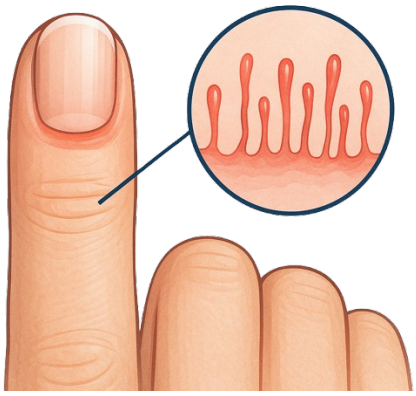
Segmentation and Counting Capillaries



Submitted By:

Malik Danial Ahmed
Somaia Elbaradey

Course:
Medical Image Analysis

Date:
June 2025

Department of Computer Vision and Robotics
Université Bourgogne Europe

Contents

List of Figures

1 Abstract

Accurate segmentation of capillaries in biomedical images is critical for diagnosing and monitoring various diseases related to microcirculation. This study proposes a semi-automated segmentation pipeline designed for enhancing and extracting capillary structures from native tongue images across four datasets: **N1, N2, S2,** and **S3**. The method incorporates preprocessing steps including CLAHE-based contrast enhancement and median filtering, followed by vessel enhancement filters such as Frangi, Sato, and Meijering to highlight vascular structures. A composite vesselness map is generated, and Otsu's thresholding is applied for binarization. Finally, morphological operations refine the segmentation by removing noise and small artifacts. The segmented results are saved and visually overlaid on original images for validation.

In addition to segmentation, we performed capillary counting on the N1 dataset using a predefined region of interest (ROI). This involved detecting reference lines, aligning the ROI, and applying connected components analysis to estimate capillary count.

Overall, this pipeline demonstrates adaptability across datasets with varying image quality and capillary visibility, making it a promising step toward standardized microvascular analysis and automated quantification in clinical and research settings.

2 Introduction

Analyzing microvascular structures like tongue capillaries is vital for diagnosing diseases such as diabetes and hypertension. Manual segmentation is slow and subjective, so automated image processing is needed.

This project presents a segmentation pipeline using contrast enhancement, noise reduction, and vesselness filters (Frangi, Sato, Meijering) to detect capillaries from tongue images under native lighting.

The method is tested on four datasets (N1 Native, N2 Native, S2 Native, S3 Native) to ensure robustness. Outputs include vesselness images, binary masks, and overlays for further analysis.

3 Methodology for the Segmentation of Capillaries

This section describes the complete image preprocessing and capillary segmentation pipeline applied to the retinal image datasets (N1, N2, S2, and S3). The goal was to enhance and extract the capillary vessels using a combination of classical image processing techniques and vesselness filtering.

3.1 Dataset Preparation

Each dataset (N1, N2, S2, S3) consists of retinal images acquired under similar imaging conditions but possibly from different subjects or experimental setups. Images were stored in standard formats such as PNG or JPEG.

For each dataset:

- A structured folder was created to store the Original, Vesselness Map, Binary Mask, and Overlay images.
- Only supported image formats were processed: `.png`, `.jpg`, `.jpeg`, `.tif`.

3.2 Image Preprocessing

1. **Grayscale Conversion**: All input images were converted to grayscale to simplify processing and reduce computational cost.
2. **Noise Reduction**: Median filtering was applied to suppress salt-and-pepper noise, which is common in medical images.
3. **Contrast Enhancement**: Contrast Limited Adaptive Histogram Equalization (CLAHE) was used to enhance local contrast and prevent over-amplification of noise in homogeneous regions. It operates by applying histogram equalization within small tiles and limiting contrast using a clip limit parameter. Mathematically, CLAHE redistributes pixel intensities based on the local histogram $H(i)$ within a tile, with clipping applied at a threshold T to limit the maximum bin height.
4. **Normalization**: Images were normalized to a 0–1 range before applying vesselness filters.

3.3 Vesselness Filtering

Three vessel enhancement filters were applied to highlight elongated, tubular capillary structures:

- Frangi Filter [1]
- Sato Filter [2]
- Meijering Filter [3]

These filters exploit the eigenvalues of the Hessian matrix to enhance line-like structures.
The final vesselness map was computed by averaging the outputs of the three filters:

$$\text{Vesselness} = \frac{1}{3}(\text{Frangi} + \text{Sato} + \text{Meijering})$$

3.4 Thresholding

The enhanced vesselness response was binarized using Otsu's Thresholding, an automatic global thresholding method that determines the optimal threshold by minimizing intra-class variance between the foreground (vessels) and background pixels. This step is essential to convert the vesselness map into a binary image, clearly separating capillary structures from the background for further morphological processing and analysis.

3.5 Morphological Post-Processing

To refine the segmentation mask:

- **Closing** was applied to fill small gaps and connect nearby vessel segments.
- **Small Object Removal**: Noise and artifacts smaller than a threshold size were removed.
- **Small Hole Filling**: Holes within segmented vessels were filled to ensure continuity.

Each dataset had slightly different morphological parameters to adapt to the image quality and noise level.

3.6 Overlay Generation

A visual overlay was created by marking the segmented vessels in red over the original image for qualitative evaluation.

3.7 Output Storage

For each image, the following results were stored in separate subfolders:

- Original image
- Vesselness map (grayscale)
- Binary segmentation mask
- Overlay image (original + mask)

4 Methodology for Capillary Counting (N1 Dataset)

Once the segmentation was completed, we proceeded to estimate the number of individual capillaries within a defined Region of Interest (ROI) in the N1 dataset. The pipeline followed these key steps: [**?**]

1. **Load Region of Interest (ROI) and Corresponding Segmentation Mask**
   For each N1 image:
   - The ROI image (with a central black reference line) was loaded.
   - The corresponding binary mask (segmented vessels) was loaded using a matching filename pattern.
2. **Detection and Alignment of Black Line**
   To ensure consistency in capillary orientation:
   - The black central reference line was detected using thresholding and contour detection.
   - A minimum-area rectangle was fitted to the line.
   - Both the ROI and the vessel mask were rotated to horizontally align the black line using OpenCV's affine transformation.
3. **Define Standardized ROI Window**
   The central portion of the aligned vessel mask was cropped using a rectangular window based on the detected black line's dimensions (length and a fixed width equal to length/4). This ensures that the same anatomical region is analyzed across all images.
4. **Capillary Counting via Connected Components**
   The cropped binary mask was thresholded. OpenCV's `connectedComponents()` function was used to count distinct white regions, each corresponding to an isolated capillary structure.
5. **Visualization and Quality Control**
   For each image:
   - A visual panel was created with four subplots: the original ROI, rotated ROI, rotated vessel mask with ROI box, and the cropped ROI.
   - These visuals were saved to a dedicated folder for quality review.
6. **Result Logging**
   The final count of capillaries per image (excluding the background component) was saved in a structured CSV file (`capillary_counts_N1.csv`), facilitating further analysis or plotting.

5 N1 Native Segmentation Parameters and Results

Preprocessing

- **Median Filter (kernel size = 5)**: Applied to reduce noise while preserving edges, essential for accurate vessel detection.
- **CLAHE (clip limit = 3.0, tile grid size = 8×8)**: Used to enhance local contrast and improve visibility of capillaries in unevenly illuminated images.

Vessel Enhancement

Applied three vesselness filters: Frangi, Sato, and Meijering, combined by averaging to robustly highlight tubular structures typical of capillaries.

Thresholding and Cleaning

- **Otsu's Thresholding**: Automatically selected threshold to separate vessels from background based on image histogram.
- **Morphological Operations**:
  - Closing with a 3×3 square structuring element to fill small gaps.
  - Removed small objects smaller than 1000 pixels to eliminate noise and irrelevant details.
  - Removed small holes smaller than 200 pixels to fill gaps inside vessel regions.

