methods, black background

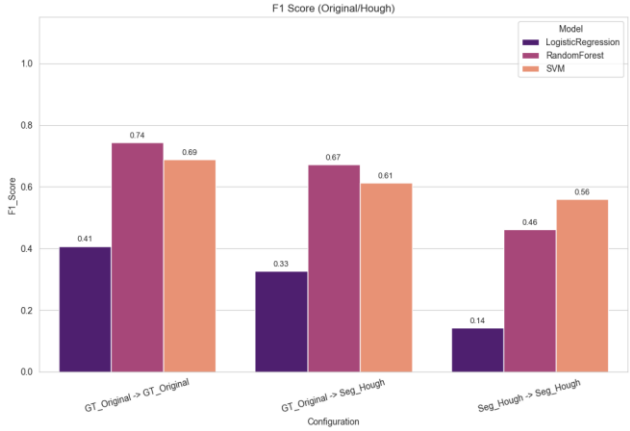

Figure 9. Comparison of various classical segmentation models and methods, original background

5.3.3 Efficiency and Parasitemia Estimation

The final clinical objective is to estimate the patient's Parasitemia Rate within a reasonable timeframe. We analyzed both the accuracy of this estimation and the total processing time per field of view.

Surprisingly, the Hough-SVM pipeline provided a more accurate clinical diagnosis with a Mean Parasitemia Error of 2.4%, compared to 3.0% for the Watershed pipeline. The Watershed algorithm tends to detect 34% more cells than actually present (over-segmentation), which artificially inflates the total cell count and biases the percentage calculation. The Hough pipeline, despite having lower

classification precision, provides a much more accurate count of total cells, leading to a balanced and lower final error.

Regarding speed, the full system demonstrated high efficiency. The average processing time was approximately 2.6 seconds per image (2583 ms for Random Forest, 2611 ms for SVM) as shown in Appendix Table 1. This speed allows for the automated screening of a typical patient slide (around 100-200 fields of view) in under 5 minutes, which fits well within standard laboratory workflows.

6. Discussion

Our analysis revealed a critical trade-off between geometric accuracy and clinical utility. The Marker-controlled Watershed algorithm achieved superior shape fidelity with an IoU of 0.79, but it suffered from massive over-segmentation, detecting 33.9% more cells than the ground truth. This noise caused the classification F1-score to drop from 0.87 (on ideal data) to 0.64. In contrast, the Circle Hough Transform was less precise in shape (IoU 0.76) but highly robust in counting, with a deviation of only 4.3%. Surprisingly, this counting stability made the Hough pipeline more accurate for the final diagnosis. It achieved a Mean Parasitemia Error of 2.4%, compared to 3.0% for Watershed. This proves that for monitoring infection rates, accurate cell counting is more important than perfect segmentation of individual cell boundaries.

To ensure the system works on low-resource hardware, we optimized the feature extraction process by replacing slow K-Means clustering with fast statistical calculations (Color Percentiles). This reduced the computational load significantly. The resource analysis showed that the Hough Transform is the most efficient solution, requiring only 168 MB of peak RAM compared to 403 MB for Watershed. With a total processing speed of approximately 1.0 second per image, the Hough-based system is suitable for deployment on standard laptops in field laboratories, fulfilling the project's "Frugal AI" objective.

The lightweight models, especially the random forest perform best across the training and testing set indicating accurate feature engineering and hyperparameter tuning can save time and computational resources allowing for edge applications to exist in the field.

Classification and analysis of many blood cells can be greatly reduced by using intelligent models to differentiate the space between blood cells of the type parasitic and non-parasitic, however blood cell images may come in forms that are not equivalent at each sample. There are often differences in lighting or plasma solution makeup which shifts the domain of the input images for blood cells possibly reducing a model's ability for correct identification. Therefore, it is critical to further develop methods that are robust to out of domain blood cell samples and therefore able to be widely used in the field.

