Towards a Deep Learning Pipeline for Measuring Retinal Bloodflow

Chrysostomos Chadjiminas

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Master of Research

of

University College London.

Department of Computer Science

University College London

December 11, 2022

I, Chrysostomos Chadjiminas, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Alterations to microvascular flow are responsible for a number of ocular and systemic conditions including diabetes, dementia and multiple sclerosis. The challenge is developing of methods to capture and quantify retinal capillary flow in the human eye. We built a bespoke adaptive optics scanning laser ophthalmoscope with two spatially offset detection channels, with configurable offset aperture detection schemes to image microvascular flow. In this research we sought to develop an automatic tool that detects and tracks erythrocytes. A deep learning convolutional neural network is proposed for classifying blood-cell from non-blood-cell patches in each frame. The patch classification is coupled with a localisation process to detect the positions of the red blood cells. A capillary segmentation method is also presented to increase the efficiency and performance of the localisation process. Finally, a technique is presented to match corresponding cells between the two channels in the raster, allowing for a fully automatic blood flow velocity measurement. Results from various experiments are reported and compared to give the most accurate configuration. The deep learning basis of the tool allows for a continual and adaptable learning that can improve the performance of the tool as more samples are collected from subjects. In addition, the modular nature of the tool allows for replacing its components, such as the capillary segmentation, with state-of-the-art techniques allowing for the software's longevity.

Acknowledgements

The author thanks Dr. Adam Dubis for the frequent meetings and for always being on-line when a certain roadblock was encountered. The helpful conversations and sound advice given from Adam helped for a smooth and fruitful thesis development. The author also thanks PhD student Mustafa Arikan for providing the computational resources for running multiple experiments which helped in finding model configurations with good performance. Last but not least, the author thanks Prof. Daniel Alexander for bringing him in contact with people that work with the NHS and for allowing him to use his master's acquired knowledge in the medical field.

Contents

List of Figures

List of Tables

Chapter 1

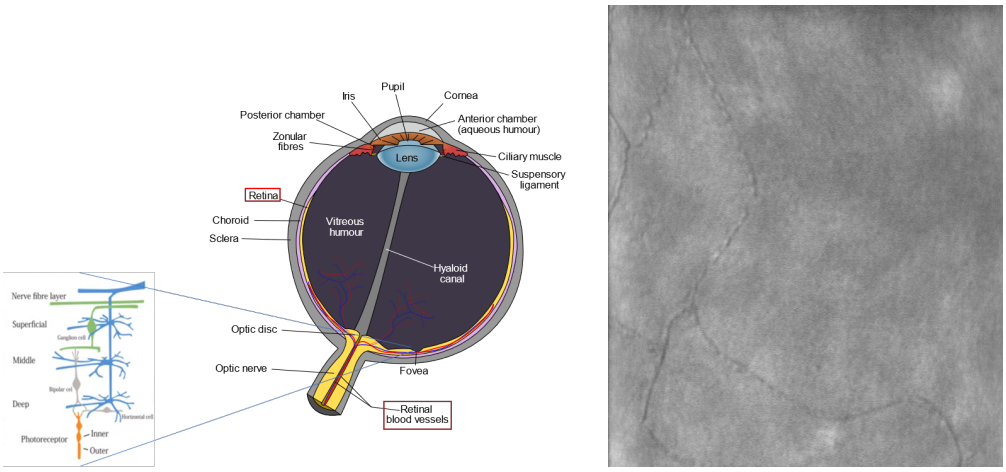
Introduction

1.1 Introduction

1.1.1 Motivation

As the vascular system gets older, or damage is done to it from different diseases such as diabetes, heart disease, chronic inflammatory diseases or other neurological diseases, the arteries are not able to dampen the rate of blood flow as healthy ones do. As a result, the pressure wave tracks down the arterial blood vessels and into the capillaries. These changes occur before the disease is detected using traditional testing. It is believed that this loss of vessel pulse dampening is responsible the disease effects of diabetes [1], dementia [2], stroke [3], hypertension [4, 5], Parkinson's and Multiple Sclerosis [6]. The capillaries of the brain could serve as an effective biomarker for the existence of such diseases. Unfortunately, examining these small vessels and the blood flow in the brain can be invasive, expensive or sometimes impossible. However, the vascular system of the eye and brain is similar to each other and different from everywhere else in the body. As a result, imaging of the retinal vasculature could accurate reflect the microvascular changes that happen in the brain [7]. "The eye is the window of the soul", and more specifically, a window to the brain[8]. The retina is unique in this way because, with the developments in adaptive optics, non-invasive imaging of the it's vascular plexus is possible [9].

In this thesis, the main goal is to provide a tool that is able to locate and track blood cells in videos of the capillaries inside the retina. Multiple videos from var-



(a) left image from [10]. Right picture from [11]. **(b)**

Figure 1.1: **(a)** An image of the eye anatomy highlighting the vessel layers in the retina. The imaging depth of field of our imaging process captures the red blood cells as they move in the capillaries in middle to deep levels of the retina. The arteries and veins are fed through the optical nerve. On the image on the left it can be seen that the retina hosts small vessels at many levels. **(b)** A still frame of the retina from our videos. The frame is a part of a bigger video from one of our subjects found in this media link.

ious subjects were captured for this purpose. Fig. 1.1 shows a sample frame from the videos that were acquired. In this thesis it was asked to detect and track the blood cells in these videos.

The work described in this thesis is mainly focused on the automatic detection of the erythrocytes in the images, but also comes with a solution for tracking the cells to allow the measuring of blood flow statistics. Manually detecting the blood cell's in these images and finding the correspondence is slow, cumbersome, and error prone. Previously published methods are semi-automated and require several subjective steps which degrades repeat-ability and generality of the method.

1.1.2 Video Acquisition

In the next section the method that is used to produce the images and videos shown in Fig 1.1 is described. To put things in context, Confocal and Offset Aperture Imaging is briefly explained. Additionally, the Adaptive Optics Scanning Laser Opthalmoscope (AOSLO) method for enhancing the quality of the imaging process is explained in short.

1.1.2.1 Confocal and Offset Aperture Imaging

In both confocal and offset aperture imaging, a laser beam of a non-harmful wavelength is shone in to the eye. The ray is then focused to narrow depth of field and is reflected back to a sensor. The beam scans the retina in a raster fashion with a fixed rate until the raster image of the desired size is created The scanning beam frequency and the size of the frame affect the frame rate of the produced video.

The difference between the confocal and offset aperture imaging is demonstrated in Fig. 1.2.

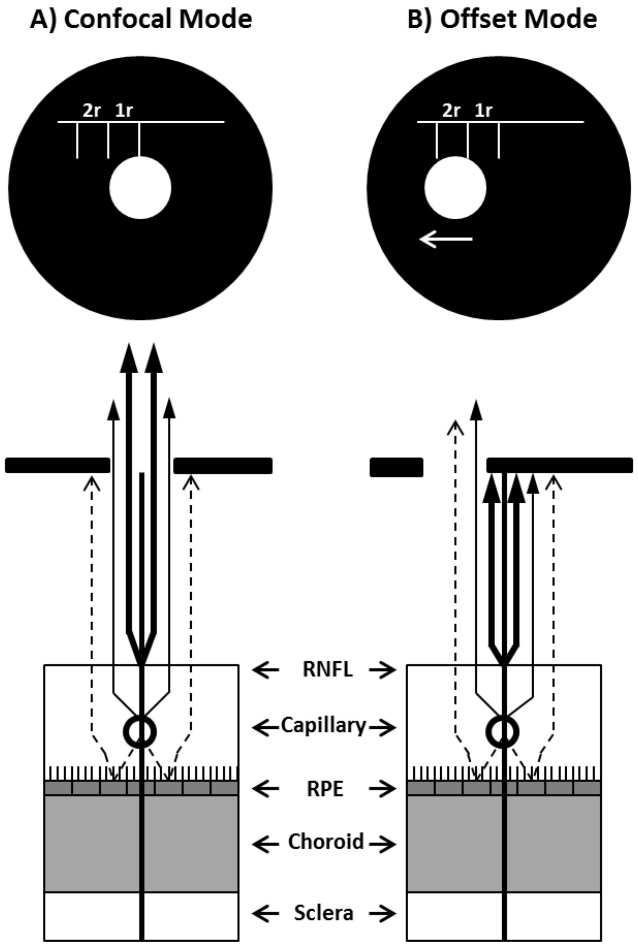
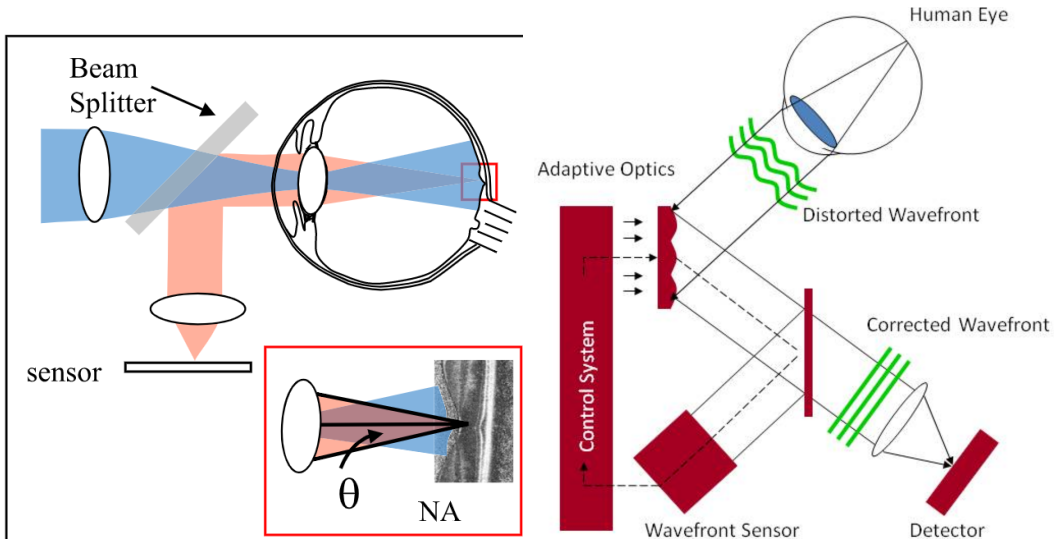


Figure 1.2: Figure from [12]. Diagram that illustrates the difference between the confocal and offset aperture imaging. **(A)** In confocal imaging, the aperture is centred allowing for light that is directly reflected to be captured by the sensor. **(B)** In offset aperture mode, a slight offset is introduced in the aperture allowing the sensor to capture diffused light.

1.1.2.2 Adaptive Optics Scanning Laser Ophthalmoscope (AOSLO)

The imaging techniques that were described, when used alone, produce distorted and noisy results. Adaptive optics introduces a deformable mirror to correct distorted wavefronts producing higher fidelity signal. Such techniques allow microvascular examination imaging of the retina [12]. Please see Fig. 1.3 for a demonstration of the imaging process and how AOSLO can improve quality.



(a) **(b)**

Figure 1.3: **(a)** Image from[13]. A laser is shone to the eye and the beam is reflected back to a sensor. **(b)** Image from [14]. Adaptive optics introduce a deformable mirror to correct distorted wave-front. The corrected signal is collected by the detector.

In our work, the videos captured use a confocal device along with two offset aperture devices that produce 3 videos of the retina. One video is captured with a confocal ophthalmoscope. The two offset aperture videos use imaging beams of **790nm** and **850nm** wavelength. The videos are synchronised and create a raster of the eye at the same time. In the next section, we explain the reason for producing multiple videos of the subjects retina.

1.1.2.3 Rapid high resolution image with dual-channel scanning technique

Our group build a system that uses The Rapid high resolution image with dual-channel scanning [15] technique. This technique utilises an Adaptive Optics Laser Opthalmoscope (AOSLO) to produce two images of the same area in the retina with a slight time difference by using two imaging beams of different wavelength.

Producing a single video of the capillaries captured at a frame rate of 32fps (based on the scan rate and frame size) does not allow for cell tracking. The reason is, because the time difference between two consecutive frames is too large, the correspondence of the frames can not be established due to the aliasing effect demonstrated in Fig. 1.4. Moving the scanners much faster is not feasible and having very fast scanners is also expensive.

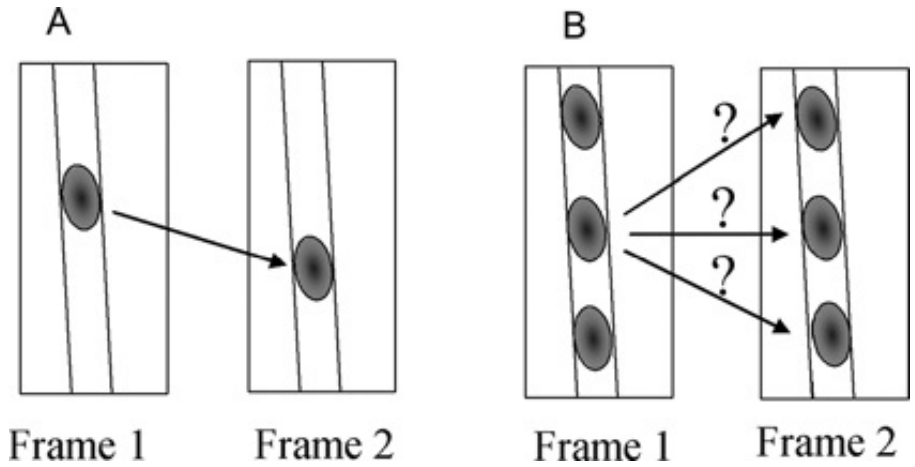


Figure 1.4: Demonstration of the aliasing effect. An image from[16]. The red blood cells are abundant in the capillaries and they move fast compared to the frame acquisition rate. As seen in B, If the frame rate is not fast enough the time difference between the between the frames is large. It's not possible to establish a correspondence between the cells in two consecutive frames.

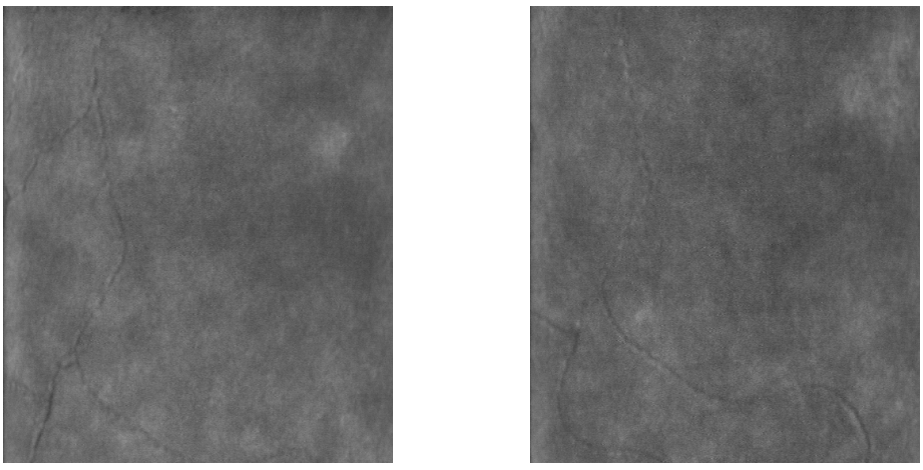
The work in [15] proposes a new technique where two videos are captured at the same time with two slightly different imaging wavelengths and slightly different spatial starting points. The vertical offset of the two scanner introduces a slight temporal difference between corresponding frames of the videos from the two imaging beams. As a result, two rasters cross the same retina area at slightly different times, providing a quicker effective frame rate, and opening up the opportunity for tracking

of individual cells without aliasing by locating and matching corresponding frames from the two videos.

The time difference between the channels in [15] is 4.7 ms while the implementation of our team currently achieves 5.3 ms due to differences between the mirror placement and the device used in the paper. The time difference is a configurable value and can change by adjusting the vertical displacement between the rasters by changing the mirrors configuration.

In our work, a 750nm offset aperture laser was used for the first video and a 890nm offset aperture wavelength was used for the second. Videos and frames from the first and second imaging beam will be referred as belonging to the 750nm channel and 850nm channel accordingly. In addition to the two offset apertures a confocal imaging video was captured that captures exactly the same retinal area with the 750nm channel but uses confocal mode instead.

Multiple subjects volunteered to have their retina scanned. As a result, numerous videos were collected from various subjects allowing for great variability in the data. For a demonstration of the results of the dual-beam scanning image please see Fig. 1.5.



(a) **(b)**

Figure 1.5: The corresponding images of the same area in the retina captured by two imaging wavelengths at the same time. The images from the two channels are vertically offset to introduce a small time difference between them. **(a)** The frame of the 790nm channel. **(b)** The frame from the 850nm channel.

1.2 Goal and challenges

In this thesis the main goal is to create an automated pipeline to facilitate the identification of retinal blood vessels, blood cells, and provide the ability to track them in time to generate maps of retinal blood flow rate. The pipeline will be applied to videos of the retinal vasculature captured using a bespoke Adaptive Optics Light Scanning Opthalmoscope which was described in the previous section. As described, this device has the ability to detect multiple reflections and wavelengths at the same time. Using this detection paradigm, we then seek to detect individual blood cells in frames of the 790nm channel. Once the cells are detected, the channels are matched and flow velocity can be calculated.

The work in [15] offered a potential solution for locating blood cells in a selected vessel segment. The process described is semi automatic because multiple thresholds need to be manually applied to get an 'ideal' velocity trace. These interventions include a threshold of Z-scores for normalisation, a Gaussian blur application, an intensity threshold and finally averaged cell trace detection. Each of these steps needed to be applied to each video pair making repeatability hard and large scale analysis near impossible. The approach used has two main downsides. First it's not automatic and is subjective to the user's input. Secondly, due to the fact that the cells do not necessarily show the same characteristics and can have great variability in intensity, even in the same frame, an intensity driven approach can produce many false positive and can miss a lot of cells. To overcome this problem, it was decided that a deep learning focused approach was going to be used, since it is known that deep learning approaches can detect features that image processing approaches can not.

The method described in [17] uses a deep learning baseline to identify retinal cone photoreceptors in selected regions of interest. In their work, a convolutional neural network was trained and used to classify photoreceptor cones from non-photoreceptors in mosaic of the retina. Their work used a pipeline in which patches from a region of interest were passed through the network and assigned a probability of cell/non-cell to every pixel. We then decided that we should adapt this

method for our use case and the images devices were identical and blood cells and photoreceptors share some characteristics. Our implementation and a more in depth description of the method is described in the methodology 3 section.

As our process and purpose were different, several significant adaptations were needed. First of all, the blood cells show weaker contrast characteristics when compared to cones cells. For this we had to experiment with different contrast enhancing techniques in an attempt to increase individual cell characteristics. In addition, the photoreceptor images are stationary while our data come from a video, allowing us to experiment with techniques that utilise the temporality of the data. The work in [17] also describes a process for generating non photoreceptor samples for training from the images. The photoreceptors are uniformly distributed in the image which provides two key pieces of information. The uniformity of space can be used as a filtering prior on probability maps, it also allows for some predictable distance between each photoreceptors and the non-photoreceptor samples extracted around each photoreceptor are significantly different. In our work, the red blood cells are usually very close together in the capillaries and many times one cell starts when the other ends making the extraction of negative patches more challenging. Multiple non-cell extraction methods were attempted to increase the performance of the network. Taking non-cell samples that are far away from cell samples had better results in the classification because the cells are very distinct from non-cell but the performance when estimating the locations were less accurate while taking non-cell samples that are close to positive lead to harder training but more accuracy when locating the cells.

To get cell positions in the videos an expert had to mark over 10000 cells in various videos. This process is cumbersome and the marker could potentially have slight errors when marking the centre of blood cells. Small errors could be beneficial to our problem by having some noise in the data but could also make the network over-fit on wrong training data. For this reason, we had to use data augmentation techniques such as small random translations on all samples to reduce this kind of over-fitting error.