Results

This approach generated clear binary masks effectively capturing the capillaries in N1 images, enabling reliable quantitative analysis.

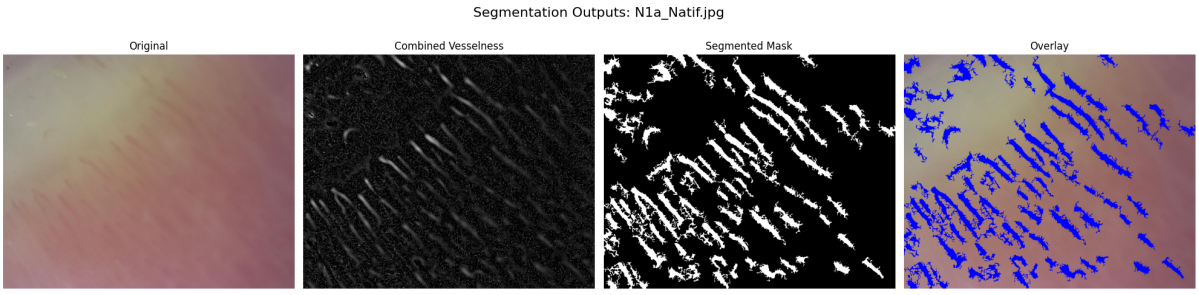


Figure 1: N1a Natif Segmented,

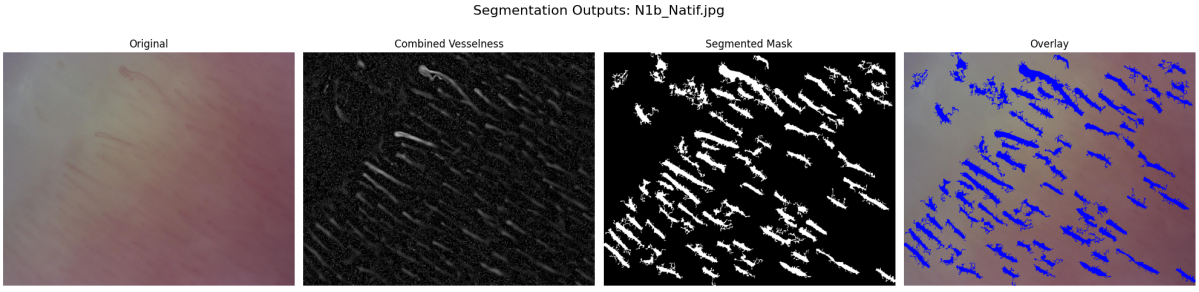


Figure 2: N1b Natif Segmented,

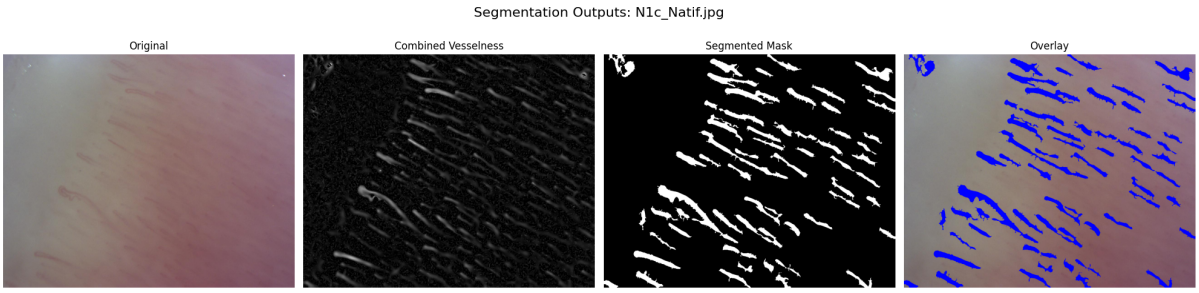


Figure 3: N1c Natif Segmented

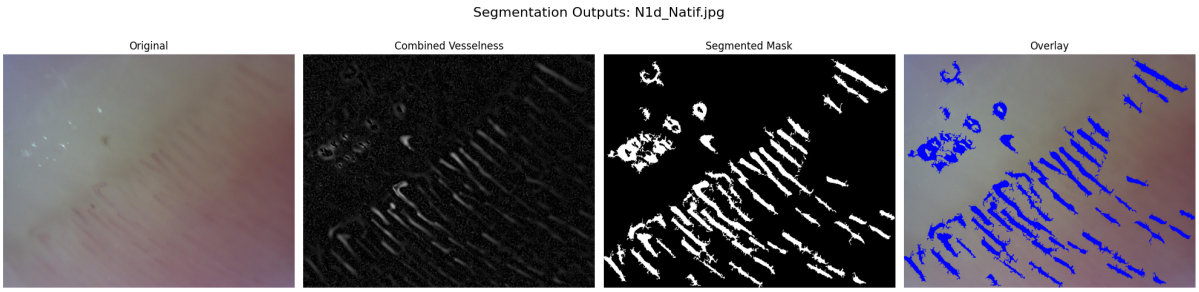


Figure 4: N1d Natif Segmented

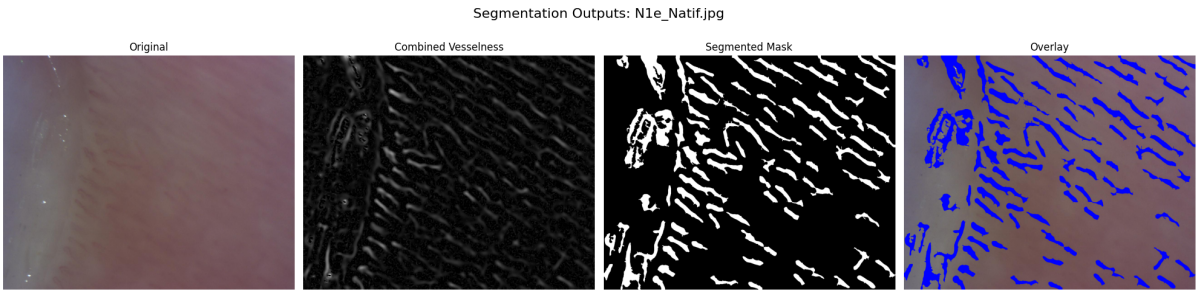


Figure 5: N1e Natif Segmented

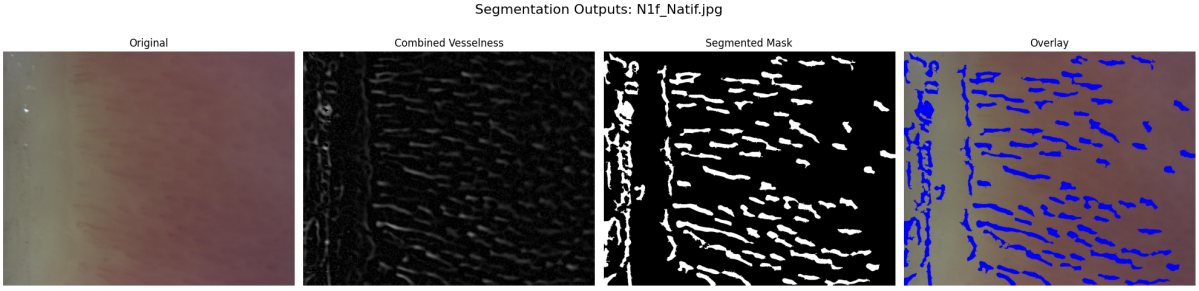


Figure 6: N1f Natif Segmented

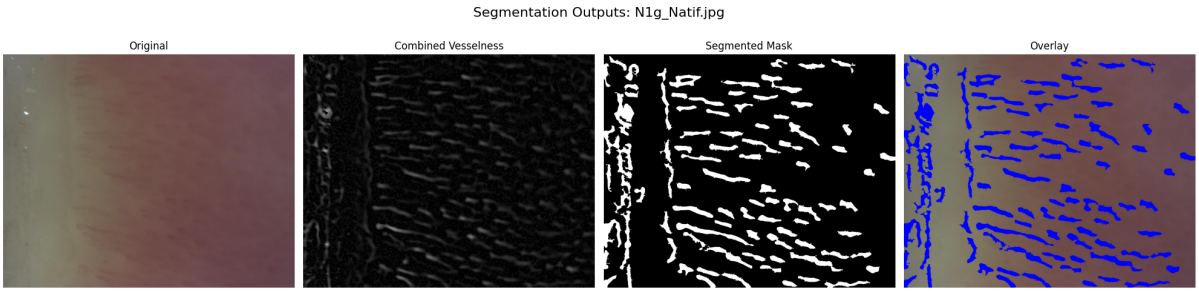


Figure 7: N1g Natif Segmented

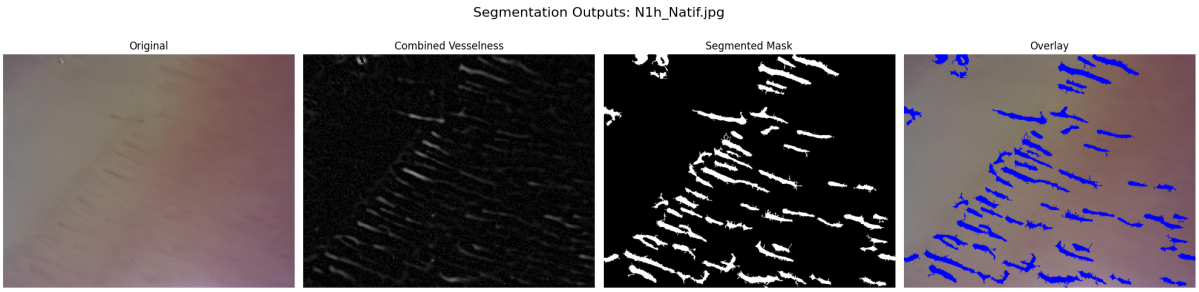


Figure 8: N1h Natif Segmented

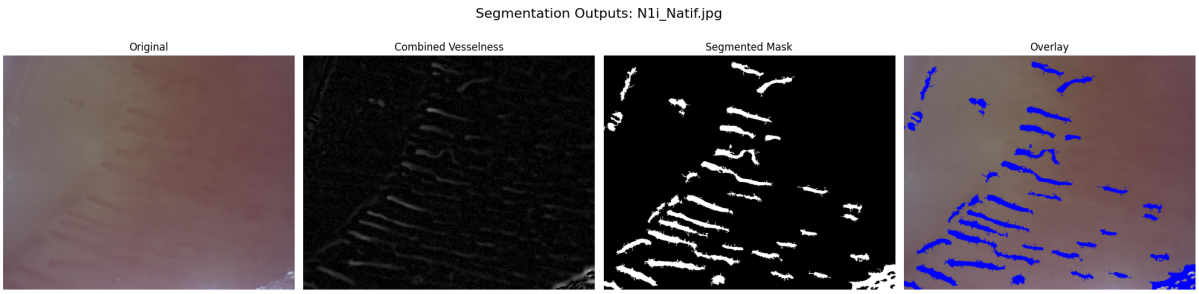


Figure 9: N1i Natif Segmented

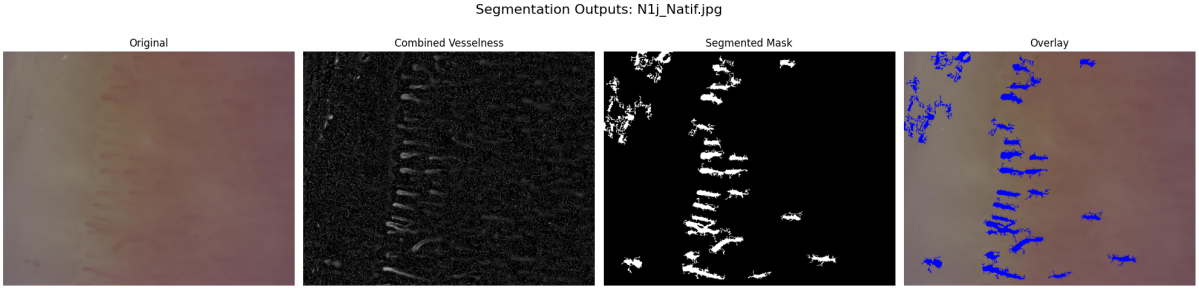


Figure 10: N1j Natif Segmented

6 N2 Segmentation Parameters and Results

Preprocessing

- **Median Filter (kernel size = 5)**: Used to reduce noise while preserving vessel edges.
- **CLAHE (clip limit = 3.0, tile grid size = 8×8)**: Enhances local contrast to better reveal capillary structures in varying illumination conditions.

Vessel Enhancement

Applied the same three vesselness filters (Frangi, Sato, Meijering) and combined their outputs by averaging to robustly highlight vessels.

Thresholding and Cleaning

- **Otsu's Thresholding**: Automatically determines the threshold separating vessels from the background.
- **Morphological Operations**:
  - Closing with a 3×3 square structuring element to fill small gaps.
  - Removed small objects smaller than 300 pixels to eliminate noise (lower than N1 since N2 images may have finer vessel details).
  - Removed small holes smaller than 50 pixels to fill small gaps inside vessel areas (also lower to preserve fine capillary structure).

Results

This adapted pipeline yielded clean binary vessel masks optimized for the N2 image characteristics, capturing finer capillary details compared to N1 segmentation.