Overall, classification efforts can be improved with online learning protocols that finetune the model as they come in. For feature engineering improvements, one could identify which types of cells give the highest false positive rates and false negative rates and try to synthesize some features that are common across the false prediction spaces.

The primary limitation is the low Recall (Sensitivity) of 49% on the segmented data. The system currently misses about half of the infected cells due to segmentation artifacts that confuse the classifier. While the system accurately estimates the overall percentage of infection (parasitemia), it is not yet sensitive enough for the initial diagnosis of patients with very low viral loads. Future work must focus on reducing segmentation noise to improve the detection of individual parasites.

Contributions

Polina: Parts 2, 3, 5.1, 5.3
Tyler: Parts 4, 5.2

References

[1] World Health Organization, World malaria report 2022. Geneva: World Health Organization, 2022.

[2] Y. M. Kassim et al., "Clustering-Based Dual Deep Learning Architecture for Detecting Red Blood Cells in Malaria Diagnostic Smears," IEEE Journal of Biomedical and Health Informatics, vol. 25, no. 5, pp. 1735–1746, May 2021

[3] F. B. Tek, A. G. Dempster, and I. Kale, "Computer vision for microscopy diagnosis of malaria: a comprehensive review," Malaria Journal, vol. 8, no. 153, 2009.

[4] D. K. Das, M. Ghosh, M. Pal, A. K. Maiti, and C. Chakraborty, "Machine learning approach for automated screening of malaria parasite using light microscopic images," Micron, vol. 45, pp. 97–106, 2013.

[5] C. Di Ruberto, A. Dempster, S. Khan, and B. Jarra, "Segmentation of blood images using morphological operators," in Proceedings of the 15th International Conference on Pattern Recognition (ICPR), vol. 3, pp. 397–400, 2000.

[6] Arunava, I. Cell Images for Detecting Malaria. Kaggle, 2018, https://www.kaggle.com/datasets/iarunava/cell-images-for-detecting-malaria.

[7] Poostchi, Mahdieh, et al. "Image Analysis and Machine Learning for Detecting Malaria." *Translational Research*, vol. 194, Apr. 2018, pp. 36–55, https://doi.org/10.1016/j.trsl.2017.12.004.

[8] Al Kafi, Md Abdullah, Raka Moni, and Sumit Kumar Banshal. *Less Is More: An Explainable AI Framework for Lightweight Malaria Classification*. arXiv, 22 Nov. 2025, arXiv:2511.18083, https://doi.org/10.48550/arXiv.2511.18083.

[9] M. K. Bashar, "Improved Classification of Malaria Parasite Stages with Support Vector Machine Using Combined Color and Texture Features," *2019 IEEE Healthcare Innovations and Point of Care Technologies, (HI-POCT)*, Bethesda, MD,

USA, 2019, pp. 135-138, doi: 10.1109/HI-POCT45284.2019.8962686.
[10] Zhang, Xiangyu, Xinyu Zhou, Mengxiao Lin, and Jian Sun. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. 2017. arXiv, arXiv:1707.01083

