DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

IMAGE PROCESSING METHODS FOR RETINAL VESSEL SEGMENTATION

by
Ozan YÜKSEL

July, 2023
İZMİR

IMAGE PROCESSING METHODS FOR RETINAL VESSEL SEGMENTATION

A thesis submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University in
Partial Fulfilment of Requirements for the Degree of Master of Science in
Computer Science, Computer Science Program

by

Ozan YÜKSEL

July, 2023

İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled "**IMAGE PROCESSING METHODS FOR RETINAL VESSEL SEGMENTATION"** completed by **OZAN YÜKSEL** under supervision of **PROF. DR. ÇAĞIN KANDEMİR ÇAVAŞ** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

.....

Prof. Dr. Çağın Kandemir Çavaş

Supervisor

.....

Prof. Dr. Emel Kuruoğlu Kandemir

Jury Member

.....

Prof. Dr. Urfat Nuriyev

Jury Member

Prof. Dr. Okan FISTIKOĞLU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENT

I’d like to thank Dokuz Eylül University, Graduate School of Natural and Applied Sciences for giving the opportunity to students from different backgrounds to pursue a Master of Science degree in Computer Science. I’d like to thank “PROF.DR. ÇAĞIN KANDEMİR ÇAVAŞ” for her continuous support and mentorship during all stages of the thesis process. Our collaboration even persisted throughout the global COVID-19 pandemic. In respect to this, I’d like to praise all the healthcare professionals and researchers, especially “Prof. Dr. Uğur Şahin” and “Dr. Özlem Türeci” for their extraordinary contributions to the fight against the global pandemic. Finally, I’d like to thank Utrecht University Image Sciences Institute for the fundus image dataset used in this study.

Ozan YÜKSEL

IMAGE PROCESSING METHODS FOR RETINAL VESSEL SEGMENTATION

ABSTRACT

In general, reflecting the latest developments in computer science to other scientific fields is mutually beneficial. Biomedical image processing is a good example of where worlds of medicine and computer science collaborate to solve common problems of humanity such as detecting diseases and authentication by using biometric information. In this study, analysis of several image processing techniques including channel extraction, smoothing, thresholding, histogram equalization and morphological operations are being presented. Those methods are combined to mark retinal vessel regions on the retinal images and differentiate non-vessel regions. Results of this study can later be used as a preliminary work for biometric identification or an automated eye disease detecting system.

Keywords: Retinal image processing, vessel segmentation, biometric identification, fundus image, CLAHE, DRIVE

RETİNA DAMARLARININ AYRIŞTIRILMASI İÇİN GÖRÜNTÜ İŞLEME YÖNTEMLERİ

ÖZ

Bilgisayar bilimleri alanındaki son gelişmelerin diğer bilimsel araştırma alanlarına yansıtılması genellikle iki alan için de fayda sağlamaktadır. Biyomedikal görüntü işleme, bilgisayar bilimleri ve tıp alanlarının, hastalık tespiti, güvenlik amaçlı biyometrik şifreleme gibi insanlığın ortak sorunlarına çözüm bulmak için iş birliği yaptığı, buna iyi bir örnektir. Bu çalışma kapsamında, kanal ayrıştırma, düzleme, limitleme, histogram eşitleme ve şekilsel operasyonlar gibi görüntü işleme yöntemleri analiz edilmektedir. Bu yöntemler birlikte kullanılarak retina damarlarının, retinadaki damar olmayan bölgelerden ayrıştırılması amaçlanmaktadır. Bu çalışmanın çıktıları biyometrik kimlik doğrulama veya göz hastalıklarının otomatik tespiti alanlarında yapılacak çalışmalar için bir ön çalışma olarak kullanılabilir.

Keywords: Retinal görüntü işleme, damar ayrıştırma, biyometrik doğrulama, göz dibi görüntüsü, CLAHE, DRIVE

CONTENTS

Page

THESIS EXAMINATION RESULT FORM ii
ACKNOWLEDGEMENT iii
ABSTRACT iv
ÖZ v
CONTENTS vi
LIST OF FIGURES ix
LIST OF TABLES x
ABBREVIATIONS xi

CHAPTER ONE – INTRODUCTION 1

1.1 Technology and Medicine 1
1.2 Scope of the Study 2
1.2.1 Anatomy of the Human Eye 3
1.2.2 Medical Imaging of the Human Eye 4

CHAPTER TWO – PREVIOUS WORK 6

2.1 Biomedical Image Processing Applications in Various Fields of Medicine 6
2.2.1 Biomedical Image Processing Research in Ophthalmology 8
2.2.2 Retinal Vessel Segmentation Research 9

CHAPTER THREE – MATERIAL AND METHODS 14

3.1 Material 14
3.1.1 Used Fundus Image Dataset 14
3.1.2 Image representation in Digital World 15
3.2 Methods 16
3.2.1 Image Processing Fundamentals 16
3.2.1.1 Convolution 16
3.2.1.2 Hadamard Product 17

3.2.1.3 Discrete 2-Dimensional Convolution 18
3.2.1.4 Padding 20
3.2.1.5 Stride 20
3.2.1.6 Image Histogram 22
3.2.2 Methods for Retinal Vessel Segmentation 23
3.2.2.1 Colour Channel Extraction 23
3.2.2.2 Smoothing 24
3.2.2.3 Histogram Equalization 26
3.2.2.3.1 Uniform Histogram Equalization 27
3.2.2.3.2 Contrast Limited Adaptive Histogram Equalization 28
3.2.2.4 Thresholding 29
3.2.2.4.1 Global Binary Thresholding 29
3.2.2.4.2 Adaptive Mean Thresholding 30
3.2.2.5 Masking and Morphological Operations 31
3.2.2.5.1 Structuring Element 32
3.2.2.5.2 Morphological Dilation and Erosion 32
3.2.2.5.4 Masking 33
3.2.2.5.5 Morphological Opening and Closing 35

CHAPTER FOUR - APPLICATION 37

4.1 Selection of the Programming Language 37
4.2 Selection of the Computer Vision Framework 38
4.3 Design Decisions and Architecture 38

CHAPTER FIVE – RESULTS 39

5.1 Execution Time 39
5.2 Performance Metrics 39

CHAPTER SIX – CONCLUSION 42

REFERENCES 43

APPENDICES 51

LIST OF FIGURES

Page

Figure 1.1 Horizontal cross section of the human eye 4

Figure 1.2 A photograph of a retinal camera 5

Figure 1.3 Example fundus image from Digital Retinal Images for Vessel Extraction (DRIVE) dataset 5

Figure 3.1 Graphical representation of the mentioned signals 17

Figure 3.2 Global Image Histogram 22

Figure 3.3 Red (top left), green (top right) and blue (bottom) channels of image 24

Figure 3.4 Fundus image after green channel extraction and smoothing 26

Figure 3.5 Global image histogram on single channel 26

Figure 3.6 Fundus image and image histogram after green channel extraction, smoothing and uniform histogram equalization 27

Figure 3.7 Fundus image and histogram after green channel extraction, smoothing and CLAHE 28

Figure 3.8 Global binary thresholding with different threshold values 30

Figure 3.9 Fundus image after green channel extraction, smoothing, CLAHE and inverse adaptive thresholding 31

Figure 3.10 Mask image and fundus image after masking step 34

Figure 3.11 Eroded mask image (left) and fundus image after masking step (right) 34

Figure 3.12 Morphologically eroded fundus image after masking 35

Figure 3.13 Morphologically opened fundus image after masking (left), fundus image after result is merged to the original image (right) 36

LIST OF TABLES

Page

Table 5.1 Performance Metrics for the Proposed Algorithm..... 42

ABBREVIATIONS

MRI	: Magnetic Resonance Imaging
ECG	: Electrocardiogram
SARS-CoV-2	: Severe Acute Respiratory Syndrome Coronavirus-2
COVID-19	: Coronavirus Disease 2019
ROI	: Regions of Interest
DRIVE	: Digital Retinal Images for Vessel Extraction
CT	: Computed Tomography
HMRF-EM	: Hidden Markov Random-Field Expectation Maximization
FCNs	: Fully Convolutional Networks
WP-CNN	: Weighted Path Convolutional Neural Network
STARE	: Structured Analysis of the Retina
K-NN	: K-Nearest Neighbour
SVM	: Support Vector Machines
MFR	: Matched Filter Response
PCA	: Principal component analysis
CLAHE	: Contrast Limited Adaptive Histogram Equalization
ISODATA	: Iterative Self-Organizing Data Analysis Technique
RGB	: Red Green Blue
CMYK	: Cyan Magenta Yellow Key
CNN	: Convolutional Neural Networks
CPU	: Central Processing Unit
TP	: True Positive
TN	: True Negative
FP	: False Positive
FN	: False Negative

CHAPTER ONE
INTRODUCTION

In this chapter the entanglement between medicine and technology will be discussed chronologically. Also, it will be mentioned how recent challenges faced by humanity were overcome by the collaboration of medicine and technology along with a short introduction to the scope of the thesis such as problem definition and use cases.

1.1 Technology and Medicine

In this day and age, it is hard to deny that computers and smart devices have evolved into companions which help us accomplish a variety of tasks. They help us with driving better, landing spacecrafts, receiving on-the-job training, engineer better systems and so much more. One of the fields that utilizes these devices to its advantage is medicine. Taking a glimpse at the wide range of technological developments in medical diagnostics in the 19th century shows how much can change within a century in the field of medicine. These developments range from the invention of stethoscope to utilization of x-rays in radiographic laboratories at the end of the century, it is also the century where first fundus photograph was taken using early assemblies of fundus cameras (Bernstein, Bilheimer, & Makuc, 2010). Moreover, coming to the last century, inventions like x-ray computed tomography allowed physicians to investigate targeted areas of the human body, whereas another invention, magnetic resonance imaging (MRI), allowed doctors to take images of the organs without using any ionizing radiation (Schwan, 2023). On top of these, electrocardiogram (ECG) machines got commercialized which enabled observing electrical activity of the heart (Schwan, 2023). All the mentioned technological developments are still widely used in hospitals as part of many diagnostic processes.

Furthermore, it is very uncommon for an invention to stagnate and not get effected by any the recent developments of the century. All the mentioned inventions are continuously being improved to this day, compared to their initial states in terms of

efficiency, accuracy, and ease of use. The century that we live in, has been a great one in terms of developments in artificial intelligence, computer vision and image processing domains. It is not surprising to see reflections of those developments and improvements on fields such as agriculture, finance, education and finally, medicine.

Advancement in technology for medicine have greatly impacted the diagnosis and treatment processes for certain diseases. The recent introduction of SARS-CoV-2 (Severe Acute Respiratory Syndrome Coronavirus-2) to the world, also commonly known as COVID-19 (Coronavirus Disease 2019), and the following pandemic, shifted the focus to how technology can help medical practitioners diagnose, monitor, protect against and cure diseases with the help of computer science as part of the global effort to reduce the mortality rate of the disease. Complex problems posed by the disease steered doctors and medical specialists in the direction to look for new technologies and methods to employ, some institutions rapidly adapted AI based technologies to tackle these challenges. (Syeda et al., 2021). Another example to the exciting research being conducted is the rapid development of drugs for diseases. With huge biomedical datasets that are available to the researchers in the recent years, AI tools can be developed that help extract meaningful data out of the experiment-based data which would result in faster drug development (Réda, Kaufmann, & Delahaye-Duriez, 2020). Considering that developing a drug and getting it approved by authorities can take up to 15 years (English, Lebovitz, & Giffin, 2010), accelerating this process is sure beneficial for patients that suffer from the disease who would get access to the drug earlier and maybe even cheaper since companies that develop the drug would spend less resources.

1.1 Scope of the Study

There are lots of ongoing research on designing systems which assist medical experts in the diagnosis processes. The assistance could be in form of marking regions of interest (ROI) which can be extremely useful considering the ROI could be a diminutive portion of the original data (Cicerone & Camp, 2019). These systems can in the future evolve into systems that directly inform patients about a potential disease.

In fact, it is estimated that 20% of the diabetes patients in the United States do not even know they have diabetes (Centers for Disease Control and Prevention, 2022). Additionally, one of the risks that diabetes patients have is the risk of developing diabetic retinopathy: an eye disease which puts diabetes patients at risk of visual deterioration and even blindness (National Eye Institute, 2022). This data highlights the importance of regular eye examinations and even automated systems to detect diseases as fast and as early as possible.

The fundus image also can give insights about a person. For example, the distribution of retinal vessels is different for each person, thus it can be used for biometric identification (Mariño, Penedo, Penas, Carreira, & Gonzalez, 2006). Moreover, according to Nelson (2019), director of engineering at Google, their deep learning-based model (designed to detect diabetic retinopathy disease) was also able to accurately determine the gender and approximate age of the owner of the retinal image. However, they were not able to explain how the correlation was formed for some of the factors such as gender, which is very important in machine learning studies. Finally, he proceeds to explain that fundus images can contain information about cardiovascular risk factors and even neurodegenerative diseases. Similarly, this study focuses on segmenting retinal blood vessels from fundus images which can be beneficial in the diagnosis processes of such diseases using automated systems. The common view across the medical community is in order to achieve development of future computer-aided diagnostic systems for eye diseases, autonomy in retinal vessel analysis is key (Fraz et al., 2012). However, manual retinal vessel segmentation is required for training supervised machine learning models and measuring the performance metrics of any algorithm developed for retinal vessel segmentation. Moreover, the manual segmentation process is a laborious task that requires skill and training (Fraz et al., 2012).