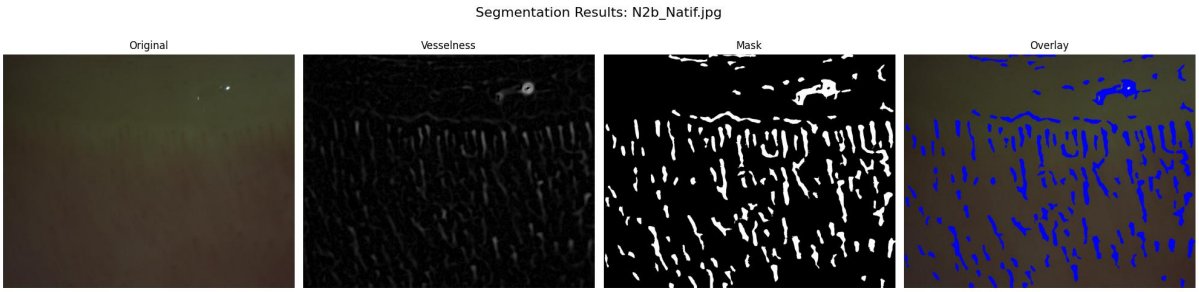


Figure 11: N2b Natif Segmented

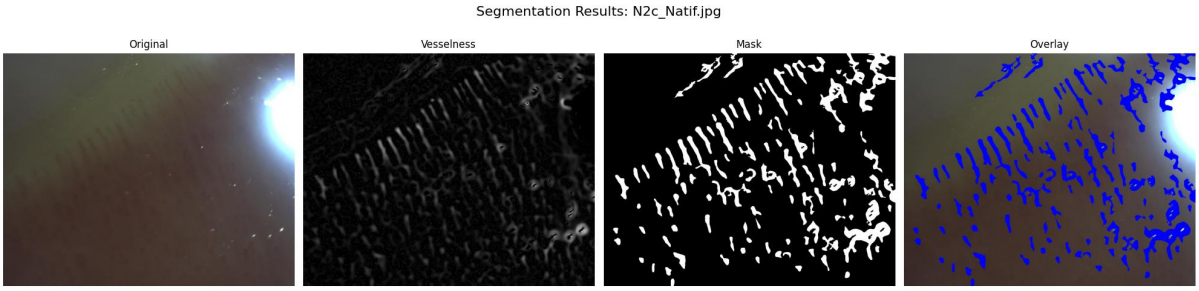


Figure 12: N2c Natif Segmented

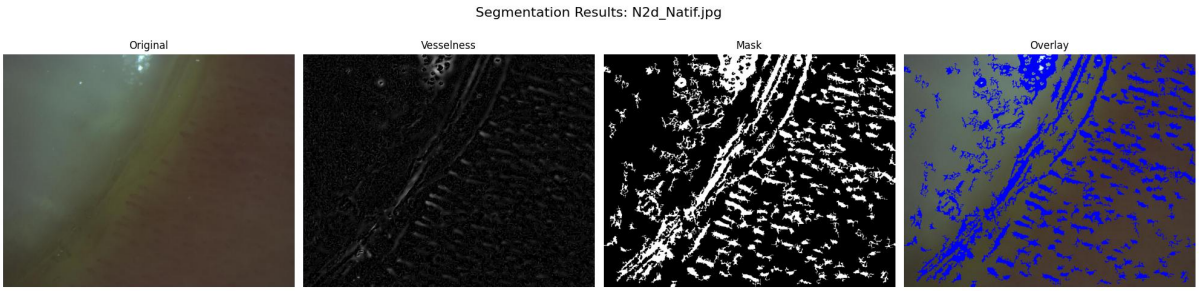


Figure 13: N2d Natif Segmented

7 S2 Segmentation Parameters and Results

Preprocessing

- **No median filtering**: Skipped median blur because the S2 images were less noisy or to preserve fine vessel details.
- **CLAHE (clip limit = 2.0, tile grid size = 8×8)**: Contrast enhancement with slightly lower clip limit than N2 to avoid over-amplifying noise, fitting the image characteristics.

Vessel Enhancement

Applied the same three vesselness filters (Frangi, Sato, Meijering) and averaged their outputs to robustly highlight vessels.

Thresholding and Cleaning

- **Otsu's Thresholding**: Automatically calculates the threshold for vessel segmentation.
- **Morphological operations**:
  - Closing with a 3×3 square structuring element to fill gaps.
  - Removed small objects smaller than 100 pixels (smaller than N2/N1, because S2 images likely have finer vessels or less noise).
  - Removed small holes smaller than 50 pixels to fill small gaps inside vessels.

Results

This pipeline was tailored to maintain fine vessel structures with minimal smoothing and a conservative small object removal size, providing clean and accurate segmentation for S2 images.