Model	Train Source	Test Target	F1 Score	Recall	Precision	Mean Parasitemia Error, %	Mean Time Per Image, ms
LogisticRegression	GT_Black	GT_Black	0,790	0,764	0,817	1,66	2223
RandomForest	GT_Black	GT_Black	0,840	0,745	0,963	1,28	2226
SVM	GT_Black	GT_Black	0,869	0,786	0,970	1,04	2265
LogisticRegression	GT_Black	Seg_Watershed	0,479	0,434	0,534	4,36	1917
RandomForest	GT_Black	Seg_Watershed	0,614	0,462	0,917	3,25	1920
SVM	GT_Black	Seg_Watershed	0,593	0,441	0,906	3,42	1941
LogisticRegression	GT_Original	GT_Original	0,408	0,286	0,708	3,56	2223
RandomForest	GT_Original	GT_Original	0,745	0,659	0,856	1,27	2225
SVM	GT_Original	GT_Original	0,689	0,577	0,854	1,85	2244
LogisticRegression	GT_Original	Seg_Hough	0,327	0,250	0,474	4,05	2581
RandomForest	GT_Original	Seg_Hough	0,673	0,587	0,789	1,46	2583
SVM	GT_Original	Seg_Hough	0,614	0,520	0,747	2,09	2611
LogisticRegression	Seg_Watershed	Seg_Watershed	0,557	0,490	0,645	3,58	1917
RandomForest	Seg_Watershed	Seg_Watershed	0,617	0,462	0,930	3,33	1920
SVM	Seg_Watershed	Seg_Watershed	0,641	0,490	0,927	2,98	1948
LogisticRegression	Seg_Hough	Seg_Hough	0,143	0,089	0,366	4,57	2581
RandomForest	Seg_Hough	Seg_Hough	0,462	0,330	0,771	2,98	2583
SVM	Seg_Hough	Seg_Hough	0,560	0,443	0,761	2,42	2629

Appendix

Table 1. Comparison of various classical segmentation models and methods

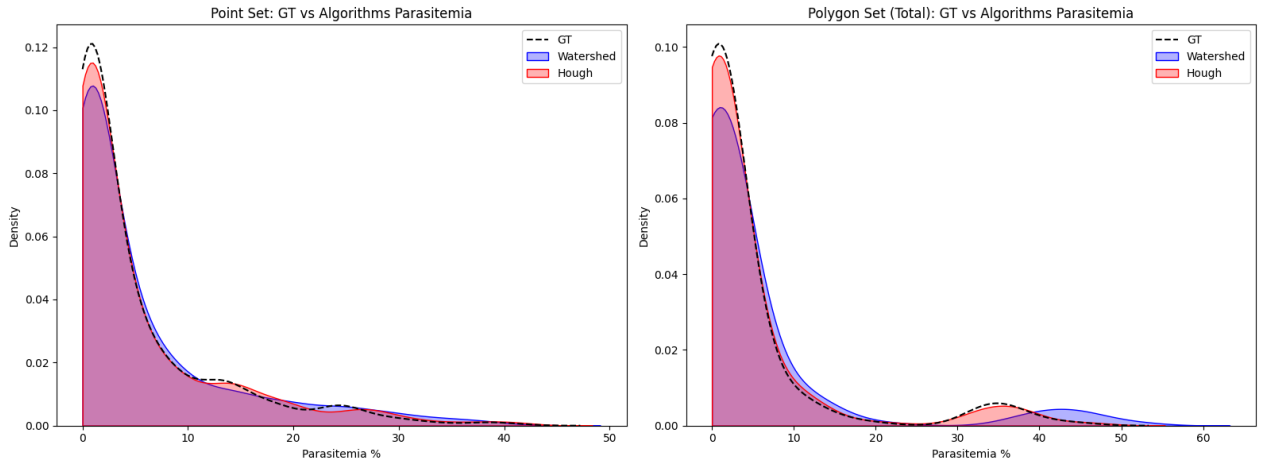

Figure.1. Distribution of final parasitemia after assigning labels to segmented data