1.1.1 Anatomy of the Human Eye

There is no doubt that the human eye is one of the wonders of the nature. To understand the scope of this study, an introduction to the human eye anatomy is

required. Anatomical components of the human eyeball can be grouped into three essential layers which are: the outermost layer encapsulating cornea, sclera and their intersection named limbus, middle uveal layer which includes iris, choroid, and ciliary body and finally the internal layer called retina (Kels, Grzybowski, & Grant-Kels, 2015). Figure 1.1 displays the horizontal cross section of the human eye and the locations of the mentioned components. As the title suggests, this study will be focused on the retina layer of the human eye.

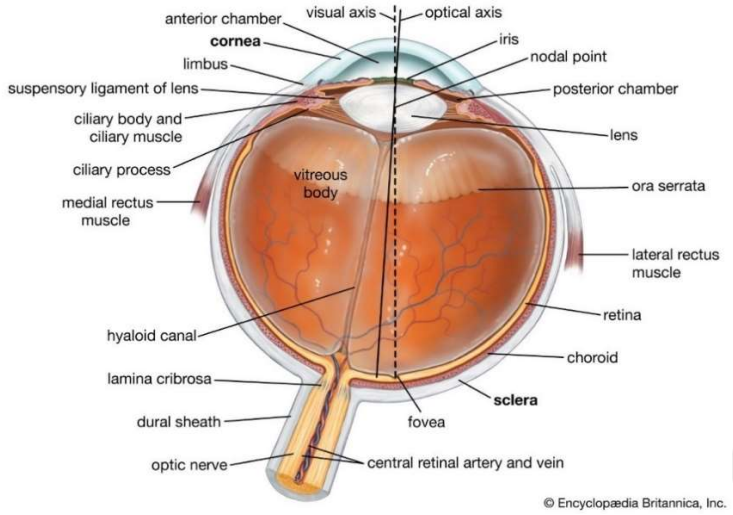


Figure 1.1 Horizontal cross section of the human eye (Encyclopedia Britannica, n.d.)

1.1.2 Medical Imaging of the Human Eye

Encyclopedia Britannica (2019) defines ophthalmology as "medical specialty dealing with the diagnosis and treatment of diseases and disorders of the eye" (para. 1). It is one of the many medical specialties that can benefit from biomedical image processing research. Retinal (fundus) image processing, which this study can also be included in, is a good example to one of those fields. It can be defined as the process of using reflection of light to map the 3D retinal tissues present in the human eye into a 2D projection as an image (Abràmoff & Kay, 2013). Figure 1.3 displays an example of a fundus image of a healthy retina. This part of the human eye can be viewed through an ophthalmoscope or a retinal camera by an ophthalmologist (Kolb, 2005).



Figure 1.2 A photograph of a retinal camera (Ruck, 2007)

The ophthalmologist looks for symptoms in the fundus image to understand if the patient has developed or started developing an eye disease. Main artifacts that can be seen in a healthy fundus image are macula, fovea, optic disc (or optic nerve) and retinal vessels (Kolb, 2005). In figure 1.3, the shiny spot on the left side of the retinal image is the optic disc, the darker area on the centre of the image is macula, at the centre of the macula is the fovea, and finally the red web of vessels coming out of the optic disc are the retinal vessels (Kolb, 2005), which this study focuses on segmenting.

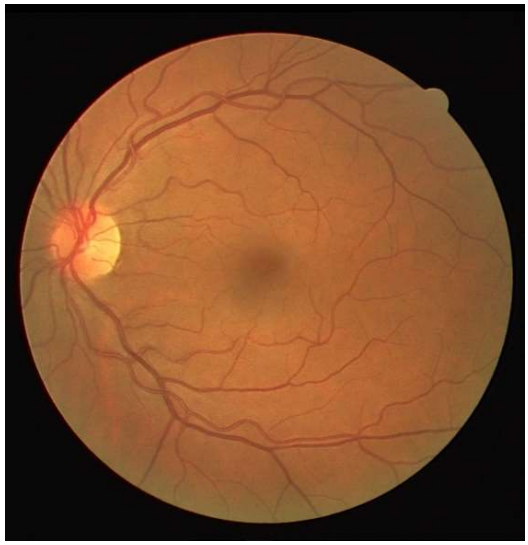


Figure 1.3 Example fundus image from Digital Retinal Images for Vessel Extraction (DRIVE) dataset (Niemeijer, Staal, Ginneken, Loog, & Abramoff, 2004)

CHAPTER TWO
PREVIOUS WORK

It's been mentioned earlier that lots of research is being conducted in the field of computer aided diagnostic systems. In this chapter, previous research will be analysed on the topic of biomedical image processing and its use cases.

2.1 Biomedical Image Processing Applications in Various Fields of Medicine

There are many attempts by researchers to segment meaningful portions of medical data with various purposes, which is not always easy. There are several methodologies used to process biomedical images such as using morphological processing or machine learning techniques. Despite not being used in this study it is logical to define machine learning since it is a very popular tool that researchers use to solve various problems. Machine learning can be defined as the generic term for algorithms which make smart predictions based on datasets, such as: linear regression, supervised/unsupervised learning, and deep learning (Nichols, Herbert Chan & Baker, 2019). Similarly, many of the studies use neural networks so it is essential to define them. As the name suggests, neural networks are a type of machine learning algorithm based on neurons in the human body (Han, Kim, Kim, & Youn, 2018). They contain various number of layers and within those layers, nodes that have weights which determine how much they effect the resulting calculation. Moreover, the learning is done by adjusting the weights of each node based on the calculated error after each iteration during the training so that the error is as low as possible (Han et al., 2018).

To start off with the example applications of biomedical image processing, a study in radiology field can be analysed. One example study would be brain tumour segmentation. A brain tumour is an abnormal group of either cancerous or non-cancerous cells that form on the brain. Depending on the region of the brain where it is present, it can affect a person's brain functions. Some types of brain tumours such as metastatic tumours have serious lethality rates and can cause symptoms such as

vomiting, seizures, headaches, focal signs, trauma, and alteration of mental status (Comelli, Lippi, Campana, Servadei, & Cervellin, 2017).

Moreover, Abdulbaqi et al. (2015) also highlighted the importance of timely and accurate diagnosis of brain tumours in a study to segment and estimate brain tumour volume in computed tomography (CT) scan images. The aim of their study was using Hidden Markov Random Field Expectation Maximization (HMRF-EM) which is a graphical probability model and threshold method, to estimate volume of brain tumours in CT images. They also used common image processing methods such as applying gaussian blur to pre-process the image before applying HMRF-EM algorithm and thresholding as one of the final steps, which are also common with this study. Their study resulted in 94% accuracy which they claim to be more useful than using the manual method.

A more recent example would be segmentation of COVID-19 caused lung infection from CT slices, a study by Ma, Song, Guo, Tan, & Xu (2022). They highlighted one problem with the creation of datasets for this kind of study which is the requirement of manual segmentation from radiologists in order to be able to train and test their model. This requires time from physicians which is a very important resource during a pandemic. Manual segmentation required from physicians is common during the creation of datasets, even for the scope of this thesis, a manually segmented or in other words labelled dataset was required, however the size of the required dataset depends on the method to be used. Unlike the scope of this thesis, which is segmentation using image processing, their method involved a deep learning focused approach which would ideally require a larger dataset. Deep learning is a sub-branch of machine learning which emerged from artificial neural network research with the key characterization being the deep neural network that contains multiple hidden layers (Sarker, 2021). The method Ma et al. (2022) proposed is called “CAPA-ResUNet” standing for content-aware pre-activated residual U-Net. This is a deep learning method that utilizes the popular U-Net architecture developed in Freiburg university by Ronneberger, Fischer, & Brox (2015), for biomedical image segmentation. Ma et al. (2022) combined this network with concepts such as an activated residual block for

down-sampling, and area loss function based on falsely segmented regions. Moreover, they used a public dataset of CT images to train and test the model which resulted in a sensitivity rating of 0.972, a dice (f1-score) rating of 0.775, and specificity rating of 0.646. These results were better, compared to several other known models. What these scores mean will be discussed in the upcoming chapters of this study since those concepts are widely used to understand and measure the performance of the algorithms and were also employed in this thesis as metrics to understand the selected algorithms performance.

Another example study that utilizes a U-Net like architecture is segmentation of both the liver and liver lesions by Vorontsov, Tang, Pal, & Kadoury (2018). They propose a model based on fully convolutional networks (FCNs) to achieve this goal. The FCN architecture uses only locally connected layers while avoiding dense connections (Shelhamer, Long, & Darrell, 2014). The word convolution in convolutional networks is derived from a linear matrix operation in mathematics (Albawi, Mohammed, & Al-Zawi, 2017), which will also be discussed later in the methods chapter. As part of their study, Vorontsov et al. (2018) trained their network using 115 manually segmented CT images without almost no additional pre-processing steps. The pre-processing they used was just in terms of normalizing and clipping the input intensity values to make sure they are within proper range for the neural network. They also used concepts such as skip connections, up-sampling, down-sampling, rectified linear unit, batch normalization, max pooling and sigmoid activation function as layers in their network architecture. Their results were promising for segmentation with a dice score (f1-score) of 0.773. However, for detection it decreases to 0.446 along with maintained high recall and precision scores. Similar metrics such as precision, recall and f1-score were used to evaluate the performance of the algorithm used in this study as well.

2.2.1 Biomedical Image Processing Research in Ophthalmology

As mentioned earlier, ophthalmology is one of the key medical fields that can benefit from biomedical image processing research. One important topic that can be

listed under this category is the automated diagnosis of diabetic retinopathy. An important study in this field belongs to Liu, Li, Xu, Li, & Liang (2019). They refer to diabetic retinopathy as leading cause of blindness in middle-aged patients and they propose an automated assessment system based on fundus images to compensate for the insufficient screening rates to detect the disease. Their proposed method for achieving this task is the development of a weighted path convolutional neural network (WP-CNN). They mention their proposed WP-CNN structure relies on convolutional layers as feature extractors. However, they mention that it also includes a multi-path strategy which has several different CNN feature extractors in each path with a weight assigned to it. Finally, results of these weighted paths are combined with weighted averaging. Liu et al. (2019) use the Structured Analysis of the Retina (STARE) dataset provided by Hoover, Kouznetsova, & Goldbaum (2000) to train and test the performance of WP-CNN algorithm which resulted in accuracy score of 0.9423, sensitivity score of 0.9094, and specificity score of 0.9574.

2.2.2 Retinal Vessel Segmentation Research

According to a survey by Fraz et al. (2012) where they reviewed 69 publications published between 1984 and 2011, retinal blood vessel segmentation methodologies can be grouped under the following categories: “pattern recognition techniques, matched filtering, vessel tracking/tracing, mathematical morphology, multiscale approaches, model based approaches and parallel/hardware based approaches” (p. 5).

One of the early studies that can be categorized under pattern recognition techniques is “Back-propagation network and its configuration for blood vessel detection in angiograms” (Nekovei & Sun, 1995, p.1). One important difference between the dataset used by Nekovei & Sun (1995) and the dataset used in this study is the fact that they used angiograms. An angiogram is an image of blood vessels in the human body, usually used by medical experts to examine the blood flow and vessel health (National Health Service, 2023). Ophthalmologists use a method called Fluorescein Angiography where they inject fluorescein and photograph the flow of it in retinal vessels to achieve this goal (Novotny & Alvis, 1961). Nekovei & Sun (1995)

used images obtained by this method to train a neural network via back-propagation algorithm. In their study they mention the limitations caused by the available hardware back then where the angiogram image being high definition made it impossible for their neural network to process the whole image at once. Back-propagation is a popular and relatively simple method to train neural networks. Using back-propagation means that, during training of the neural network, after each execution, errors are calculated and propagated backwards to adjust the weights used in the neural network (Wang, Zhang, & Fu, 2011). Nekovei & Sun (1995) used various topologies for their network within the scope of their study, however they achieved the best results with 121 nodes in the first layer, 17 nodes in the second layer and 2 nodes in the output layer. Their results show that their well-generalized algorithm with 92% accuracy is superior compared to several other algorithms used back then. They also concluded that in order to achieve faster learning, the parameters used to configure learning such as learning rate and momentum rate shall be adjusted by picking a small learning rate and a medium momentum rate.

The scope of this study, which is detecting if a pixel of an input image belongs to a vessel or not, can be considered a classification task. K-Nearest Neighbour (K-NN) algorithm is another widely used supervised method for classification tasks. As the name suggests, inputs are grouped based on the assumption that related data points will lie close to each other in a given data matrix. There is a recent study on retinal vessel extraction using the K-NN algorithm by Rehman et al. (2022). Within the scope of their study, they focused on combining this method with support vector machines (SVM) which is a type of supervised deep learning method. They utilize several pre-processing techniques, which are also used in this study, such as green channel extraction, smoothing, and histogram equalization. Rehman et al. (2022) state that the purpose of the pre-processing steps was, quality improvement of images that had poor quality due to unintentional head and eye movements during retinal imaging. After pre-processing, they apply several feature extraction techniques to form a feature matrix which is then classified by K-NN and SVM. They used the same dataset as in this study, which is the DRIVE dataset (Niemeijer et al., 2004), to measure performance of their method, which resulted in a 92% dice score.

Line tracking is an interesting unsupervised method which earns its own spot in the category list made by Fraz et al. (2012). In their study, Vlachos & Dermatas (2010) utilized this method to segment retinal vessels and proposed a multi-scale single channel method. They explain that choosing an unsupervised method like this requires pixel by pixel processing just like it does for the proposed method in this study. However, their method is even more process-heavy compared to the proposed method in this study. It relies on mathematical formulas to find a seed pixel and navigate through the retinal vessel by continuously evaluating candidate pixels, it does so until a certain condition becomes no longer valid, marking the end of the vessel. Vlachos & Dermatas (2010) also apply pre-processing step local brightness normalization and post-processing with morphological operations. They also used the same images from DRIVE dataset (Niemeijer et al., 2004) to see the performance of their proposed algorithm and their results look promising. Vlachos & Dermatas (2010) report an accuracy score of 0.929, sensitivity score of 0.747 and a specificity score of 0.955.