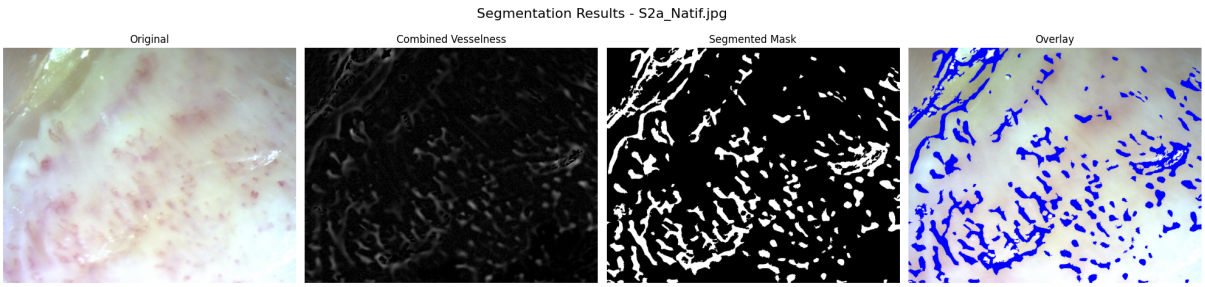


Figure 14: S2a Natif Segmented

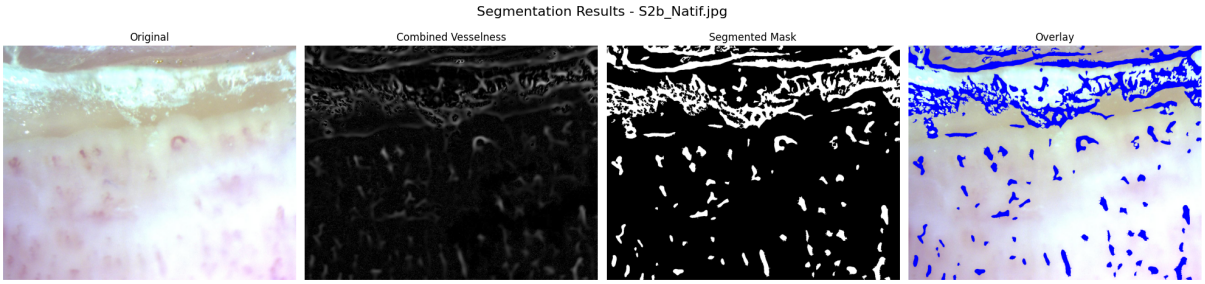


Figure 15: S2b Natif Segmented

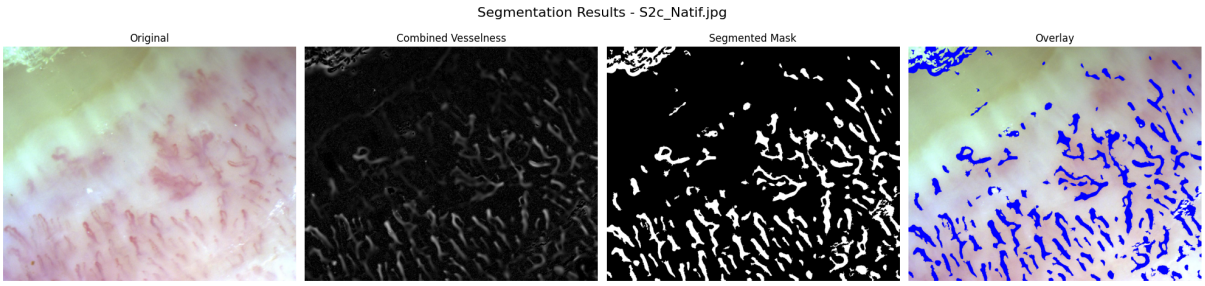


Figure 16: S2c Natif Segmented

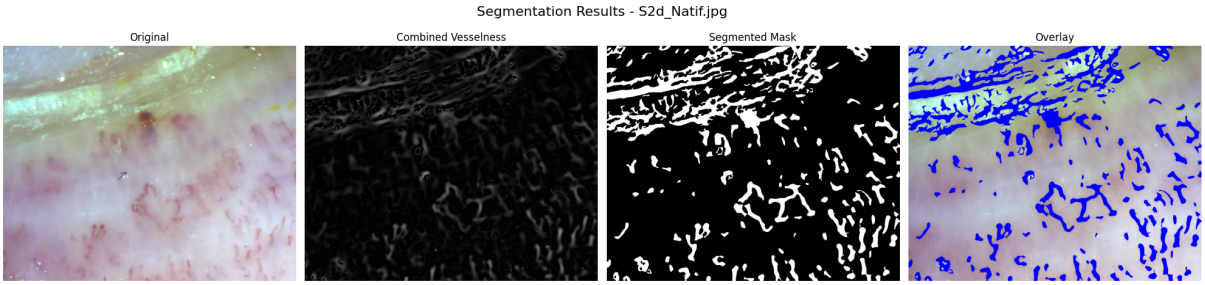


Figure 17: S2d Natif Segmented

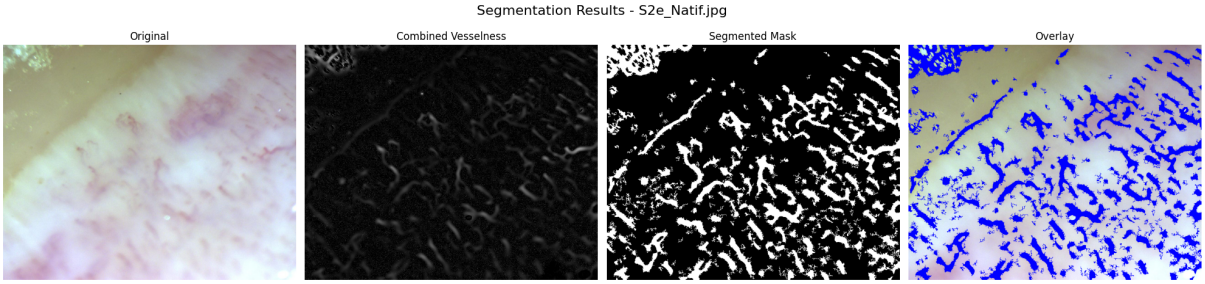


Figure 18: S2e Natif Segmented

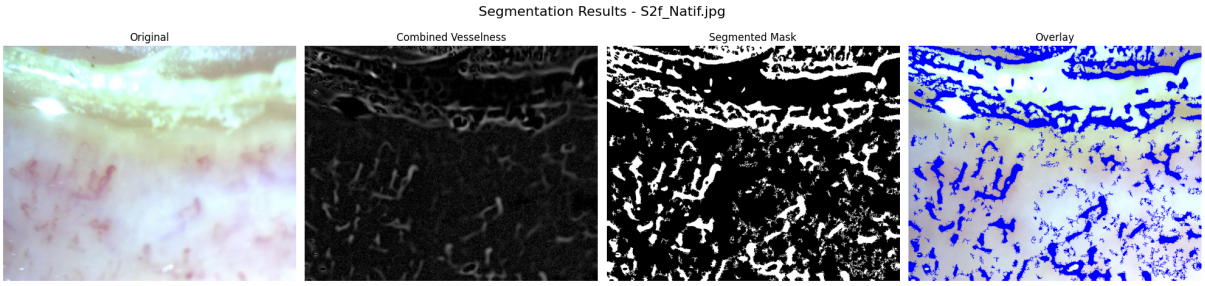


Figure 19: S2f Natif Segmented

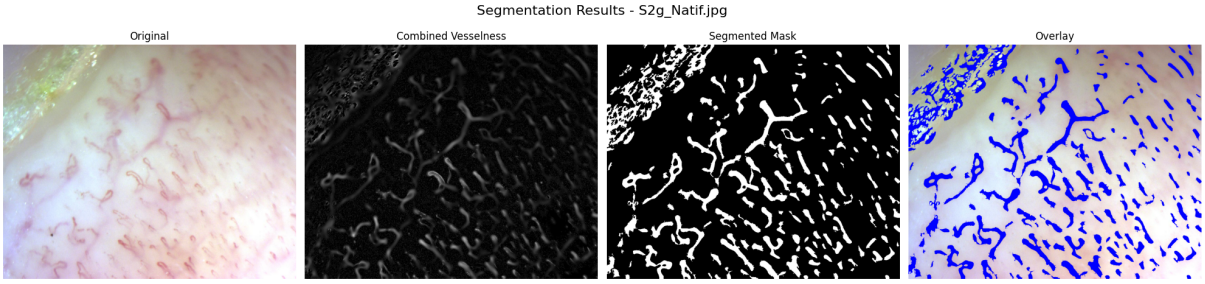


Figure 20: S2g Natif Segmented

8 S3 Native Parameters and Results

Preprocessing

- **Median Filtering**: Applied a median blur with kernel size 5 (`cv2.medianBlur(img_gray, 5)`) to reduce noise while preserving edges. This is more aggressive than for S2, indicating that S3 images might be noisier.
- **CLAHE (Contrast Limited Adaptive Histogram Equalization)**:
  - Clip limit: 4.0 (higher than S2 and S1) to enhance local contrast strongly.
  - Tile grid size: 4×4, smaller tiles for finer local contrast adjustment — useful for highlighting small vessels in potentially lower contrast images.

Vessel Enhancement

Applied Frangi, Sato, and Meijering vesselness filters on the normalized image (range 0–1). The three filter responses were averaged to obtain a robust combined vesselness map.

Thresholding

Otsu's thresholding: Automatically determines the threshold to binarize the vesselness image. The threshold value is printed for insight/debugging.

Morphological Cleaning

- **Closing**: Morphological closing with a 3×3 square structuring element to fill small gaps inside vessel structures.
- **Remove small objects**: Minimum object size set to 300 pixels, larger than S2 and S1, likely to exclude noise and keep only significant vessel regions.
- **Remove small holes**: Fills holes smaller than 70 pixels to smooth segmented vessels.

Results

Saves original images, vesselness maps (scaled 0–255), binary masks, and overlay images with segmented vessels highlighted in red. Visualizes results using `matplotlib` for quick inspection.