One additional challenge that had to be faced is that the background of the video frames is some times similar in contrast to blood cells due to noise. In addition, because the non-cell samples are picked close to the cell samples, the network is not trained on such background samples. As a result, many pixels that are not on the capillaries and in fact belong to the background are falsely classified as cells. To circumvent this, a method to segment the capillary sections from the background had to be devised. Deep learning segmentation methods could can been used to segment capillaries from background but this would require the manual creation of the segmentation for each video to provide for learning. To avoid this an image processing solution that utilised the temporal nature of the videos was designed. Through trial and error, this solution gave sufficiently good segmentation for the needs of our tool.

1.2.1 Thesis outline

In the next chapter the literature around classification in retinal images, deep learning, segmentation and blood cell tracking is explored. Following the literature review, the methodology chapter gives with an in depth description of the training and classification process, the segmentation and the tool implementation in general. In the evaluation chapter, multiple configurations of the classifier are evaluated and compared using test videos that were manually marked only for testing. Finally, we conclude with final remarks and future work. In the appendix different model configurations are demonstrated (Tab. A), along with their performance on classifying and locating the cells (Tabs. B, C D). Additionally, in the appendix sample results from runs on test videos are presented E.

Chapter 2

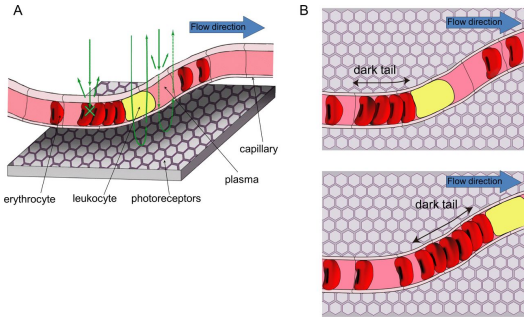
Literature Review

2.1 Retinal blood flow quantification

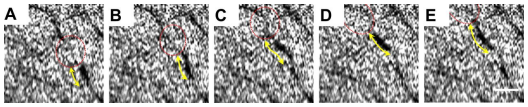
In this section, literature concerning techniques on imaging the retinal capillaries, quantifying blood flow and hemodynamics are presented.

In [18] blood flow hemodynamics in the retina were measured using AOSLO. In their technique, blood cells are observed as shadows with dark-tail like patterns. This shadow effect was caused when red blood-cells track behind a leukocyte in the capillaries causing the light to be obstructed and stopping it from being reflected from the photoreceptors. This tail of erythrocytes is detected as a dark 'tad-pole' like tail in videos (Fig.2.1).

The paper provides methods for calculating the speed of the tail to get an estimation on leukocyte velocity, which is reported to be close previously measured leukocyte velocities. The technique described doesn't allow for measuring erythrocyte velocities. This is a problem if you are trying to understand vascular hemodynamics as not all capillaries are large enough to allow leukocytes to pass, so no information would be recorded by this technique in these vessels. Additionally, erythocytes and not leukocytes are responsible for the metabolic exchange in the retina, and so their motion is the more essential. Furthermore, the capillaries of the confocal videos are visualised by using motion-contrast enhancement that was introduced in [9]. This method shows that statistics for the hemodynamics of the eye can be extracted using noninvasive AOSLO direct monitoring of the retina. However only



(a) Image from [18]



(b) Image from [18]

Figure 2.1: **(a)** Leukocytes move slowly in capillaries in relation to erythrocytes. Consequently, erythrocytes form a tail behind them. The tail blocks light from hitting the reflective photoreceptors.
(b) Five consecutive images from confocal video. In confocal imaging, the tail is detected as a dark tail (near yellow arrow), and the erythrocyte as a bright spot (red circle).

some cells are visible with this technique.

In [19], AOSLO is also used to characterise single file flow in parafoveal capillaries. In this publication, spatiotemporal plots were used to visualise the hemodynamics of the vessels. The work provides an analysis on the plots, in combination with other biometrics that were collected, to obtain information about flow dynamics such as frequency, flow direction, speed, and pulsarity. In addition, the capillaries were visualised using the motion-contrast enhancing method first presented in [9] which gave good quality images of the vessel maps, showing more capillaries than red-free fundus photography. The work mainly evaluates leukocyte flow dynamics.

The work in [20] complements previously discussed tools by providing a direct and noninvasive visualisation of erythrocyte movement in capillaries. A combination of adaptive optics, a fast sCMOS camera, and a full field illumination allowed the direct visualisation of individual erythrocytes in the capillaries. Then particle image velocimetry was used to quantify the flow. The erythrocyte velocities with this method were reported to agree with previously published results. While both this camera and the one in this study use adaptive optics to improve image quality,

the acquisition methods are completely different, so only some techniques from this study are applicable to this thesis.

In the introduction it was mentioned that part of the work in [15] was used to create the bespoke AOSLO video acquisition tool that creates frames with a very small time difference using a dual-beam imaging technique that was described. In the same publication, a method for computing the velocity of the erythrocytes is also described. The cells must be first detected. By first normalising the frames of the video for each channel and then applying a threshold to the images a binary image is created for each frame. The threshold is N times the standard deviation of the pixel intensity over the all the frames of the video. N is a value that must be manually configured by the user for each video. Then a morphological open operator is applied to the binary images to produce a set of locations of the cells. Instead of comparing the cell locations directly between the channels, an "average cell image" is created by averaging patches around the cells found within a manually selected segment. By measuring the displacement of the "average cell images" between the two channels the average displacement and hence the velocity can be calculated (Fig. 2.2).

This method allows for measuring a bigger range of retinal capillary velocities than was previously possible as shown in the results from high frame rate fundus illumination system [20], or on leukocytes using either the entoptic phenomenon [21][22], or other single channel imaging techniques with AOSLO [23][24][25][19] as reported in the paper. The method can measure velocities higher than was previously possible because red blood cells can be faster than white blood cells when travelling through capillaries due to the small vessel size which slows the latter cells down. It also produces comparable results to labelling with fluorescein [26][27] which is an invasive imaging method since it requires adding the artificial dye to the eye. Additionally, this method is better for smaller vessels (capillaries) while other techniques such as [28] or [29] are better for larger vessels. One additional benefit of this method is that it allows for a bigger area of the retina to be imaged. The benefit is twofold because it allows for the study of a bigger retina area and also

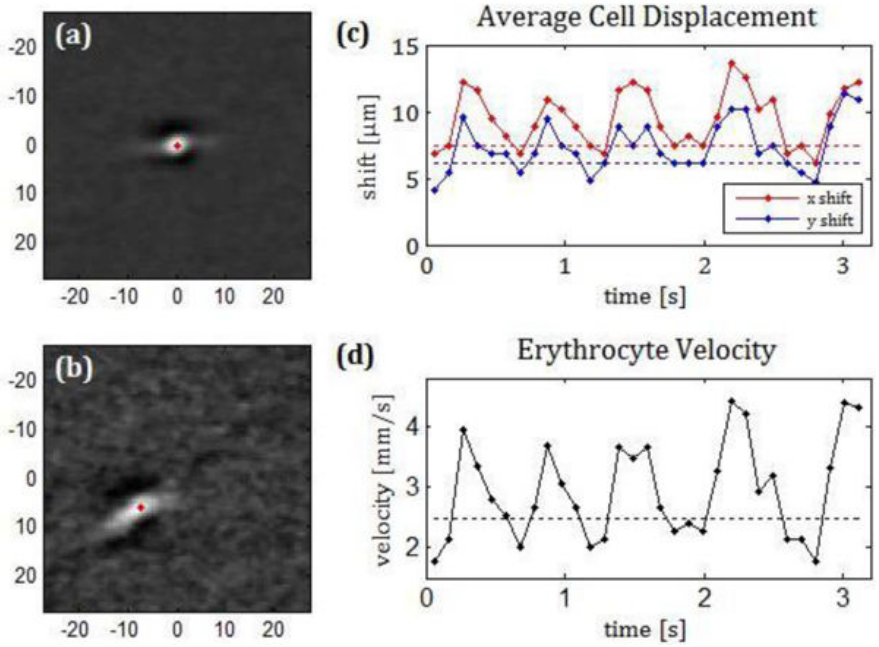


Figure 2.2: Figure from [15]. **(a)** The the average cell is created from a vessel segment in the first channel and in **(b)** The corresponding average cell for the second channel that is 4.7 ms ahead of the first. A vertical and horizontal displacement can be observed. **(c)** The displacement of in the x and y axis over the course of 3 seconds in the selected segment. **(d)** The blood velocity within the vessel segment

.

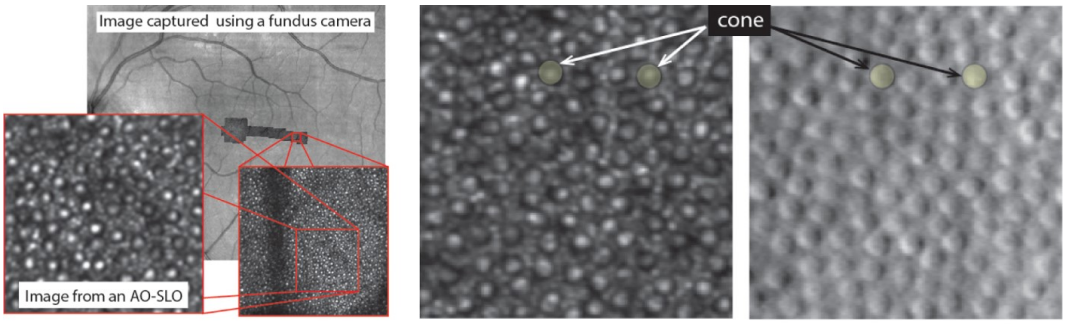
artefacts from eye movement can be corrected more easily using techniques such as [30] by registration.

2.2 Cell recognition and localisation

In this section, literature around classification and localisation of objects in retinal images is explored.

In this light, work done in image recognition and localisation of cone photoreceptors in AOSLO images is reviewed.

In [31] the cone photoreceptors are classified and located using an image processing pipeline that uses SVN is. In brief, the begins by estimating the photoreceptor's size using an automated process (Yellott's ring[32] [33]) or by manual selection. The next step includes a bilateral and a Gaussian filter, an adaptive histogram equalisation, an adaptive sliding-window enhancement that is applied to enhance the contrast between the left/dark and right/bright sides of the cones, an aggressive



(a) (b)

Figure 2.3: Figure of photoreceptors mosaic from [31]. **(b)** Photoreceptor captured with two different AOSLO imaging techniques.

thresholding to create a binary image, a non-overlapping extremal region detection to detect the centroids, a preliminary model for cone extraction and finally a model for refinement, and cone detection process using a Support Vector Machine (SVM). The process is described with great detail in the paper. There are 11 configurable hyper-parameters. The results reported in the paper show that the model is precise and it's results are comparable to a human grader for images of healthy retinas and have a positive precision and recall on images of retinas with Stargardt disease. It's also reported that the model can outperform the, at the time, state of the art model developed in[32].

Although the model shows promising results, it might not be easily generalised making it hard to be adapted for objects of other shapes. The method creates a cone-model that accounts for the characteristic shape and intensity of the photoreceptor cone cells. In addition, the model is dependent on the spatial arrangement of the photoreceptor mosaic. As a result, when the model is presented with images of the photoreceptor mosaic of people with Stargardt disease, which is more sparse, it's performance drops and both the precision and recall of the model are significantly lower.

Deep learning was shown to outperform traditional machine learning techniques for a lot of imaging tasks including image classification. Deep networks learn features directly from the data and do not rely to ad hoc rules that are unique to the data-set. As a result, such networks are more adaptable. This is more appealing because deep learning models can be generic and a model that is a trained

for a particular dataset can be adapted to be used for other datasets. One drawback of deep learning is that it can be outperformed from traditional machine learning techniques when the the amount of training data is sparse. This can be circumvented with techniques such as data augmentation [34] which was shown to work with great success in the past[35]. Deep Convolution Neural Networks (CNNs) have been shown to work for different image analysis tasks in many fields including optical imaging[36, 34, 37, 38, 39] with great performance.

For this reason, the methods described [17] and in [40] are reviewed. Both of this methods try to classify and locate cone photoreceptors in AOSLO images using image classification deep neural networks. Due to the deep learning baseline, the implementation for cell detection and localisation is not strictly dependent on the structure and appearance of the cone cells. The work in [17] is briefly described: First, images of the photoreceptor mosaic are captured and experts mark the centres of each cone to create a training set for the binary classifier we need images for cone and non-cone cells. A window of a fixed size is drawn around the marked centres to extract patches of cones. Negative patches are extracted around the cone cell in a voronoi diagram that is created from the cone centres. Then random points from the edges of the voronoi pattern are selected as centres for the non-cone patches. This process is demonstrated in Fig. 2.4.

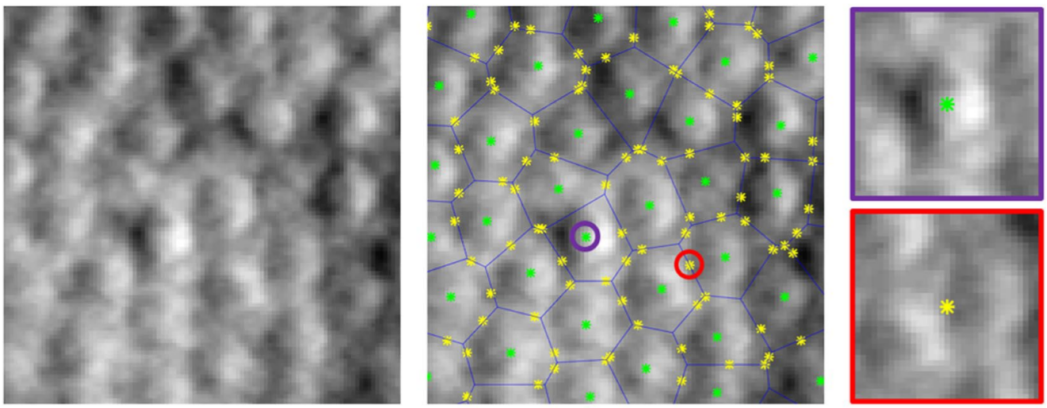


Figure 2.4: Image from [17]. Training data generation process for the CNN training. **(a)** The original image of the photoreceptor mosaic, **(b)** Image (a) with the centre of the photoreceptors marked by an expert and a voronoi diagram around them. Points on the voronoi edges are randomly selected as centres of non-photoreceptor patches. **(c)** An example of a cone (top) and non-cone (bottom) patch extracted from the marked image (b).

Once the training set is created the CNN classifier can be trained to detect cone from non-cone images. The CNN described in[17] is a modification of the famous AlexNet [41]. It consists of convolutional layers to generate feature maps, pooling layers which are known to improve computational performance and increase invariance to small distortions [42], Rectified linear units (ReLU) activation layers to introduce non-linearities [42], batch normalisation layers which are reported to increase training speed and reduce over-fitting by normalising layer inputs learning their mean statistic [43]. In the end there are fully connected layers, again with ReLU activations, for the final classification part. The last fully connected layer has two outputs and is fed to a soft-max layer that turns the output to probability distribution for cone and non-cone[44]. For the training process the weights are randomly initialised. Each iteration, also called an epoch, a subset of the training set, called a mini-batch, is fed to the network. For each image in the mini-batch a label is predicted with a probability distribution from the soft-max layer, i.e 75 per cent cone and 25 per cent non-cone. In the training process, because the label is known for each image, a loss function measures the distance of the predicted output and the actual label.

Proportionally to this distance, the network adjusts the weights, with a process called back propagation. The training process is called gradient descent because during the forward pass a gradient is calculated for each weight with the goal to minimise the training loss.

With sufficient training data the network can potentially classify cones from non-cones with a high accuracy. It should mention that, part of the dataset should be used just for testing the performance of the classifier and should not be used for training. The photoreceptors can then be localised in the image. In [17], a patch is created around every pixel of the image and all the patches are classified with the trained CNN. The result is a grayscale image with a probability of each pixel being the centre of a cone. Next, a Gaussian smoothing is applied on the probability map followed by an extended-maxima transform. The resulting binary image is filtered by removing connected components that have low intensity (in the probability map)

in the under a threshold. Finally, the centre of the remaining clusters is considered to be the centre of the cone. For additional information for the parameter values see [17].

The results are compared against the 'gold' standard for detecting photoreceptors that uses graph theory and dynamic programming (GTDP) and a previous work that we mentioned earlier that uses adaptive filtering and local detection (AFLD) [32]. In both cases the results are comparable with a true positive rate ranging from 0.943 to 0.989 and a false discovery rate of 0.034 and as low as 0.003. For a more detailed description of the evaluation method please refer to [17].

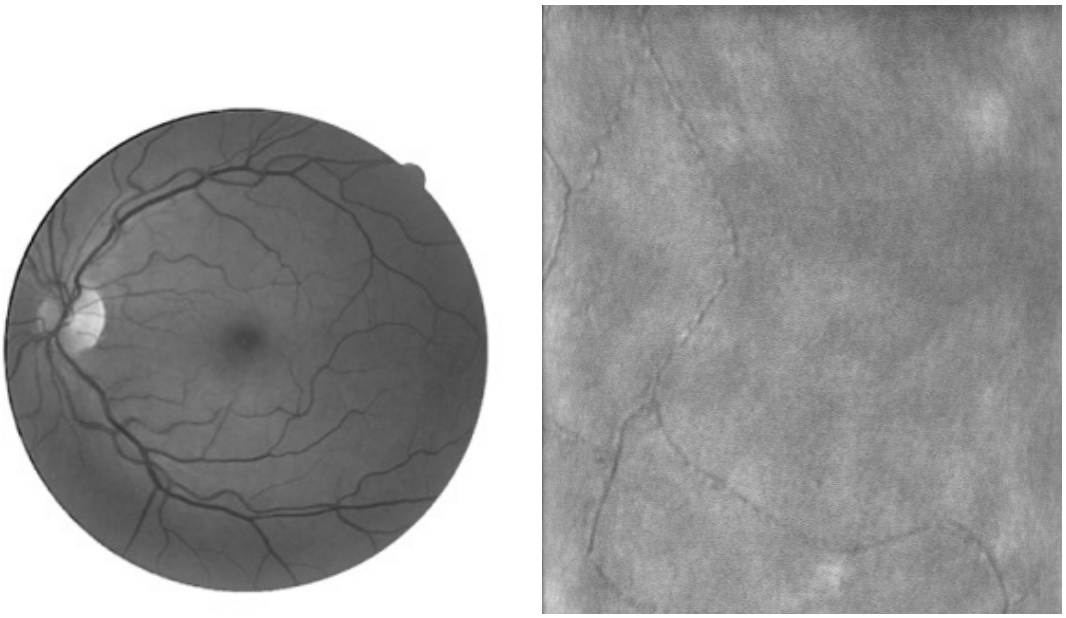
The results are appealing since it shows that the model can compete with state of the art methods. Contrary to the other techniques, it has the advantage that it can be adapted for applications with different modalities since it uses a modified generic CIFAR network. The GTDP and AFLD models were created with assumptions and rules that are specific for the photoreceptor mosaic.

2.3 Capillary segmentation

Capillary segmentation methods are useful for reducing the search-space in many blood flow quantification methods since all the movement is limited within the capillaries. Examining areas outside those vessels can introduce noise to the results and reduce performance. In addition, limiting the search-space to the capillaries can greatly reduce the amount of computations and time needed to quantify blood flow.

