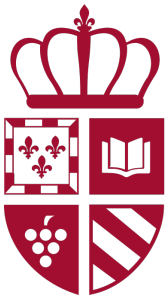
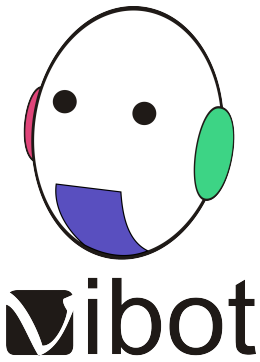


UNIVERSITÉ BOURGOGNE EUROPE



Nailfold Capillaroscopy Capillaries Segmentation

Course: Medical Image Analysis

Authors: Muhammad Usama Javaid
Arham Sajjad

Instructor: Prof. Dr. Stephanie Bricq

Date: June 9, 2025

ViBOT - Vision and Robotics

CONDORCET UNIVERSITY CENTER, LE CREUSOT

UNIVERSITE DE BORGOGNE, DIJON

Contents

1 Introduction **1**
1.1 Background . . . 1
1.2 Non-Machine Learning-Based Approaches for Nailfold Capillaroscopy Capillary Segmentation 2

2 Thresholding-based methods **3**
2.1 Global Thresholding . . . 3
2.2 Adaptive Thresholding . . . 3
2.3 Otsu's Thresholding . . . 4

3 Methodology **5**
3.1 Line detection . . . 6
3.2 Crop and rotate image . . . 7
3.3 RGB channel splitting and Green channel enhancement . . . 7
3.4 Denoising . . . 8
3.5 Capillary Segmentation and Contouring . . . 8

4 Implementation and Results **9**
4.1 Test data . . . 9
4.2 Line detection . . . 11
4.3 Cropping and Rotating . . . 12
4.4 Green Channel enhancement and Denoising . . . 13
4.5 Capillary segmentation and contouring . . . 14

5 Discussion **15**
5.1 Results . . . 15
5.2 Advantages . . . 16
5.3 Limitations . . . 16
5.4 Future works . . . 16

6 Conclusion **17**

7 Appendix - I - Refrences **i**

List of Figures

1 Nailfold capillaroscopy 1
2 Test data with accurate results 9
3 Test data with almost accurate results 10
4 Test data with inaccurate results 10
5 Test data masked with the detected line 11
6 Test data cropped and rotated with reference to the detected line 12
7 Enhanced Green Channel and Denoised enhanced green channel images of the test data . . . 13
8 Segmented and contoured capillaries 14

1 Introduction

1.1 Background

Nailfold capillaroscopy is a simple, non-invasive technique used to examine the tiny blood vessels (capillaries) in the skin at the base of the fingernails or toenails. Using a microscope or a handheld device, doctors can get a close-up view of these capillaries, making it easier to detect abnormalities in blood flow or vessel structure. This method has become especially valuable in fields like rheumatology, dermatology, and vascular medicine because of its ability to help diagnose and monitor a range of conditions.The capillaries in the nailfold offer a unique glimpse into how the body's microvascular system is functioning. Changes in their shape, size, or arrangement—such as being enlarged, twisted, or even missing—can point to underlying health issues. Nailfold capillaroscopy is particularly helpful in diagnosing connective tissue diseases like systemic sclerosis (SSc), where specific capillary changes are closely linked to how advanced the disease is and how quickly it may be progressing.where specific capil- lary changes are associated with disease severity and progression.

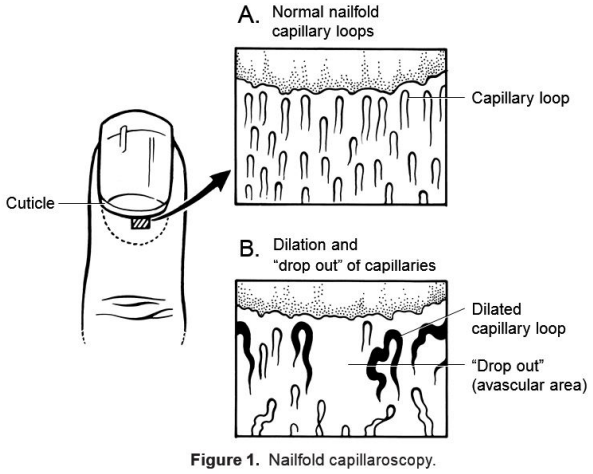


Figure 1: Nailfold capillaroscopy

Traditionally, examining nailfold capillaroscopy images has depended on experts visually interpreting the images by hand. This process, however, is subjective, takes a lot of time, and can vary between different observers. Recently, there's been increasing interest in creating automated segmentation methods that can objectively measure and analyze the shape and structure of capillaries. These automated approaches aim to deliver more precise and consistent results, helping with early diagnosis, tracking how diseases evolve, and evaluating how well treatments are working.

Although machine learning and deep learning techniques have shown great potential in segmenting medical images, this report concentrates on capillary segmentation methods that do not use these advanced algorithms. Instead, it focuses on more traditional image processing techniques to identify and extract capillary features from nailfold images.

Traditional Methods for Nailfold Capillaroscopy Capillary Segmentation (Without Machine Learning)

The goal of this report is to review and assess one of the existing non-machine learning methods used for segmenting capillaries in nailfold capillaroscopy images. By looking into the underlying principles, strengths, and weaknesses of these techniques, we hope to offer a clearer understanding of their reliability, suitability, and where improvements might be needed. Ultimately, this work aims to support the development of more accurate and dependable segmentation methods that can improve the clinical value of nailfold capillaroscopy across different medical fields.

After thorough research, we identified four main approaches for non-machine learning-based capillary segmentation in nailfold capillaroscopy. Below, we provide a brief overview of each method:

1. **Thresholding-based Methods:** These techniques involve choosing a threshold value to separate the capillaries from the background in the image. Various thresholding approaches—like global thresholding, adaptive thresholding, or Otsu's method—are used to segment the capillaries. The underlying idea is that capillaries usually have different intensity or color values compared to the surrounding area.