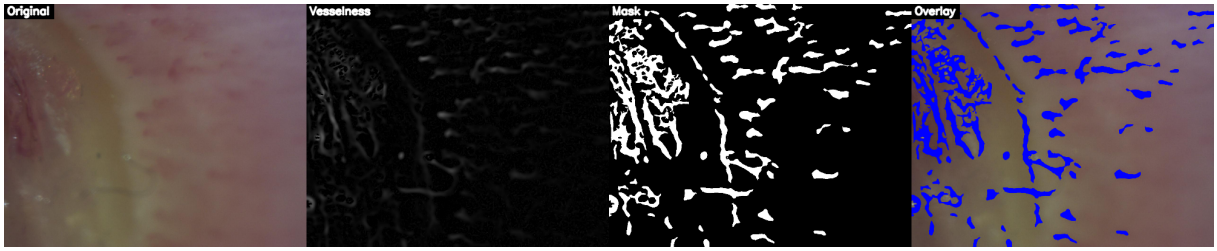


Figure 21: S3a Natif Segmented

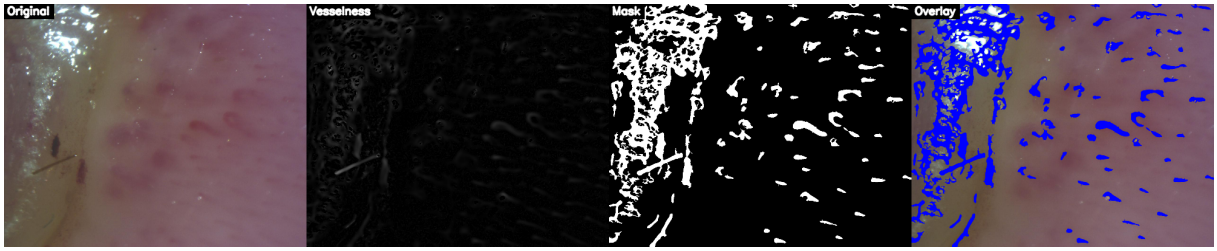


Figure 22: S3b Natif Segmented

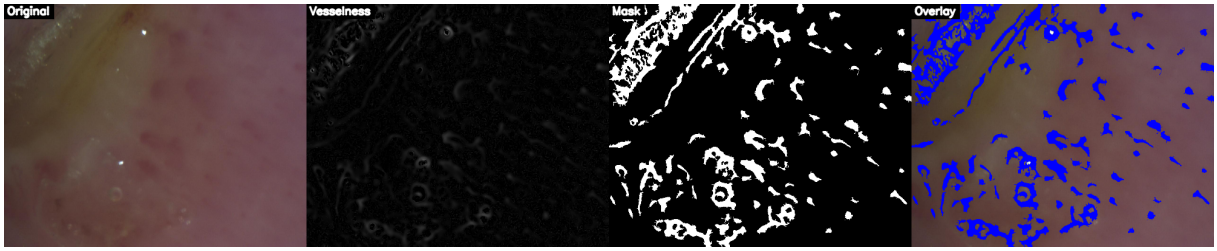


Figure 23: S3c Natif Segmented

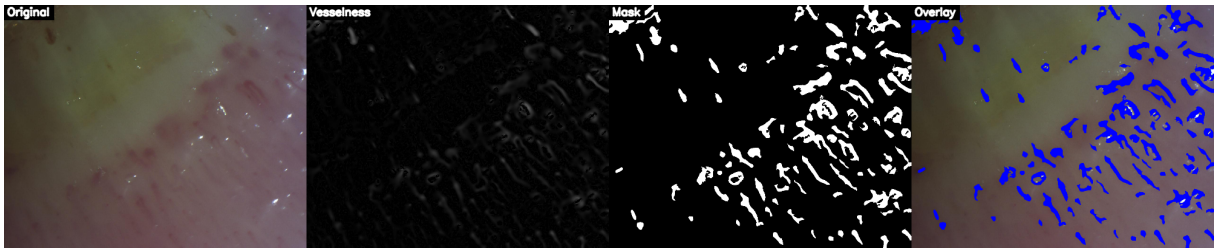


Figure 24: S3d Natif Segmented

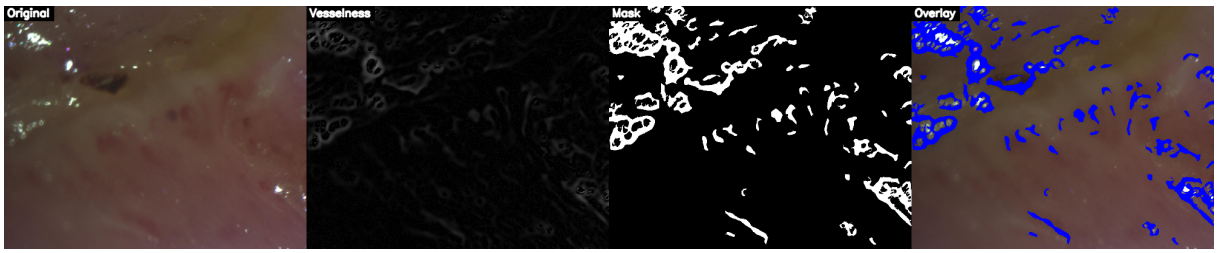


Figure 25: S3e Natif Segmented

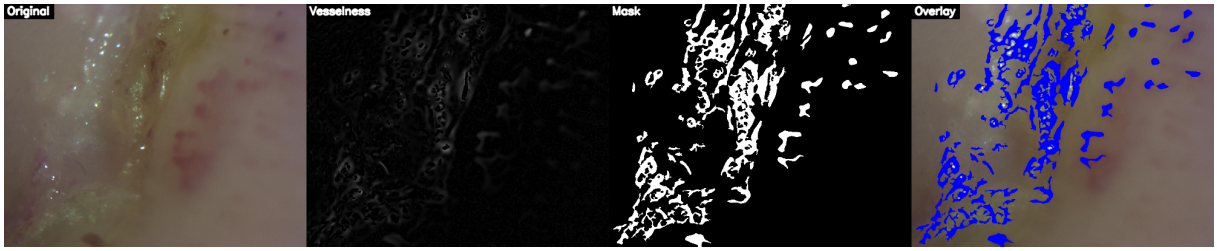


Figure 26: S3f Natif Segmented

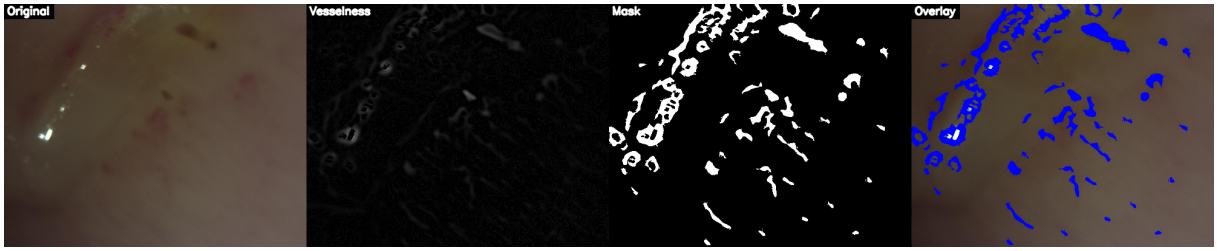


Figure 27: S3g Natif Segmented

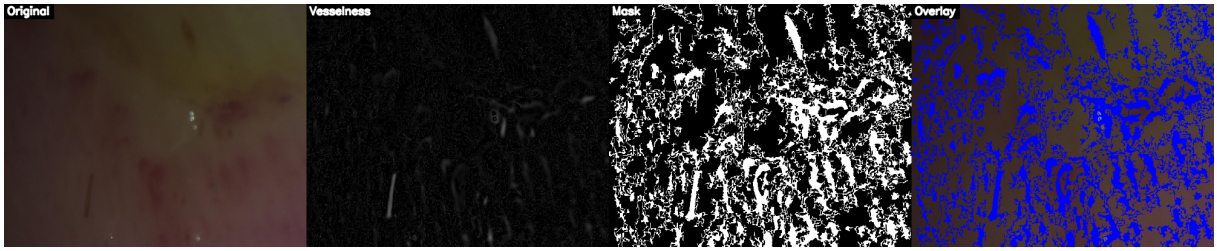


Figure 28: S3h Natif Segmented

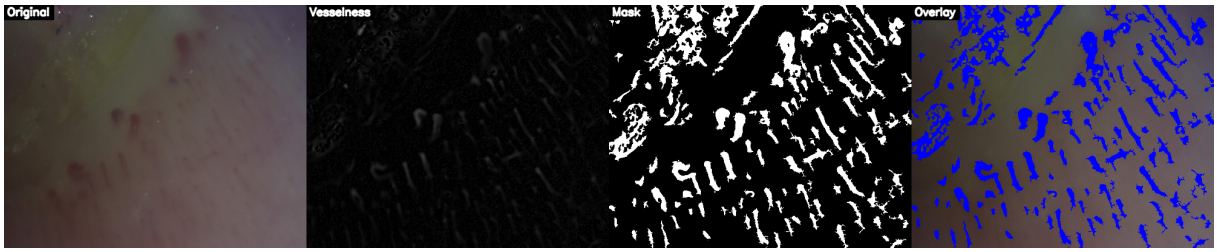


Figure 29: S3i Natif Segmented

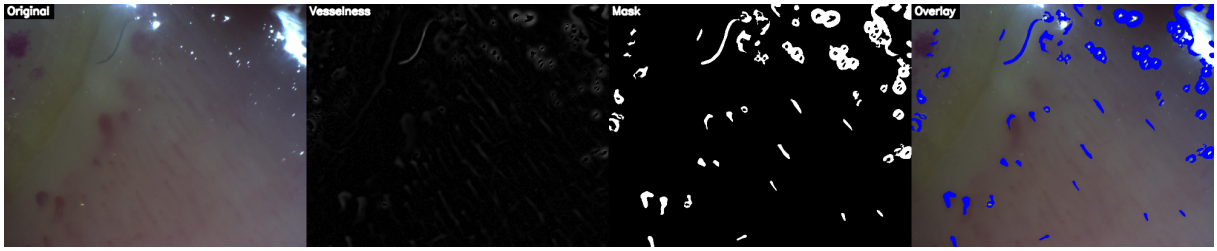


Figure 30: S3j Natif Segmented

9 Capillary Counting on Dataset N1

After performing segmentation on the N1 dataset, we applied a capillary counting algorithm to quantify the number of capillaries in each region of interest (ROI).