Currently, there is a growing literature for blood vessel segmentation in retinal images [45][38][37]. The data-set for this problems focuses on fundus imaging[46] and is focused on big blood vessels in the retina (arteries and veins) and not on capillaries. Fundus imaging is different in contrast and intensity to AOSLO imaging rasters (Fig. 2.5). AOSLO videos could be flattened by creating a standard deviation image of the video (explained more in depth in the methodology chapter)

Deep learning could also be used for segmentation tasks. One promising such segmentation network is DeepVessel[48] which produces state of the art results. To



(a) Fundus image from[47] **(b)** AOSLO image of a capillary

Figure 2.5: (a) Fundus and **(b)** AOSLO image comparison.

train for segmentation using a supervised machine learning method it is required to have a ground truth for the capillaries

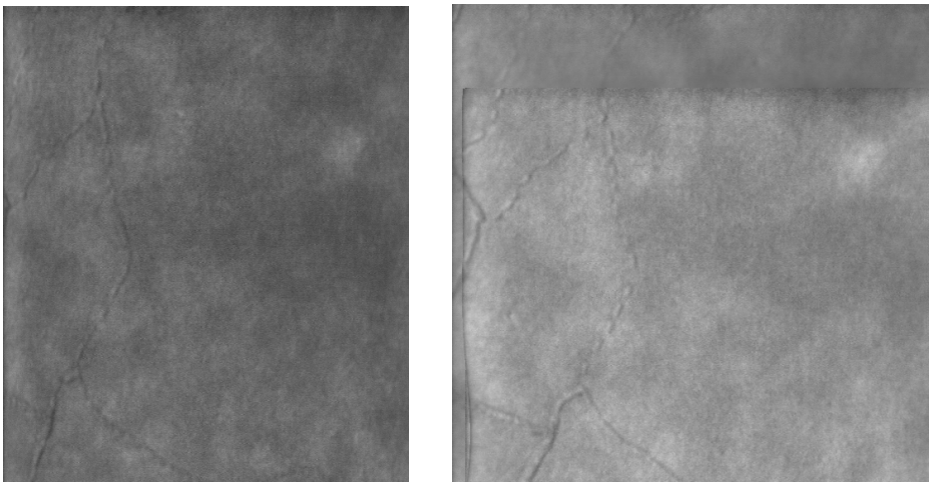
Chapter 3

Method

In this section, the pipeline from patch extraction to cell localisation and matching are described.

3.1 Video stabilisation

The videos acquired are not stabilised and are hard to work with. As a first step the videos should be stabilised. The videos were acquired over a period of 20 - 30 seconds. During this time the eye natural moves, which makes analysis of spatial features difficult. To compensate for this eye motion, a proprietary motion stabilisation step was applied. This algorithm is described in detail in [30]. In short, the stack of images are cross-correlated with the image producing the highest cross correlation of the stack being assigned as the best frame. All other frames are then stabilised in respect of that frame. As scanning devices construct the image pixel by pixel, motion can occur during the acquisition of a single frame. To compensate for this, images are broken into strips and each strip is them registered to the reference frame. A matrix of strip locations is recorded. After all strips in a frame are aligned to the reference frame, a spline fit and bi-cubic interpolation is applied to rejoin and map the strips back to a full frame. This process is applied to all frames in the stack. Strips and frames that do not achieve a desired threshold cross correlation are removed from further analysis. Fig. 3.1 shows a frame before and after stabilisation and corresponding media links of the videos.



(a) **(b)**

Figure 3.1: **(a)** A frame from 790nm video before stabilisation. This media link shows the non-stabilised video. **(b)** The same frame after stabilisation. Video is provided with masks that keep only the valid part of the image. The following media link shows the stabilised video.

3.2 Blood Cell Classifier

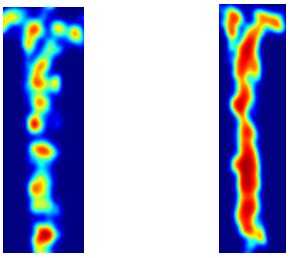
The first task was to classify blood cell patches from non blood cells. To do this a convolutional neural network was used that was configured with adaptive patch size input to account for different fields of view and cell appearance.

3.2.1 Patch Extraction

3.2.1.1 Single channel patch extraction

To train a binary neural network classifier it is required to have positive and negative samples. For this case patches that represent blood cells and patches that don't are required to be extracted from the videos. To be more precise we define positive blood cell patches as windows of certain patch size (i.e 21 x 21) that are centred around a blood cell while non blood cell patches are windows of same size that, although blood cells may be in them, they are not centred around blood cells. The blood cell patches were hand picked by an expert by going through the frames of the 790nm channel videos and marking the blood cells. The non blood cell patches are extracted with one of two strategies implemented. The first strategy was to select random points from the perimeter of the positive patch to serve as the centre for the non-cell patch. From these positions the non cell blood cell positions

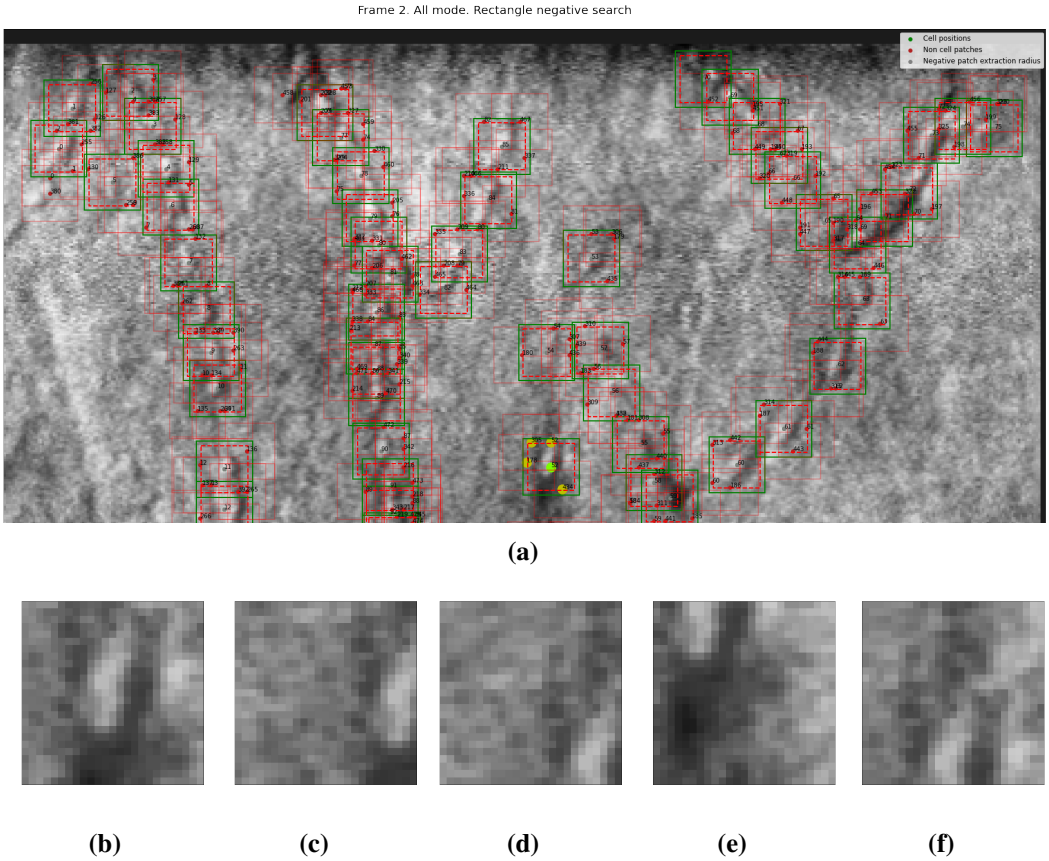
picked. These patch sizes were the same as the positive patches were extracted. This strategy requires that the user picks a radius for the search window. A radius of 21 pixels was most commonly used. An example of such extraction can be seen in Fig. 3.3. Bigger values select patches that are further from the blood cell making it easier for the network to learn positives from negative. On the other hand, the localisation of patches is less accurate since the classifier isn't trained for patches that are close to blood cells. This has two main negative effects. The first is that classifier is much more sensitive, because it picks up cells near the blood cell as positive as well. This is expected since convolutional neural networks are translation invariant by construction, meaning that they can classify the same object correctly regardless of where it is in the image. The second issue, that closely relates to the first one, is that blobs of high intensity in the generated probability map merge together making it harder for the probability map binarisation to pick them up as individual cells. When the negative search window is too small, learning becomes considerably harder because the cell patches are just translations of the non cell patches. This is sometimes preventively hard since the network can not discern between positive and negative patches. The effects of negative patch extraction radius sizes can be see in Fig.



(a) **(b)**

Figure 3.2: The effects of using different negative search window sizes. The images show a heat-map of a vessel segment where hot areas represent areas where the centre of a cell is more probable. **(a)** When using a small negative search window the network learns to classify patches close to positive less confidently, producing cleaner blobs. Estimated cell locations can be extracted from each blob. **(b)** When using a large search window for negatives, the network classifies all patches close to the patch as positive confidently. As a result the blobs merge together. It's not possible to extract correct cell locations from the merged blobs.

The rectangle extraction process is presented in Fig. 3.3



(a)

(b) **(c)** **(d)** **(e)** **(f)**

Figure 3.3: Rectangle negative patch extraction.**(a)** The patch extraction process. The green rectangles represent positive patches while the solid red rectangles represent the negative patches. Points around the dashed red rectangles are picked as negative patch centres. **(b)** A positive patch extracted around the light green point in (a). **(c - f)** Negative patches. Patches around yellow points from (a).

The second negative selection strategy is similar to the first but doesn't require a negative selection radius manually specified, since it's automatically computed for each cell. The non cell positions are picked around a circle of each blood cell location. The radius of the circle of each position is calculated as half the distance between the blood cell's closest neighbouring blood cell. This method guarantees that the negative cell positions are never on top of a positive position. This method also doesn't require an extra parameter for the search radius. The closest neighbours are picked by using a KD tree implementation by the sklearn library[49]. The negative positions are picked as random points along the perimeters of the circles. If more than two points are to be picked from the circle then they are picked uni-

formly. For example if 4 points are picked then the first point is picked at random and the other 3 points are picked at 90, 180, and 270 degrees away from the first point. In addition, a point that is at the middle between every two consecutive points is picked. The reason for these points are for the network to train on hard cases between two blood cells so that blood cell blobs in the probability map are not merged together. This extraction process can be seen in Fig. 3.4.

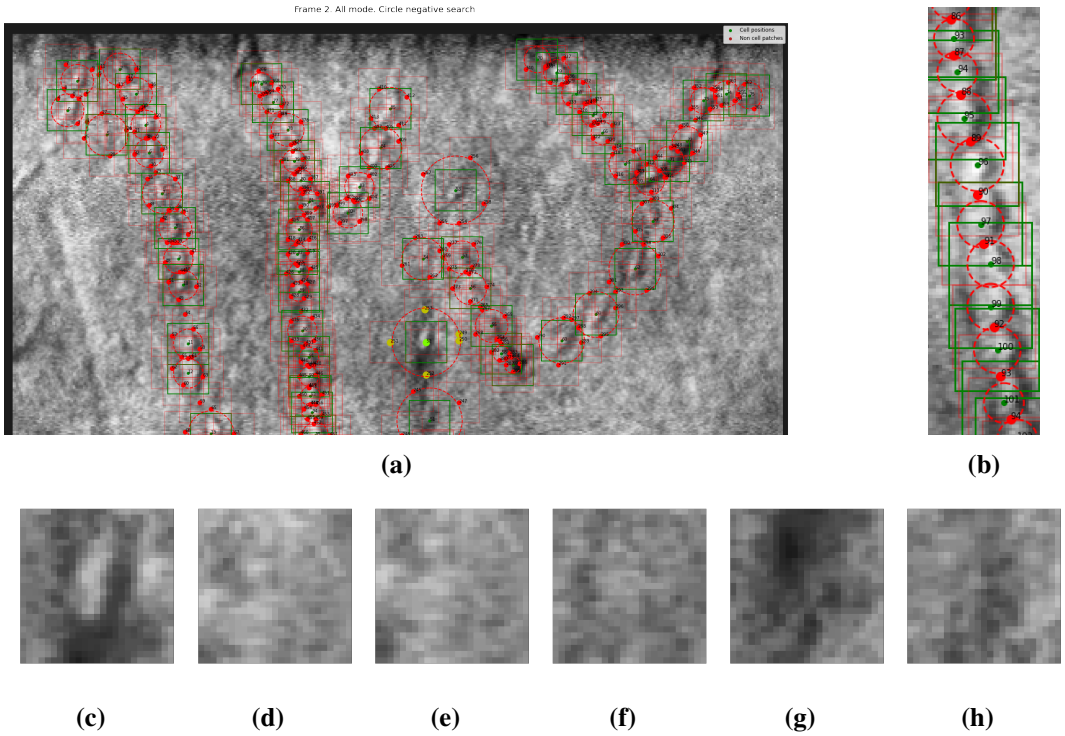


Figure 3.4: Circle negative patch extraction. **(a)** The patch extraction process. The green rectangles represent positive patches while the solid red rectangles represent the negative patches. Points around the dashed red circle are picked as negative patch centres. The circle radius for each point is half the distance to its closest neighbour. The radius is capped to 21 pixels. The negative points are picked uniformly around the circle with small epsilon added or subtracted from the radius. **(b)** In addition to points around the circle, a negative point is always picked at the exact middle between every two consecutive positive points. **(c)** A positive patch extracted around the light green point in (a). **(d - h)** Negative patches. Patches around yellow points from (a).

These two methods are the main methods used for picking positive and negative patches. Until now we talked about picking one dimensional patches from each frame. The problem is that our goal is to measure flow within the vessels, which means we need to incorporate temporal information. The current positive/negative

patch detection does not accomplish this and doesn't take into account additional temporal information that can be used for classifying the patches. In contrast to the classifier in [17], the retinal images in this case come from a video where temporal information can be used. In the next two sections we discuss about temporal patches and mixed channel patches that try to take advantage of the temporal nature of the data.

3.2.2 Temporal patches

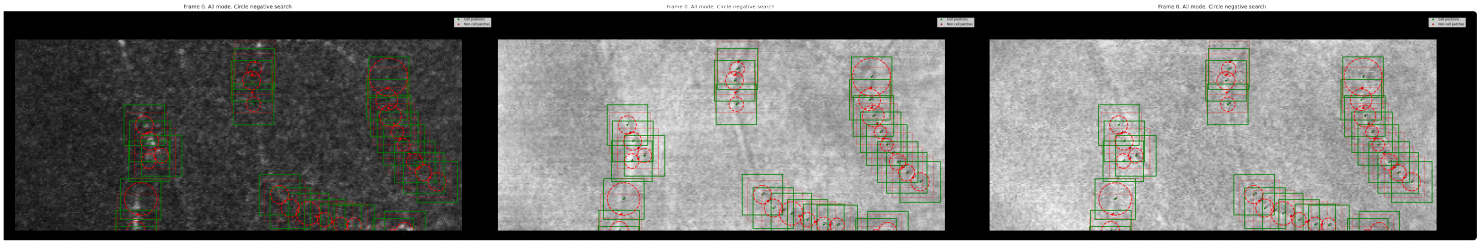
The temporal patches are similar to the spatial patches previously discussed with the difference that they are comprised, at their simplest form, of three channels, one for the previous frame, one for the current, and one for the frame after.

To be more precise, for each frame that has marked cell positions, we take positive and negative positions with the techniques that were previously mentioned but at the same positions we take patches from previous and latter frames. This is to include temporal information that is missed from the spatial patches. One challenge that is faced is the speed of the moving cells relative to the image acquisition speed. This challenge creates a high probability of aliasing, or complication due to similarity in appearance of the cells and cell spacing. To overcome this, the patch sizes must be bigger to include the temporal information. The number of previous and latter frames is adjustable and is called temporal width. For example, a temporal width of 2 would produce a 5 channel patch with the current frame patch in the middle, the first two channels for the patches at the same position in the previous frames and the last two channels from the patches at those potions from the 2 frames after.

A side effect of this method is that we can not use frames that are too close to the start or the end of the video. To be more precise, the frames that closer than temporal width to the start or the beginning can not be used because there are not enough frames to create the patch. This means that such frames can not be used neither in learning, discarding all training samples for those frames, nor in classification. Fortunately, this is an acceptable compromise since the videos recorded are long and losing a couple of frames of data is insignificant.

3.2.3 Mixed channel patches

Another way for including temporal information is to use channels that contain information from the confocal video, the 790nm channel video, and the 850nm channel video. In a similar fashion to the temporal patches, each channel is a patch from a different frame, but in this case the not from the same video. The patches are picked from the positive and negative cell positions in the 790nm channel video. At these positions, the corresponding patches from the confocal and 850nm channel video are picked, then the final patch is the concatination of those 3 channels with the first channel of the patch being from the confocal video, the middle channel being from the 790nm channel and the last channel being from the 850nm channel. The confocal and 790nm channel videos happen to have the blood cells at the same image positions but the 890nm channel video has a vertical displacement in relation to the 790 video so in order to pick the corresponding position from the 850nm video, the 850nm channel frame has to be registered with a method that will be discussed in a later section. Similarly to the temporal patches, the patch size should be bigger to account for the displacement of the cell at the 850nm channel. See Fig. 3.5 for the mixed channel patch extraction process.



(a)

(b) **(c)** **(d)** **(e)**

Figure 3.5: Mixed channel patch extraction. **(a)** The mixed patch extraction process. The patches are extracted from the confocal, the 790nm channel, and the 850nm channel video with a patch size of 35 to 45 pixels. The 850nm channel is vertically aligned with 790nm frame before patch extraction. **(b)** The confocal patch. **(c)** The 790nm channel patch. **(d)** The 850nm channel patch. **(e)** The mixed channel patch, the confocal patch as the red channel, the 790nm patch as the green channel, and the 850nm channel as the blue channel.

A side effect of this method, is that, due to the registration of the 890nm channel frames, some parts of the images can not be used for training or classification. This is not a big problem for the classification since the end goal is to calculate the displacement of the blood cell between the 790nm and the 850nm channel and so only the blood cell locations that are present in both videos need to be located.

3.2.3.1 Limiting negative patch selection within capillaries

In a future section, a process for the creation of a vessel mask that limits the search space for potential cells in the capillaries of the retina is described. Because the search space is limited to patches inside vessels, the negative patches should also be extracted within the capillaries. For this reason the vessel mask is also used to remove any negative cell positions that are outside the capillaries. This process can help the network learn more difficult negative sample cases to boost accuracy when locating the blood cells.

3.2.4 Convolutional Neural Network classifier

To classify the patches it was decided to use a convolutional neural network. The architecture of the network is similar to the one described in [40]. It is comprised of a convolutional part and a dense part that ends up to two nodes for the two classes, blood cell and not blood cell. The convolutional network includes average and max pooling layers, batch normalisation layers and dropout layers. The network classifier structure was build with pytorch. The network structure was not fixed since the input dimension for changed to account for multiple patch sizes and channel numbers. For a model that classifies 23x23 grayscale images, the following structure was used (as printed from pytorch):

```
CNN(
  (convolutional): Sequential(
    (0): Conv2d(1, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): MaxPool2d(kernel_size=(3, 3), stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(32, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): AvgPool2d(kernel_size=3, stride=2, padding=1)
    (7): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (8): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (9): ReLU()
    (10): AvgPool2d(kernel_size=3, stride=2, padding=1)
  )
  (dense): Sequential(
    (0): Linear(in_features=576, out_features=64, bias=True)
    (1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=64, out_features=32, bias=True)
    (5): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU()
    (7): Dropout(p=0.5, inplace=False)
    (8): Linear(in_features=32, out_features=2, bias=True)
)
```

3.2.5 Learning

The network is then trained using the dataset extracted with one of the methods previously described, with a label 1 for the positive and 0 for the negative patches. The learning is done with a gradient descent method with back propagation. The Adam optimiser is used with a learning rate that stars with 0.01 and then gradually drops by 10^{-1} when the validation accuracy stop seeing an improvement for 20 epochs, in order to induce faster learning at the beginning and more controlled

learning at the later stages. The dataset is split into a validation and training set. The validation set size is 0.2 of the whole dataset size. The training happens only using the training set while the validation set is used for measuring the validation loss and the validation accuracy. Early stop is used to prevent the network from over-fitting, and the network is stopped when the validation accuracy stops increasing for an amount of epochs. Because the training dataset is imbalanced with more negative patch samples than true positive patch samples, the negative patches are undersampled while the positive patches are oversampled to induce balanced sampling. The performance of the model is evaluated at fixed intervals. The validation accuracy, specificity, and sensitivity of the classifier is measured. For a binary classifier, *accuracy* is the ratio of correct predictions to all predictions. *Sensitivity*, is the ability of the classifier to classify the positive results correctly while *specificity* is the ability of the classifier to predict negative samples[50]. The sensitivity of a binary classifier is measured as:

$$\text{sensitivity} = \frac{\text{number of true positive}}{\text{number of positive samples}}$$

While the specificity is calculated as:

$$\text{specificity} = \frac{\text{number of true negatives}}{\text{number of negative samples}}$$

While the accuracy is calculated as:

$$\text{accuracy} = \frac{\text{number of correct predictions}}{\text{total number of samples}}$$

While accuracy can be a good performance metric for a classifier when having balanced classes, it can not correctly measure the performance of a classifier when the classes are severely imbalanced. For our case, the negative samples greatly outweight the positive samples due to the extraction method. For example, a classifier that is very specific but with a very low sensitivity could have a high accuracy. To circumvent this, the *balanced accuracy*[51] metric is used. The balanced accuracy

is calculated as:

$$\text{balanced accuracy} = \frac{\text{specificity} + \text{sensitivity}}{2}$$

The weights for the model with the highest validation accuracy and balanced accuracy are saved.