Another important study for retinal vessel segmentation literature was by Hoover et al. (2000). What makes this study different is that they collected the fundus image dataset required for the vessel segmentation and formed the STARE dataset. This dataset contains 20 manually segmented images which are separately segmented by 2 observers who have at least 2 years of retinal imagery experience. STARE dataset is also widely used across other studies about retinal vessel segmentation (Fraz et al., 2012). The method proposed by Hoover et al. (2000) was based on match filtering, it also has its own category in the list by Fraz et al. (2012). However, they mention they built their method based on an older study by Chaudhuri, Chatterjee, Katz, Nelson, & Goldbaum (1989). The proposed method by Chaudhuri et al. (1989) was to approximate the morphological characteristic of a retinal vessel by matching it with a kernel shaped like a gaussian curve. In their study, after pre-processing the image with smoothing, Chaudhuri et al. (1989) applied the mentioned gaussian kernel in all directions with a resolution of 15 degrees and maximum match rates are retained. They did not provide metrics like in other studies because they did not have the datasets of this century. However, they compared their results with resulting images from other

methods and concluded their results are sufficient. On the other hand, Hoover et al. (2000) proposed to apply green channel extraction followed by convolving the image with a filter bank of 12 2-d filters, and finally calculating the Matched Filter Response (MFR) based on the kernel with maximum score for each pixel. After the MFR is obtained, their method continues processing with a technique called Threshold Probing, where they iterate over the image piece by piece to decide which structure is to be considered as a vessel. The algorithm tests a given region based on predefined properties and if the result is positive, it marks the composing pixels as vessels and recycles the remaining pixels to be considered again. This property enables pixels to be considered for multiple regions, which is one of the strengths of this study compared to the others. Hoover et al. (2000) also mention that it takes 5 minutes to process for each retinal image due to the available computing power to them at the time.

As mentioned earlier, another medical imaging method in ophthalmology is angiogram images. Unlike the proposed method which uses fundus images in this study, Sun, Chen, Jiang, & Wang (2011) propose a morphology-based method to extract the retinal tree from retinal angiograms. Their approach starts processing the image by multiscale enhancing of the image by a top-hat operator. This method uses morphological opening (which is also used in this study) with structuring elements of different sizes to estimate the background of the angiogram. Moreover, they further process the angiogram image by using a fuzzy filter to eliminate remainder noise after enhancement. They refer that step of the processing as fuzzy opening since that method is also mased on morphological opening. Beforehand, to be able to treat the intensity values as fuzzy sets, they used intensity normalization to get the intensity values from 0 to 255 to 0 to 1. After fuzzy opening, they use fixed thresholding to approximately mark the vessel regions. Finally, they use watershed techniques to determine the retinal vessel boundaries.

Finally, the most recent publication during the time of this study was by Sidhu, Sachdeva & Katoch (2023). Their focus was extraction of the vessels for diabetic retinopathy detection. They used a different dataset from other studies which was provided by a medical education institute in India. Moreover, their method was based

on 2 main pillars: Principal Component Analysis (PCA) and Contrast Limited Adaptive Histogram Equalization (CLAHE). PCA is a statistical technique used to extract essential information from a given dataset and represent it in form a variable set called principal components (Abdi & Williams, 2010). Details for CLAHE will be discussed in the methods chapter as it is essential for this study as well. Sidhu et al. (2023) proposes the initial step of acquiring the grayscale form of the retinal image using PCA. They follow by applying CLAHE and smoothing. Finally, they threshold their image using Iterative Self-Organizing Data Analysis Technique (ISODATA). Their implementation resulted in a sensitivity of 0.9234, a specificity of 0.9955 and an accuracy of 0.9682, which they deem superior compared to other methods in literature.

CHAPTER THREE
MATERIAL AND METHODS

In this chapter, the used material such as the dataset and the computer vision framework will be discussed. Moreover, several image processing methods will be explained along with their underlying mathematical background.

3.1 Material

3.1.1 Used Fundus Image Dataset

The dataset used in this study was the publicly available dataset for the online Digital Retinal Images for Vessel Extraction challenge hosted by Image Sciences Institute of University Medical Center Utrecht as a grand challenge (Niemeijer et al., 2004). This dataset consists of 40 retinal images collected during a screening in the Netherlands from people of ranging age between 25 to 90. These images are divided into 2 categories as test images which have a corresponding image with vessels marked, and training images which do not have an image with vessels marked. One of the reasons for this separation is to be able to train supervised machine learning models using the train data and test the results by running the model on test data. Moreover, separation is done to test against the overfitting problem of the machine learning algorithms which can briefly explained as the machine learning algorithm performing very well on one dataset that it trained on and cannot generalize for other datasets. On the other hand, the provided dataset was relatively small compared to data that is usually used to train machine learning models. Liu et al. (2019) addresses this issue on their experience with deep models and the STARE dataset (Hoover et al., 2000) which included 131 fundus images. As a result of these factors along with the additional processing load machine learning algorithms usually introduce and, this study being about methods that are considered as preliminary operations before the image can be processed for the detection of a disease, using unsupervised image processing methods were preferred over machine learning methods in this study.

3.1.2 Image representation in Digital World

To segment the retinal vessels which are the regions of interest of this study: channel extraction, smoothing, histogram equalization, thresholding and morphological operations were applied in that exact order. Before explaining these image processing methods to process the image, basics of digital image storage and colour spaces need to be understood. The primary concept that is to be explained is representation of images in the digital world. There are several colour spaces that are used worldwide. The most popular ones are RGB (Red, green, blue) and CMYK (Cyan, magenta, yellow, key) which take their names from the essential colours they use to form other colours. CMYK is mostly used in printing since when all the essential colours combined the resulting colour is black and since paper is white you do not often need to produce the colour. This is out of the scope for this study.

However, RGB colour space is relevant considering that it is used in monitors and digital cameras. Digital images can be represented as 3-dimensional matrices in RGB colour space in which each cell in each dimension represent their corresponding colour density value on a scale of 0 to 255 for red, green, and blue components. It is possible to represent this with a mathematical equation like below.

$$p(x,y,c) = \begin{cases} \mathrm{r(x,y)}, & \mathrm{c} = 1 \\ g(x,y), & c = 2 \\ b(x,y), & c = 3 \end{cases} \middle| \begin{array}{c} 3 \geq c \geq 1 \\ w > x \geq 0 \\ h > y \geq 0 \\ 255 \geq p(x,y,c) \geq 0 \\ x,y,c,w,h \in N \end{array} \tag{3.1}$$

In this equation, p represents the colour image matrix, c represents the colour channel index taking index 1 as red, 2 as green and 3 as blue channel, “r” represents red, “g” represents green, and “b” represents the blue channel matrix. Variables “x” and “y” represent the coordinates of the pixel, whereas “w” and “h” represent the colour image width and height respectively where “N” stands for natural numbers. For each matrix cell in the colour image, colour density values at the same “x” and “y” coordinates in each colour channel are combined to produce a specific colour. For example, considering first value for red, second for green and last for blue, a cell with

values 255, 255, 255 would form a white, a cell with 255, 255, 0 values would be yellow and a cell with values 128, 0, 128 would form a purple pixel.

3.2 Methods

3.2.1 Image Processing Fundamentals

3.2.1.1 Convolution

Prior to explaining any method related to image processing, it is essential to have theoretical background on the mathematical operation, convolution. Convolution is another way of combining two functions such as addition, subtraction or multiplication. In continuous mathematics domain, the convolution operation can be characterized with the equation below.

$$f(t) * g(t) = \int_{-\infty}^{\infty} f(T)\, g(t-T)\, dT \tag{3.2}$$

In equation 3.2, “f” and “g” represent two continuous functions, “t” represents the independent variable which corresponds to time in signal processing, “T” represents variable of integration and “*” represents the convolution operation. However, since this study is not about continuous signals, the discrete version of this equation 3.2 should be introduced.

$$f[t] * g[t] = \sum_{T=-\infty}^{\infty} f[T]\, g[t-T] \tag{3.3}$$

In equation 3.3, “f” and “g” represent two discrete functions, “t” represents the independent variable which corresponds to sample time in signal processing.

To observe the behaviour of this operation, suppose 2 signals were sampled with the same rate of 1 millisecond within the time window of 0 to 2 milliseconds with the corresponding amplitude values shown below in figure 3.1.

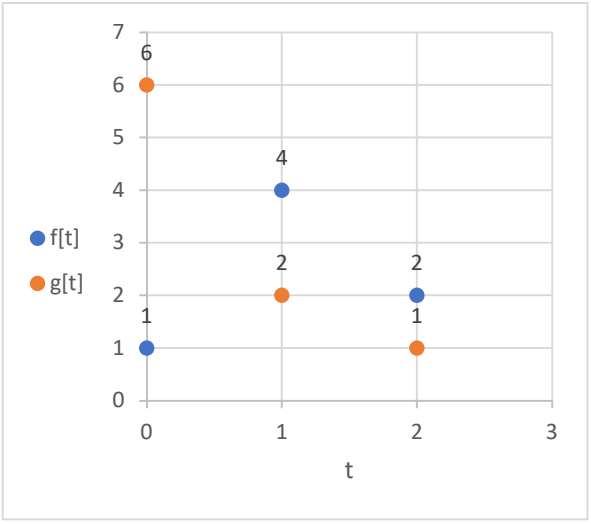


Figure 3.1 Graphical representation of the mentioned signals

If the values displayed in figure 3.1 are plugged in to the equation 3.3, we would get the following solution shown in equation 3.4.

$$
\begin{gathered}
t < 0 => (f * g)[t] = 0 \\
t = 0 => (f * g)[0] = f[0] \times g[0] = 6 \\
\mathrm{t} = 1 => (f * g)[1] = f[0] \times g[1] + f[1] \times g[0] = 26 \\
\mathrm{t} = 2 => (f * g)[2] = f[0] \times g[2] + f[1] \times g[1] + f[2] \times g[0] = 21 \\
t = 3 => (f * g)[3] = f[1] \times g[2] + f[2] \times g[1] = 8 \\
t = 4 => (f * g)[4] = f[2] \times g[2] = 2 \\
t > 4 => (f * g)[t] = 0
\end{gathered} \tag{3.4}
$$

Moreover, if the pattern of calculation is derived from equation 3.4, it can be concluded that this can be considered as taking the inverse of the second signal in time domain (g[-t]) and step by step moving it towards f[t] while multiplying each value that overlap for each timestep. This is the behaviour of convolution that is valuable to this study. However, our signal at hand is a grayscale image, that is a 2-dimensional time independent signal which makes 1-dimensional convolution methods non applicable. This is where 2-dimensional discrete convolution methods come in handy.

3.2.1.2 Hadamard Product

To explain 2-dimensional convolution method better in further chapters, Hadamard product should be considered beforehand. Contrary to a regular matrix multiplication,

Hadamard product or element-wise product is a mathematical operation performed on two matrices with identical dimensions where elements of matrices with the same indexes are multiplied to calculate the element with the same index in the resulting matrix. It can be formulated as below in equation 3.5, where "f" and "g" represent two independent matrices and "." represent the operation of Hadamard product.

$$f = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$
$$g = \begin{bmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{bmatrix}$$
$$f.g = \begin{bmatrix} f_{11} \times g_{11} & f_{12} \times g_{12} & f_{13} \times g_{13} \\ f_{21} \times g_{21} & f_{22} \times g_{22} & f_{23} \times g_{23} \\ f_{31} \times g_{31} & f_{32} \times g_{32} & f_{33} \times g_{33} \end{bmatrix} \quad (3.5)$$

Most image processing methods use this matrix operation in combination with other operations to apply various characteristics to images or regions of images. One of the biggest problems of applying Hadamard product is the requirement of same size for the matrices. In the image processing field, the image is rarely the same size of the kernel making them two matrices with different dimensions which makes operations like Hadamard product not applicable. This is not a concern since the usual application is to apply the kernel to the image piece by piece where each piece has the same dimension as the kernel, sliding it across the image. This and other related concerns will also be discussed in the next chapter.

3.2.1.3 Discrete 2-Dimensional Convolution

In the previous chapter, it was explained that 1-dimensional convolution is not sufficient considering the scope of this study. Luckily, it is also possible to apply convolution to 2-dimensional structures with 2-dimensional discrete convolution. Mathematical representation of 2-dimensional discrete convolution is shown below in equation 3.6.

$$[f * g](x, y) = \sum_{a=-\infty}^{\infty} \sum_{b=\infty}^{\infty} f[a, b]\, g[x - a, y - b] \quad (3.6)$$

In the image processing domain, equation 3.6 can be interpreted as follows: “f” can be interpreted as the image, “g” can be interpreted as the filter kernel. Moreover, the whole formula can be considered as a 2-dimensional kernel “g”, being flipped in both vertical and horizontal dimensions and slid across the 2-dimensional image matrix “f” and each value being multiplied with the overlapping weight from the kernel (Hadamard product), afterwards, results of all multiplications being summed and the central cell value “f [x, y]” being updated with the calculated value. This makes it possible to apply a certain characteristic determined by kernel “g” to image “f”. To visualise this better consider an image “f” and a filter kernel “g” with values shown below in equation 3.7.

$$f = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad g = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$f * g = 1 \times 1 + 1 \times 5 + 1 \times 9$$
$$f * g = 15 \tag{3.7}$$

It can be inferred from the result of calculation in equation 3.7 that, the result of a 2-dimensional discrete convolution operation shrinks the image since the dimensions of the result is always smaller than the dimensions of the input. In this specific case, both the image and kernel dimensions were 3 by 3, and the resulting dimensions were 1 by 1. In general, dimension or shape of a convolution operation can be calculated with the formula given in equation 3.8.

$$z = (w_f - w_g + 1) \times (h_f - h_k + 1) \tag{3.8}$$

In equation 3.8, “z” represents the shape of the output, “w” represents the width or number of columns in the matrix specified in the subscript “h” represents the height or number of rows in the matrix specified in the subscript.