2. **Morphological Operations:** Morphological operations such as erosion, dilation, opening, and closing are applied to enhance and clean up the capillary structures. These steps help remove noise, close gaps, and separate clusters of capillaries. When combined with thresholding, morphological operations can significantly improve segmentation resul**ts.**

3. **Edge Detection:** Edge detection techniques are used to find the borders of capillaries by detecting changes in intensity. Popular methods like the Sobel operator, Canny edge detection, or the Laplacian of Gaussian (LoG) help identify the edges of the capillary network**s.**

4. **Region Growing:** This approach begins with a seed point and gradually expands the area by including neighboring pixels that meet specific criteria, such as similar intensity values. By continuously adding pixels, the capillary regions are outlined. Region growing works well when capillaries have fairly consistent intensities across their structur**es.**

2 Thresholding-based methods

After careful consideration, the thresholding-based method was selected as the preferred approach for capil- lary segmentation in this study. Thresholding-based methods offer several advantageous features for nailfold capillaroscopy analysis. Firstly, they demonstrate computational efficiency and relative ease of implementa- tion, rendering them accessible to researchers and clinicians without extensive expertise in advanced image processing techniques. By establishing a threshold value, these methods enable the discrimination between capillaries and the background, thus facilitating the extraction of pertinent information pertaining to capil- lary density, tortuosity, and other essential morphological characteristics. Furthermore, thresholding-based methods do not necessitate a substantial volume of training data or intricate model training procedures, which are often time-consuming and resource-intensive. Instead, they leverage the inherent intensity or color discrepancies between capillaries and the surrounding tissue. This simplicity and efficiency render thresholding-based methods an appealing option for capillary segmentation, particularly in scenarios where the adoption of machine learning or deep learning approaches may prove impractical due to limited resources or specific research constraints. By employing thresholding-based methods, researchers can obtain depend- able and interpretable capillary segmentations, thereby facilitating subsequent analysis and aiding in the diagnosis and monitoring of microvascular diseases.

There are three main thresholding techniques that we can use i.e., **global thresholding**, **adaptive thresh- olding** and **otsu's thresholding method**. A brief explanation of the before mentioned methods are as following:

2.1 Global Thresholding

Global thresholding is a simple thresholding technique where a single threshold value is applied to the entire image to separate foreground (object of interest) from the background. The threshold value is chosen based on certain criteria, such as maximizing inter-class variance or minimizing intra-class variance.

The basic idea behind global thresholding is to select a threshold value that optimally separates the fore- ground and background pixels based on a global characteristic of the image intensity histogram.

Let I represent the input image and T be the threshold value. The segmentation function S for global thresholding can be defined as:

$$\begin{cases} 1 & \text{if } I(x,y) \geq T \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

2.2 Adaptive Thresholding

Adaptive thresholding is a technique that uses a local threshold value that varies across different regions of an image. It is particularly useful when the image has uneven illumination or varying contrast. Instead of applying a single threshold value to the entire image, adaptive thresholding computes a threshold value for each pixel based on its local neighborhood.

The fundamental idea behind adaptive thresholding is to adaptively determine the threshold value based on the local characteristics of the image. This ensures that the thresholding process is robust to local variations in illumination and contrast.

Let I epresent the input image, and $T(x,y)$ denote the threshold value at pixel (x,y) computed from the local neighborhood. The segmentation function S for adaptive thresholding can be defined as:

$$\begin{cases} 1 & \text{if } I(x,y) \geq T(x,y) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

2.3 Otsu's Thresholding

Otsu's thresholding is a widely used method to automatically determine an optimal threshold value for image segmentation. It aims to minimize the intra-class variance of the foreground and background pixels, making it useful when the histogram of the image is bimodal (contains distinct peaks).

Otsu's thresholding assumes that the image contains two classes of pixels (foreground and background) and aims to find the threshold that minimizes the weighted sum of intra-class variances.

Let I represent the input image and T be the threshold value computed by Otsu's method. The segmentation function S for Otsu's thresholding can be defined as:

$$\begin{cases} 1 & \text{if } I(x,y) \geq T \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

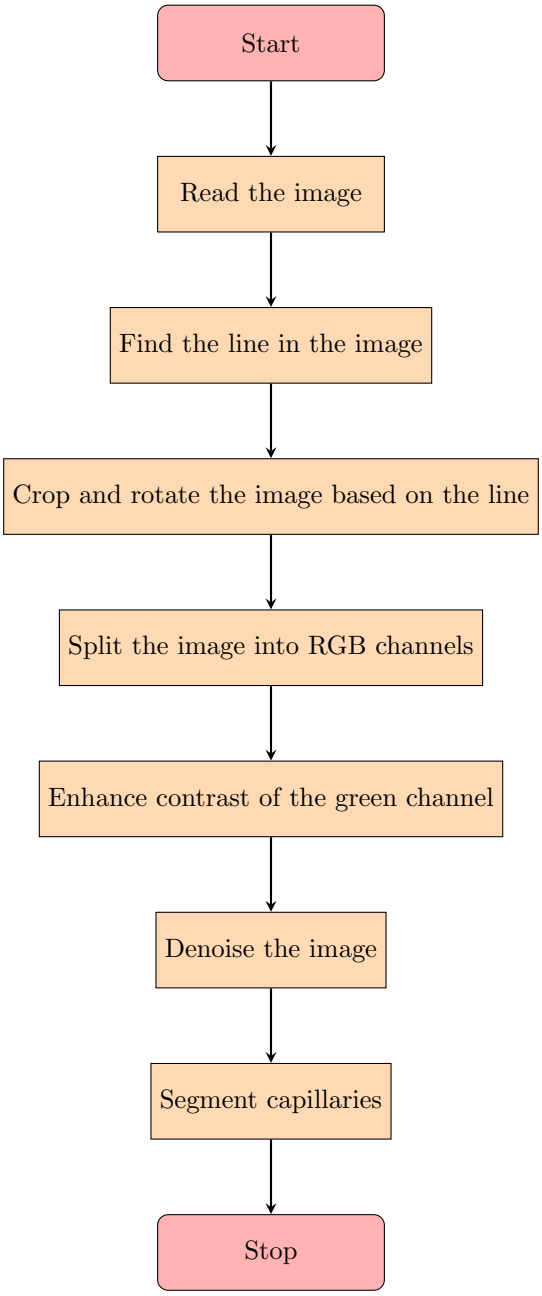
For segmenting the capillaries, we chose the adaptive thresholding method because it offers clear advantages over both global thresholding and Otsu's method when working with nailfold capillaroscopy images. While global thresholding is simple, it applies a single threshold value to the entire image, which can be problematic when the image has uneven lighting, varying contrast, or noise—common issues in nailfold capillaroscopy due to the imaging process and the complex structure of capillaries.

Adaptive thresholding, by contrast, calculates a threshold for each pixel based on its local surroundings. This makes it much better suited to handle local differences within the image. Since capillaries in nailfold images vary widely in shape, size, and density across different areas, this local adaptation helps distinguish capillaries from the background more accurately, resulting in better segmentation.