Random small translation transformations are added to the training set to account for small errors of the manual marking of the blood cells and to further decrease over-fitting (Fig. 3.6).

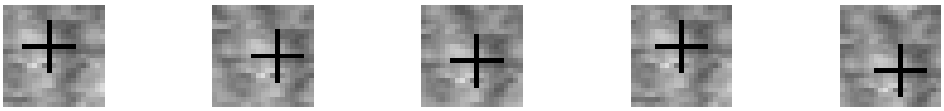


Figure 3.6: Random translations on a cell patch. Small transformations are applied during sampling when training to reduce over-fitting.

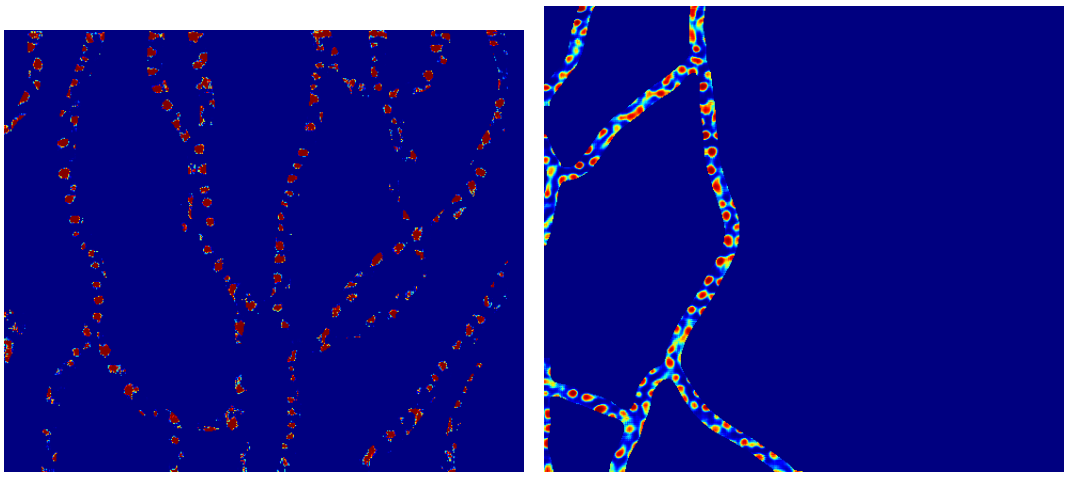
The network and it's learning routine are both implemented using pytorch which is a python machine learning library based on the Torch library that supports automatic differentiation [52].

3.3 Blood cell localisation

The next step is the localisation of the blood cells within the frames of the 790nm channel video. First of all, the capillaries of each video must be detected by using video processing processing methods that are described in the next section. We call pixels that are located on the detected capillaries *vessel pixels*.

3.3.1 Probability Map generation

Afterwards, a patch should be extracted for every vessel pixel of the frame. The frame is padded, by reflection, to get patches that are close to the borders. The trained classifier then assigns a probability for each vessel pixel. The resulting image is called the *probability map* of the frame. Fig. 3.7 shows two examples of probability maps produced for a training and a validation video.



(a) **(b)**

Figure 3.7: Two probability map examples. The values in a probability map range from 0 to 1 and are produced by applying softmax on the output of the classifier for patches extracted around each pixel. The probability for the positive class prediction is assigned to each pixel. The probabilities are only calculated for vessel pixels by using a vessel mask. **(a)** An example of a training video probability map. It can be observed that the network is fairly confident. **(a)** An example of a test video probability map. The predictions are not as confident but areas in the centres of the blobs have higher intensities than in the edges.

3.3.2 Probability Map binarisation and cell localisation

Finally, the probability map is binarised and the *estimated cell locations* are extracted from the centres of connected components of the binarised image by taking the area of the blob and the intensities of the blob from the probability map into account.

The probability map is a gray-scale image with values from 0 to 1 with pixels close to one when near estimated blood cell locations. Because of translation-invariant, areas around the estimated blood cell locations are expected to also have high intensities forming blobs around potential blood cell locations. The purpose of this step is to calculate the blood cell locations from the probability map that maximise

The location extraction process is tantamount to the process described in[32]. First the probability map is blurred with a sigma σ. Afterwards, an extended-maxima transform is applied. The extended-maxima transform detects regional maxima of the H-maxima transform. Regional maxima, or local maxima, are connected components that have higher intensity that the neighbouring pixels. The H-maxima transform filters the regional maxima of an image based on a threshold H. The local maxima that have a height difference from their neighbours smaller or equal than H are being suppressed, leaving only regional maxima who have a height difference greater than H[53]. The transform is implemented using scickit-image's morphology reconstruction function and mahotas'[54] regmax function. The result of the extended-maxima transform is a binary image. The connected components binary are potential locations for the blood cell locations. Weak candidates are discarded when the maximum intensity in the region from the filtered probability map is less than a threshold T. Finally, the centres of the remaining regions are considered as the blood cell estimated locations. To be more precise, the estimated cell locations can be the weighted centre of the regions where the intensities of the probability map at the blobs are considered, or the max intensity pixel in the region or simply the geometric centre of mass of the regions. Please see Fig. 3.8 for the binarisation and cell localisation process.

To tune the hyper-parameters σ, H, and T a grid search on a set of potential values is performed to maximise dice coefficient on manually marked validation videos. Dice coefficient and the performance evaluation of the tool will be discussed in the evaluation chapter.

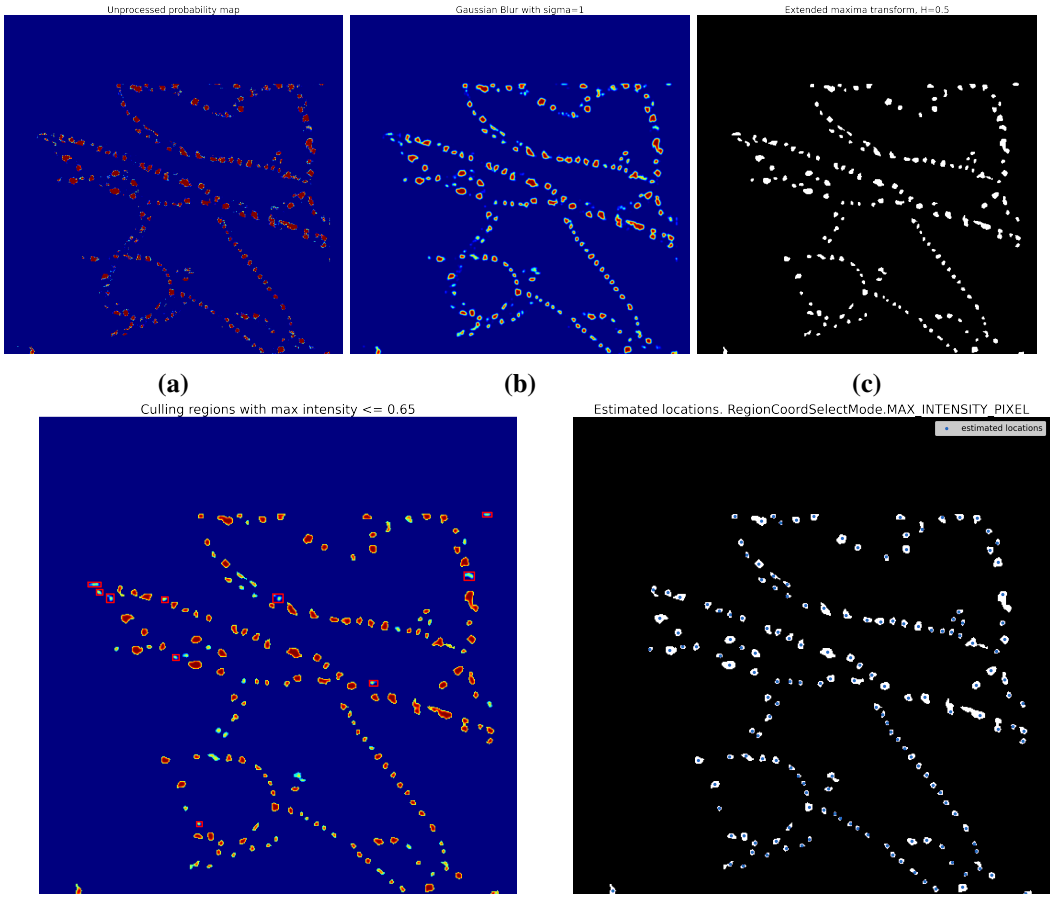


Figure 3.8: The cell localisation process from probability map. **(a)** The probability map as is produced by classifying each pixel. **(b)** Applying Gaussian blur. **(c)** Applying extended maxima transform. **(d)** Regions that have a maximum intensity less than a threshold are discarded. (Regions inside red squares). **(e)** Select the maximum intensity pixel as the estimated location for the cell.

3.4 Vesselness - Capillary detection

Because the network is trained with blood cell samples and non blood cell samples that are close to the positive samples, the classifier has a low accuracy on pixels that are away from the vessels. As a result, areas that are away from the capillaries can be classified randomly as positive. For this reason, the detection of the capillaries within a video is important, both for improving the accuracy of the tool, and for making the classification process faster by reducing the search space within the image.

3.4.1 Standard Deviation Image

Having a registered video of the capillaries of the retina, the vessel image of the video can be created by taking the standard deviation of of each pixel across the frames. Calculating the standard deviation of each pixel alone can yield good images where the capillaries are highlighted but the results are better by applying a video processing step to the frames to increase the motion contrast.

One such method is described in [9]. In brief, the method is similar to difference video technique that is used for background subtraction but uses frame division instead. For every pair of consecutive frames j, j + 1 the division image D(x, y) is calculated as:

$$D_j(x,y) = \frac{I_j(x,y)}{I_{j+1}(x,y)}$$

Then, multi-frame division frames M(x, y) are calculated as:

$$M_j(x,y) = \frac{D_j(x,y) + D_{j+1}(x,y)}{2}$$

In [19] it is claimed that, this the multi-frame division method can help remove unchanging parts of the frame, while highlighting areas with high relative motion, in our case the capillaries where the erythrocytes move through. In this work, the technique gave better results by applying adaptive histogram equalisation to each multi-frame division frame. Histogram equalisation is used to to increase contrast in images without losing information[55]. For this purpose scikt-image[56] implementation of Contrast Limited Adaptive Histogram Equalisation (CLAHE) was

used. Lastly, the standard deviation image is calculated. The resulting image usually highlights the capillaries better than by computing the standard deviation image of the unprocessed video.

Another method for highlighting the capillaries is described in [15]. Using this video normalisation method, the frames are first smoothed using a Gaussian filter. Then, each frame is normalised by being divided by it's mean. Next, each pixel is divided by the average of the pixel's intensity across the frames. According to [15], this method can correct changes in intensity that appear from minor variations in the tear film or the patients positions. Finally, just as before, the standard deviation image is calculated by taking the standard deviation of each pixel across the frames of the processed video.

Both these methods can be combined, usually by applying the latter method before the first and then calculating the standard deviation image.

Both these methods could also be used as a processing step prior to patch extraction but unfortunately the learning doesn't benefit much.

3.4.2 Vessel Mask generation

By having the standard deviation image, the next step is to segment the image to get the *vessel mask* where al background pixels are 0. For this purpose we use the following image processing techniques: First, we use frangi filters ridge operator that is known for it's ability to detect vessel-like structures[57]. Skcikit-image's implementation of the filter is used. Next, optionally an additional adaptive histogram equalisation step. Afterwards, the frangi image is thresholded to create a binary image. The threshold is calculated automatically using Otsu's method [58]. Usually at this stage, the binary image capture the vessel structures pretty well but the thresholding process introduces some blob-like structure that are produced from either noise in the standard deviation image or small vessel segments. To remove these blobs, a morphological open operator is used. The morphological opening operator first erodes the image to remove small connected components and then, using the same structuring element, dilates the image to keep the other structures to have the same shape. The binary image at this phase captures the biggest vessels

but doesn't necessary connect vessels that are normally connected. This can occur because at some parts of the capillaries there isn't much relative activity, and so the response in the standard deviation is not as strong. For this reason, a morphological close operator is used. The operator is the opposite of the opening operator since it first dilates the image and then, using the same structuring element, erodes the image to bring the structures to their original shape. This operator is used to fill in small holes and close gaps while maintaining the shape of the objects in the image. Because at this stage the vessel mask can sometimes be too thin, an optional binary dilation step is followed. As a final step, connected components of area smaller than a predefined value are detected and removed. The results of this process can be see in Fig. 3.9 and Fig. 3.10.

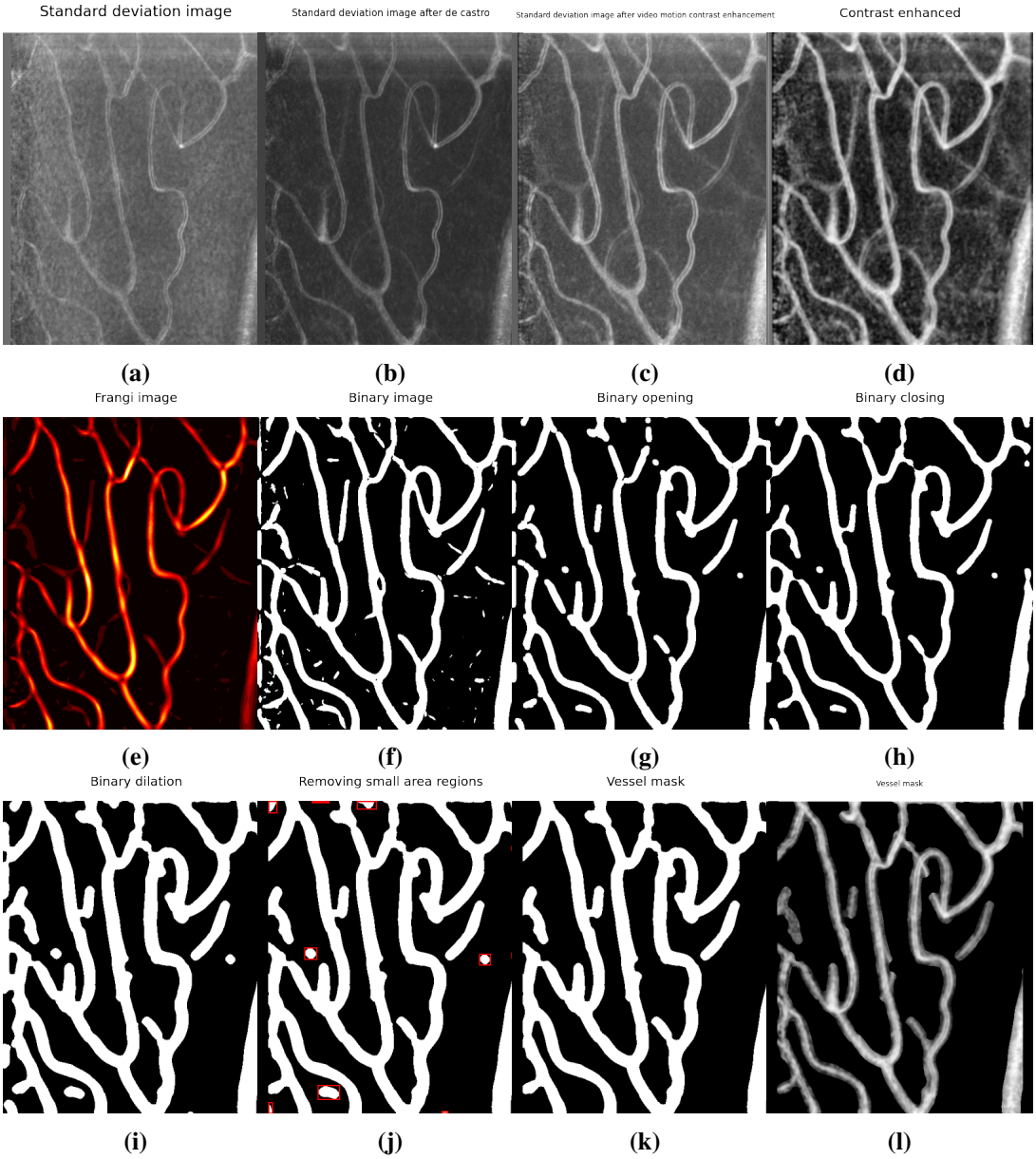


Figure 3.9: The vessel mask creation process. **(a)** Simple standard deviation image doesn't highlight vessels very well. **(b)** Using the method in [15] gives good results for this video. **(c)** The standard deviation image after video motion contrast enhancement. This is done after applying both the video normalisation described in [15] and the motion contrast enhancement described in [9] with the extra step of adaptive histogram equalisation applied to each multi-frame. **(d)** After applying adaptive histogram equalisation to (c). **(e)** After running the frangi filter on (d). Hot areas score high in vesselness. **(f)** After applying otsu's threshold to create binary image. **(g)** After applying morphological opening to remove small speckles and thin offshoots from the main capillaries. **(h)** After applying morphological closing to fill small holes and connect capillaries that should be connected. **(i)** After applying optional binary dilation to (h). **(j)** Detect and remove small connected components (that have an area smaller than a value that is predefined). **(k)** The final vessel mask. **(l)** The vessel mask applied on (d) showing that the vessel mask captures the areas with a lot of movement well.