Number of elements in the result of convolution operation is an important property to consider. For some applications it might be useful to have a smaller matrix as a result, compared to the given input. For example, in Convolutional Neural Networks (CNN) the convolutional layers where convolution operations are performed is used the extract features from input data where each layer output will have more intense

information stored in each output cell. Moreover, adding more layers will contribute to the ability of the network to learn more complicated features (Liu et al., 2019).

On the other hand, shrinking of the output is an unwanted event in image processing since when a filter is applied to the image, unless the purpose is resizing, it is not expected that the dimensions of the resulting image to be different from the dimensions of the input image. In the following chapters, concepts related to overcoming this problem are explained.

3.2.1.4 Padding

One of the solutions to the problem of image resizing while applying the 2-dimensional discrete convolution operation would be applying "Padding". In image processing domain, padding is the operation of encircling the image matrix with selected values, making the image larger and more compatible with the kernel at hand in terms of convolution and filtering operations.

There are various methods where you can pad the image such as surrounding it with average of neighbouring cells, value of the closest cell or all zeros. In equation 3.9 an example of "Zero padding" is shown where the image matrix from equation 3.7 is padded with zeros.

$$f = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 4 & 5 & 6 & 0 \\ 0 & 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.9}$$

3.2.1.5 Stride

Another important feature that contributes to size of the output of 2-dimensional discrete convolution is "Stride". Stride is the step size of the so-called movement of the kernel inside the image matrix after the summation of each Hadamard product

calculation which was described previously. It will determine where the central cell will be for each calculation.

If 2-dimensional discrete convolution operation is manipulated with the properties of padding and stride, equation 3.8 can no longer be used to determine the shape of the convolution output. However, generalizing the properties of stride and padding with the formula provided in equation 3.8 would result in equation 3.10 which can be used for this purpose. In equation 3.10, "z", "w" and "h" represent the same concepts as in equation 3.8 whereas, "p" represents number of paddings and "s" represents the number of strides.

$$z = \left(\frac{w_f - w_g + 2\mathrm{p}}{\mathrm{s}}\right) \times \left(\frac{h_f - h_g + 2\mathrm{p}}{\mathrm{s}}\right) \tag{3.10}$$

Considering these properties, a more meaningful 2-dimensional discrete convolution operation can be applied with the image and kernel matrices described in equation 3.7. Equation 3.11 shows image matrix "f" zero padded with a size of 1, steps and result of the 2-dimensional discrete convolution operation is shown considering stride as 1. From the formula presented in equation 3.10, the shape of the output can already be calculated as 3 by 3 and the result of equation 3.11 confirms that.

$$f = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 4 & 5 & 6 & 0 \\ 0 & 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad g = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{gathered} (f * g)_{11} = 1 \times 0 + 1 \times 1 + 1 \times 5 = 6 \\ (f * g)_{12} = 1 \times 0 + 1 \times 2 + 1 \times 6 = 8 \\ (f * g)_{13} = 1 \times 0 + 1 \times 3 + 1 \times 0 = 3 \\ (f * g)_{21} = 1 \times 0 + 1 \times 4 + 1 \times 8 = 12 \\ (f * g)_{22} = 1 \times 1 + 1 \times 5 + 1 \times 9 = 15 \\ (f * g)_{23} = 1 \times 2 + 1 \times 6 + 1 \times 0 = 8 \\ (f * g)_{31} = 1 \times 0 + 1 \times 7 + 1 \times 0 = 7 \\ (f * g)_{32} = 1 \times 4 + 1 \times 8 + 1 \times 0 = 12 \\ (f * g)_{33} = 1 \times 5 + 1 \times 9 + 1 \times 0 = 14 \end{gathered}$$

$$f * g = \begin{bmatrix} 6 & 8 & 3 \\ 12 & 15 & 8 \\ 7 & 12 & 14 \end{bmatrix} \tag{3.11}$$

3.2.1.6 Image Histogram

It's been explained in previous chapters how an image is represented in the RGB colour space by each pixel having intensity values of red, green and blue channels that are being combined to determine its colour. In image processing, it usually is an advantage to know how intensity values have been distributed across the image for each channel. This can easily be achieved by producing and examining the global image histogram, which contains the three essential channels, will hold global information about the image (Tan & Isa, 2011). This information can help determine the image processing strategies to be used further in the process.

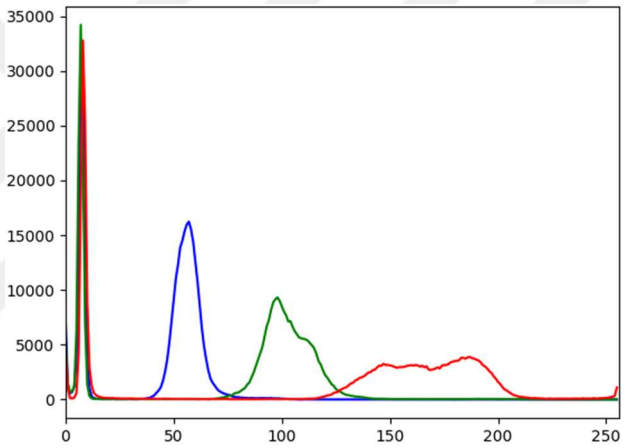


Figure 3.2 Global Image Histogram

Figure 3.2 is a global image histogram acquired from the image shown in figure 1.3, red, green and blue lines represent their corresponding channels, values on the horizontal axis represent the intensity values stored in each cell of the digital image, values on vertical axis represent the number of cells with that value. It can be seen from the global histogram that the image has a lot of values between 0 and 25. The reason for that is the black pixels encircling the fundus image which is in fact not part of the retinal image gathered from the patient. On the other hand, other values are distributed across the histogram. It can be observed that for each channel, ranges for intensity values differ. It is visible that blue channel values are centred roughly around 55, green channel values are centred roughly around 100 and red channel values are

centred roughly around 175. What this means for the image will be discussed in the next chapter.

3.2.2 Methods for Retinal Vessel Segmentation

3.2.2.1 Colour Channel Extraction

Now that the fundamentals are covered, more advanced methods, where one or more of the previously explained methods are used in combination to achieve the results, can be covered. In this study, the first processing step applied to the digital fundus image, is a method called colour channel extraction. This is one of the methods in image processing which produces a grayscale image. Other grayscale conversion methods usually achieve this through taking all red, green, and blue components into account and then producing the output accordingly. As an example, the averaging method averages the intensity values of the pixel in red, green and blue channels to produce the grayscale value. However, by using the Colour Channel Extraction method, intensity values in one channel of the channels are picked, and the values in the other 2 channels are ignored completely. This brings up the question of which one of the red, green and blue channels to extract.

In this study, green channel was chosen to be extracted by the colour channel extraction step. The reason for selecting the green channel is digital cameras having 2 green sensors for each red and blue sensor, making the green channel the channel which has the least noise compared to others (Vikram, Nomula & Srinivas, 2022). Hoover et al. (2000) also support that statement by identifying the blue channel as usually empty and the red channel as saturated. However, these factors also depend on the fundus camera technology, as it can be seen from figure 3.3 the blue channel is not empty for the sample image.

On the other hand, even with the bare eye, it can be seen that the green channel has the better use case by having a look at figure 3.3. It can be concluded that retinal vessels have a higher contrast and more visibility in the green channel compared to red and

blue channels. This phenomenon is also mentioned by Vlachos & Dermatas (2010) in their study based on line tracking algorithm.

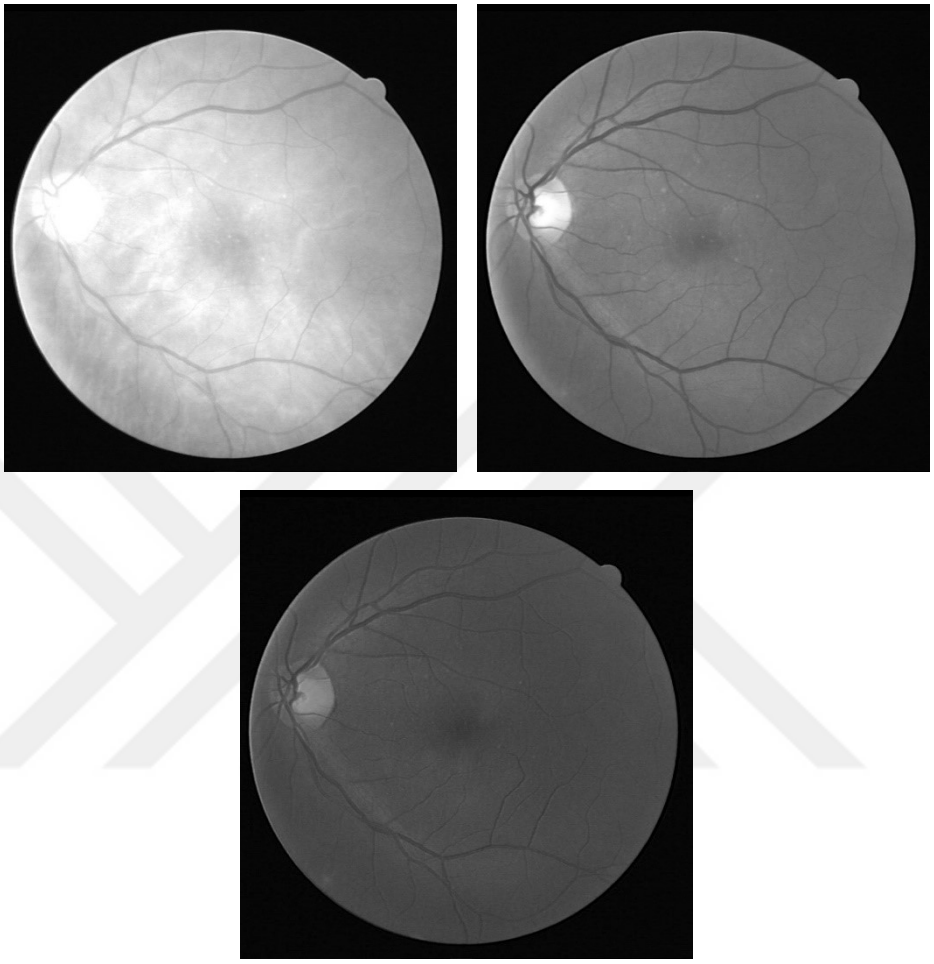


Figure 3.3 Red (top left), green (top right) and blue (bottom) channels of image

3.2.2.2 Smoothing

Following the green channel extraction step, smoothing was applied. Smoothing is the correspondence of low pass filtering in digital signal processing in the image processing domain. As the name suggests, smoothing is done to have a smoother image where drastic changes between density values in other words “noises” or “high frequency components” are reduced. This operation will result in a blurry image and usually improve performance of algorithms following smoothing. There are several

methods to smoothen the image, some examples would be averaging, median filtering and gaussian filtering.

In this study, gaussian filtering or gaussian blurring is applied to the image by using 2-dimensional convolution which was explained in previous chapters. In order not to reduce the image size after convolution, padding and stride values were configured automatically by the computer vision framework.

$$g(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma}} \tag{3.12}$$

Mathematical representation of a 2-d gaussian kernel is expressed in equation 3.12. Where "x" and "y" are the coordinates that indicate the location compared to the central pixel and "σ" is the variable which determines the level of blur applied to a given pixel and the neighbouring pixels, so an increase in "σ" will result in less high frequency components and more blur around a pixel region (Misra & Wu, 2020).

Equation 3.13 shows examples for a gaussian kernel and an averaging kernel which can be applied to images using 2-d convolution. As you can see the gaussian kernel shown by "g" has increased weight for the central pixel and less weight for corner pixels. On the other hand, the averaging filter kernel shown by "a" have homogeneous weights across the kernel.

$$g = \begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix}, \qquad a = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} \tag{3.13}$$

Figure 3.4 shows the application of a gaussian kernel to the image acquired after green channel extraction step which is shown in figure 3.3.



Figure 3.4 Fundus image after green channel extraction and smoothing

3.2.2.3 Histogram Equalization

Earlier, definition and importance of a global image histogram was discussed. After examining the image histogram, further processing methods can be picked based on the information gathered. A quick examination of the histogram shown in figure 3.2 showed the global information on red, green and blue channels. Nonetheless, this information is now outdated considering that green channel extraction and smoothing steps are applied.