Additionally, Otsu's method assumes that pixel intensities form a bimodal distribution, which isn't always true for nailfold capillaroscopy images. This assumption can limit Otsu's ability to capture the subtle and intricate details of the capillary patterns. Adaptive thresholding doesn't depend on this assumption and can adapt to a wide range of intensity distributions, allowing it to detect finer details and more precisely outline the complex capillary networks.

3 Methodology

In this study focused on segmenting capillaries in nailfold capillaroscopy images, we worked with a diverse collection of images that varied in lighting, environmental conditions, and included both healthy and diseased patients. Our main goal was to precisely segment and measure the capillaries that cross or touch a designated line within each image. To accomplish this, we developed a structured approach, which is illustrated in the flowchart below, highlighting the main steps of our method.



The flowchart above visually outlines the step-by-step process we followed for segmentation. Using this organized approach helped us maintain consistency and achieve reliable, repeatable results across the different types of images included in the study.

It is important to note that this flowchart serves as a general overview of our methodology, highlighting the major stages of our capillary segmentation process. Detailed descriptions of each step, including the specific techniques employed for image preprocessing, thresholding, and contouring, are elaborated upon in subsequent sections of this report. The flowchart serves as a guide, aiding readers in understanding the overall framework of our approach while enabling them to delve into the specifics of each stage for a comprehensive understanding of our methodology.

3.1 Line detection

The first step is to detect the drawn line in the image and for this, we used the Probabilistic Hough Line Transform. The Probabilistic Hough Line Transform is a technique used in computer vision and image processing to detect lines in an image. It is particularly useful when dealing with images that contain lines with gaps or breaks. The transform is based on the Hough Transform, which is a general method for detecting shapes in an image by representing them as parameterized mathematical forms.

Given an image $I(x, y)$, the Hough Line Transform maps each point (x, y) in the image space to a sinusoidal curve in the parameter space (ρ, θ) using the equation:

$$\rho = x \cdot cos(\theta) + y \cdot sin(\theta) \tag{4}$$

where ρ represents the perpendicular distance from the origin to the line, and θ denotes the angle between the line and the x-axis.

The steps involved in Probabilistic Hough Line Transform are summarized as follows:

1. **Edge Detection**: Prior to applying the Hough Line Transform, we performed an edge detection operation on the image using the Canny edge detector. This highlights the regions of significant intensity gradients, emphasizing potential edges that may correspond to lines.
2. **Apply the Probabilistic Hough Line Transform:** The Probabilistic Hough Line Transform differs from the standard Hough Transform by not considering all possible combinations of ρ and θ values. Instead, it randomly selects a subset of edge points and finds the lines that pass through them.
3. **Define the parameters:** Set the desired parameters for the Probabilistic Hough Line Transform, such as the minimum line length, maximum gap between line segments, and the threshold for line detection. These parameters depend on the specific characteristics of your image and the lines you want to detect.
4. **Detect lines**: Apply the Probabilistic Hough Line Transform algorithm to the edge image, using the defined parameters. This will return the detected lines as line segments represented by their endpoints.

Once the lines are detected using the chosen method, we sort them by their length to prioritize the longest one. This helps ensure that the most significant and relevant line is selected first. After that, we draw a red mask along the detected line to visually confirm that the algorithm has accurately captured the entire length.

This sorting and masking step acts as a way to validate the detection results. The red mask makes the line stand out clearly, making it easier to verify that the algorithm successfully identified the full line.

3.2 Crop and rotate image

Having successfully detected the line of interest and visually validated its accuracy, the subsequent objective is to perform image rotation and subsequent cropping based on the detected line. The sequential steps involved in this process are outlined below:

1. **Image Cropping:** To crop the image, we first calculates the minimum and maximum values for the x and y coordinates of the region of interest (ROI) based on the given input points (the co-ordinates of the starting and ending points of our detected line) (x_1, y_1) and (x_2, y_2), and a margin value.
2. **Angle Calculation:** We then calculates the angle between the points $(x1, y1)$ and $(x2, y2)$. The resulting angle is converted from radians to degrees.
3. **Image Rotation:** The rotation is performed by defining a rotation matrix (M) using the calculated angle and the center of the image. The center of the image is computed as half the width (cols/2) and half the height (rows/2) of the image.
4. **Points Rotation and Image Cropping:** Next, we defines a rotation matrix (M) to rotate the four points (ROI coordinates) based on the center of the image and the calculated angle.
5. **Final Cropping:** Finally, the rotated image is cropped again based on the rotated points.

3.3 RGB channel splitting and Green channel enhancement

First, the image is separated into its three color channels: Red, Green, and Blue. We then focus on the green channel because it is particularly important for our analysis. The green channel provides useful information and has proven effective in enhancing features related to capillary structures. Several reasons support the choice of the green channel, which are explained bel**ow:**

1. **Better Contrast for Vascular Structures:** The green channel tends to exhibit enhanced contrast for vascular structures, including capillaries. This is because the green channel is less affected by the absorption of hemoglobin, which can distort the appearance of blood vessels in the red and blue channels. By isolating the green channel, it is possible to achieve clearer and more distinguishable capillary structures.
2. **Effective Discrimination of Capillaries from Surrounding Tissues:** Capillaries are typically surrounded by different types of tissues, and differentiating them from the background can be challenging. The green channel often provides a good balance between highlighting capillaries and suppressing unwanted features from surrounding tissues. This can be particularly useful when performing segmentation tasks, as it helps separate capillary structures from the rest of the image.

Upon isolating the green channel from the image, a crucial step in our methodology involves enhancing the contrast of this channel by performing histogram equalization. This process plays a pivotal role in capillary segmentation for the following reasons:

1. **Improved Visibility of Capillaries:** Capillaries are tiny, delicate blood vessels that are often hard to see clearly in medical images because of their small size and low contrast against surrounding tissues. By boosting the contrast in the green channel, the capillaries become much easier to spot, standing out more clearly from the background and other nearby structures. This improved visibility supports more accurate detection and segmentation of capillaries, which in turn leads to better analysis and diagno**sis.**
2. **Enhanced Differentiation:** The contrast enhancement of the green channel allows for better differentiation between capillaries and surrounding tissues or structures. By increasing the contrast, capillaries can stand out more prominently, enabling their differentiation from nearby vessels, background noise, or other anatomical features. This differentiation is crucial for extracting capillary-specific information and facilitating subsequent analysis or measurement tasks.