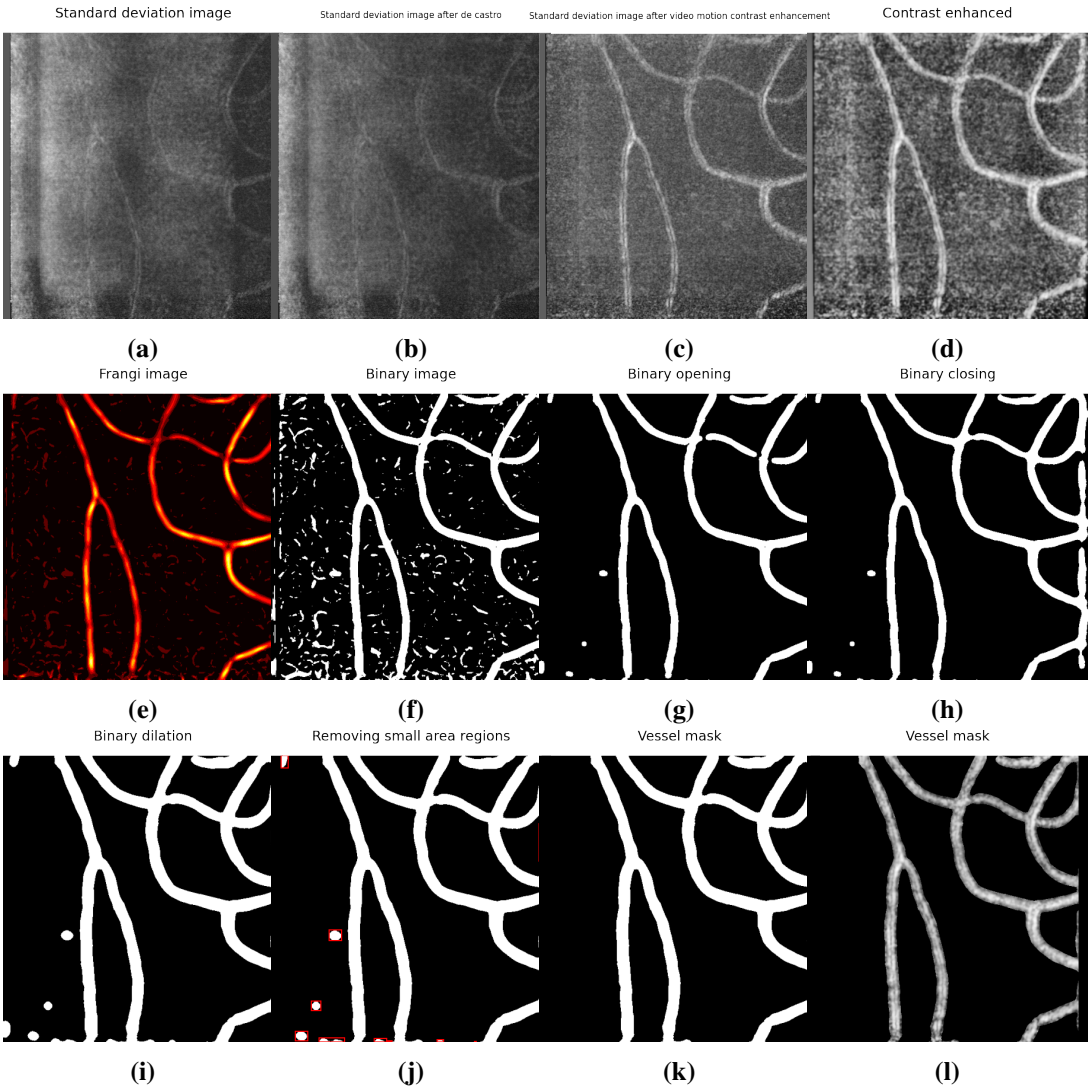


Figure 3.10: The vessel mask creation process. Same process as the one in described Fig. 3.9. In this case it can be observed that a simple standard deviation image on the unprocessed video (a) or after applying the video normalisation method in described in [15] (b) doesn't highlight the capillary segment very well. Also applying the motion contrast enhancement step (c) and finally applying an extra adaptive histogram equalisation step (d) highlights the capillaries better.

3.5 Channel registration

Because there's a spatial disparity between the two channels, the frames from the 850nm must be aligned to the frames from the 790nm channel. It is known that the displacement is strictly vertical. This fact is an important constraint since the search-space is limited only to the vertical axis. Because the vessel masks for videos of both channels can be accurately produced with the method previously discussed, they're almost the same but with a vertical displacement. To align the 850nm channel all possible displacements, from 1 pixel of vertical displacement to the height of frame, of the 850nm video vessel mask are tried while also calculating the overlap with the 790nm video vessel mask. The overlap is measured using the Sørensen–Dice coefficient which is a metric that can measure how close two sets of data are. It can be used with discrete data, for example when comparing the outputs of the classifier to the actual output, or to check the overlap of two binary segmentation results[59]. The Dice Similarity Coefficient (DSC) is calculated as:

$$DSC = 2\frac{|X \cap Y|}{|X| + |Y|} \tag{3.1}$$

Where, for our case, $|X|$ and $|Y|$ is the number of pixels that are true for the two vessel masks correspondingly and $|X \cap Y|$ are the number of pixels that are true in both masks. The displacement that maximises the Dice coefficient is selected as the translation needed for the alignment of the two channels. Examples of the registration can be seen in 3.11.

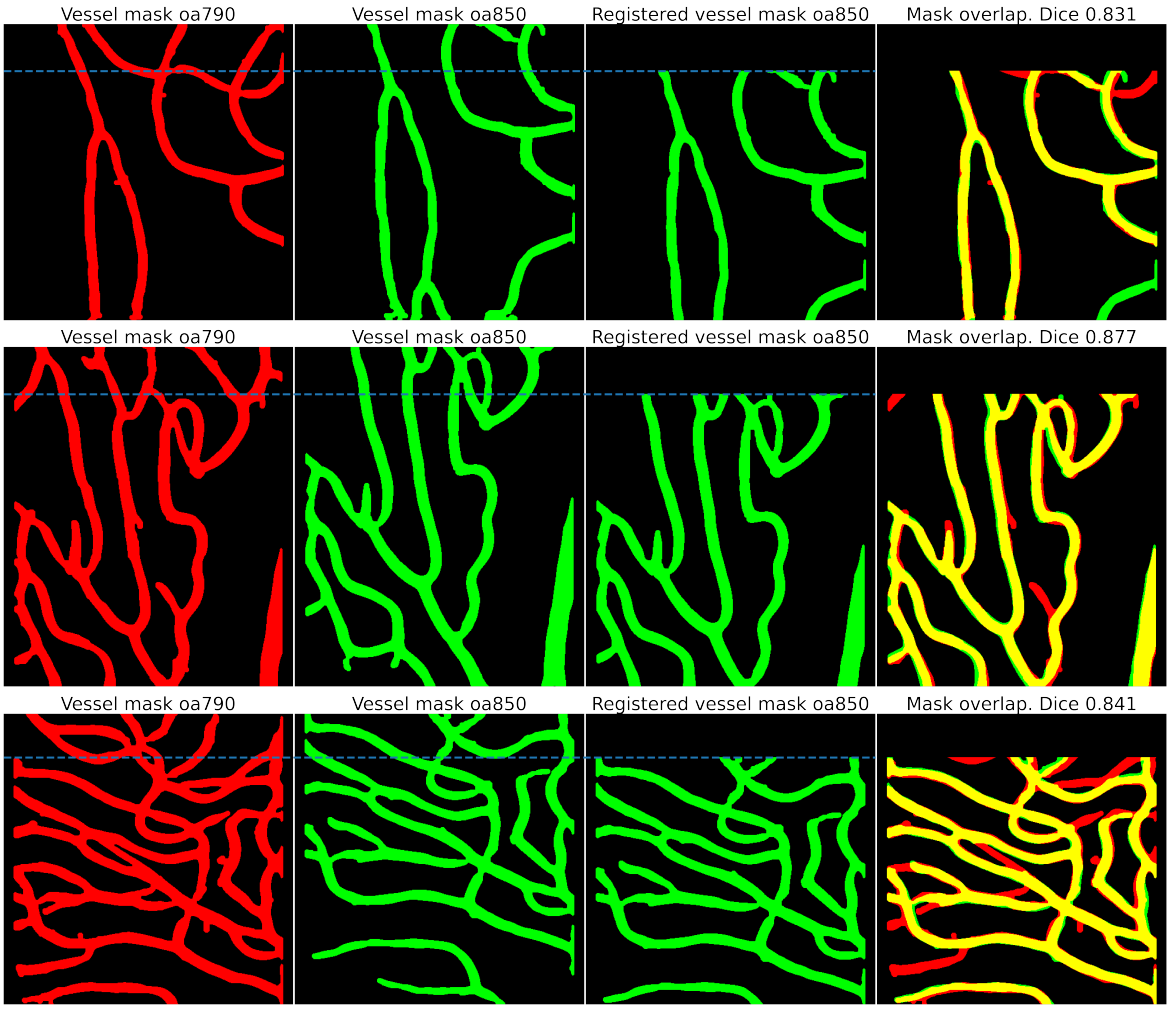


Figure 3.11: Registration examples. Each row shows a registration example for a different video. The red mask is the vessel mask generated for the 790nm channel video while the green vessel mask is the vessel mask generated for the 850nm channel video. The third column shows the 850nm vessel mask aligned to match the vessel mask of the 790nm channel. The last column shows the overlap between the two masks with yellow parts showing parts where the masks match. Even if the vessel mask detection process can have slight differences for the two channels, the registrations are consistently correct.

3.6 Blood cell matching between channels

Once the blood cell locations are estimated, the blood cell locations from the 790nm channel video should be matched with the corresponding cell locations in the 850nm channel. That would allow for calculating the translation of the blood cell and, since the time disparity is known, the velocity.

Because the final goal of this work is to estimate the blood flow velocity, we can calculate the average displacement of the erythrocytes along a straight capillary and then divide by the time difference of the two channels. To do this we follow the method of calculating the *average cell* along the capillary which is similar to the method described in [15]. For a single video, a straight and long vessel segment should be selected because the displacement should be parallel to the flow. If the vessel segment has a curve, then the velocity measurement would be inaccurate because the erythrocytes will no longer move in a straight line and so the distance travelled between the two channels would be unknown, even if the cells were correctly matched . At the moment, the vessel segment selection should be done by the user for better accuracy. For our use case, this is not a big problem since the user should just select one vessel segment per video session. Methods for automatic vessel segment selection were attempted but the lack of robustness in the results does not excuse the automisation of this simple process.

Once the vessel segment is selected, a trained network runs over the pixels in the vessel to estimate the erythrocyte locations with the process previously discussed. For each estimated cell location in the vessel segment, a patch of 51x51 pixels is extracted around the location. For the same locations, patches of the same size are extracted in the **registered** 850nm channel frame. Next, the patches for each channel are averaged. This produces an average cell in the centre of the average790nm patch. This is because the patches averaged are expected to be centred around blood cells. On the other hand, for the average 890nm patch the average cell is at a distance from the centre because. This is due to the fact that the patches were extracted at the same location in the 890nm channel meaning that, during the time between the two channels were captured, the blood cell moved. Because it was

observed that the two average cells shared similar characteristics like the roundness and the change of intensity from high to low, template matching was used to find the displacement of the average cell. A smaller patch of 21x21 pixels is extracted in the centre of the average 780nm patch and then the template is matched to the average 890nm patch. The centre of highest similarity part of the 890nm patch is considered the matched average cell location. The vertical and horizontal displacement of the matched location from the centre of the patch is considered the distance that the cell has moved. Please see Fig. 3.12 for a representation of this process.

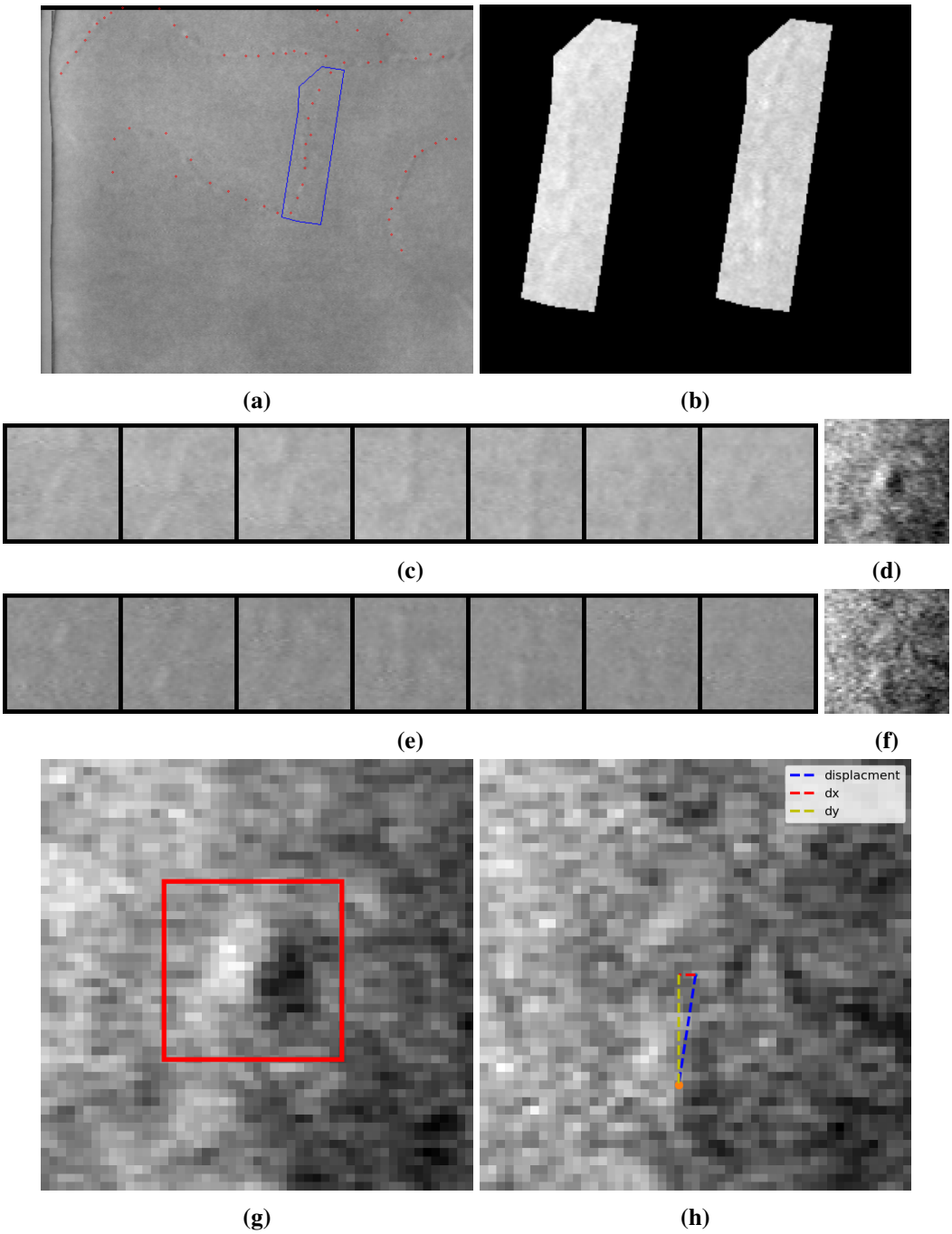


Figure 3.12: Average cell matching using template matching. **(a)** First a straight vessel segment with blood cells is selected. **(b)** The vessel segment in the 790nm channel (left) and 850nm channel (right). **(c)** The 51x51 patches extracted around the cell locations from the 790nm channel. **(d)** The average patch of the 790nm patches. It can be observed that the average cell is located at the centre of the patch. **(e)** The patches extracted at same locations from the 850nm channel. **(f)** The average patch of the 790nm patches. This time, the average cell is displaced from the centre. **(g)** The template image extracted from the centre of the average 790nm patch. **(h)** Performing template matching on the average 890nm patch, we find the part of the image with the highest similarity with the template. The centre of this part is considered as the matched location (orange point in image). The distance from the centre of the patch is considered as the displacement of the the cell.

In addition to template matching, feature matching was also used to match blood cell locations between the two channels. In this case, the average cell doesn't need to be centred in the average 790nm patch. This can happen when the estimations are not exactly centred in the middle of cells. The process is almost identical to the one previously described but this time, instead of extracting a template from the middle of the average 790nm patch and doing template matching on the 850nm patch, feature matching is directly performed on the patches. Feature matching is a process of detecting highly characteristic spatial features, called **key points**, and matching them between two images. The key points in algorithms such as SIFT [60] are represented as descriptors that are usually translation, rotation, scale, and illumination invariant. In this case the SURF[61] feature detector was used because it was shown that it can outperform many of it's competitors and it's comparable to SIFT. In addition, an open source implementation of SURF is offered with opencv [62] in xfeatures2d class.

Performing SURF on the two average patches one or more key points can be matched. The first key point match is taken as the match of the blood cell in the two channels. Instead of taking the distance from the centre of the patch, the displacement is calculated as the vertical and horizontal difference from the matched key point positions (Fig. 3.13).

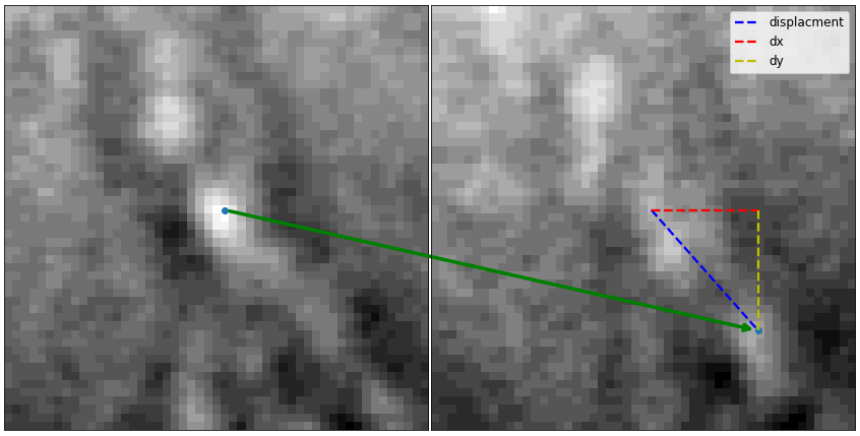


Figure 3.13: Average cell matching using feature matching (SURF). On the left is the 790nm channel average patch and on the right is the 850nm channel average patch. The key point location on the left image is not necessarily at the exact centre of the image because the feature detector may find a location in the 790nm patch with a more characteristic feature that is not exactly in the middle. The displacement of the cell is set to be the distance between the matched keypoints.

3.7 Cell selector GUI

Because the process of picking cells can be cumbersome and error-prone, it was decided to create a GUI that would show the networks estimated positions. A user can then keep or add new points. The selection can be saved the future training (Fig. 3.14).

This tool makes it easier to pick cells that the network missed. We believe that the false negative samples can share some characteristic that the network hasn't learn. By specifically picking cells that the network missed, we believe that the classification performance of the network could be significantly improved.

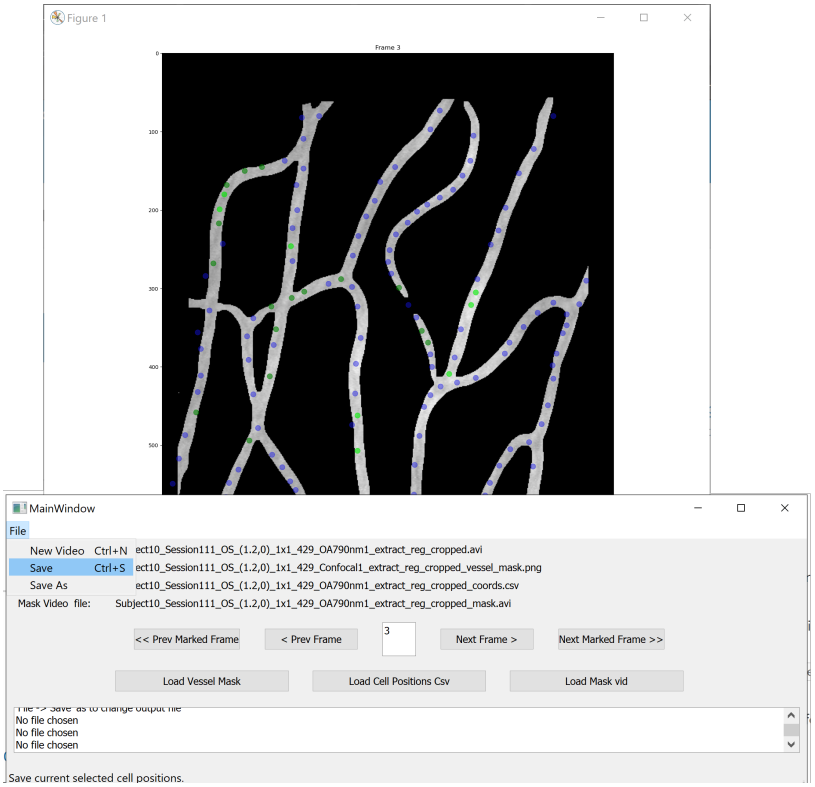


Figure 3.14: User loads a video and the estimated positions as a csv. The estimates are shown as blue points. The vessel and stabilisation mask are automatically found and used. The user can go through frames freely. For each frame the user can click on the blue points to mark them as suitable for training. User can also pick points that are missed by the network (light green points). Right-clicking can undo a wrong selection. Finally, the selected green locations can be saved a csv.