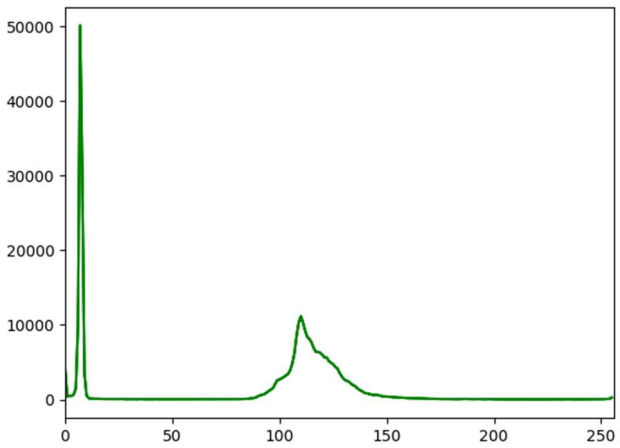


Figure 3.5 Global image histogram on single channel

Figure 3.5 shows the global image histogram on green channel after smoothing. As discussed previously, the outer region of the image where black pixels are dominant cause the left peak on the image histogram whereas most of the meaningful data is in range 75 to 150. However, we can widen that range using a method called histogram equalization. Histogram equalization is the process of stretching the image histogram values to a wider range allowing more contrast to be observed in the image. This is especially helpful when pixel values in the image are stuck in a certain range. There are multiple ways of applying histogram equalization to an image.

3.2.2.3.1 Uniform Histogram Equalization. One of the most used methods is conventional or uniform histogram equalization. This method stretches all the ranges considering all the intensity values in an image histogram. Uniform histogram equalization method offers a more homogenous distribution of intensity values across the global image histogram. On the other hand, it can introduce over-enhancement of the contrast in some undesirable regions (Lin et al., 2015). Figure 3.6 covers a good example of this over-enhancement. It is conspicuous from the fundus image that contrast is somewhat improved in the regions of interest. However, noise have been introduced to the dark regions that are not part of the collected retinal image and brightness of the regions hugely differ across the different regions of the fundus image. This is also visible in the presented histogram, the sharp peak that was present in the lefthand side of the histogram split into 3 peaks and the value range has been widened.

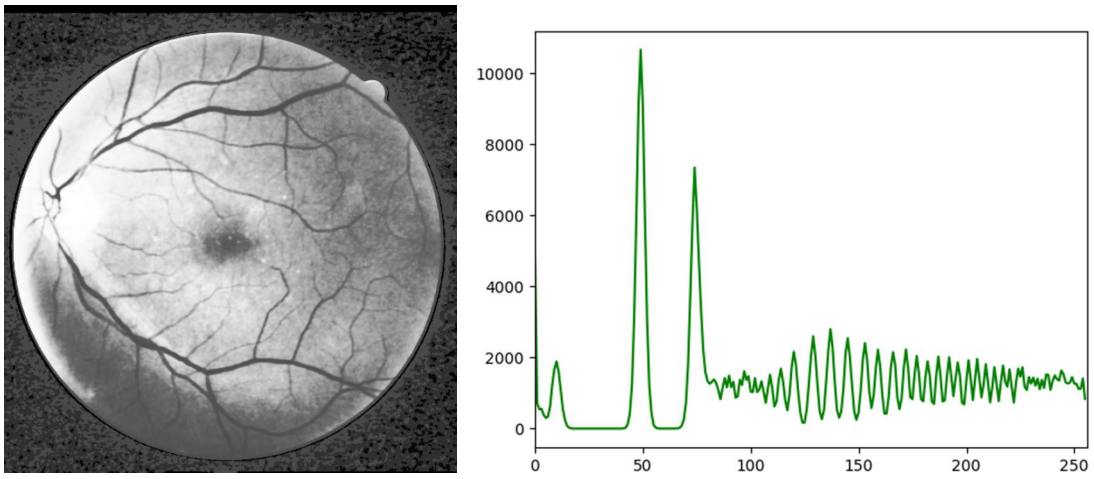


Figure 3.6 Fundus image and image histogram after green channel extraction, smoothing and uniform histogram equalization

3.2.2.3.2 Contrast Limited Adaptive Histogram Equalization. As discussed earlier, uniform histogram equalization is a widely used method to enhance contrast. However, it does not have a structure to control the level of enhancement (Yang, Oh & Park, 2003). This is where adaptive histogram equalization methods come in handy. These methods split the image into smaller tiles and equalize the histogram inside the tile boundaries. It calculates the equalization constants based on neighbouring pixels in a tile, preventing regions which are not inside the same tile from affecting the calculation. In this study, following the smoothing step, Contrast Limited Adaptive Histogram Equalization (CLAHE) is applied. CLAHE is a type of adaptive histogram equalization which in addition to splitting the image into tiles and equalizing the histogram, it also limits the level of contrast enhancement for each intensity value (Mehdizadeh, Tavakoli Tafti, & Soltani, 2023).

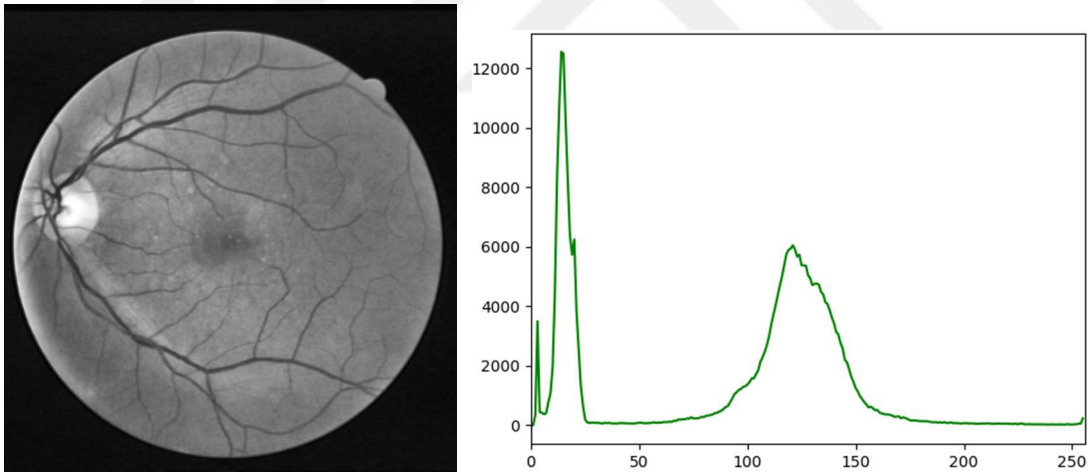


Figure 3.7 Fundus image and histogram after green channel extraction, smoothing and CLAHE

By observing the difference between figure 3.7 and figure 3.6 it can be concluded that the CLAHE method produces a better result compared to uniform histogram equalization in terms of choosing the right amount of enhancement for different regions. The produced image also has a reasonable distribution of brightness unlike the result produced by the uniform histogram equalization. Moreover, looking at the

image histogram, it is visible that number of intensity values on the peaks decreased significantly compared to the initial single channel histogram shown in figure 3.5, confirming intensity values have been stretched to a wider range.

3.2.2.4 Thresholding

One of the essentials of data segmentation from digital images is the thresholding step. In this study, succeeding CLAHE step, thresholding was applied to the fundus image. Thresholding is a method that enables meaningful data to be separated from the rest of the image. The basic logic behind it is if a certain region that needs to be segmented from the image has a different range of intensity values compared to its background, it can be extracted from the image via thresholding (Davis, Rosenfeld & Weszka, 1975). Thresholding has many variants as well, such as adaptive mean thresholding, adaptive gaussian thresholding, truncated thresholding, and binary thresholding. In this study, several thresholding methods were applied to pick the one with the most sufficient results.

3.2.2.4.1 Global Binary Thresholding. To understand thresholding, global binary thresholding is a good start. Global binary thresholding is a very basic operation which can be explained by assigning all pixel values surpassing a threshold to an intensity value, and rest to another intensity value. These values are usually picked as the maximum and minimum values possible to create a visible difference between regions of interest and other parts of the image. This relation has been formulized in equation 3.14 where "T" represents the threshold, "x" and "y" represent the pixel coordinates and "g" represent the global binary thresholding function.

$$g(x,y) = \begin{cases} 255, & g(x,y) > T \\ 0, & otherwise \end{cases} \tag{3.14}$$

Figure 3.8 shows two fundus images after binary thresholding with thresholds picked as 90 (left) and 150 (right). It can be inferred from these images that different global threshold values serve different purposes. For example, lowering the threshold includes less non-vessel artifacts to the result at the cost of losing some portion of the

vessel data. Nonetheless, increasing the threshold includes more vessel data along with more non-vessel artifacts. Also, both resulting images include the dark region outside the retinal image which is not part of the region of interest for this study.

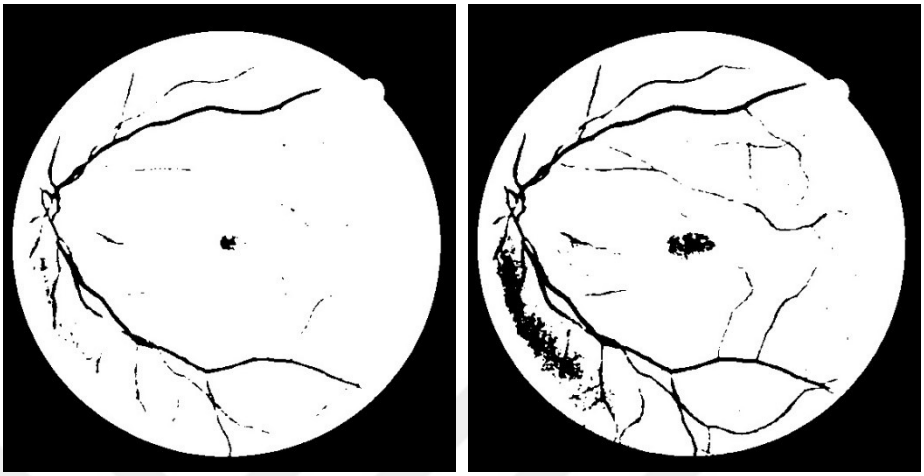


Figure 3.8 Global binary thresholding with different threshold values

$$g(x,y) = \begin{cases} 0, & g(x,y) > T(x,y) \\ 255, & otherwise \end{cases} \tag{3.15}$$

3.2.2.4.2 Adaptive Mean Thresholding. In equation 3.15 inverse adaptive thresholding, which was used in this study is formulized. As it can be understood from the formula, any cell value that is above a certain threshold is set to 0 and otherwise it is set to 255, this is also the reason it is called as "inverse" since values surpassing the threshold are assigned to the minimum value and otherwise, they are assigned to maximum value. Moreover, in this equation "T" represents the function which is used to determine the threshold of which above values are equated to zero. In this study, adaptive mean thresholding was used, so result of "T" is calculated by taking the neighbouring values of a pixel in a specified region, calculating the mean value for that range and applying it as a threshold to pixels in the region. Moreover, it can be adjusted as a last step by adding or subtracting an offset after mean calculation to calculate the threshold. The main difference between adaptive methods and other methods is, as discussed earlier in CLAHE method, adaptive thresholding being "adaptive" which indicates threshold value being calculated separately for each cell using neighbouring cells instead of using a fixed value like in binary thresholding.

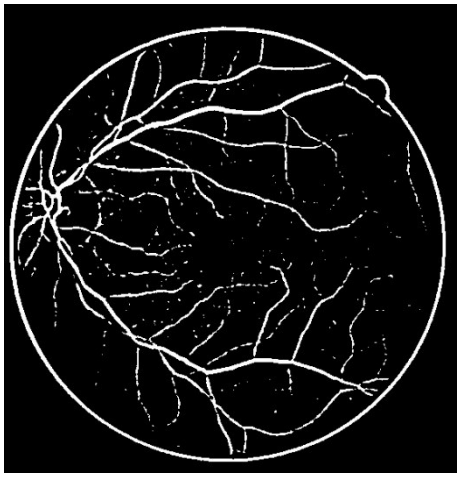


Figure 3.9 Fundus image after green channel extraction, smoothing, CLAHE and inverse adaptive thresholding

In Figure 3.9 the resulting image is shown. Now that the image has been thresholded, it can be considered as a 2-dimensional logical matrix consisting of intensity values 255 (true) and 0 (false). It is seen that most of the retinal vessels are segmented and marked as “true” along with several artefacts that are not of interest. One of the erroneous inclusions to this result is the region which separates medical data from the dark background. How this problem is handled is explained in the next chapter.

3.2.2.5 Masking and Morphological Operations

Even though methods of denoising were applied to the image, there still might be the need to eliminate unwanted artifacts, usually after thresholding step. Morphological operations are a good way to achieve that after certain processing steps are already applied to the image. Szeliski (2010), explains the basic idea behind morphological operations is to convolve the image using a kernel called “structuring element” and then select the output to be either true or false for pixels that are affected based on thresholded convolution result. There are several types of morphological operations, those are morphological erosion, morphological dilation, morphological opening and morphological closing. The essential operations are morphological

erosion and dilation since morphological opening and closing are derived from these two operations. To be specific, morphological opening is applying erosion then dilation whereas morphological opening is applying dilation then erosion.

3.2.2.5.1 Structuring Element. Structuring element is a kernel just like the other kernels explained in this study. The main difference is it holds binary values of true or false instead of intensity values. Structuring element can be structured in a way that a certain shape can be given to the image (Szeliski, 2010). Some examples of these shapes are represented in the equation 3.16 below where "d" represents diamond, "r" represents rectangular, "t" represents triangular, "e" represents elliptical, "s" represents square, and "c" represents cross kernels.

$$d = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad \mathrm{r} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathrm{t} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathrm{e} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathrm{s} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \mathrm{c} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (3.16)$$

Structuring element is slid across the image as explained earlier and pixel values which are overlapping with the structuring element are morphed according to the chosen algorithm.

3.2.2.5.2 Morphological Dilation and Erosion. As mentioned earlier, one of the two essential morphological operations is dilation. Dilation can be thought as sliding the structuring element across the image while for each "true" or "white" pixel hit by a "true" pixel in the structuring element, the pixel hit by the central pixel of the structuring element will be turned into "true". In other words, for an image "I" and structuring element "S" which has origin at "n", the result of I diluted by S is a set of

every point of “n” where S intersecting I is not empty (Chudasama, Patel, Joshi, & Prajapati, 2015).