Ground Truth
GT: IMG_20150819_133236
GT: IMG_20150820_162153
GT: IMG_20151106_112252
GT: IMG_20151201_142846
GT: IMG_20150728_123237
Otsu_Green (Raw)
IoU: 0.77 | F1: 0.719
IoU: 0.804 | F1: 0.618
IoU: 0.823 | F1: 0.885
IoU: 0.755 | F1: 0.829
IoU: 0.791 | F1: 0.88
Otsu_Green (+ Refine)
IoU: 0.819 | F1: 0.729
IoU: 0.802 | F1: 0.618
IoU: 0.827 | F1: 0.885
IoU: 0.785 | F1: 0.825
IoU: 0.834 | F1: 0.878
Otsu_Hue (Raw)
IoU: 0.786 | F1: 0.549
IoU: 0.808 | F1: 0.521
IoU: 0.837 | F1: 0.834
IoU: 0.752 | F1: 0.835
IoU: 0.683 | F1: 0.955
Otsu_Hue (+ Refine)
IoU: 0.779 | F1: 0.553
IoU: 0.801 | F1: 0.521
IoU: 0.837 | F1: 0.834
IoU: 0.782 | F1: 0.835
IoU: 0.751 | F1: 0.952
Watershed_Green (Raw)
IoU: 0.699 | F1: 0.885
IoU: 0.687 | F1: 0.897
IoU: 0.823 | F1: 0.921
IoU: 0.685 | F1: 0.913
IoU: 0.74 | F1: 0.945
Watershed_Green (+ Refine)
IoU: 0.763 | F1: 0.881
IoU: 0.688 | F1: 0.89
IoU: 0.831 | F1: 0.92
IoU: 0.728 | F1: 0.911
IoU: 0.807 | F1: 0.945