Libraries used

For the modules and libraries used please see appendix F

Chapter 4

Evaluation

4.1 Classification evaluation

In this chapter, the results from from different network configurations are evaluated and compared. Each network is to be evaluated in two stages. First, the ability of the classifier to classify positive from negative cell patches is evaluated. Positive and negative patches are extracted from all the training videos and then they are split to a training and a validation set. The dataset was randomly split into training and validation set with a 5 to 1 ratio, meaning that, for every training sample there is 1 validation sample. An additional and independent test set from patients not used in the training and validation set was developed. The reason for this, is to give a better idea how the networks generalised to data of the same type but with potentially slightly different characteristics than were used in training. The test videos, along with sample results for those videos, can be found in E. Although the validation dataset from the split is never used in training, it can be similar enough to the training dataset to give false indications of the classifier's performance. The classifier is also evaluated with patches extracted only from the test videos. In the evaluation results, both the validation performance and test performance will be shown. It must be noted that the patch extraction method for the test performance evaluation will be the same for all networks, regardless the extraction method used for training. This is to provide a fair evaluation on the classification on the test patches. The

classification performance of the network is evaluated with the following metrics:

$$accuracy = \frac{number\ of\ true\ positives + number\ of\ true\ negatives}{number\ of\ samples}$$

$$sensitivity = \frac{number\ of\ true\ positives}{number\ of\ positive\ samples}$$

$$specificity = \frac{number\ of\ true\ negatives}{number\ of\ negative\ samples}$$

$$balanced\ accuracy = \frac{sensitivity + specificity}{2}$$

A true positive sample is when the prediction and the ground truth are both positive while a true negative is when both the prediction and the ground truth are negative. The most important metric for our case is that of balanced accuracy since the datasets are imbalanced, with the negative samples greatly outweighing the positives. Simply using the normal accuracy metric can not evaluate the classifier correctly since a specific classifier with a low sensitivity would still score a high accuracy.

4.2 Localisation evaluation

The second evaluation stage,is to evaluate the network on it's ability to localise the blood cell withing the frames of the tests videos. Although a high balanced accuracy can be a good indicator whether the network can create good probability maps, this is not always the case. For example, if during the negative patch extraction process patches are picked in a wide radius around the positive patches, because the positive and negative samples will have very distinct features, the classifier will be accurate but won't be able to localise blood cells correctly. The translation in-variance of the convolutional neural network would make it so that patches that have a cell in them would be classified as true positive, regardless whether it's in the centre of the patch or not. In the other hand, if negatives are picked in between and close to the positive samples, it is more difficult for the classifier to have high balanced accuracy but will be able to estimate the locations of the blood cells more accurately. For this reason, this second evaluation step is necessary.

The cell position estimation is evaluated similar way as in [17]. In short, a cell location estimation is considered to be true positive if it's in within a distance d of a manually marked position in the test video frame, where d is .75 of the median spacing between the ground truth positions. If more than one estimated position is within distance d to a ground truth position, then only the one closest to the manually marked cell is considered a true positive while the other one is considered a false positive. In addition, estimated points that are not within a distance d of any ground truth positions are also considered false positives. Finally, manually marked positions that are not matched to any estimated location are considered as false negative. An example of this evaluation can be seen in seen Fig. 4.1.

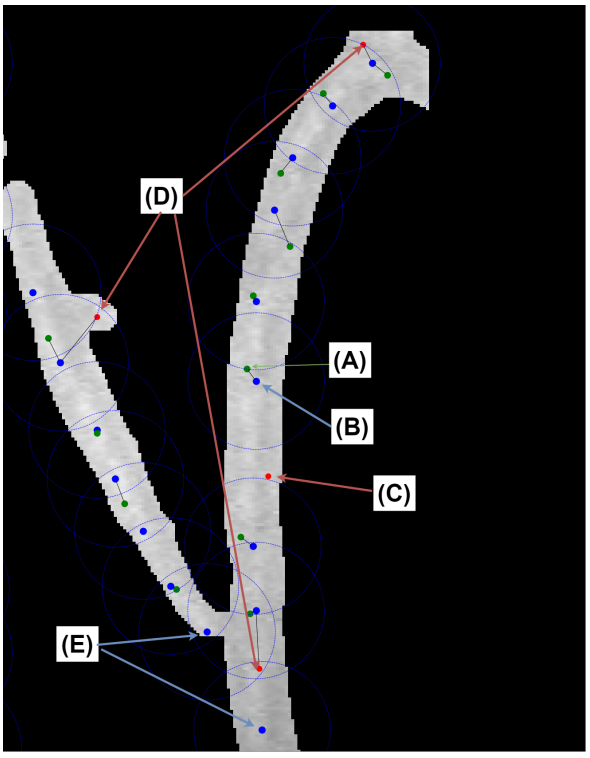


Figure 4.1: Blue points are manually marked cell locations while green and blue point are estimates. The blue circle shows the maximum accepted distance for an estimation to be matched to a ground truth position. The radius d is 0.75 of the median spacing between the manually marked positions. **(A)** A correctly estimated cell position that counts as a true positive. It is inside the accepted distance d of a ground truth position. A true positive. **(B)** A manually marked point that is matched to an estimated point. **(C)** An estimated point that is too far away from any ground truth positions. A false positive. **(D)** Estimated positions that inside acceptance radius d but other estimates are closer to the matched marked positions. Counted as false positives. **(E)** Manually marked positions that are not matched to an estimation. Counted as false negatives.

With the number of true positive (N_{TP}), number of false negatives (N_{FN})), and number of false positives (N_{FP}) defined this way, the number of manually marked positions N_{manual} is:

$$N_{manual} = N_{TP} + N_{FN}$$

And the number of estimated locations N_{est} is:

$$N_{est} = N_{TP} + N_{FP}$$

the following metrics are used to evaluate the blood cell location estimation:

$$\text{true positive rate} = \frac{N_{TP}}{N_{manual}}$$

$$\text{false discovery rate} = \frac{N_{FP}}{N_{est}}$$

$$\text{Dice's coefficient} = \frac{2N_{TP}}{N_{manual} + N_{est}}$$

In an ideal system, the true positive rate and a dice coefficient should be 1 while the false discovery rate should be 0. Adding to these metrics, not all classifications of the same Dice are equal since for some the distance of the estimations to the ground truth can be smaller than others. For this we also include the mean distance of every true positive to it's closes ground truth.

As mentioned in a previous section, the binarisation of the probability map and the localisation of the cells from the binary image requires 3 parameters. The σ for the gaussian blurring, the height difference H for the extended maxima transform, and a threshold T for filtering regions with low max intensity. The configuration of this three parameters can greatly affect the performance of the localisation. To evaluate the performance of the networks with the same localisation parameters, the σ, H, and T for each test video were fixed. The values were fixed by maximising Dice's coefficient, by performing a grid search over a set of possible parameters, on the first experiment. To be more precise, for each marked frame of each test video the parameters the σ, H, and T that maximise the Dice's coefficient for that frame

are picked and then, the accumulated parameters from each frame are averaged for each test video. As a result, for each video a fixed σ, H, and T is used for all the networks.

4.3 Observations from the results

To show the results 4 tables were created. In the first table the different model configurations are linked to a specific unique id (Tab. A.1). In the next table, the classification performance of the models on the validation patches (Tab. B.1) is displayed. The third table demonstrates the classification performance of the networks on the patches extracted from the test videos (Tab. C.1). In the last table, the evaluations of blood cell localisation of the models are presented (Tab. D.1). All the tables can be found in the appendix.

In this section only a subset of the results is shown in the tables Tab. 4.1 and Tab. 4.2. The model unique ids are the same with the ones in the appendix, just a subset shown here. When referring to model's by it's unique id then it's configuration, validation, and test performance can be seen in the corresponding tables. The conclusions and observations are extracted from all the experiments but only some of the models are shown here for an easier demonstration for the reader.

From the experiments presented in tables Tab.4.1 and Tab. 4.2 we can see that temporal and mixed channel patches do not offer any particular benefits. It seems that single channel patches can outperform both these patch extraction techniques.

From the models 35, 37, and 41, (For configurations see Tab. 4.1) all trained with temporal patches, one can see that using a larger patch size (55 pixels) is beneficial. We believe that particularly large patch sizes can offer temporal information since the cells move further between individual image captures within a single channel, meaning that large patches should be used to capture that displacement. We think that an even bigger patch size is required to capture the temporal information since the time difference between the frames of the same channel are particularly large and the displacement of a cell can be too big to fit in smaller patch size. On the other hand, using a bigger patch size would make the classification harder when

picking positives close to negatives since the majority of the patches would overlap.‘ Smaller patches can still have a positive Dice’s coefficient but they seem to eliminate useful temporal cues. In addition, when no temporal cues are included then the 850nm channel only adds random and irrelevant information. For example, comparing model 4 and 35, both with patch size 21 (Tab. 4.1) but the latter with a temporal width, the first outperforms the latter while also being simpler and faster in training (Tab. 4.2).

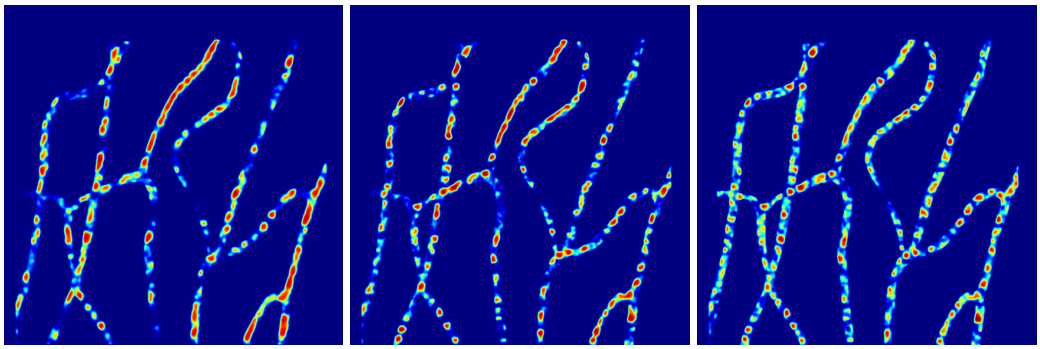
When it comes to mixed channel patches, the observations are similar to the the temporal patches when only the 790nm and 850nm channels are used. Because there is a some time difference between the channels, the patch size must be larger 31 pixels in order to capture temporal information. When comparing models trained with mixed channel patches and a small patch size, then the model trained with all 3 channels performs better than the one trained only with the 790nm and 850nm channel. For example, comparing model 11 and 16 (for configurations see Tab. 4.1), the first one was trained only on 2 channels while the latter on all channel patches, both with a patch size of 21. The second performed better than the other (Tab. 4.2). We believe that the confocal channel can give useful information when the patch size is small, since there is no time difference between the confocal and 790nm channel, while the 850nm channel includes random irrelevant information that do not help the classification in a meaningful way. For small patches the cell displacement is not captured. It is presumed that only using the confocal and the 790nm channel with a small patch size would be even more beneficial but this was not implemented because it is believed it won’t give significant benefits over the single channel patches since, due to the optics of confocal videos, the blood cells are not always visible. On the other hand, when the patch size is large the model performs better for the two channel case. Comparing model 15 and 11 (for configuration see Tab. 4.1), both trained with 2 channel patches, it can be observed that model 15 which has a large patch size of 45 pixels outperforms the latter (Tab. 4.2). This is an indication that including the 890nm channel can give useful temporal information given that patch size is large enough.

Although using temporal patches and especially mixed channels can include additional useful temporal information, the methods do not show any benefits when being compared to single channel techniques.

Another way to use temporal information is to increase the contrast of the images using video normalisation method described in [15] prior to training and classifying. Model 22 with a patch size of 23 showed no significant improvement and so that idea was also scrapped since it added an additional layer of complexity without benefiting learning.

A bigger factor for getting better localisation performance was proved to be a random vertical and horizontal translation when sampling the patches during training. In particular, the combination of a patch size of 23 and a random translation of 5 pixels gave the two best localisation results with the lead being on model 6 with a Dice of 0.789 and a 19 having a 0.786 as a close second 4.2. The patch size must be sufficiently large enough so that the random translations do not completely alter the sample. For example, when using a patch size of 21 with a random translation of 5 pixels such as model 10 then the localisation performance is bad. A bigger patch size of 25 pixels can also support a big random translation, like model 7, but a patch size of 23 pixels is shown to be better.

All the previous examples used the circle negative patch extraction strategy limited within the vessel mask. One of the biggest factors of the localisation performance is the way is the negative are picked prior to learning. By comparing models 6, 42, and 43 (for configurations see 4.1) one can see how much the negative patch extraction process can affect the final localisation performance of the network (4.2). The three models were trained with the same configuration except the way the negatives are picked before training. In short, picking negative samples at a big distance from the positives creates bad probability maps while a smaller radius have a better separation. Finally, the smart negative sample picking described in the previous chapter allows for a much cleaner probability map with a higher blob separation. This effect is demonstrated in Fig. 4.2.



(a) **(b)** **(c)**

Figure 4.2: A comparison of the probabilities map created by 3 different patch extraction processes. **(a)** Using a rectangle patch extraction with a radius of 31 pixels. Because negative patch samples are picked in a square 31 pixels away from the blood cells the network doesn't classify pixels close to blood cells as negative. As a result, the blobs of high intensities are merged together leading to a probability of a bad quality with no clear separation between the cells. **(b)** Rectangle negative patch extraction in a 21 pixel radius around the positive cell. The results are better but some blobs are not clearly separated because not enough negative samples are picked between cells. **(c)** Using a circle patch extraction and limiting the negative positions within the vessel mask of the training videos. The separation between the blobs is much cleaner because the network was trained on many hard negative examples between the cells and parallel to the capillaries instead of around it.

Model uid	Channels	Patch Size	Translation Pixels	Preprpocess
12	2	31	2	FALSE
14	2	45	2	FALSE
15	2	45	2	FALSE
10	2	21	5	FALSE
11	2	21	2	FALSE
16	3	21	2	FALSE
18	3	35	2	FALSE
21	3	27	2	FALSE
4	1	21	2	FALSE
5	1	23	2	FALSE
22	1	23	2	TRUE
6	1	23	5	FALSE
19	1	23	5	FALSE
7	1	25	5	FALSE
8	1	21	2	FALSE
9	1	21	5	FALSE
35	temporal	21	2	FALSE
37	temporal	55	5	FALSE
41	temporal	35	2	FALSE
42[(1)]	1	23	3	FALSE
43[(1)]	1	23	3	FALSE

Model uid	Balanced Accuracy	Accuracy	Sensitivity	Specifiticy
12	0.823	0.822	0.825	0.821
14	0.808	0.813	0.803	0.814
15	0.807	0.829	0.783	0.832
10	0.772	0.731	0.819	0.725
11	0.811	0.818	0.802	0.819
16	0.806	0.784	0.830	0.781
18	0.825	0.833	0.816	0.834
21	0.805	0.827	0.779	0.830
4	0.830	0.821	0.840	0.820
5	0.824	0.833	0.813	0.835
22	0.780	0.788	0.771	0.790
6	0.814	0.801	0.828	0.800
19	0.809	0.797	0.821	0.796
7	0.822	0.828	0.815	0.828
8	0.821	0.790	0.858	0.785
9	0.772	0.788	0.752	0.791
35	0.813	0.779	0.853	0.774
37	0.773	0.733	0.818	0.727
41	0.815	0.802	0.830	0.801
42[(3)]	0.832	0.813	0.862	0.803
43[(4)]	0.832	0.832	0.833	0.832

Table 4.1: **(left)** Model configurations. For a full explanation of column names please see A.1 in the appendix.
(right) Classification performance on validation patches.
(1) Model trained with negatives extracted in a rectangle negative search window with radius of 31 pixels.
(1) Trained with a rectangle negative search window with radius of 31 pixels.
(3, 4) Validation patches extracted with the same method as the training patches, hence the good validation accuracy.

Model uid	Balanced Accuracy	Accuracy	Sensitivity	Specificity
12	0.803	0.825	0.777	0.828
14	0.759	0.823	0.684	0.834
15	0.798	0.823	0.769	0.826
10	0.780	0.804	0.752	0.809
11	0.801	0.860	0.731	0.871
16	0.787	0.827	0.737	0.837
18	0.819	0.800	0.843	0.795
21	0.774	0.807	0.734	0.813
4	0.806	0.785	0.831	0.782
5	0.818	0.796	0.843	0.792
22	0.774157	0.804	0.738	0.810
6	0.794	0.772	0.820	0.769
19	0.807	0.780	0.838	0.775
7	0.795	0.773	0.820	0.769
8	0.823	0.801	0.847	0.798
9	0.801	0.806	0.794	0.807
35	0.807	0.797	0.818	0.796
37	0.808	0.773	0.850	0.767
41	0.807	0.775	0.845	0.768
42[(1)]	0.690	0.561	0.836	0.543
43	0.768	0.723	0.818	0.717

Model uid	Dice's coefficient	True positive rate	False discovery rate	True Positive Mean Distance
12	0.742	0.782	0.270	5.047
14	0.709	0.778	0.329	4.795
15	0.768	0.838	0.281	4.538
10	0.695	0.617	0.184	5.328
11	0.703	0.716	0.249	4.441
16	0.776	0.854	0.283	4.148
18	0.774	0.853	0.274	4.142
21	0.723	0.850	0.357	4.452
4	0.765	0.879	0.318	4.245
5	0.772	0.903	0.316	4.313
22	0.753858	0.883	0.337	4.770
6	0.789	0.84572	0.252	4.423
19	0.786	0.850	0.259	4.566
7	0.773	0.856	0.284	4.278
8	0.776	0.880	0.299	4.317
9	0.712	0.646	0.188	4.818
35	0.759	0.875	0.326	4.725
37	0.765	0.766	0.223	4.764
41	0.732	0.906	0.380	4.569
42	0.621	0.508	0.189	5.430
43	0.768	0.723	0.818	0.717

Table 4.2: **(left)** Classification performance on patches from test videos.
(right) Localisation performance on test patches.

4.4 Best pipeline configuration

Following careful experimentation, we have arrived at a pipeline including vessel detector, cell classifier and positive/negative bounding boxes that has shown the most promising Dice's coefficient and shows the greatest ability to generalise across data that it has never seen, the parameters for this classification network are outlined here.

For the vessel masks the following parameters were used: The frames were first blurred with a Gaussian blur with sigma 1. Then we used the normalisation from [15] described in the methodology chapter and then applied motion-contrast enhancement on the video with the histogram equalisation as described. The video stack was then flattened to create a single image representing the standard deviation in intensity at each location within the image.

The standard deviation image then has it's contrast enhanced with adaptive histogram equalisation after being Gaussian blurred with a sigma 1. The frangi filter is applied on the with an alpha and beta of .5 (for an explanation of the parameters please see skimage.filters.frangi). The frangi image is then contrast enhanced with an adaptive equalisation histogram. The frangi image is then thresholded with Otsu's method and a threshold sensitivity of .5 meaning we take half of the threshold value otsu's method returns. The morphological opening and closing happens with a square structuring element of size 5. After the opening two morphological binary dilation are applied. In the end small objects are removed with skimage.morphology.remove_small_objects with an area smaller than 700.

For training the network the following parameters were decided: The network should be trained with a patch size of 23 pixels. The circle negative patch extraction process is followed with the negative position limited inside the vessel mask. The dataset should be standardised to a mean and variance of .5. A small random vertical and horizontal translation of 5 pixels during sampling was found to give the best result. The batch size used was 512 samples. The learning rate starts from 0.001 and lowers by 10^{-1} every 10 epochs the validation accuracy doesn't improve. After 20 epochs of no validation accuracy improvement the training is stopped. The model

structure used is (as printed from the pytorch module) :

```
CNN(
  (convolutional): Sequential(
    (0): Conv2d(1, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): MaxPool2d(kernel_size=(3, 3), stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(32, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): AvgPool2d(kernel_size=3, stride=2, padding=1)
    (7): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (8): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (9): ReLU()
    (10): AvgPool2d(kernel_size=3, stride=2, padding=1)
  )
  (dense): Sequential(
    (0): Linear(in_features=576, out_features=64, bias=True)
    (1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=64, out_features=32, bias=True)
    (5): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU()
    (7): Dropout(p=0.5, inplace=False)
    (8): Linear(in_features=32, out_features=2, bias=True)
  )
```

For the binarisation of the probability map and the cell location extraction process, the following parameters should be used: $\sigma = 1.57$, $H = 0.34$, $T = .59$. These values were produced as the mean of the σs Hs and Ts that maximised each test video.