3. **Mitigation of Lighting and Imaging Artifacts:** Medical images, including those of capillaries, can be susceptible to various artifacts, such as uneven lighting, shadows, noise, or artifacts introduced during the imaging process. These artifacts can obscure or distort the appearance of capillaries, hindering accurate analysis and interpretation. Contrast enhancement of the green channel helps to mitigate the impact of such artifacts, making it easier to distinguish capillaries from background noise, correct uneven illumination, and minimize the influence of imaging artifacts.

By enhancing the contrast of the green channel, we improve the visibility, differentiation, sensitivity, and robustness of capillary structures in medical images. This enhancement enables more accurate segmentation, analysis, and interpretation of capillary networks, contributing to advancements in medical diagnosis, treatment planning, and research related to various diseases and conditions.

3.4 Denoising

After obtaining the enhanced isolated green channel from the image, a series of preprocessing steps are applied to refine the quality of the image. Firstly, a Gaussian blur is employed to effectively reduce white noise, thereby enhancing the clarity of the image. Subsequently, the smoothed image is subjected to a median filter, which effectively suppresses any impulsive noise present in the image. These sequential preprocessing steps collectively contribute to improving the overall quality and reliability of the image, enabling more accurate and robust subsequent analysis.

3.5 Capillary Segmentation and Contouring

Upon obtaining the denoised image, our pre-processing phase is completed, allowing us to proceed with capillary segmentation, contour generation, and the enumeration of capillaries intersecting or touching the designated line within the original image. To accomplish this, we employ the adaptive-thresholding technique, as elucidated in the preceding section. The ensuing steps are delineated below:

1. For each pixel in the denoised isolated green channel, a threshold value is calculated based on the local neighborhood of the pixel.
2. The local neighborhood is defined by the ***block size***, which determines the number of neighboring pixels considered for threshold calculation. A larger block size includes a larger area surrounding each pixel.
3. The threshold value is calculated as the mean (average) intensity of the pixels in the local neighborhood minus the constant $\boldsymbol{C}$.
4. If the intensity of the pixel is greater than or equal to the calculated threshold, it is set to the maximum intensity value (255 in our case). Otherwise, it is set to zero, hence returning a binary image.

After obtaining the binary image, the subsequent step involves contouring and counting the capillaries. The contouring process can be summarized as follows:

1. We analyzes the edges of the binary image to identify continuous curves or contours. We detect and extract the contours that separate regions of non-zero pixel values.
2. The contour detection algorithm traverses the edges of the image, identifying and linking adjacent edge pixels to form complete contours. It segments the image into distinct regions based on the contours.
3. The contours detected are returned as a list of curves or sets of connected points, where each contour is represented as a sequence of points or vertices.
4. The final step involves counting the capillaries present in the image. This is accomplished by determining the length of the list obtained in the preceding step. Since each capillary is stored as a distinct entry within the list, the total length of the list directly corresponds to the overall number of capillaries contained within it.

4 Implementation and Results

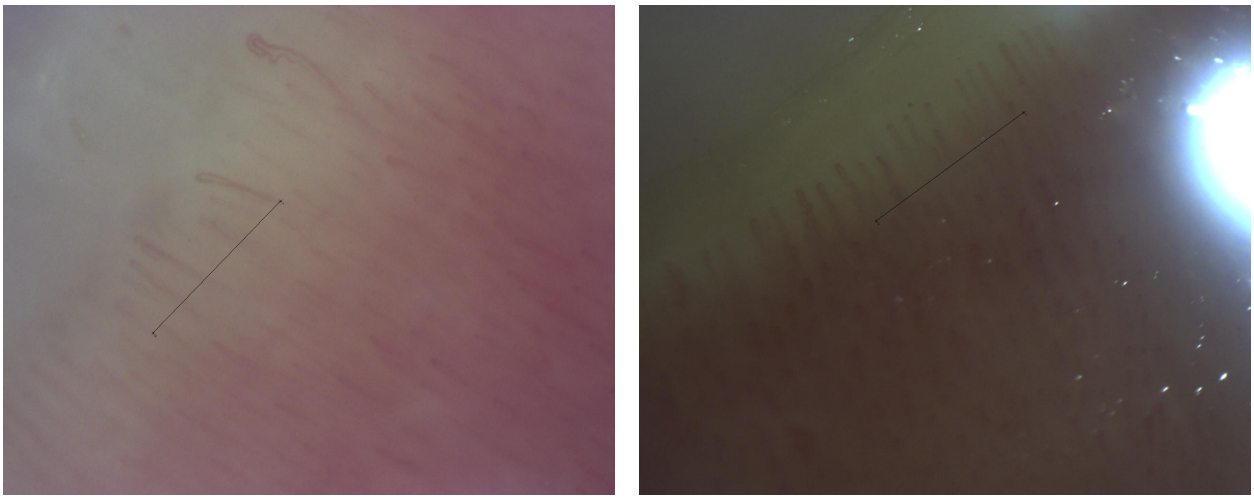
In this section, we'll explain the implementation steps based on the methodology described earlier. The project was developed using Python, selected for its wide range of reliable libraries with built-in functions that suited our needs. In particular, we made use of popular libraries like Numpy, OpenCV, and Matplotlib. The following subsections will walk through each step of the implementation in detail, along with the results we achieved.

4.1 Test data

For the purpose of this report, we are selecting 6 test images which showcase 3 cases. They are as following:

1. Accurate results
2. Near to accurate results
3. Wrong results

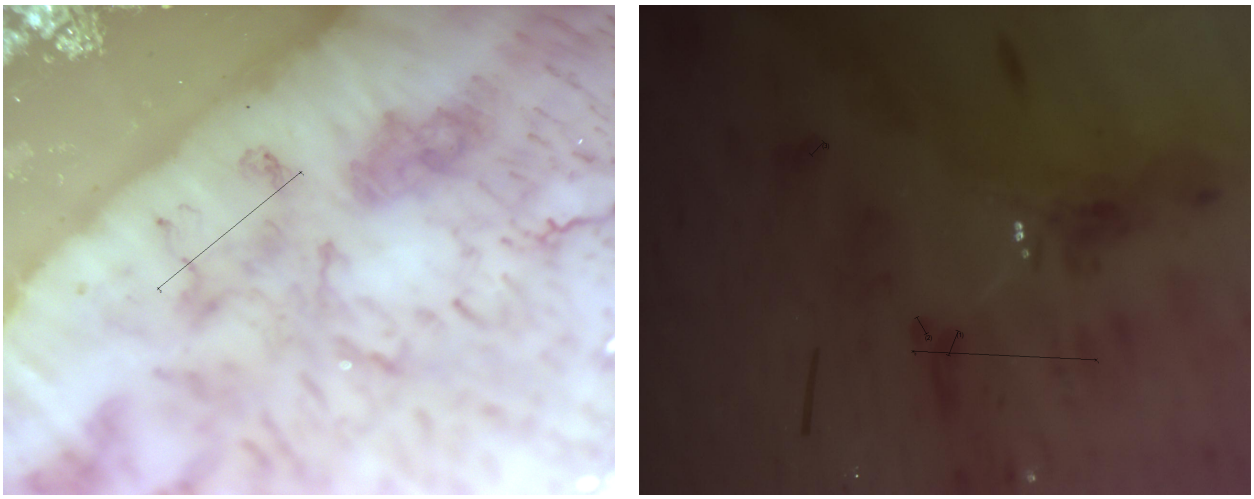
The test data is shown in the figures bellow:



(a) N1b (b) N2c

Figure 2: Test data with accurate results

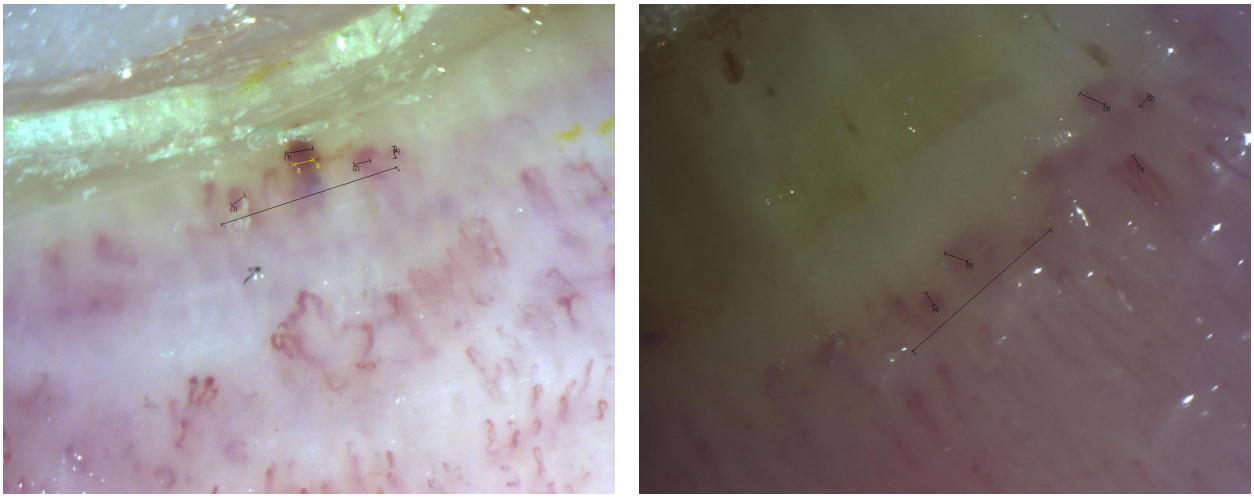
The above figure (2) shows two figures (2a) and (2b). These two figures, among others, showed accurate results when passed to our algorithm.



(a) S2e (b) S3h

Figure 3: Test data with almost accurate results

The above figure (3) shows two figures (3a) and (3b). These two figures, among others, showed almost accurate results, meaning they were off by 1 or 2 counts when passed to our algorithm.



(a) S2d (b) S3d

Figure 4: Test data with inaccurate results

The above figure (4) shows two figures (4a) and (4b). These two figures, among others, showed inaccurate results when passed to our algorithm.

4.2 Line detection

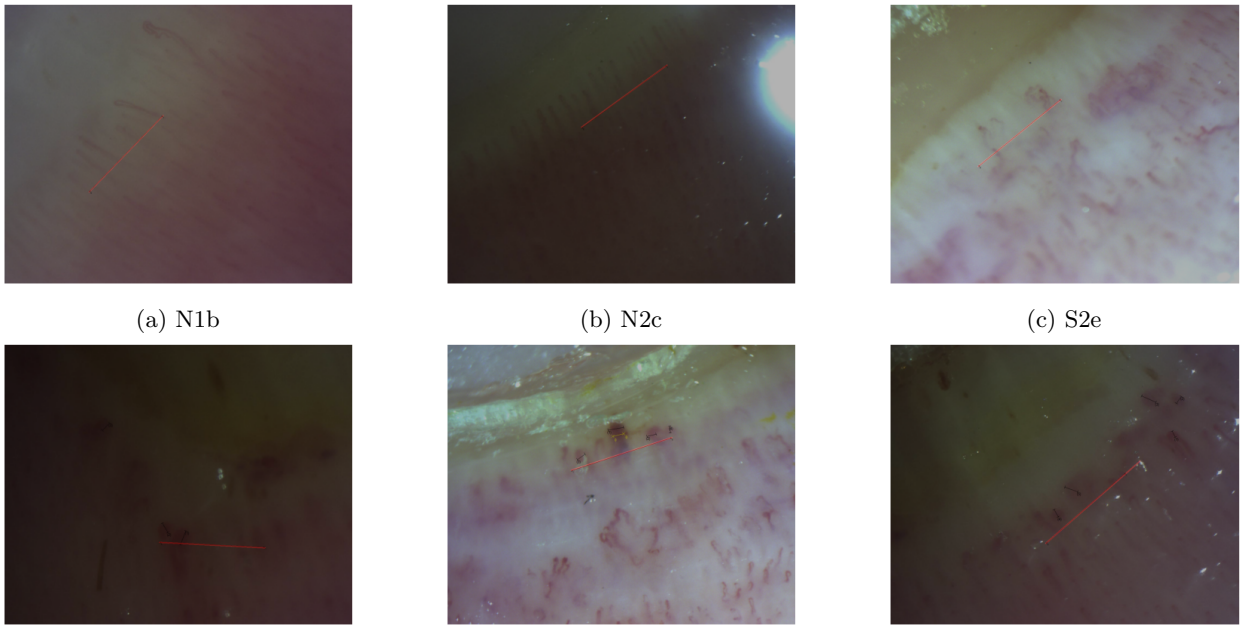
As discussed in the methodology section, for line detection we used the Probabilistic Hough Line Transform. For the implementation, we used:

```
cv2.HoughLinesP(image, rho, theta, threshold, minLineLength, maxLineGap)
```

Where,

- **Image** is the edge image found by using the canny edge detector
- **Rho** is the distance resolution of the accumulator in pixels (set to 1).
- **Theta** is the angle resolution of the accumulator in radians. (set tp π)
- **Threshold** is the accumulator threshold parameter. (set to 25)
- **minLineLength** is the minimum line length. Line segments shorter than that are rejected. (set to 100)
- **maxLineGap** is the maximum allowed gap between points on the same line to link them. (set to 65)

After passing our test images through this function, the results we get are as following:



(a) N1b (b) N2c (c) S2e

(d) S3h (e) S2d (f) S3d

Figure 5: Test data masked with the detected line

The above figure (5) shows that in each test image we used for this report, our algorithm is detecting the line accurately. We visually validate our results, we masked the detected line, which can be seen in the said figure.