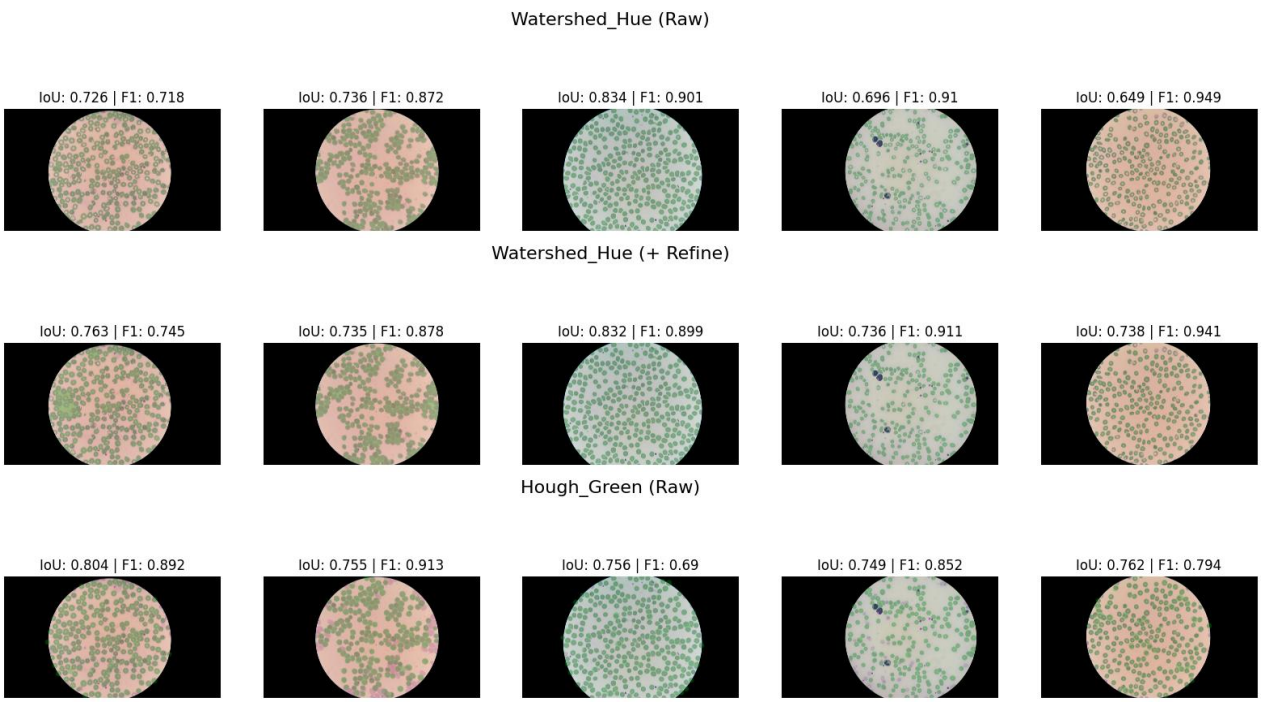

Figure. 2. Best performance of segmentation methods on a sample data subset

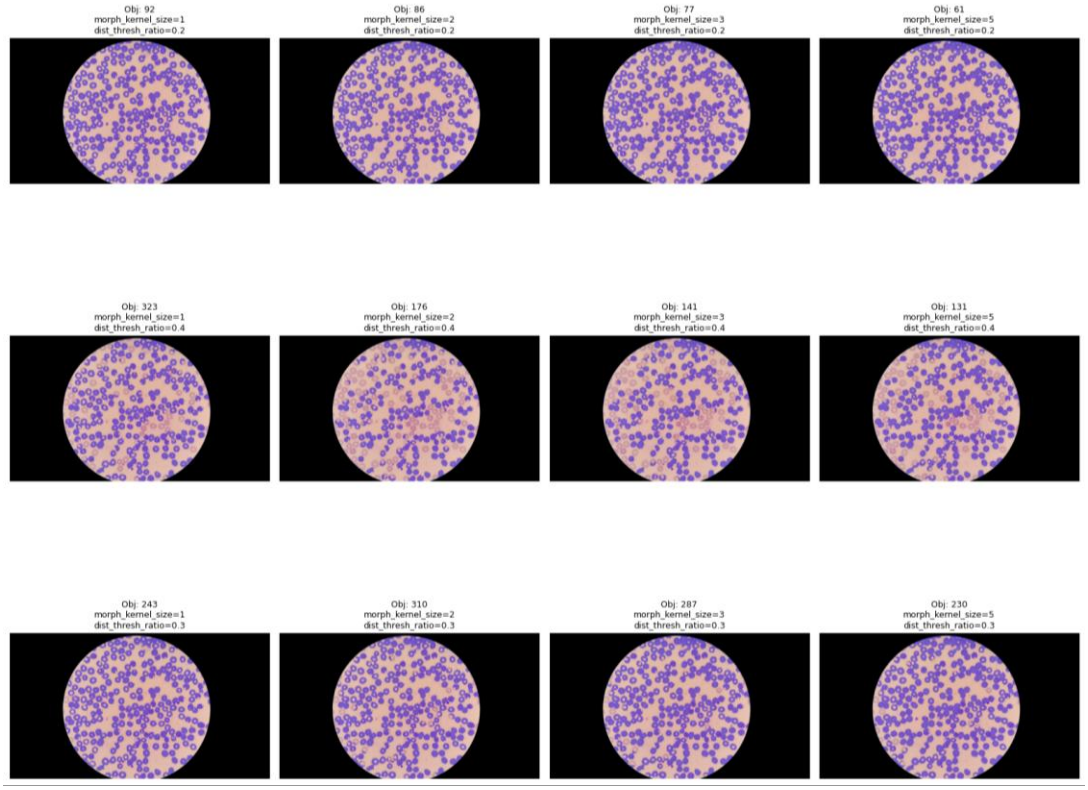

Figure. 3. An example of parameter selection for the Watershed segmentation algorithm