Chapter 5

General Conclusions

In this thesis we present a pipeline that takes stabilised retinal images, extracts vessel locations, identifies the individual cells within the vessels and then is able to track them with time. This pipeline has been implemented as a fully automated tool where one has never been produced before. This automated pipeline also replaces a very time intensive process which prevented this power imaging test from becoming clinically viable. With the improved pipeline the imagining technique can now be expanded. We believe that our application could be have a great impact in the health of people and we hope it can serve as an useful early biomarker for doctors to administer treatments for people with multiple diseases at an early stage, where intervention is still possible, to slow down their progression.

5.0.1 Future work

This project faced a classic deep learning challenge, there is plenty of existing data but few labels. Further the labelling process is laborious. So while a significant number of labels were generated for this project, the network could still not generalise as much as we wanted. This suggests that there were characteristic differences between the training and validation sets which need to be captured for the network to perform more robustly. In the future, we hope that collecting samples from a greater variability of subjects could improve the performance of the network.

In our labels for this project most cells have some characteristics that the networks can identify. The classifier can some times incorrectly classify cell patches as negatives leading to a false negative sample. We believe that the false negative ex-

amples share common characteristics that confuse the classification process. A few steps that we hope to investigate, would be to collect and analyse the false negative examples in order to include more samples that share these feature in the training process. We hope that the GUI tool that was build for picking up missed cells from the estimated locations can make this process easy.

In the evaluation chapter it was demonstrated that the negative patch extraction process can have a great impact in the localisation process. By picking points in between cells there is a cleaner separation in the estimations while by picking patches around the cells the probability blobs merge together. We believe that we could combine two negative patch extraction processes for a more accurate network. For example, if one network only picks negative samples perpendicular to the flow then the network could have good segmentation properties. By running a network trained on negative patches parallel to the flow on the segmentation the previous network gives we believe that the number of false positives and the distance of the estimates from the ground truth could be decreased.

Appendix A

Model configurations

Model uid	Channels	Patch Size	Translation Pixels	Preprpocess
0	1	17	2	FALSE
1	1	19	2	FALSE
2	1	21	2	FALSE
3	1	25	2	FALSE
4	1	21	2	FALSE
5	1	23	2	FALSE
6	1	23	5	FALSE
7	1	25	5	FALSE
8	1	21	2	FALSE
9	1	21	5	FALSE
10	2	21	5	FALSE
11	2	21	2	FALSE
12	2	31	2	FALSE
13	2	45	2	FALSE
14	2	45	2	FALSE
15	2	45	2	FALSE
16	3	21	2	FALSE
17	3	29	3	FALSE
18	3	35	2	FALSE
19	1	23	5	FALSE

Model uid	Channels	Patch Size	Translation Pixels	Preprpocess
20	3	29	3	FALSE
21	3	27	2	FALSE
22	1	23	2	TRUE
23	2	41	2	FALSE
24	2	39	2	FALSE
25	3	29	2	FALSE
26	3	29	2	TRUE
27	3	29	2	TRUE
28	3	35	2	FALSE
29	1	23	7	FALSE
30	1	23	5	FALSE
31	1	21	5	FALSE
32	1	23	5	FALSE
33	1	17	5	FALSE
34	1	25	5	FALSE
35	temporal	21	2	FALSE
36	temporal	35	2	FALSE
37	temporal	55	5	FALSE
41	temporal	35	2	FALSE
$42^{(1)}$	1	23	3	FALSE
$43^{(2)}$	1	23	3	FALSE

Table A.1: Model configurations.

uid: Unique model id that links this table to the performance evaluation tables.
channels: The number of channels for the patch extraction process. When 3 then it's the mixed channel case with the confocal, 790nm, and the registered 850nm channel. When 2 then it's just the 790nm and the registered 850nm. When 1 then only the 790nm channel is used. When 'temporal' then temporal patches were extracted with temporal width 1.
Translation pixels: The number of random vertical and horizontal translation in pixels when the patch is sampled.
Preprocess: True when the video data were processed with [15] prior to training and classification.
(1) Model trained with rectangle negative patch extraction with negative search radius of 31.
(2) Model trained with rectangle negative patch extraction with negative search radius of 21.

Appendix B

Model classification evaluation with validation data

Model uid	Balanced Accuracy	Accuracy	Sensitivity	Specificity
0	0.820	0.820	0.819	0.820
1	0.822	0.802	0.846	0.799
2	0.824	0.841	0.804	0.843
3	0.825	0.881	0.761	0.889
4	0.830	0.821	0.840	0.820
5	0.824	0.833	0.813	0.835
6	0.814	0.801	0.828	0.800
7	0.822	0.828	0.815	0.828
8	0.821	0.790	0.858	0.785
9	0.772	0.788	0.752	0.791
10	0.772	0.731	0.819	0.725
11	0.811	0.818	0.802	0.819
12	0.823	0.822	0.825	0.821
13	0.812	0.815	0.809	0.815
14	0.808	0.813	0.803	0.814
15	0.807	0.829	0.783	0.832

Model uid	Balanced Accuracy	Accuracy	Sensitivity	Specificity
16	0.806	0.784	0.830	0.781
17	0.836	0.826	0.847	0.825
18	0.825	0.833	0.816	0.834
19	0.809	0.797	0.821	0.796
20	0.816	0.813	0.820	0.812
21	0.805	0.827	0.779	0.830
22	0.780	0.788	0.771	0.790
23	0.818	0.807	0.831	0.806
24	0.811	0.806	0.817	0.805
25	0.811	0.816	0.805	0.817
26	0.783	0.750	0.820	0.746
27	0.814	0.788	0.844	0.784
28	0.815	0.778	0.857	0.773
29	0.784	0.764	0.807	0.761
30	0.781	0.737	0.832	0.730
31	0.763	0.696	0.839	0.686
32	0.803	0.830	0.773	0.834
33	0.769	0.789	0.747	0.792
34	0.788	0.764	0.816	0.761
35	0.813	0.779	0.853	0.774
36	0.815	0.802	0.830	0.801
37	0.773	0.733	0.818	0.727
41	0.815	0.802	0.830	0.801
42	0.832	0.813	0.862	0.803
43	0.832	0.832	0.833	0.832

Table B.1: Classification performance with the patches from the validation dataset.

Appendix C

Model classification evaluation with test data

Model uid	Balanced Accuracy	Accuracy	Sensitivity	Specificity
0	0.796	0.820	0.767	0.826
1	0.790	0.791	0.788	0.793
2	0.802	0.844	0.751	0.853
3	0.801	0.875	0.712	0.889
4	0.806	0.785	0.831	0.782
5	0.818	0.796	0.843	0.792
6	0.794	0.772	0.820	0.769
7	0.795	0.773	0.820	0.769
8	0.823	0.801	0.847	0.798
9	0.801	0.806	0.794	0.807
10	0.780	0.804	0.752	0.809
11	0.801	0.860	0.731	0.871
12	0.803	0.825	0.777	0.828
13	0.753	0.734	0.776	0.729
14	0.759	0.823	0.684	0.834
15	0.798	0.823	0.769	0.826

Model uid	Balanced Accuracy	Accuracy	Sensitivity	Specificity
16	0.787	0.827	0.737	0.837
17	0.808	0.810	0.806	0.810
18	0.819	0.800	0.843	0.795
19	0.807	0.780	0.838	0.775
20	0.784	0.767	0.803	0.766
21	0.774	0.807	0.734	0.813
22	0.774157	0.804	0.738	0.810
23	0.775	0.715	0.846	0.704
24	0.791	0.787	0.797	0.786
25	0.778	0.812	0.738	0.818
26	0.779	0.747	0.816	0.742
27	0.773	0.717	0.840	0.706
28	0.763	0.681	0.858	0.668
29	0.793	0.787	0.801	0.786
30	0.786	0.741	0.839	0.733
31	0.768	0.688	0.862	0.674
32	0.804	0.828	0.775	0.833
33	0.769	0.794	0.739	0.800
34	0.789	0.757	0.829	0.749
35	0.807	0.797	0.818	0.796
36	0.811	0.790	0.835	0.787
37	0.808	0.773	0.850	0.767
41	0.807	0.775	0.845	0.768
$42^{(*)}$	0.690	0.561	0.836	0.543
$43^{(*)}$	0.768	0.723	0.818	0.717

Table C.1: Classification performance from patches extracted from the test videos.
(*) Although the negative patch extraction was different than the rest videos during training, the negative patch extraction process for testing is the same for all videos. That's the reason model 42 has a particularly bad score since it wasn't trained with negative cells that are close to positives.

Appendix D

Model localisation evaluation with test data

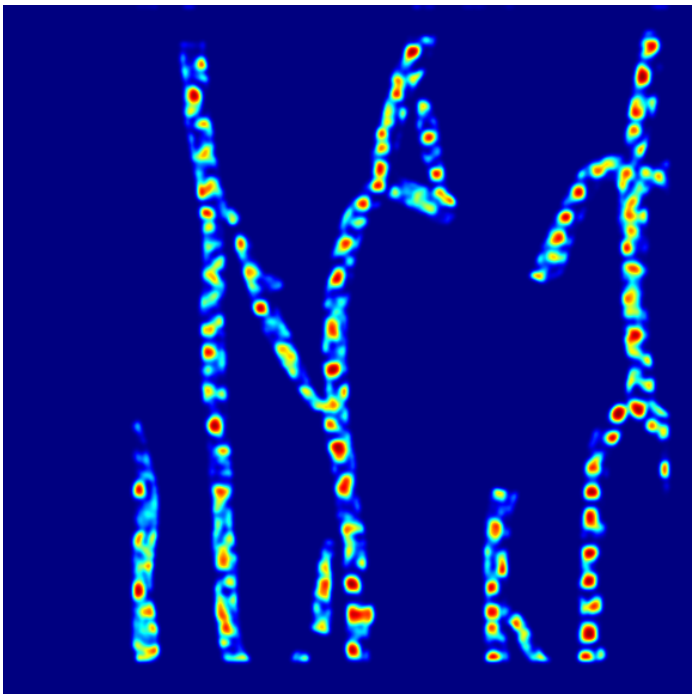
Model uid	Dice's coefficient	True positive rate	False discovery rate	True Positive Mean Distance
0	0.754	0.337	0.882	4.448
1	0.754	0.882	0.337	4.448
2	0.774	0.830	0.271	3.981
3	0.766	0.876	0.301	4.142
4	0.765	0.879	0.318	4.245
5	0.772	0.903	0.316	4.313
6	0.789	0.84572	0.252	4.423
7	0.773	0.856	0.284	4.278
8	0.776	0.880	0.299	4.317
9	0.712	0.646	0.188	4.818
10	0.695	0.617	0.184	5.328
11	0.703	0.716	0.249	4.441
12	0.742	0.782	0.270	5.047
13	0.711	0.858	0.386	5.414
14	0.709	0.778	0.329	4.795
15	0.768	0.838	0.281	4.538

Model uid	Dice's coefficient	True positive rate	False discovery rate	True Positive Mean Distance
16	0.776	0.854	0.283	4.148
17	0.771	0.832	0.263	4.764
18	0.774	0.853	0.274	4.142
19	0.786	0.850	0.259	4.566
20	0.746	0.815	0.297	5.016
21	0.723	0.850	0.357	4.452
22	0.753858	0.883	0.337	4.770
23	0.742	0.847	0.326	4.918
24	0.747	0.818	0.302	4.651
25	0.703	0.755	0.298	4.622
26	0.753	0.821	0.291	5.117
27	0.732	0.829	0.336	5.287
28	0.714	0.787	0.320	5.574
29	0.745	0.709	0.199	4.462
30	0.730	0.694	0.214	4.724
31	0.699	0.606	0.162	5.127
32	0.769	0.764	0.216	4.511
33	0.674	0.589	0.191	5.757
34	0.723	0.659	0.182	5.232
35	0.759	0.875	0.326	4.725
36	0.732	0.906	0.380	4.569
37	0.765	0.766	0.223	4.764
41	0.732	0.906	0.380	4.569
42	0.621	0.508	0.189	5.430
43	0.768	0.723	0.818	0.717

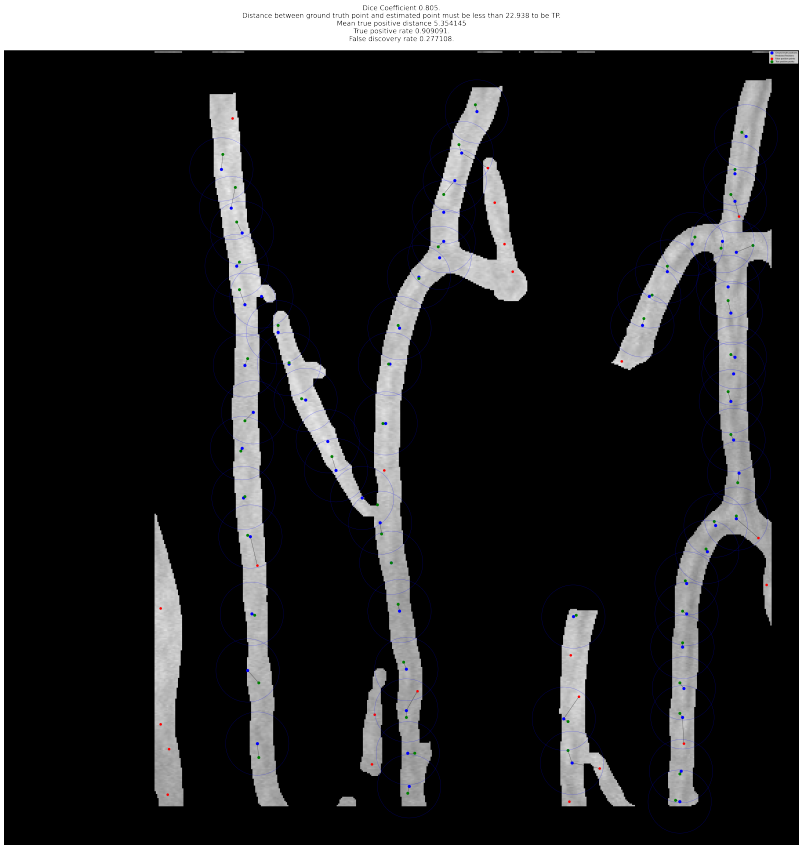
Table D.1: Model localisation performance on the test videos.

Appendix E

Sample results on test videos

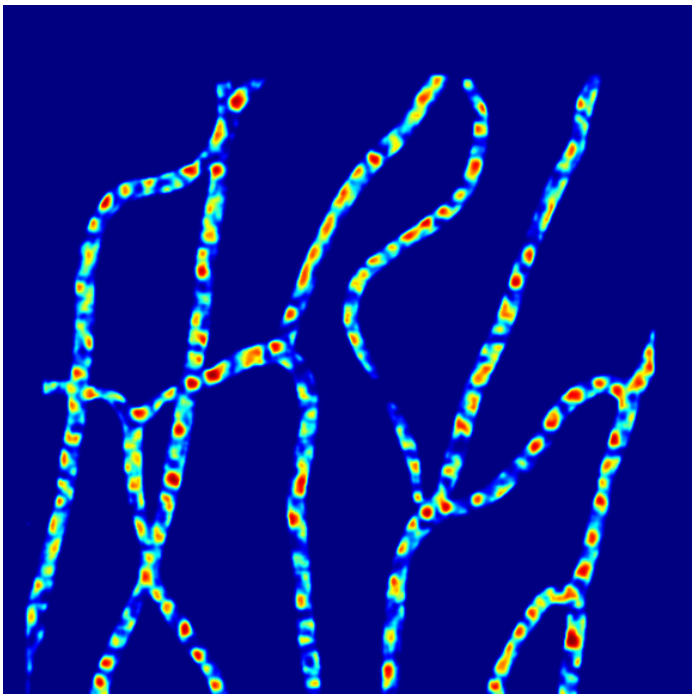


(a)

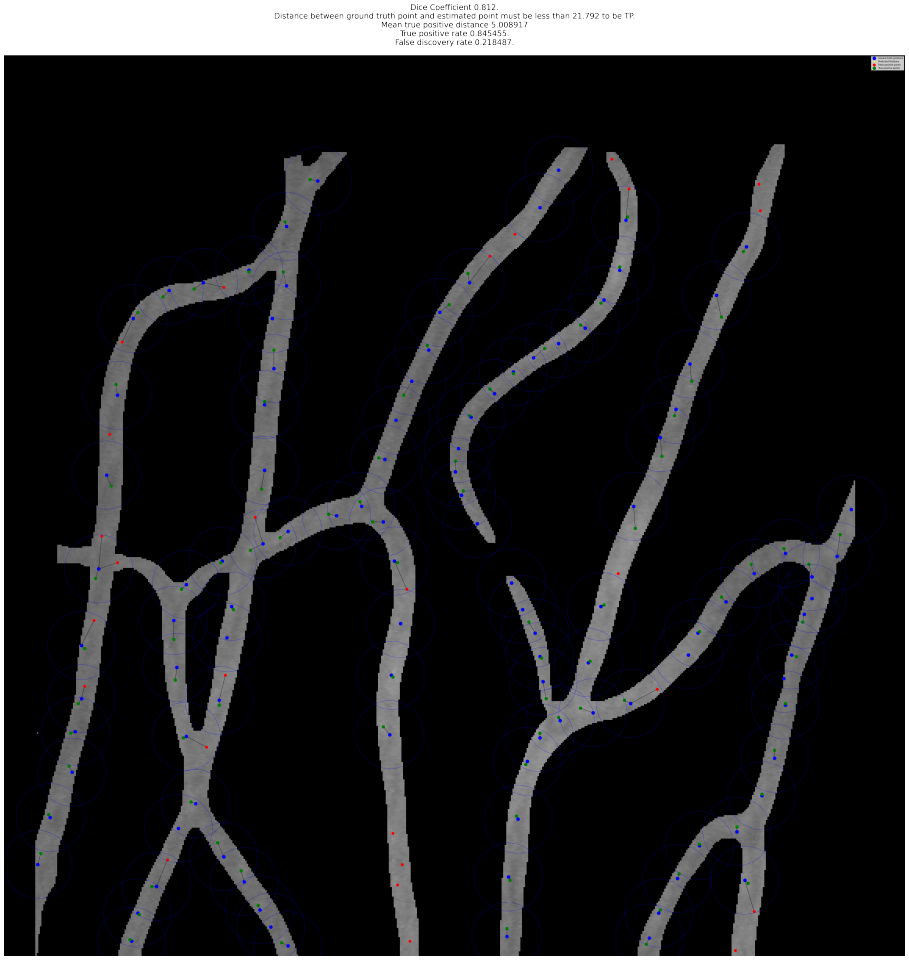


(b)

Figure E.1: Location estimation on the video shown in this media link. Because part of the frames were never marked, even with the existence of cells, some vessels were manually cropped to avoid a bad score from incomplete manual marking.
(a) The probability map. For the the probability map video click this media link. For the probability map with the estimated positions marked click this media link.
(b) Evaluation results for a frame of the video. For a video with the estimated locations click this media link

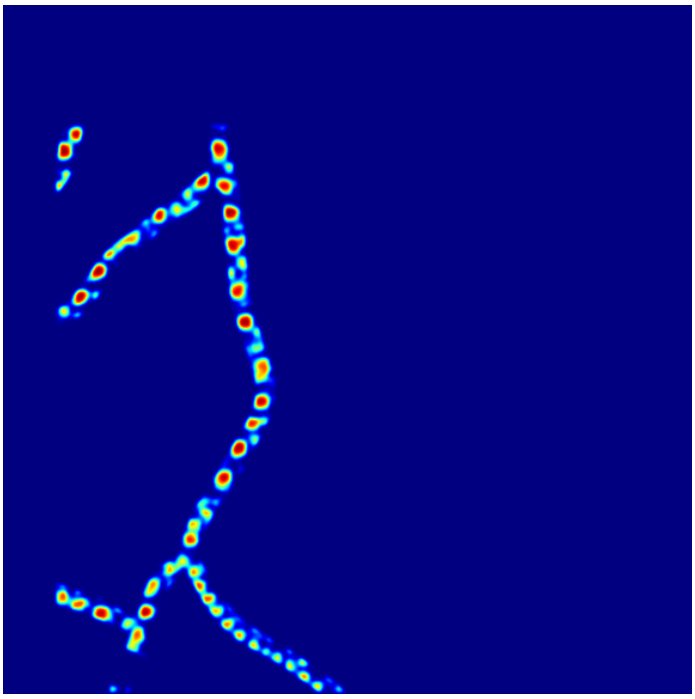


(a)