4.3 Cropping and Rotating

Following the identification of the starting and ending coordinates of the detected lines, the subsequent step involved the rotation and cropping of each test image based on the detected line.

The rotation process commenced by determining the angle of the line within the image. This was accom- plished by employing an appropriate mathematical formulation, which allowed us to calculate the angle of inclination. Subsequently, the image was subjected to a rotation operation utilizing a rotation matrix. The rotation matrix, applied through an affine transformation, facilitated the adjustment of the image's orien- tation in accordance with the calculated angle. The implementation of this rotational adjustment employed the following algorithmic pro**ce**dure:

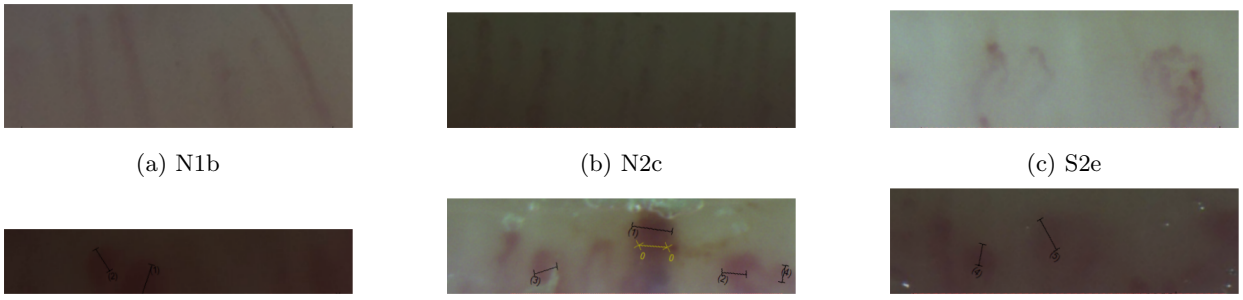
```
angle = np.arctan2(y2 - y1, x2 - x1) * 180 / np.pi
cv2.getRotationMatrix2D((cols/2, rows/2), angle, 1)
cv2.warpAffine(image, rotation matrix, (cols, rows))
```

The resulting rotation of the image facilitated a more aligned representation of the detected line, aiding in subsequent analysis and interpretation of the relevant features within the image.

Subsequently, the image was cropped by defining a boundary with an added margin. The cropping process was implemented using the following technique:

```
min_x = max(min(x1, x2) - margin, 0)
max_x = min(max(x1, x2) + margin, image.shape[1])
min_y = max(min(y1, y2) - margin, 0)
max_y = min(max(y1, y2) + margin + space_below_line, image.shape[0])
cropped = image[min_y:max_y, min_x:max_x]
```

The margin and space below line had to be set for each test data as each image is unique.



(a) N1b (b) N2c (c) S2e

(d) S3h (e) S2d (f) S3d

Figure 6: Test data cropped and rotated with reference to the detected line

The above figure (6) shows the result of rotating and cropping the image. The transformation of all the images is successfully, as seen visually.

4.4 Green Channel enhancement and Denoising

As discussed in the previous section, we use the green channel and for better results, we enhanced the contrast of the channel by using histogram equalization, which is implemented using OpenCV:

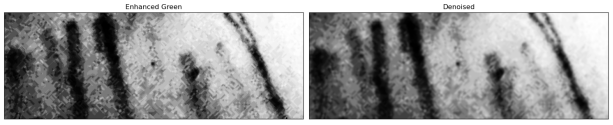
```
cv2.equalizeHist(image)
```

Histogram equalization is a technique used to enhance the contrast of an image by spreading out the intensity values over the entire dynamic range. It redistributes the pixel intensities in such a way that the histogram of the equalized image is approximately uniform.

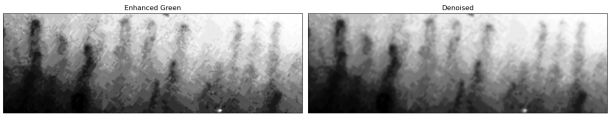
For the denoising and smoothing of the enhanced green channel, we used Gaussian blur and a median filter. The implementation is as following:

```
blur = cv2.GaussianBlur(enhanced green channel image, kernel size, sigma)
denoised = cv2.medianBlur(smoothed enhanced green channel image, kernel size)
```

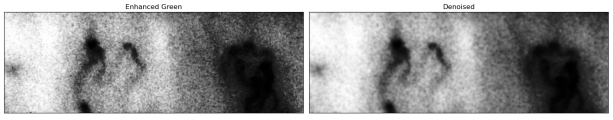
The results of green channel enhancement and denoising are as following:



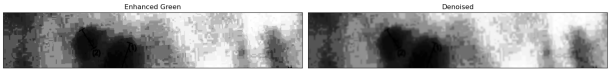
(a) N1b



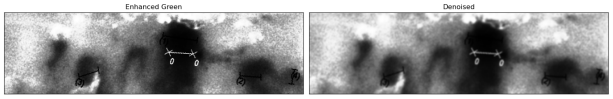
(b) N2c



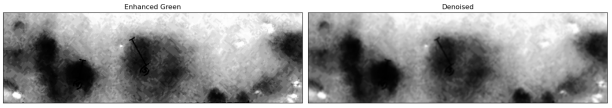
(c) S2e



(d) S3h



(e) S2d



(f) S3d

Figure 7: Enhanced Green Channel and Denoised enhanced green channel images of the test data

From the above figure (7), we can confirm that our algorithm is working and giving us accurate results.

This completes our pre-processing pipeline and now we are ready to segment and contour the capillaries.