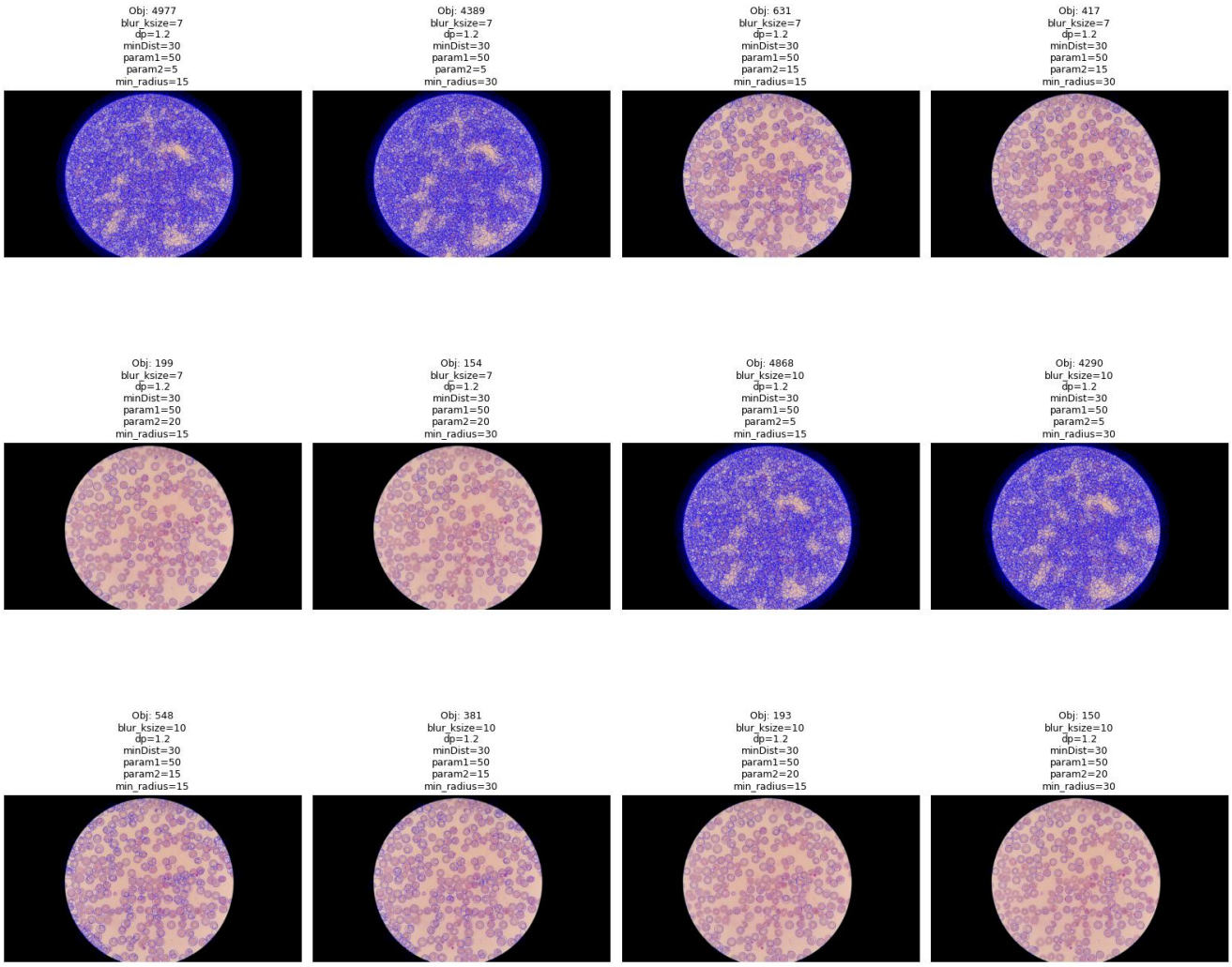

Figure 4. An example of parameter selection for the Hough segmentation algorithm

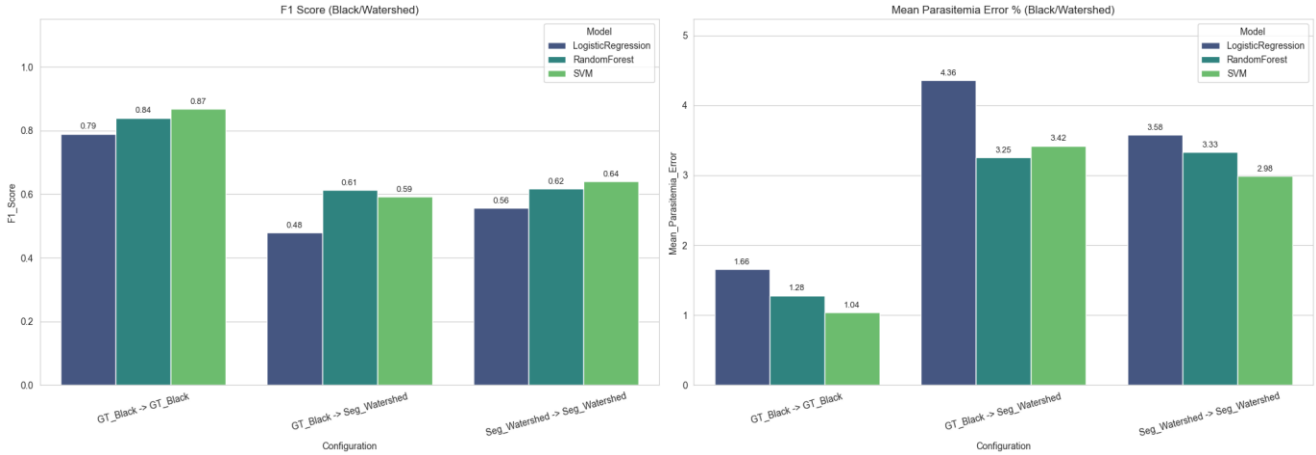

Figure 5. Comparison of various