Moreover, applying dilation operation will thicken or in other words, expand the boundaries of an object in an image while filling small gaps (Raid, Khedr, El-Dosuky, & Aoud, 2014). Hence, this property is especially useful when image at hand has small noises inside the region of interest. Looking at the resulting image after adaptive thresholding, figure 3.9, it can be seen that regions of interest, which are the retinal vessels, do not have noticeably high noise component within their boundaries thus this method is not useful to remove noise from the thresholded image. However, it is visible there is noise outside of the vessel boundaries.

The other one of the two essential morphological operations is erosion. Erosion operation can be applied to the image similarly to the dilation operation. However, the difference is, erosion turns the pixel that is overlapping with the central pixel to “false” or “black”. This results in making the object thinner, shrinking it in size. This property is useful to this study in several ways. First of all, the mask that was provided was not sufficient for the removal of the noise withing the boundary of the retinal image and the dark background, so it needed to be eroded, and secondly the remainder noise within the retinal image background is a good candidate for removal by erosion.

3.2.2.5.4 Masking. It’s been previously discussed that the result of the inverse adaptive mean thresholding had several problems. One of those problems was that it included the boundary between the retinal image and the dark background. To overcome this issue, the mask that was provided by the dataset was used. This mask was provided to separate regions of interest from the background.

Masking operation was performed by iterating over both mask and the image. For each iteration, pixel values were checked in the same coordinates. If the pixel value for the mask image is “true” or “white” the data from the image was used and otherwise the pixel value on the image was set to “false” or “black”.

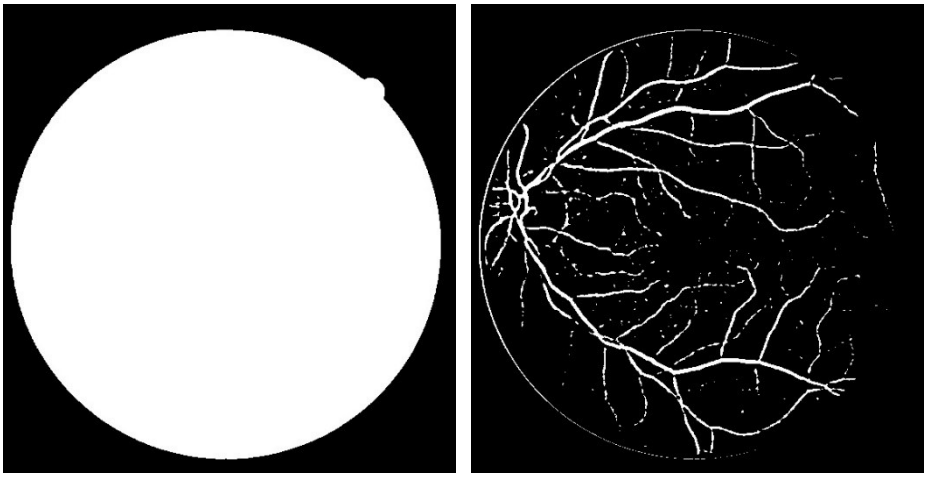


Figure 3.10 Mask image and fundus image after masking step

Figure 3.10 shows the attempt to mask out the boundary noise included in the image after adaptive thresholding. It can be seen from this result that the mask was able to cancel out some of the boundary but was not able to completely remove it from the resulting image. Moreover, it has similar characteristics to a vessel so applying morphological operations to the image itself without also harming the vessel data is not an option. To solve this issue the provided mask was morphologically eroded 3 times by a cross shaped kernel of size 3 by 3. As discussed earlier, this made the object boundaries thinner and shrunk the mask by a small amount.

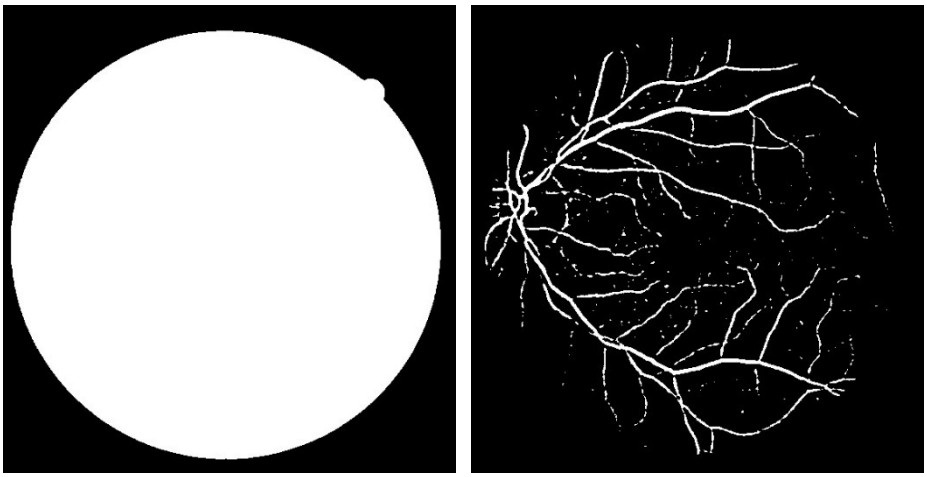


Figure 3.11 Eroded mask image (left) and fundus image after masking step (right)

Looking at the result at Figure 3.11 it can be seen that, the erosion of the mask made it possible to cancel out the noise at the boundary of the retinal image while not harming any of the vessel data. However, it can also be observed that there is still some visible noise along the spaces between the retinal vessels.

3.2.2.5.5 Morphological Opening and Closing. One of the derived morphological operations is morphological opening. This operation is achieved by applying erosion and then dilation to the image with the same structuring element. If the vice versa is applied, (dilation first, erosion second) than it is called morphological closing. In this study morphological opening was used.

The benefit of using morphological opening operation in this study can explained as removing the noise that lies between vessel regions without losing the existing vessel data. In other words, it eliminates smaller artefacts than the used structuring element (Sun et al., 2011). This means that if the structuring element is chosen with a bigger diameter than the vessel, it can result in loss of vessel data (Sun et al., 2011). As mentioned before, since erosion is applied first in this operation, noise will be reduced at the cost of losing data at vessel boundaries since they will be thinned out. This behaviour is displayed in figure 3.12 as thinner vessels and less noise.

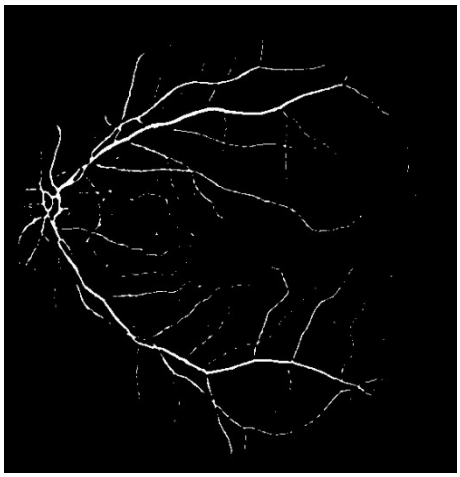


Figure 3.12 Morphologically eroded fundus image after masking

Hence, to have a more accurate image, erosion was followed by dilation resulting in a morphological opening operation. Result is displayed in figure 3.13 it is visible that, both the noises and loss of vessel data are reduced. This is because since the noise artefacts that are removed by erosion are no longer present in the image, the following dilation operation will only dilate or thicken the vessel data and the remaining small amount of noise.

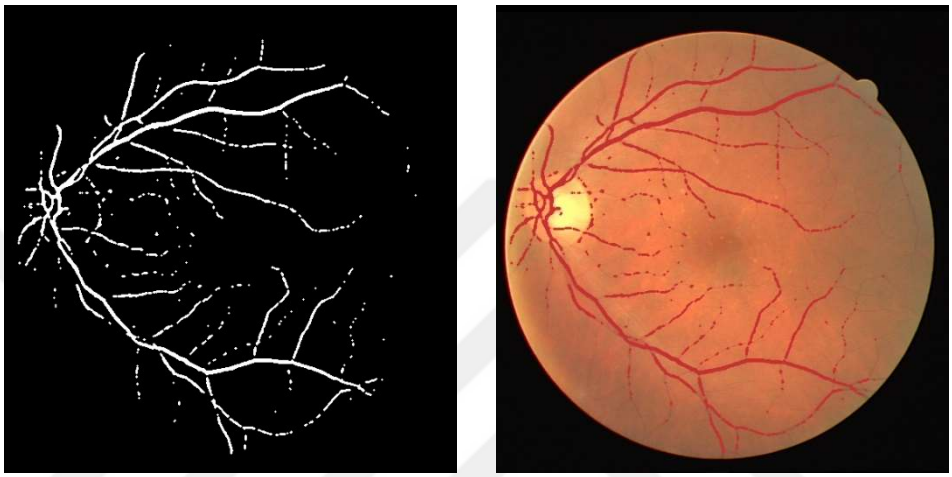


Figure 3.13 Morphologically opened fundus image after masking (left), fundus image after result is merged to the original image (right)

CHAPTER FOUR
APPLICATION

In the previous chapter, theoretical background, and applied methodologies to segment retinal vessels from fundus images were discussed. In this chapter, the implementation will be demonstrated and discussed from a software engineering perspective.

4.1 Selection of the Programming Language

After finding the topic to research and the dataset to work with, the structural analysis of the project started. The first step was to determine the programming language to work with. During this phase 3 programming languages were considered: C, C++, and Python. The C programming language is a general-purpose language, which is considered the grandfather of the popular programming languages of today. It was introduced by Dennis Ritchie in 1970 (Ritchie, 1993). It also persists its popularity for applications where resources such as power, memory, CPU (Central Processing Unit) load are strictly limited. Recently it lost its edge to programming languages like JavaScript, Python and C# as the needs of the modern world developers have changed compared to when C was introduced in the 20th century. It is now the 9th most popular language across projects on GitHub (GitHub, 2023). The C programming language was not designed with an object-oriented mentality, so it does not support classes like the modern languages of today. In the 21st century, developers have access to much more powerful hardware, thus being efficient, while keeping its importance, became optional. C++ is the modern correspondence of C. It takes its name from the increment operator in C, which is used to increment an integer by one. Unlike C, C++ was designed with an object-oriented mindset by Bjarne Stroustrup in early 1980s (Stroustrup, 2020), he even called his invention “C with classes” in the initial phases of development (Wikipedia, 2023). It is also the language the OpenCV framework was introduced in. However, even though it supports object orientation, it is considered harder to learn and usually blamed for having an uglier syntax compared to languages like Python. For this study, Python was chosen as the development language. It was

introduced by Guido van Rossum in 1991 so it is a younger and more modern language. Python is easy to read and learn and has a huge developer community. This results in a lot of library frameworks being available for use, so the developer does not have to reinvent the wheel every time, reducing the development time by a huge amount. It is also supported by many integrated development environments and is easy to measure execution time metrics with the profiler it comes with.

4.2 Selection of the Computer Vision Framework

OpenCV is one of the frameworks mentioned in the previous chapter that are available in Python. It is an Open-Source Computer Vision framework that was developed using C++ (OpenCV Team, 2023). Moreover, it keeps being used and improved by developers across the world in newer versions today. It introduces pre-built functions such as edge detectors, filters, histograms and many more. The user has the freedom to configure the behaviour of the used algorithms within the pre-built functions using the input parameters.

4.3 Design Decisions and Architecture

Within the scope of this study, a file was introduced as a wrapper around OpenCV library to access it easier and keep the project structure clearer (see Appendix 2). These wrapper functions made available were executed in the "main" file with the order described in the Methods chapter with an iterator to loop over each image (see Appendix 1). With each iteration, a performance metric was calculated for each image. Finally, when the iterator reaches the final image, the loop is terminated, and the average performance metric is calculated (see Appendix 3). The details of these metrics will be discussed in upcoming chapters. Moreover, to ease the understanding of the software, classes and variables were defined with camel case and methods or functions were defined with snake case. Using camel case format means that each word used in the variable name starts with a capital letter except for the first word, whereas snake case uses underscores to separate those words.

CHAPTER FIVE
RESULTS

5.1 Execution Time

As mentioned earlier, the built in profiler of the Python programming language was used to time the execution rates of the functions. The profiling measured the execution time as 28.75 seconds (see Appendix 4). However, approximately 13.3 of those seconds were spent for calculating the performance metrics and 7.2 seconds were spent for writing the intermediate image files to the disk after each performed processing step. This means that actual execution time for the proposed algorithm is 8.25 seconds. Considering 20 images from the DRIVE dataset (Niemeijer et al., 2004), are being processed during this period, the processing time for each image can be derived as approximately 0.4125 second. Compared to the study by Hoover et al. (2000) where their proposed algorithm takes 5 minutes to process each retinal image, the proposed algorithm is a lot faster. However, considering the available hardware today compared to available hardware back then, that difference is not surprising.

5.2 Performance Metrics

An important factor when proposing an algorithm for a specific problem is measuring how well it performed. This can be accomplished by calculating the key performance metrics which were also calculated in other studies in the literature to compare the performance of the proposed algorithm against other algorithms. For this study, chosen metrics to evaluate performance were accuracy, precision, recall, f1(dice) score and specificity. These metrics are calculated using 4 essential parameters: True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). Any pixel on the image is categorized as a member of one of these 4 categories. A pixel is categorized as a TP when a pixel is decided to belong to a vessel by both the proposed algorithm and the manually segmented image. Moreover, if the algorithm marked it as a pixel that belongs to a vessel however, on the manually

segmented image it is not marked as a vessel, it is considered a FP. If the pixel is not marked as a pixel that belongs to a vessel by the proposed algorithm however, it is considered to be a part of the vessel in the manually segmented image, then it is considered a FN. Finally, if the pixel is not considered to part of any vessel structure by both the proposed algorithm and the manually segmented image, then it is considered a TN.