4.5 Capillary segmentation and contouring

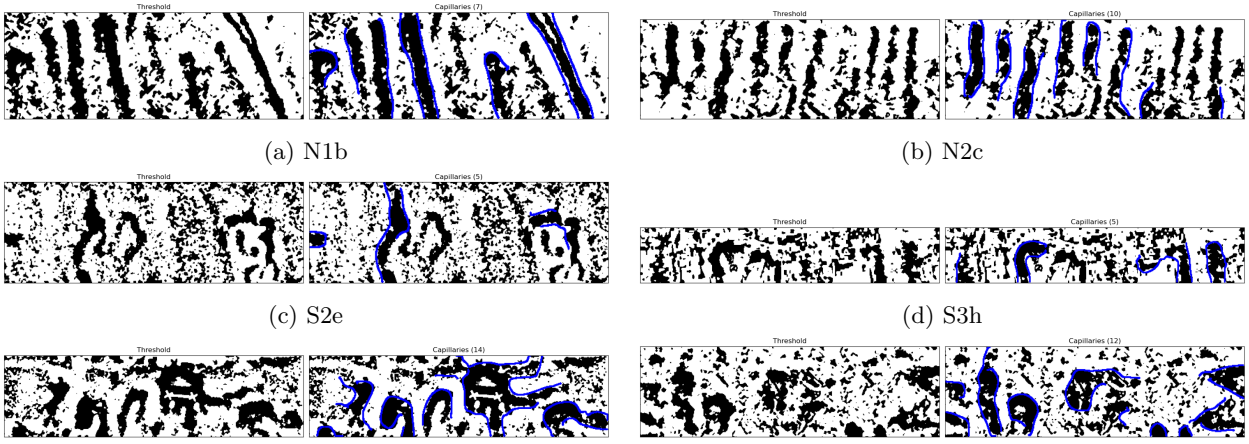
Having successfully obtained the denoised and enhanced isolated green channel for each of the test data samples, the next step involves the application of adaptive thresholding to perform capillary segmentation, as previously discussed. This process entails generating a binary image wherein pixels below the adaptive threshold within their respective neighborhoods are eliminated, thereby retaining only the pixels with values exceeding the adaptive threshold. Consequently, the resulting binary image will facilitate the precise delineation and extraction of capillary structures, enabling further analysis and interpretation of relevant morphological characteristics. The implementation is as following:

```
cv2.adaptiveThreshold(image, maxValue, adaptiveMethod, thresholdType, blockSize, C)
```

After obtaining the binary image, we can now apply the contouring algorithm. The implementation is as following:

```
contours, hierarchy = cv2.findContours(binary image, mode, method)
```

The results of the segmentation and contouring are as following:



(a) N1b (b) N2c

(c) S2e (d) S3h

(e) S2d (f) S3d

Figure 8: Segmented and contoured capillaries

In the above figure (8), we have displayed the segmented and contoured capillaries of 6 images.

Figure (8a) and (8b) show accurate segmentation of capillaries as the results match with the ground truth.

Figure (8c) and (8d) show results which were not accurate but were near to the ground truth can be fine tuned to accuracy.

Lastly, (8e) and (8f) show results which were inaccurate by a huge margin and this shows that our approch failed.

Further discussion of the results is done in the next section.

5 Discussion

In this section of the report, we will discuss the results, strengths, and weaknesses of the method we used for segmenting capillaries in nailfold capillaroscopy images.The results obtained from applying the adaptive thresholding technique were generally satisfactory, although not all outcomes perfectly matched the provided ground truth. Our method successfully detected and outlined the specified line in each image, which then allowed us to rotate and crop the images according to the detected line's orientation. Additionally, we were able to separate the images into their Red, Green, and Blue channels, and focused on enhancing the contrast of the green channel, which produced the best results.The adaptive thresholding approach proved effective in segmenting the capillaries, producing a binary image where the capillary structures were clearly visible. However, difficulties were encountered during the contouring step, which introduced some limitations to our overall methodology.

5.1 Results

The table bellow compares the results we achieved compared to the ground truth provided of all the data provided to us:

Image	Achieved Result	Ground Truth
Image N1a	6	6
Image N1b	7	7
Image N1c	3	9
Image N1d	10	8
Image N1e	9	8
Image N1f	13	9
Image N1g	14	9
Image N1h	19	7
Image N1i	10	7
Image N2b	8	8
Image N2c	10	10
Image N2d	7	9
Image S2a	7	1
Image S2c	3	2
Image S2d	14	5
Image S2g	4	5
Image S3c	2	2
Image S3d	12	5
Image S3e	12	5
Image S3f	5	4
Image S3g	5	6
Image S3h	5	4
Image S3i	4	2
Image S3j	3	5

Table 1: Comparison of Achieved Results vs. Ground Truth

From the analysis shown in the previous table, it's clear that our implementation produced results with varying levels of accuracy. Some images achieved satisfactory outcomes, with results closely matching the ground truth. In other cases, the results were reasonably close, suggesting that adjusting the algorithm's parameters could improve performance. However, it's also important to recognize that some data produced highly inaccurate results, pointing to areas that require further investigation and refinement.

5.2 Advantages

The advantages of this approch are:

1. **Computational Efficiency and Ease of Implementation:** Adaptive thresholding methods provide a computationally efficient and straightforward approach for capillary segmentation. These methods can be readily implemented, making them accessible to researchers and clinicians with varying levels of expertise in advanced image processing techniques.

2. **Discrimination of Capillaries from Background:** By establishing a threshold value, adaptive thresholding methods effectively discriminate capillaries from the background. This enables the extraction of relevant information regarding capillary density, tortuosity, and other essential morphological characteristics for subsequent analysis.

3. **Applicability in Resource-Constrained Scenarios:** Thresholding-based methods present an appealing option for capillary segmentation, especially in situations where the adoption of machine learning or deep learning approaches may be impractical due to limited resources or specific research constraints. The simplicity and efficiency of thresholding methods allow for reliable and interpretable capillary segmentations without the need for extensive training or complex models.

5.3 Limitations

The limitations of our approch are as following:

1. **Absence of explicit capillary definition or template:** The contouring algorithm employed in our approach is not reliant on predefined knowledge or templates that explicitly define capillaries. Consequently, the algorithm lacks intrinsic awareness of capillaries as distinct anatomical structures. In order to enhance accuracy, the integration of deep learning techniques becomes imperative, wherein comprehensive and precise datasets can facilitate the algorithm in recognizing and delineating capillaries more effectively. The current contouring algorithm relies on edge-based contour detection, which may inadvertently identify noise or background objects that exhibit similar edge characteristics, thereby potentially compromising the accuracy of capillary segmentation.