classical segmentation models and methods, black background

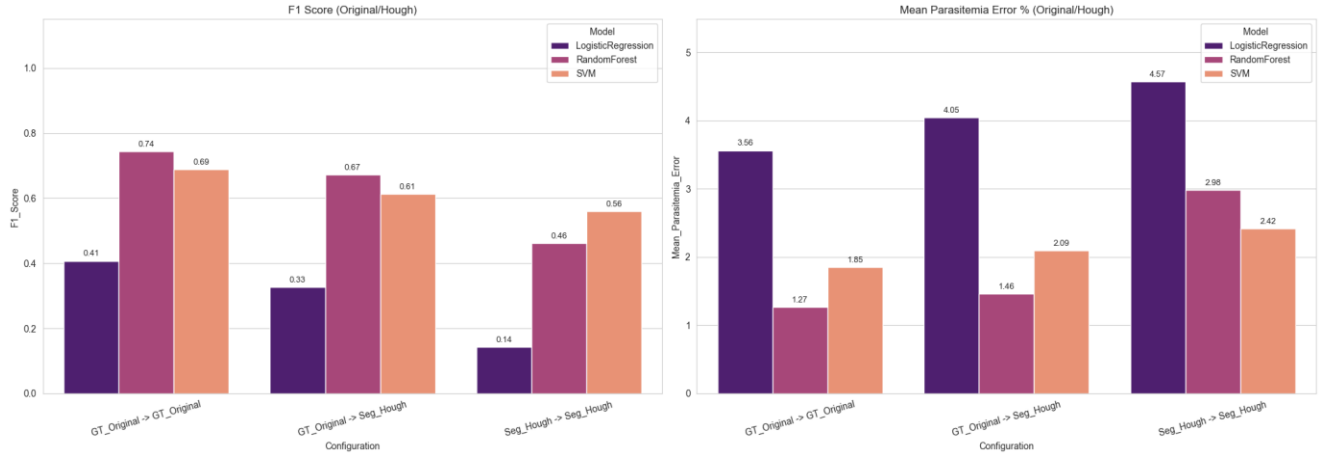

Figure 6. Comparison of various classical segmentation models and methods, original background