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FN} + \mathrm{FP}}$$

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

$$Recall\ (Sensitivity) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

$$Specifity = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

$$F1 - Score = \frac{2\ \times Recall \times Precision}{Recall + Precision} \tag{5.1}$$

Equation 5.1 displays the formulas for the calculation of the used performance metrics. Accuracy is calculated by the sum of TP and TN divided by the number of all the pixels in the image. Recall or sensitivity is the performance of algorithm for marking the vessel pixels (Fraz et al., 2012). Moreover, according to Ghoneim (2019), precision is the rate of correctly classified pixels among all pixels that are marked as a vessel, f1 score also known as dice score is the way balance precision and recall in the system. As visible in equation 5.1, it is calculated by taking the harmonic average of those metrics. Finally, specificity is the metric for measuring the rate of correctly labelled pixels as non-vessel.

Table 5.1 displays the output of the metric calculation function which exports the metrics for each image in the DRIVE database (Niemeijer et al., 2004) into a table. From this table, it can be concluded that the algorithm performs better for the non-vessel regions that it does for the vessel regions. This is due to non-region-based approach that was followed with the algorithm, this makes the algorithm execute faster, however marked vessel performance is not as good as the previous studies.

Table 5.1 Performance Metrics for the Proposed Algorithm

Image ID	*Accuracy*	*Precision*	*Recall*	*F1-Score*	*Specifity*
21	0.9596	0.9022	0.5155	0.6561	0.9955
22	0.9501	0.9368	0.4796	0.6344	0.9968
23	0.9432	0.5698	0.5570	0.5634	0.9704
24	0.9387	0.9586	0.4923	0.6505	0.9972
25	0.9303	0.9592	0.2862	0.4409	0.9987
26	0.9468	0.8971	0.4104	0.5632	0.9957
27	0.9498	0.9048	0.4813	0.6284	0.9951
28	0.9476	0.9239	0.5051	0.6531	0.9955
29	0.9514	0.9097	0.4684	0.6184	0.9957
30	0.9449	0.9548	0.3127	0.4711	0.9987
31	0.9638	0.7810	0.5556	0.6493	0.9900
32	0.9563	0.9283	0.5046	0.6538	0.9965
33	0.9568	0.9196	0.5108	0.6568	0.9961
34	0.9264	0.6621	0.5060	0.5736	0.9720
35	0.9566	0.9119	0.5530	0.6885	0.9949
36	0.9397	0.9239	0.4858	0.6368	0.9951
37	0.9536	0.8795	0.5437	0.6720	0.9929
38	0.9531	0.9011	0.5127	0.6536	0.9947
39	0.9508	0.9053	0.4776	0.6253	0.9953
40	0.9595	0.9070	0.5181	0.6595	0.9956
Average	0.9490	0.8818	0.4838	0.6174	0.9931

CHAPTER SIX
CONCLUSION

Within the scope of this study, studies with different approaches for retinal vessel segmentation were investigated and one approach was proposed. The proposed method involved application of following image processing techniques in the exact order: green channel extraction, smoothing, CLAHE, inverse adaptive mean thresholding, masking, and morphological opening. The main purpose with doing so was to differentiate vessel structures in the retinal images of the DRIVE database (Niemeijer et al., 2004), from non-vessel artefacts. In figure 3.13 the image on the right displays the last output of these steps on an exemplary image. As it can be seen, retinal vessel regions were marked with high accuracy and precision. However, the average recall metric result shows that there can be improvements such as segmenting smaller vessel branches. The improvement can be in the form of designing a new adaptive histogram equalization algorithm to enhance vessel contrast compared to retinal background. Also, statistical analysis tools such as Principal Component Analysis (PCA), can be investigated for improvement of retinal vessel feature detection (Sidhu et al, 2023). Moreover, investigation of transfer-learning based methods, which involve taking a pre-trained model to detect vessel structures and training it on retinal images to specialize in detecting vessels can be a good option. The good performance of the proposed algorithm by Ma et al. (2022) for COVID-19 caused lung infection segmentation can be considered a corroborative example to that statement.

In further studies, the optical disc and other retina regions can be segmented using image processing techniques. The corelation between retinal features and medical data such as: age, blood pressure and disease risk factors can also be further investigated in other studies (Nelson, 2019). Moreover, after the processing steps mentioned in this study are applied, the output can be used as an input to another algorithm which detects ophthalmologic diseases or identifies the user using the biometric data (Fraz et al., 2012). Depending on performance metrics, the best performing algorithm can be chosen and a full system on detecting eye diseases can be developed. However, for this to be possible different datasets with much larger data is required.

REFERENCES

Abdi, H. & Williams, L.J. (2010), Principal component analysis. WIREs Comp Stat, (Vol. 2, pp. 433-459). https://doi.org/10.1002/wics.101

Abdulbaqi, H. R., Jafri, M. Z. M., Omar, A., Mutter, K. N., Abood, L. K., & Mustafa, I. S. (2015)."Segmentation and estimation of brain tumor volume in computed tomography scan images using hidden Markov random field Expectation Maximization algorithm," 2015 IEEE Student Conference on Research and Development (SCOReD), Kuala Lumpur, Malaysia, (pp. 55-60), https://doi.org/10.1109/SCORED.2015.7449396

Abràmoff, M & Kay, C. N. (2013). Image Processing. In Ryan, S. J., Sadda, S. R., Hinton, D. R., Schachat, A. P., Wilkinson, C. W., & Wiedemann, P. (Eds.), Retina (5th ed., pp. 151–176). https://doi.org/10.1016/c2010-1-69566-1

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. 2017 International Conference on Engineering and Technology (ICET) (pp. 1-6). Antalya, Turkey. https://doi.org/10.1109/icengtechnol.2017.8308186

Encyclopedia Britannica. (2019, June 17). ophthalmology. https://www.britannica.com/science/ophthalmology

Encyclopedia Britannica. (n.d.). [Image]. Cross section of the human eye. Retrieved April 23, 2023, from https://www.britannica.com/science/human-eye#/media/1/1688997/100415

Centers for Disease Control and Prevention. (2022, October 17). National Diabetes Statistics Report website. https://www.cdc.gov/diabetes/data/statistics-report/index.html

Chaudhuri, S., Chatterjee, S., Katz, N., Nelson, M. T., & Goldbaum, M. H. (1989). Detection of blood vessels in retinal images using two-dimensional matched filters.

IEEE Transactions on Medical Imaging (Vol. 8(3), pp. 263–269). https://doi.org/10.1109/42.34715

Chudasama, D., Patel, T., Joshi, S., & Prajapati, G. I. (2015). Image Segmentation using Morphological Operations. International Journal of Computer Applications (Vol. 117(18), pp. 16–19). https://doi.org/10.5120/20654-3197

Cicerone, M. T., & Camp, C. V. (2019). Potential Roles for Spectroscopic Coherent Raman Imaging for Histopathology and Biomedicine. In A. Robert & S. Lingyan, (Eds.), Neurophotonics and Biomedical Spectroscopy (pp. 547–570). Elsevier. https://doi.org/10.1016/b978-0-323-48067-3.00021-4

C++. (2023, May 14). In Wikipedia. https://en.wikipedia.org/wiki/C%2B%2B

Comelli, I., Lippi, G., Campana, V., Servadei, F., & Cervellin, G. (2017). Clinical presentation and epidemiology of brain tumors firstly diagnosed in adults in the Emergency Department: a 10-year, single center retrospective study. Annals of translational medicine, 5(13), 269. https://doi.org/10.21037/atm.2017.06.12

Davis, L. S., Rosenfeld, A., & Weszka, J. S. (1975). Region Extraction by Averaging and Thresholding. IEEE Transactions on Systems, Man, and Cybernetics (Vol. SMC-5(3), pp. 383–388). https://doi.org/10.1109/tsmc.1975.5408419

English R., Lebovitz, Y., & Giffin, R. (2010). Institute of Medicine (US) Forum on Drug Discovery, Development, and Translation. Transforming Clinical Research in the United States: Challenges and Opportunities: Workshop Summary. Washington (DC): National Academies Press (US). Available from: https://www.ncbi.nlm.nih.gov/books/NBK50892/ doi: 10.17226/12900

Fraz, M. M., Remagnino, P., Hoppe, A., Uyyanonvara, B., Rudnicka, A. R., Owen, C. J., & Barman, S. (2012). Blood vessel segmentation methodologies in retinal images – A survey. Computer Methods and Programs in Biomedicine, (Vol. 108(1), pp. 407–433). https://doi.org/10.1016/j.cmpb.2012.03.009

GitHub. (2023, March 13). The top programming languages. https://octoverse.github.com/2022/top-programming-languages

Ghoneim, S. (2019, Apr 2). Accuracy, Recall, Precision, F-Score & Specificity, which to optimize on? Towards Data Science - Medium. https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124

Han, S., Kim, K. H., Kim, S., & Youn, Y. H. (2018). Artificial Neural Network: Understanding the Basic Concepts without Mathematics. Dementia and Neurocognitive Disorders, (Vol. 17(3), pp. 83-89). https://doi.org/10.12779/dnd.2018.17.3.83

Hoover, A., Kouznetsova, V. L., & Goldbaum, M. H. (2000). Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response. IEEE Transactions on Medical Imaging (Vol. 19(3), pp. 203–210). https://doi.org/10.1109/42.845178

IBM. (2023, 06 May) What is the k-nearest neighbors algorithm? https://www.ibm.com/topics/knn

Kels, B. D., Grzybowski, A., & Grant-Kels, J. M. (2015). Human ocular anatomy. Clinics in Dermatology (Vol. 33(2), pp. 140–146). https://doi.org/10.1016/j.clindermatol.2014.10.006

Kolb, H. (2005). Simple Anatomy of the Retina. In Kolb, H., Fernandez E., Nelson R. (Eds.), Webvision: The Organization of the Retina and Visual System. University of Utah Health Sciences Center. https://www.ncbi.nlm.nih.gov/books/NBK11533/

Lin, S.C.F., Wong, C.Y., Rahman, M. A., Jiang, G., Liu, S., Kwok, N., Shi, H., Yu, Y., & Wu, T. (2015). Image enhancement using the averaging histogram equalization (AVHEQ) approach for contrast improvement and brightness preservation. Computers & Electrical Engineering (Vol. 46, pp. 356–370). https://doi.org/10.1016/j.compeleceng.2015.06.001

Liu, Y., Li, Z., Xu, C., Li, J., & Liang, R. (2019). Referable diabetic retinopathy identification from eye fundus images with weighted path for convolutional neural

network. Artificial Intelligence in Medicine (Vol. 99, 101694). https://doi.org/10.1016/j.artmed.2019.07.002

Ma, L., Song, S., Guo, L., Tan, W., & Xu, L. (2022). COVID -19 lung infection segmentation from chest CT images based on CAPA-ResUNet. International Journal of Imaging Systems and Technology, (Vol. 33(1), pp. 6–17). https://doi.org/10.1002/ima.22819

Mariño, C., Penedo, G., Penas, M., Carreira, J. A., & Gonzalez, F. J. (2006). Personal authentication using digital retinal images. Pattern Analysis and Applications, 9(1), 21–33. https://doi.org/10.1007/s10044-005-0022-6

Mehdizadeh, M., Tavakoli Tafti, K. & Soltani, P. (2023). Evaluation of histogram equalization and contrast limited adaptive histogram equalization effect on image quality and fractal dimensions of digital periapical radiographs. Oral Radiol (Vol. 39, pp. 418–424). https://doi.org/10.1007/s11282-022-00654-7

Misra, S., & Wu, Y. (2020). Machine learning assisted segmentation of scanning electron microscopy images of organic-rich shales with feature extraction and feature ranking. In S. Misra, H. Li, J. He, (Eds.) Machine Learning for Subsurface Characterization, Gulf Professional Publishing, (pp. 289–314). https://doi.org/10.1016/b978-0-12-817736-5.00010-7

Bernstein, A., Bilheimer, L.T., & Makuc, D.M. (2010). Health, United States, 2009: with special feature on medical technology. https://stacks.cdc.gov/view/cdc/11555

National Eye Institute (NEI). (2022, July 8) Diabetic Retinopathy. https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/diabetic-retinopathy

National Health Service (NHS). (2023, April 30) Angiography. https://www.nhs.uk/conditions/angiography/

Nekovei, R., & Sun, Y. (1995). Back-propagation network and its configuration for blood vessel detection in angiograms. IEEE Transactions on Neural Networks (Vol. 6(1), pp. 64–72). https://doi.org/10.1109/72.363449

Nelson, P. (2019). Deep Learning on Retinal Fundus Images, and Lessons Learned [Video] https://www.youtube.com/watch?v=ViSfhPE6q6Q

Niemeijer, M., Staal, J.J., Ginneken, B.V., Loog, M., & Abramoff, M.D. (2004). Digital Retinal Images for Vessel Extraction: DRIVE database. http://www.isi.uu.nl/research/databases, https://drive.grand-challenge.org

Nichols, J. A., Herbert Chan, H. W., & Baker, M. A. B. (2019). Machine learning: applications of artificial intelligence to imaging and diagnosis. Biophysical reviews (Vol. 11(1), pp. 111–118). https://doi.org/10.1007/s12551-018-0449-9

Novotny, H. R. & Alvis, D. L. (1961). A Method of Photographing Fluorescence in Circulating Blood in the Human Retina. Circulation (Vol. 24 (1), pp. 82-86).