2. **Dependency on Proper Parameter Selection:** The performance of adaptive thresholding methods relies on the appropriate selection of parameters, such as block size, constant values, or adaptive methods. Improper parameter selection may lead to suboptimal segmentations or difficulty in achieving desired results.

3. **Limited Adaptability to Complex Capillary Structures:** Adaptive thresholding methods may encounter difficulties when segmenting capillaries with complex structures, such as overlapping or intertwined vessels. The simplicity of thresholding techniques may not fully capture the intricate details or nuances present in such capillary networks.

4. **Confirmation of Results by Experts:** Given the non-autonomous nature of this approach, it is imperative to seek validation from domain experts before placing full reliance on the obtained results or making any consequential decisions. In order to ensure the accuracy and reliability of the findings, the expertise and judgment of knowledgeable professionals are essential. Expert confirmation or potential modifications to the findings are necessary steps to establish the credibility and trustworthiness of the outcomes.

5.4 Future works

Despite the progress made in capillary segmentation using the proposed approach, several limitations warrant further investigation and improvement. The identified limitations serve as valuable pointers for future research endeavors in the field. The following areas present potential avenues for enhancing the accuracy, robustness, and applicability of capillary segmentation:

1. **Incorporation of Explicit Capillary Definition or Template:** The current approach relies on an edge-based contouring algorithm that lacks explicit knowledge or templates specifically designed for capillaries. Integrating deep learning techniques can aid in overcoming this limitation by leveraging comprehensive and precise datasets to train the algorithm in recognizing and delineating capillaries more effectively. By incorporating capillary-specific knowledge or templates, the algorithm can enhance its understanding of capillaries as distinct anatomical structures, thereby improving segmentation accuracy and reducing the risk of erroneous identification of noise or background objects.

2. **Optimization of Parameter Selection for Adaptive Thresholding:** The performance of adaptive thresholding methods employed in capillary segmentation is dependent on proper parameter selection, such as block size, constant values, or adaptive methods. Future work should focus on devising strategies for automated or data-driven selection of optimal parameters to ensure consistent and reliable results across different datasets and imaging conditions. This optimization process can contribute to minimizing user intervention and subjectivity while achieving optimal segmentation outcomes.

3. **Advanced Techniques for Complex Capillary Structures:** The segmentation of capillaries with complex structures, such as overlapping or intertwined vessels, remains a challenge for existing methods. To address this limitation, future research should explore advanced techniques capable of capturing the intricate details and nuances present in such capillary networks. This may involve the development of novel algorithms incorporating contextual information, shape priors, or spatial constraints to improve the segmentation accuracy and delineation of complex capillary structures.

4. **Integration of Expert Evaluation and Validation:** Given the non-autonomous nature of the proposed approach, it is crucial to integrate expert evaluation and validation into the capillary segmentation pipeline. Future works should involve close collaboration with domain experts to assess the accuracy and reliability of the obtained results. Expert feedback and expert-guided modifications can significantly enhance the credibility and trustworthiness of the segmentation outcomes, ensuring that the technique is effectively aligned with clinical or research requirements.

Addressing these areas of improvement will contribute to the advancement of capillary segmentation techniques, enabling more accurate and reliable analysis of micro-vascular structures. By refining the approach and overcoming the identified limitations, researchers can pave the way for enhanced diagnostic capabilities, better understanding of micro-vascular diseases, and improved monitoring of treatment outcomes in various clinical and research settings.

6 Conclusion

In this project, our goal was to segment capillaries in nailfold capillaroscopy images using traditional image processing methods. We carried out several preprocessing steps to isolate the region of interest, focusing on extracting the green channel and applying noise reduction techniques. Next, we applied adaptive thresholding to the denoised green channel, producing a binary image that clearly emphasized the capillaries. We then performed contour detection on the binary image, which allowed us to count the number of capillaries identified. The results achieved by our approach were satisfactory, especially given the limitation of not using automated methods. However, to improve the accuracy and precision of capillary segmentation, incorporating deep learning techniques is strongly advised.

The successful completion of this project demonstrates the effectiveness of classical image processing techniques for capillary segmentation. Our approach enabled the identification and outlining of capillaries, offering valuable information about their morphological features. Still, integrating deep learning methods could further enhance the segmentation by utilizing large datasets and advanced neural network models. This combination could significantly improve the ability to detect complex capillary patterns and boost segmentation accuracy.

In summary, although our project achieved notable results using non-automated techniques, the future incorporation of deep learning methods offers great potential to improve capillary segmentation in nailfold capillaroscopy images. Leveraging the capabilities of deep learning algorithms can provide researchers and clinicians with more accurate and efficient tools for analyzing capillaries, ultimately advancing the study of microvascular diseases and supporting better diagnosis and treatment follow-up.

7 Appendix - I - Refrences

[1] Ruaro B, Sulli A, Alessandri E, et al. Application of different normalization methods for the analysis of nailfold videocapillaroscopic images. Microvasc Res. 2015;99:58-63.

[2] Kurzinski K, Torok KS. Digital microscopy and imaging analysis application for nailfold capillaroscopy. Methods Mol Biol. 2019;1982:263-277.

[3] Kurzinski, K., & Torok, K. S. (2019). Digital microscopy and imaging analysis application for nailfold capillaroscopy. Methods in molecular biology (Clifton, N.J.)

[4] Ruaro, B., Sulli, A., Alessandri, E., Pizzorni, C., Ferrari, G., Taraborelli, M., ... & Cutolo, M. (2015). Application of different normalization methods for the analysis of nailfold videocapillaroscopic images. Microvascular research, 99, 58-63.

[5] Cutolo, M., Pizzorni, C., Tuccio, M., & Burroni, A. (2004). Nailfold videocapillaroscopic patterns and serum autoantibodies in systemic sclerosis. Rheumatology, 43(6), 719-726.

[6] Smith, V., Herrick, A. L., Ingegnoli, F., & Distler O. (2020). Standardisation of nailfold capillaroscopy for the assessment of patients with Raynaud's phenomenon and systemic sclerosis. Autoimmunity reviews, 19(2), 102458.

[7] Gonzalez, R. C., & Woods, R. E. (2008). Digital image processing (3rd ed.). Prentice Hall.

[8] Pratt, W. K. (2001). Digital image processing (3rd ed.). Wiley.

[9] Otsu, N. (1979). A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cyber