The counting process begins by detecting a reference black line in the ROI image, which allows us to correct the orientation through rotation for consistent analysis. The segmentation mask is then rotated by the same angle to align with the corrected ROI. A bounding box is calculated around the region of interest, and the mask is cropped accordingly.

Connected component analysis is applied to the cropped binary mask to identify individual capillaries as distinct connected regions. The total number of connected components, minus the background, gives the capillary count for that sample.

Visualization

Figure 40 The four panels illustrate: (1) the original ROI image, (2) the ROI after rotation, (3) the rotated segmentation mask with the bounding box indicating the counting area, and (4) the cropped mask with the detected capillaries highlighted.

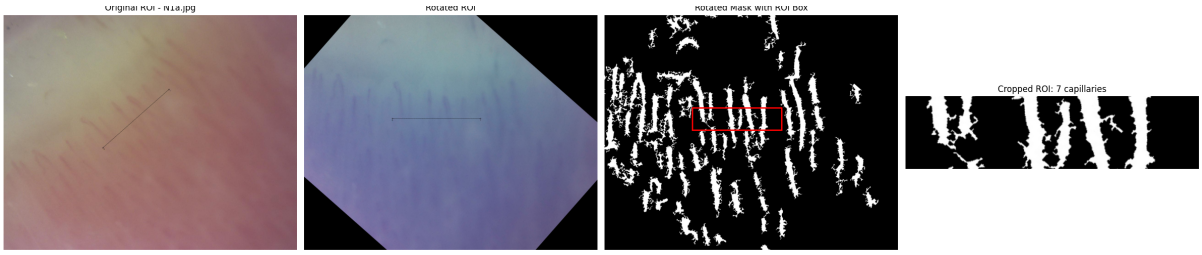


Figure 31: Capillary counting visualization for N1a.

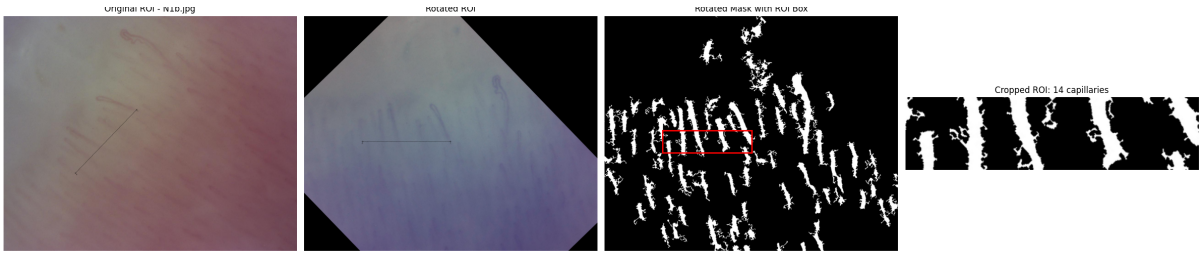


Figure 32: Capillary counting visualization for N1b.

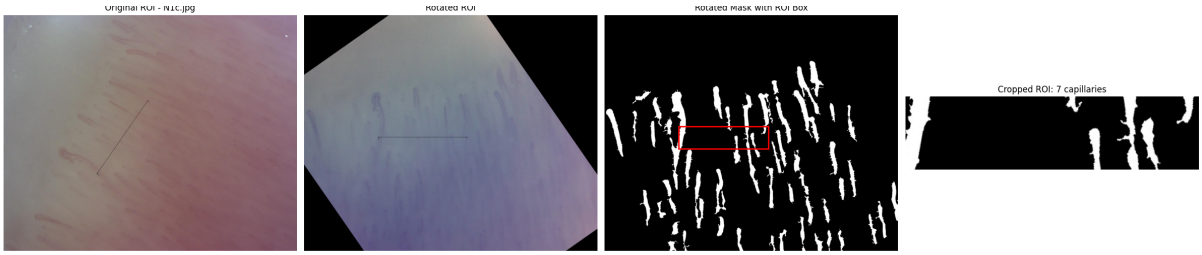


Figure 33: Capillary counting visualization for N1c.

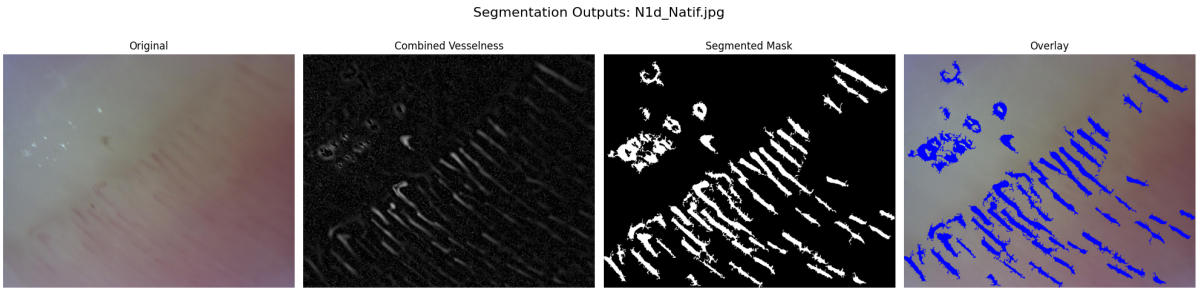


Figure 34: Capillary counting visualization for N1d.

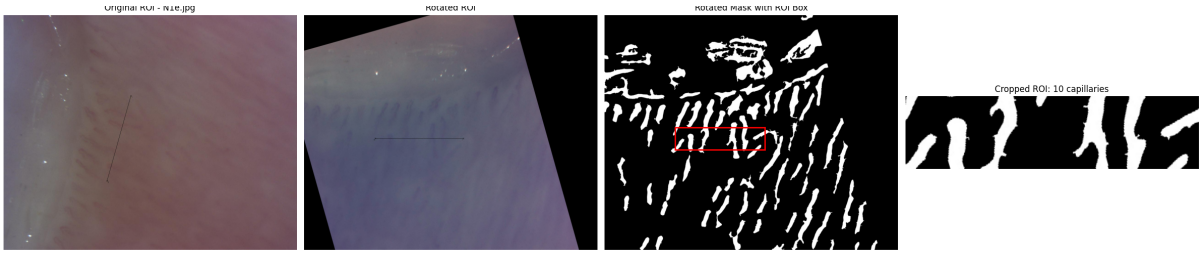


Figure 35: Capillary counting visualization for N1e.

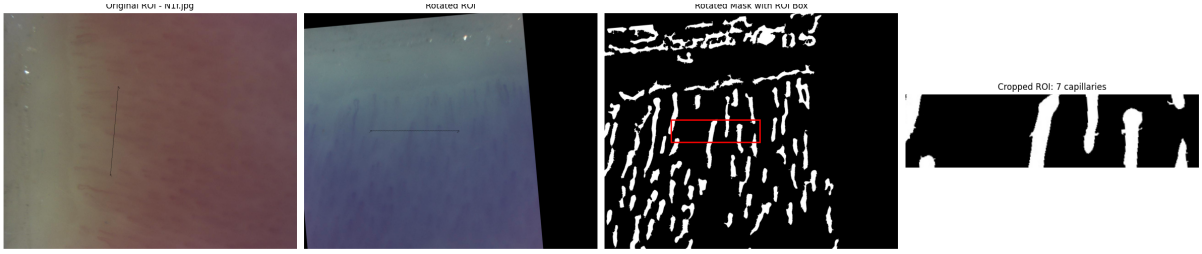


Figure 36: Capillary counting visualization for N1f.