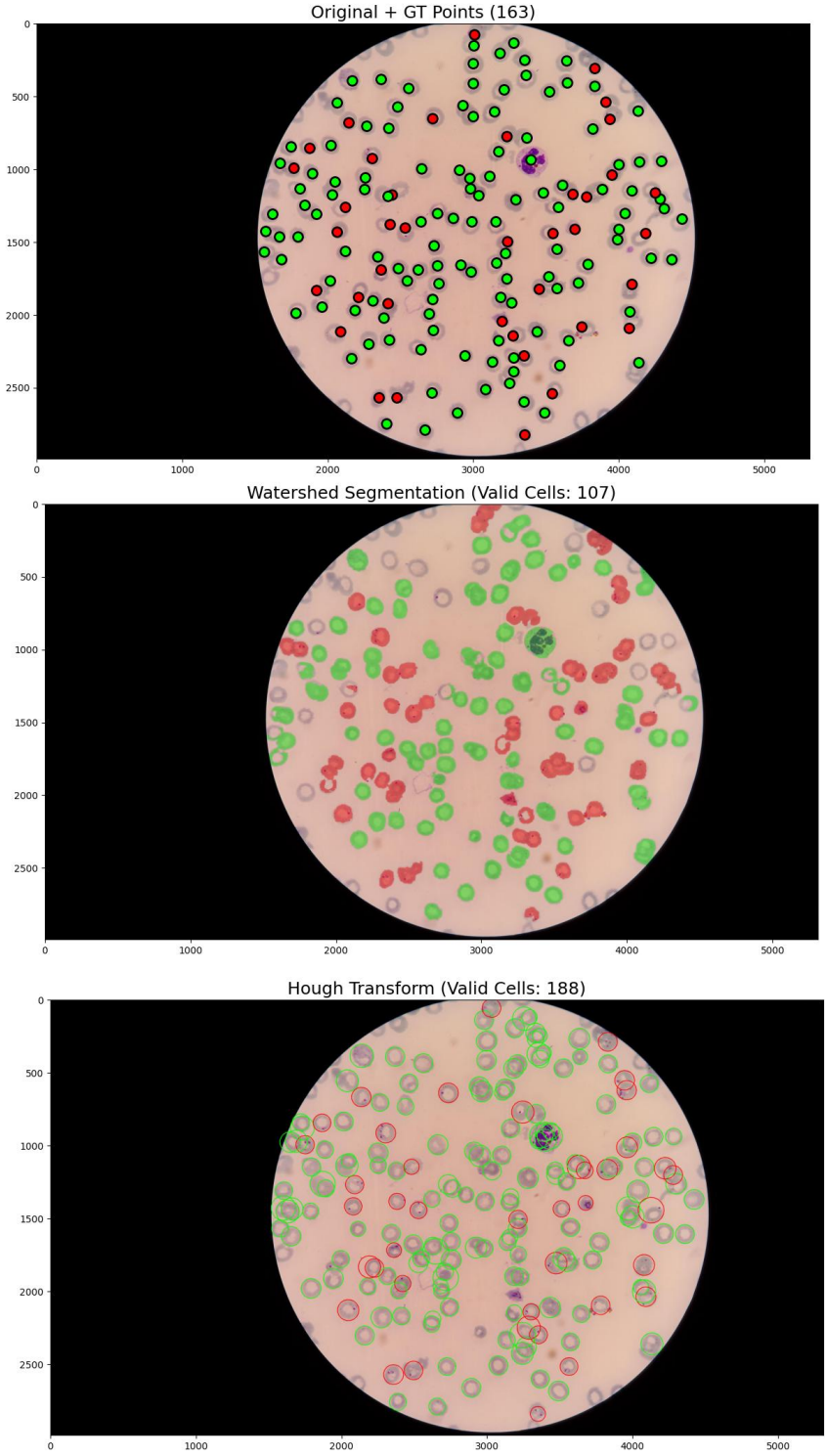

Figure 7. An example of assigning labels to segmented data, true labels are represented as dots, red ones are infected, green ones are healthy.