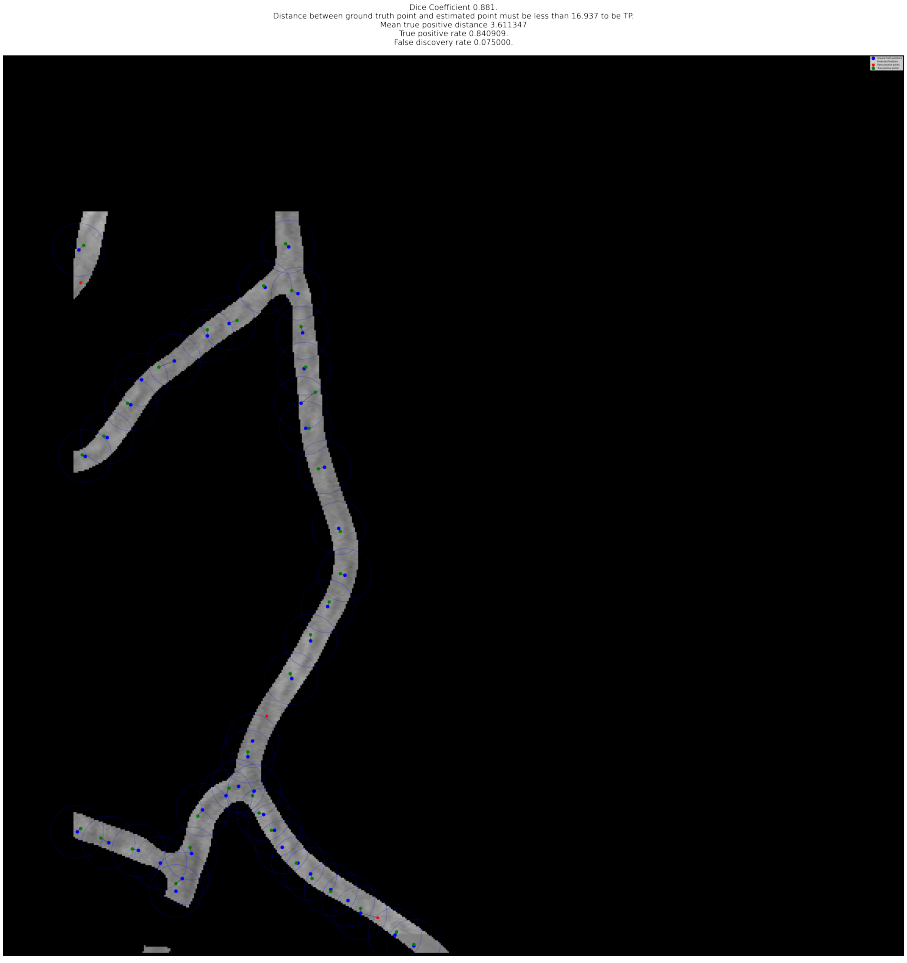


(b)

Figure E.2: Location estimation on the video shown in this media link.
(a) The probability map. For the the probability map video click this media link. For the probability map with the estimated positions marked click this media link.
(b) Evaluation results for a frame of the video. For a video with the estimated locations click this media link



(a)



(b)

Figure E.3: Location estimation on the video shown in this media link.
(a) The probability map. For the the probability map video click this media link. For the probability map with the estimated positions marked click this media link.
(b) Evaluation results for a frame of the video. For a video with the estimated locations click this media link

Appendix F

Colophon

The code for this thesis was developed using python. In addition to the standard python libraries the following modules have been used:

- numpy [63]
  Provides useful n-dimensional array operations. Used throughout the project.
- pytorch
  A python machine learning library based on the Torch library that supports automatic differentiation [52]. Used for creating and training the models.
- torchvision
  A package that complements pytorch. Was used to for batching, transformations, dataset creation.
- Opencv [62]
  A computer vision library with many build in algorithms. Was used for image processing, template matching, and feature matching.
- scickit-image [64]
  Provides image processing algorithms. Used for image processing such as morphological operations and contrast enhancement.
- scickit-learn [65]
  A machine learning library for python. Has a fast and easy to use nearest neighbour implementation that was used in the thesis.

- PIL

  The Python Imaging Library. Useful for easily saving images.

- pandas [66]

  A data analysis and manipulation tool. Used mainly for displaying results during training.

- mahotas [54]

  An image processing library used for a regmax function to implement extended-maxima transform

- tqdm [67]

  A library that provides a progress meter. Was used for multiple occasions where some operations would take a certain time so the user can have a visual feedback.

- shutil

  Offers high level operations on files. Used for removing directories.

- PyQt5 [68]

  Used for creating GUI tools.

Bibliography

[1] M. Mizutani, T. S. Kern, and M. Lorenzi. Accelerated death of retinal microvascular cells in human and experimental diabetic retinopathy. 97(12):2883–2890. Publisher: American Society for Clinical Investigation.

[2] Jack C de la Torre. Is alzheimer's disease a neurodegenerative or a vascular disorder? data, dogma, and dialectics. 3(3):184–190.

[3] Leif Østergaard, Sune Nørhøj Jespersen, Kim Mouridsen, Irene Klærke Mikkelsen, Kristjana Ýr Jonsdottír, Anna Tietze, Jakob Udby Blicher, Rasmus Aamand, Niels Hjort, Nina Kerting Iversen, Changsi Cai, Kristina Dupont Hougaard, Claus Z Simonsen, Paul Von Weitzel-Mudersbach, Boris Modrau, Kartheeban Nagenthiraja, Lars Riisgaard Ribe, Mikkel Bo Hansen, Susanne Lise Bekke, Martin Gervais Dahlman, Josep Puig, Salvador Pedraza, Joaquín Serena, Tae-Hee Cho, Susanne Siemonsen, Götz Thomalla, Jens Fiehler, Norbert Nighoghossian, and Grethe Andersen. The role of the cerebral capillaries in acute ischemic stroke: The extended penumbra model. 33(5):635–648. Publisher: SAGE Publications Ltd STM.

[4] RM. GUNN. Ophthalmoscopic evidence of (1) arterial changes associated with chronic renal disease, and (2) of increased arterial tension. *Transactions of the Ophthalmological Society of the United Kingdom*, 12:124–125, 1892.

[5] Wolf S, Arend O, Schulte K, Ittel T H, and Reim M. Quantification of retinal capillary density and flow velocity in patients with essential hypertension. 23(4):464–467. Publisher: American Heart Association.

[6] Grant A. Bateman, Jeannette Lechner-Scott, and Rodney A. Lea. A comparison between the pathophysiology of multiple sclerosis and normal pressure hydrocephalus: is pulse wave encephalopathy a component of MS? 13.

[7] Niall Patton, Tariq Aslam, Thomas MacGillivray, Alison Pattie, Ian J Deary, and Baljean Dhillon. Retinal vascular image analysis as a potential screening tool for cerebrovascular disease: a rationale based on homology between cerebral and retinal microvasculatures. 206(4):319–348.

[8] Anat London, Inbal Benhar, and Michal Schwartz. The retina as a window to the brain-from eye research to CNS disorders. *Nature Reviews. Neurology*, 9(1):44–53, January 2013.

[9] Johnny Tam, Joy A. Martin, and Austin Roorda. Noninvasive Visualization and Analysis of Parafoveal Capillaries in Humans. *Investigative Ophthalmology & Visual Science*, 51(3):1691–1698, March 2010. Publisher: The Association for Research in Vision and Ophthalmology.

[10] Varying focus through retina AOSLO.

[11] Free image on pixabay - eye, diagram, eyeball, body, pupil. Library Catalog: pixabay.com.

[12] Toco Y. P. Chui, Dean A. VanNasdale, and Stephen A. Burns. The use of forward scatter to improve retinal vascular imaging with an adaptive optics scanning laser ophthalmoscope. *Biomedical Optics Express*, 3(10):2537–2549, October 2012. Publisher: Optical Society of America.

[13] Stephen A. Burns, Ann E. Elsner, Kaitlyn A. Sapoznik, Raymond L. Warner, and Thomas J. Gast. Adaptive optics imaging of the human retina. *Progress in Retinal and Eye Research*, 68:1–30, January 2019.

[14] The indiana adaptive optics slos.

[15] Alberto de Castro, Gang Huang, Lucie Sawides, Ting Luo, and Stephen A. Burns. Rapid high resolution imaging with a dual-channel scanning technique.

Optics Letters, 41(8):1881–1884, April 2016. Publisher: Optical Society of America.

[16] Shruti A. Japee, Roland N. Pittman, and Christopher G. Ellis. Automated Method for Tracking Individual Red Blood Cells Within Capillaries to Compute Velocity and Oxygen Saturation. *Microcirculation*, 12(6):507–515, 2005. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1080/10739680591003341.

[17] David Cunefare, Leyuan Fang, Robert F. Cooper, Alfredo Dubra, Joseph Carroll, and Sina Farsiu. Open source software for automatic detection of cone photoreceptors in adaptive optics ophthalmoscopy using convolutional neural networks. *Scientific Reports*, 7(1):1–11, July 2017.

[18] Shigeta Arichika, Akihito Uji, Masanori Hangai, Sotaro Ooto, and Nagahisa Yoshimura. Noninvasive and Direct Monitoring of Erythrocyte Aggregates in Human Retinal Microvasculature Using Adaptive Optics Scanning Laser Ophthalmoscopy. *Investigative Ophthalmology & Visual Science*, 54(6):4394–4402, June 2013. Publisher: The Association for Research in Vision and Ophthalmology.

[19] Johnny Tam, Pavan Tiruveedhula, and Austin Roorda. Characterization of single-file flow through human retinal parafoveal capillaries using an adaptive optics scanning laser ophthalmoscope. *Biomedical Optics Express*, 2(4):781–793, March 2011.

[20] Phillip Bedggood and Andrew Metha. Direct visualization and characterization of erythrocyte flow in human retinal capillaries. *Biomedical Optics Express*, 3(12):3264–3277, November 2012.

[21] C. E. Riva and B. Petrig. Blue field entoptic phenomenon and blood velocity in the retinal capillaries. *JOSA*, 70(10):1234–1238, October 1980. Publisher: Optical Society of America.

[22] J. E. Grunwald, J. Piltz, N. Patel, S. Bose, and C. E. Riva. Effect of aging on retinal macular microcirculation: a blue field simulation study. *Investigative Ophthalmology & Visual Science*, 34(13):3609–3613, December 1993. Publisher: The Association for Research in Vision and Ophthalmology.

[23] Joy A. Martin and Austin Roorda. Pulsatility of parafoveal capillary leukocytes. *Experimental Eye Research*, 88(3):356–360, March 2009.

[24] Joy A. Martin and Austin Roorda. Direct and Noninvasive Assessment of Parafoveal Capillary Leukocyte Velocity. *Ophthalmology*, 112(12):2219–2224, December 2005.

[25] Johnny Tam and Austin Roorda. Speed quantification and tracking of moving objects in adaptive optics scanning laser ophthalmoscopy. *Journal of Biomedical Optics*, 16(3):036002, March 2011. Publisher: International Society for Optics and Photonics.

[26] Yunsik Yang, Sangduck Kim, and Jaeduck Kim. Fluorescent Dots in Fluorescein Angiography and Fluorescein Leukocyte Angiography Using a Scanning Laser Ophthalmoscope in Humans. *Ophthalmology*, 104(10):1670–1676, October 1997.

[27] M. Paques, B. Boval, S. Richard, R. Tadayoni, P. Massin, O. Mundler, A. Gaudric, and E. Vicaut. Evaluation of fluorescein-labeled autologous leukocytes for examination of retinal circulation in humans. *Current Eye Research*, 21(1):560–565, January 2000. Publisher: Taylor & Francis _eprint: https://www.tandfonline.com/doi/pdf/10.1076/0271-3683%28200007%292111-ZFT560.

[28] Zhangyi Zhong, Benno L. Petrig, Xiaofeng Qi, and Stephen A. Burns. In vivo measurement of erythrocyte velocity and retinal blood flow using adaptive optics scanning laser ophthalmoscopy. *Optics Express*, 16(17):12746–12756, August 2008. Publisher: Optical Society of America.

[29] Charles Riva, Benjamin Ross, and George B. Benedek. Laser Doppler Measurements of Blood Flow in Capillary Tubes and Retinal Arteries. *Investigative Ophthalmology & Visual Science*, 11(11):936–944, November 1972. Publisher: The Association for Research in Vision and Ophthalmology.

[30] Alfredo Dubra and Zachary Harvey. Registration of 2D Images from Fast Scanning Ophthalmic Instruments. In Bernd Fischer, Benoît M. Dawant, and Cristian Lorenz, editors, *Biomedical Image Registration*, Lecture Notes in Computer Science, pages 60–71, Berlin, Heidelberg, 2010. Springer.

[31] Christos Bergeles, Adam M. Dubis, Benjamin Davidson, Melissa Kasilian, Angelos Kalitzeos, Joseph Carroll, Alfredo Dubra, Michel Michaelides, and Sebastien Ourselin. Unsupervised identification of cone photoreceptors in non-confocal adaptive optics scanning light ophthalmoscope images. *Biomedical Optics Express*, 8(6):3081–3094, May 2017.

[32] David Cunefare, Robert F. Cooper, Brian Higgins, David F. Katz, Alfredo Dubra, Joseph Carroll, and Sina Farsiu. Automatic detection of cone photoreceptors in split detector adaptive optics scanning light ophthalmoscope images. *Biomedical Optics Express*, 7(5):2036–2050, May 2016. Publisher: Optical Society of America.

[33] Robert F. Cooper, Christopher S. Langlo, Alfredo Dubra, and Joseph Carroll. Automatic detection of modal spacing (Yellott's ring) in adaptive optics scanning light ophthalmoscope images. *Ophthalmic & Physiological Optics: The Journal of the British College of Ophthalmic Opticians (Optometrists)*, 33(4):540–549, July 2013.

[34] Luis Perez and Jason Wang. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. *arXiv:1712.04621 [cs]*, December 2017. arXiv: 1712.04621.

[35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv:1505.04597 [cs]*, May 2015. arXiv: 1505.04597.

[36] Varun Gulshan, Lily Peng, Marc Coram, Martin C. Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, Ramasamy Kim, Rajiv Raman, Philip C. Nelson, Jessica L. Mega, and Dale R. Webster. Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. *JAMA*, 316(22):2402–2410, December 2016. Publisher: American Medical Association.

[37] Qiaoliang Li, Bowei Feng, LinPei Xie, Ping Liang, Huisheng Zhang, and Tianfu Wang. A Cross-Modality Learning Approach for Vessel Segmentation in Retinal Images. *IEEE Transactions on Medical Imaging*, 35(1):109–118, January 2016. Conference Name: IEEE Transactions on Medical Imaging.

[38] Huazhu Fu, Yanwu Xu, Damon Wing Kee Wong, and Jiang Liu. Retinal vessel segmentation via deep learning network and fully-connected conditional random fields. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pages 698–701, April 2016. ISSN: 1945-8452.

[39] Leyuan Fang, David Cunefare, Chong Wang, Robyn H. Guymer, Shutao Li, and Sina Farsiu. Automatic segmentation of nine retinal layer boundaries in OCT images of non-exudative AMD patients using deep learning and graph search. 8(5):2732–2744. Publisher: Optical Society of America.

[40] David Cunefare, Christopher S. Langlo, Emily J. Patterson, Sarah Blau, Alfredo Dubra, Joseph Carroll, and Sina Farsiu. Deep learning based detection of cone photoreceptors with multimodal adaptive optics scanning light ophthalmoscope images of achromatopsia. *Biomedical Optics Express*, 9(8):3740–3756, August 2018.

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[42] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pages 2146–2153, September 2009. ISSN: 2380-7504.

[43] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*, March 2015. arXiv: 1502.03167.

[44] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[45] Paweł Liskowski and Krzysztof Krawiec. Segmenting Retinal Blood Vessels With Deep Neural Networks. *IEEE Transactions on Medical Imaging*, 35(11):2369–2380, November 2016. Conference Name: IEEE Transactions on Medical Imaging.

[46] M. M. Fraz, P. Remagnino, A. Hoppe, B. Uyyanonvara, A. R. Rudnicka, C. G. Owen, and S. A. Barman. Blood vessel segmentation methodologies in retinal images – A survey. *Computer Methods and Programs in Biomedicine*, 108(1):407–433, October 2012.

[47] J. Staal, M.D. Abramoff, M. Niemeijer, M.A. Viergever, and B. van Ginneken. Ridge-based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging*, 23(4):501–509, April 2004. Conference Name: IEEE Transactions on Medical Imaging.

[48] Huazhu Fu, Yanwu Xu, Stephen Lin, Damon Wing Kee Wong, and Jiang Liu. DeepVessel: Retinal Vessel Segmentation via Deep Learning and Conditional

Random Field. In Sebastien Ourselin, Leo Joskowicz, Mert R. Sabuncu, Gozde Unal, and William Wells, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, Lecture Notes in Computer Science, pages 132–139, Cham, 2016. Springer International Publishing.

[49] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

[50] Ning Wang, Nancy N Zeng, and Wen Zhu. Sensitivity, Specificity, Accuracy, Associated Confidence Interval And ROC Analysis With Practical SAS Implementations. page 9, 2010.

[51] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M. Buhmann. The Balanced Accuracy and Its Posterior Distribution. In *2010 20th International Conference on Pattern Recognition*, pages 3121–3124, August 2010. ISSN: 1051-4651.

[52] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. October 2017.

[53] Yibo Qin, Wei Wang, Wei Liu, and Ning Yuan. Extended-Maxima Transform Watershed Segmentation Algorithm for Touching Corn Kernels. *Advances in Mechanical Engineering*, 5:268046, January 2013. Publisher: SAGE Publications.

[54] Luis Pedro Coelho. Mahotas: Open source software for scriptable computer vision. *Journal of Open Research Software*, 1(1):e3, July 2013. Number: 1 Publisher: Ubiquity Press.

[55] Priyanka Garg. A Comparative Study on Histogram Equalization and Cumulative Histogram Equalization. 3(9):3, 2017.

[56] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in Python. *PeerJ*, 2:e453, June 2014.

[57] Alejandro F. Frangi, Wiro J. Niessen, Koen L. Vincken, and Max A. Viergever. Multiscale vessel enhancement filtering. In William M. Wells, Alan Colchester, and Scott Delp, editors, *Medical Image Computing and Computer-Assisted Intervention — MICCAI'98*, Lecture Notes in Computer Science, pages 130–137, Berlin, Heidelberg, 1998. Springer.

[58] Nobuyuki Otsu. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, January 1979. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics.

[59] Kelly H. Zou, Simon K. Warfield, Aditya Bharatha, Clare M. C. Tempany, Michael R. Kaus, Steven J. Haker, William M. Wells, Ferenc A. Jolesz, and Ron Kikinis. Statistical validation of image segmentation quality based on a spatial overlap index1: scientific reports. *Academic Radiology*, 11(2):178–189, February 2004.

[60] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[61] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008.

[62] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[63] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22, 2011.

[64] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014.

[65] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[66] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[67] Casper O. da Costa-Luis. ‘tqdm‘: A fast, extensible progress meter for python and cli. *Journal of Open Source Software*, 4(37):1277, 2019.

[68] Riverbank Computing Limited. PyQt5: Python bindings for the Qt cross platform application toolkit.