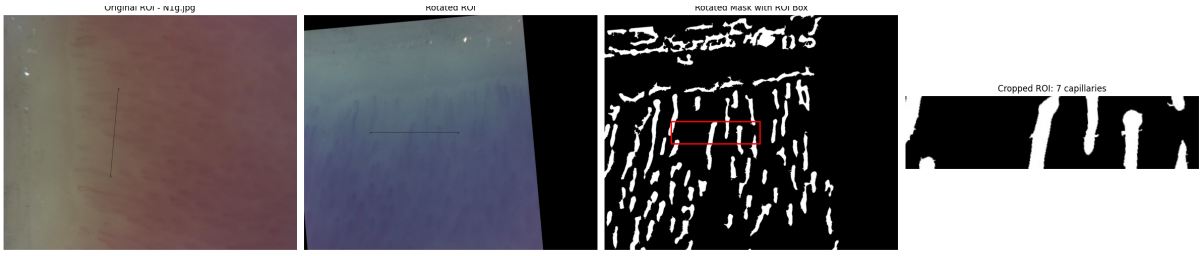


Figure 37: Capillary counting visualization for N1g.

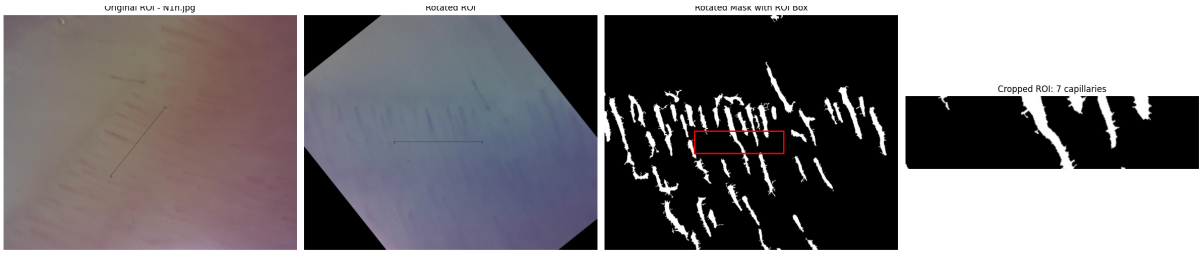


Figure 38: Capillary counting visualization for N1h.

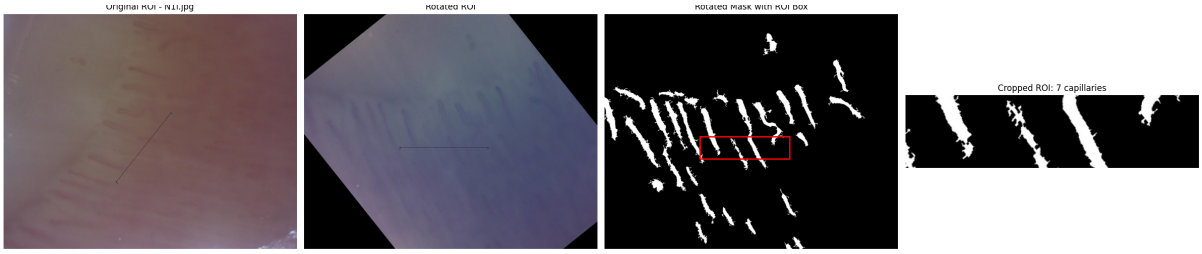


Figure 39: Capillary counting visualization for N1i.

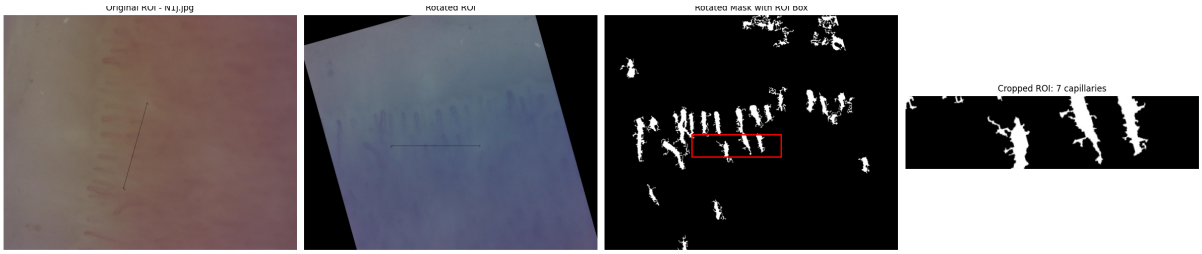


Figure 40: Capillary counting visualization for N1j.

Results Summary

The capillary counts for the N1 dataset images are summarized in Table. The number of capillaries per image varies depending on the sample and the region analyzed. These counts provide a quantitative measure that can be used for further statistical analysis or comparisons across datasets.

Table 1 shows the number of capillaries detected by the algorithm alongside the ground truth counts for each image in the N1 dataset.

Table 1: Comparison of detected capillary counts and ground truth values for N1 dataset images

Image Filename	Detected Count	Ground Truth
N1a.jpg	7	6
N1b.jpg	14	7
N1c.jpg	7	9
N1d.jpg	11	8
N1e.jpg	10	8
N1f.jpg	7	9
N1g.jpg	7	9
N1h.jpg	7	7
N1i.jpg	7	7
N1j.jpg	7	8

Discussion

The detected capillary counts show some discrepancies compared to the ground truth annotations. For some images, such as `N1b.jpg`, the detected count (14) is significantly higher than the ground truth (7), suggesting potential over-segmentation or false positives in the counting process.

Conversely, for images like `N1c.jpg` and `N1f.jpg`, the detected counts are lower than the ground truth, indicating possible under-segmentation or missed capillaries.

These differences highlight areas for improvement in the segmentation and counting pipeline, such as refining mask accuracy or post-processing to reduce false positives.

Further statistical analysis (e.g., calculating accuracy, precision, recall) can provide deeper insights into the performance of the algorithm.

Conclusion

In this project, we implemented a complete segmentation and analysis pipeline for four different capillaroscopic datasets: **N1, N2, S2,** and **S3**. Each dataset posed unique challenges in terms of image quality and capillary visibility. To address this, we experimented with multiple variations of classical image processing techniques — including vesselness filtering, morphological operations, and adaptive thresholding — and adjusted parameters accordingly for each dataset. Through iterative tuning and visual analysis, we identified the most effective method for segmenting the capillaries across different datasets.

Following successful segmentation, we conducted capillary counting specifically for the **N1** dataset, utilizing the predefined region of interest (ROI) described in the project documentation. This involved detecting and aligning the ROI based on black reference lines and applying connected components analysis to estimate the number of distinct capillaries present. Our method showed a reasonable correlation with the ground truth values, highlighting the effectiveness of our approach while also revealing potential areas for improvement.

While classical methods provided a solid foundation, the project results suggest that further enhancements could be achieved using machine learning or deep learning techniques to improve both segmentation accuracy and counting robustness. Integrating these approaches could help reduce false positives/negatives and generalize better across varying image conditions.

In summary, this project demonstrates a functional and adaptable approach to capillary segmentation and counting, providing useful insights and laying the groundwork for future development in automated biomedical image analysis.

References

[1] "scikit-image: skimage.filters.frangi — Frangi vesselness filter," https://scikit-image.org/docs/0.25.x/api/skimage.filters.html#skimage.filters.frangi, 2024, [Online; accessed 09-June-2025].

[2] "scikit-image: skimage.filters.sato — Sato vesselness filter," https://scikit-image.org/docs/0.25.x/api/skimage.filters.html#skimage.filters.sato, 2024, [Online; accessed 09-June-2025].

[3] "scikit-image: skimage.filters.meijering — Meijering vesselness filter," https://scikit-image.org/docs/0.25.x/api/skimage.filters.html#skimage.filters.meijering, 2024, [Online; accessed 09-June-2025].