OpenCV Team. (2023, May 12). About - OpenCV. https://opencv.org/about/

Raid, A. M., Khedr, W. M., El-Dosuky, M. A., & Aoud, M. (2014). Image Restoration Based on Morphological Operations. International Journal of Computer Science, Engineering and Information Technology (Vol. 4(3), pp. 9–21). https://doi.org/10.5121/ijcseit.2014.4302

Réda, C., Kaufmann, E., & Delahaye-Duriez, A. (2019). Machine learning applications in drug development. Computational and structural biotechnology journal (Vol. 18, pp. 241–252). https://doi.org/10.1016/j.csbj.2019.12.006

Rehman, A., Saba, T., Harouni, M., Karimi, M., Bahaj, & Awan, M. J. (2022). Microscopic retinal blood vessels detection and segmentation using support vector machine and K-nearest neighbors. Microscopy Research and Technique (Vol. 85(5), pp. 1899–1914). https://doi.org/10.1002/jemt.24051

Ritchie, D. M. (1993). The development of the C language. Sigplan Notices (Vol. 28(3), pp. 201–208). https://doi.org/10.1145/155360.155580

Ronneberger, O., Fischer, P., Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab, N., Hornegger, J., Wells, W., Frangi, A. (Eds.) Medical Image Computing and Computer-Assisted Intervention –

MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science (Vol. 9351, pp 234–241). Springer, Cham. https://doi.org/10.1007/978-3-319-24574-4_28

Ruck, J. (2007). A non-mydriatic Topcon retinal camera [Image]. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:Retinal_camera.jpg

Sarker, I. H. (2021). Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. SN Computer Science, 2(6), 420. https://doi.org/10.1007/s42979-021-00815-1

Schwan, P. (2023, May 27). Devices and Machinery in the Medical Field. Technology: Where it Started and Where it's Going. https://opentextbooks.clemson.edu/sts1010fidlerfall2021/chapter/devices-in-the-medical-field/

Shelhamer, E., Long, J., & Darrell, T. (2014). Fully Convolutional Networks for Semantic Segmentation. ArXiv (Cornell University). https://doi.org/10.48550/arxiv.1605.06211

Sidhu, R., Sachdeva, J., & Katoch, D. (2023). Segmentation of retinal blood vessels by a novel hybrid technique- Principal Component Analysis (PCA) and Contrast Limited Adaptive Histogram Equalization (CLAHE). Microvascular Research, (Vol. 148, 104477). https://doi.org/10.1016/j.mvr.2023.104477

Stroustrup, B. (2020). Thriving in a crowded and changing world: C++ 2006–2020. Proceedings of the ACM on Programming Languages (Vol. 4(HOPL), pp. 1–168). https://doi.org/10.1145/3386320

Sun, K., Chen, Z., Jiang, S., & Wang, Y. (2011) Morphological Multiscale Enhancement, Fuzzy Filter and Watershed for Vascular Tree Extraction in Angiogram. J Med Syst (Vol. 35, pp. 811–824). https://doi.org/10.1007/s10916-010-9466-3

Syeda, H. B., Syed, M., Sexton, K. W., Syed, S., Begum, S., Syed, F., Prior, F., & Yu, F., Jr (2021). Role of Machine Learning Techniques to Tackle the COVID-19

Crisis: Systematic Review. JMIR medical informatics (Vol. 9(1), e23811). https://doi.org/10.2196/23811

Szeliski, R. (2010), Computer Vision: Algorithms and Applications, Springer London.

Tan, K. W., & Isa, N. a. M. (2011). Color image segmentation using histogram thresholding – Fuzzy C-means hybrid approach. Pattern Recognition (Vol. 44(1), pp. 1–15). https://doi.org/10.1016/j.patcog.2010.07.013

Vikram, M. C., Nomula, S., & Srinivas, S. (2022). Blood Vessels Extraction using Green Channel Extraction. 2022 International Conference on Electronics and Renewable Systems (ICEARS) (pp. 996-1000). https://doi.org/10.1109/icears53579.2022.9751954

Vlachos, M., & Dermatas, E. (2010). Multi-scale retinal vessel segmentation using line tracking. Computerized medical imaging and graphics : the official journal of the Computerized Medical Imaging Society (Vol. 34(3), pp. 213–227). https://doi.org/10.1016/j.compmedimag.2009.09.006

Vorontsov, E., Tang, A., Pal, C., & Kadoury, S. (2018). Liver lesion segmentation informed by joint liver segmentation. 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018) (pp. 1332-1335), Washington, DC, USA. https://doi.org/10.1109/isbi.2018.8363817

Wang, Y., Zhang, W., & Fu, W. (2011). Back Propogation(BP)-neural network for tropical cyclone track forecast. 19th International Conference on Geoinformatics, Shanghai, China, 2011, (pp. 1-4)

Yang, S., Oh, J. K., & Park, Y. (2003). Contrast enhancement using histogram equalization with bin underflow and bin overflow. International Conference on Image Processing (pp. I-881). https://doi.org/10.1109/icip.2003.1247104

APPENDIX 1: Content of the main.py file

```
# Retinal Fundus Image Processing - 2023 - Ozan Y.
# Python 3.11
import cProfile, pstats, io
from pstats import SortKey

from retinalImageProcessing import retinalImage
from results import perfMetrics, resultTable

pr = cProfile.Profile()
pr.enable()

perfResults = perfMetrics()
resultTab = resultTable()

firstImgId = 21
lastImgId = 41
imgWritingEnabled = True

for imgId in range(firstImgId,lastImgId):

    rImg = retinalImage(imgId)

    rImg.extract_channel("green")
    if imgWritingEnabled: rImg.write_img("1_gchannel")

    rImg.smoothen(blurType="gaussian", kernel=(5,5),iter=1)
    if imgWritingEnabled: rImg.write_img("2_smooth")

    rImg.clahe(clipLimit=2.0, tileGridSize=(9,9))
    if imgWritingEnabled: rImg.write_img("3_clahe")

    rImg.threshold(threshType="inverse_adaptive_mean",
neighborhoodSize=11)
    if imgWritingEnabled: rImg.write_img("4_thresholded")

    rImg.apply_mask()
    if imgWritingEnabled: rImg.write_img("5_masked")

    rImg.morph_operator("open", sElementType="cross", kernelSize=(3,3))
    if imgWritingEnabled: rImg.write_img("6_morphed")

    rImg.merge_and_display(dispMergedImg=False)
    if imgWritingEnabled: rImg.write_img("7_merged",writeMerged=True)
```

```
    rImg.calculate_match(onlyMasked=False)
    rImg.pm.print_metric("Image " + str(imgId))

    perfResults += rImg.pm
    resultTab.insert_line(rImg.pm, imgId-firstImgId+2, imgId)

perfResults.avg(lastImgId-firstImgId)
resultTab.insert_line(perfResults, lastImgId-firstImgId+2, "Averages:
")
resultTab.close()

pr.disable()
s = io.StringIO()
sortby = SortKey.CUMULATIVE
ps = pstats.Stats(pr, stream=s).sort_stats(sortby)
# ps.print_stats()
# print(s.getvalue())
```

APPENDIX 2: Content of the retinalImageProcessing.py file

```
import cv2
from os import mkdir, path as p
from results import perfMetrics

class retinalImage:
    img = list
    maskImg = list
    labelImg = list
    initialImg = list
    imgHeight = 0
    imgWidth = 0
    matchPercentage = 0

    defaultKernel = (3,3)


    def __init__(self, imgId):
        #Constructor
        self.imgId = imgId
        imgPath = p.join( r"training\images", f"{imgId}_training.tif")
        maskPath = p.join(r"training\mask",
f"{imgId}_training_mask.jpg")
        labelPath = p.join(r"training\1st_manual",
f"{imgId}_manual1.jpg")

        self.img = cv2.imread(imgPath)
        self.maskImg = cv2.imread(maskPath, cv2.IMREAD_GRAYSCALE)
        self.labelImg = cv2.imread(labelPath, cv2.IMREAD_GRAYSCALE)

        if self.img.shape != self.maskImg.shape != self.labelImg.shape:
            raise Exception('DimensionErr', 'Dimension mismatch
between: Mask, original image, labeled image.')

        self.labelImg = cv2.threshold(self.labelImg, 150, 255,
cv2.THRESH_BINARY)
        self.labelImg = self.labelImg[1]

        self.maskImg = cv2.threshold(self.maskImg, 150, 255,
cv2.THRESH_BINARY)
        self.maskImg = self.maskImg[1]

        self.initialImg = self.img
        self.mergedImg = self.initialImg

        self.imgHeight = self.img.shape[0]
```

```
        self.imgWidth = self.img.shape[1]

        self.pm = perfMetrics()


    def extract_channel(self, chVal):
        #Extracts any channel of image

        b, g, r = cv2.split(self.img)
        match chVal:
            case "red":
                self.img = r
            case "green":
                self.img = g
            case "blue":
                self.img = b

    def smoothen(self,
            blurType,
            kernel = defaultKernel,
            borderType = 0,
            iter = 1):
        #Applies Averaging or Gaussian blur (smoothing)

        match blurType:
            case "averaging":
                for _ in range(iter):
                    self.img = cv2.blur(self.img,kernel)

            case "gaussian":
                for _ in range(iter):
                    self.img
=  cv2.GaussianBlur(self.img,kernel,borderType)

    def threshold(self,
                    neighborhoodSize = 20,
                    threshType="binary"):
        #Applies adaptive thresholding

        match threshType:
            case "adaptive_mean":
                self.img = cv2.adaptiveThreshold(self.img, 255,
cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_INV, neighborhoodSize,
10)

            case "inverse_adaptive_mean":
```

```
                self.img = cv2.adaptiveThreshold(self.img, 255,
cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_INV, neighborhoodSize,
10)

            case "adaptive_gaussian":
                self.img =
cv2.adaptiveThreshold(self.img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY,neighborhoodSize,2)

            case "inverse_adaptive_gaussian":
                self.img =
cv2.adaptiveThreshold(self.img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY_INV,neighborhoodSize,2)

            case "binary":
                _, self.img =  cv2.threshold(self.img, 150, 255,
cv2.THRESH_BINARY)

            case "inverse_binary":
                _, self.img =  cv2.threshold(self.img, 155, 255,
cv2.THRESH_BINARY_INV)


    def morph_operator(self,
                       morphType,
                       kernelSize=defaultKernel,
                       sElementType="rectangular",
                       iter = 1):
        #General function for applying morphological operations

        match sElementType:
            case "rectangular":
                structuringElement=cv2.getStructuringElement(cv2.MORPH_
RECT,kernelSize)

            case "cross":
                structuringElement=cv2.getStructuringElement(cv2.MORPH_
CROSS,kernelSize)

            case "ellipse":
                structuringElement=cv2.getStructuringElement(cv2.MORPH_
ELLIPSE,kernelSize)

        for _ in range(iter):
            match morphType:
                case "erode":
```

```
                    self.img = cv2.erode(self.img, structuringElement,
iterations=1)

                case "dilate":
                    self.img = cv2.dilate(self.img, structuringElement,
iterations=1)

                case "open":
                    self.img = cv2.morphologyEx(self.img,
cv2.MORPH_OPEN, structuringElement)

                case "close":
                    self.img = cv2.morphologyEx(self.img,
cv2.MORPH_CLOSE, structuringElement)

    def apply_mask(self):
        # Erodes and applies mask
        se = cv2.getStructuringElement(cv2.MORPH_CROSS,(3,3))
        self.maskImg = cv2.erode(self.maskImg, se, iterations=3)

        outputPath = p.join(r"imgout",f"{self.imgId}")
        outputPath = p.join(outputPath, r"MASK.jpg")

        cv2.imwrite(outputPath, self.maskImg)

        for y in range(0, self.imgHeight):
            for x in range(0, self.imgWidth):
                if self.maskImg[y,x] != 255:
                    self.img[y,x] = 0

    def uniform_hist_eq(self):
        #Uniform Histogram equalization
        self.img = cv2.equalizeHist(self.img)

    def clahe(self, clipLimit=2.0, tileGridSize=(8,8)):
        # Contrast limited histogram equalization
        # clipThr: Contrast limiting threshold
        claheObj = cv2.createCLAHE(clipLimit, tileGridSize)
        self.img = claheObj.apply(self.img)

    def adjust_brightness(self, brightnessValue):
        self.img = self.img + brightnessValue

    def adjust_contrast(self, contrastScaler):
        self.img = self.img*contrastScaler

    def show(self, dImg=None, title="Image View", closeAll = 1):
```

```
        if dImg == None:
            dImg = self.img

        cv2.imshow(title,dImg)
        cv2.waitKey(0) # waits until a key is pressed

        if closeAll:
            cv2.destroyAllWindows() # destroys the window showing image

    def merge_and_display(self, dispMergedImg=False):
        for y in range(0, self.imgHeight):
            for x in range(0, self.imgWidth):
                if self.img[y,x] == 255:
                    self.mergedImg[y,x] = (50,50,200)

        if dispMergedImg:
            self.show(self.mergedImg)

    def calculate_match(self, onlyMasked=False):
        tp = 0 # true positive
        tn = 0 # true negative
        fp = 0 # false positive
        fn = 0 # false negative

        for y in range(0, self.imgHeight):
            for x in range(0, self.imgWidth):
                if self.maskImg[y,x] == 255 or not onlyMasked:
                    if self.img[y,x] == self.labelImg[y,x]:
                        if self.img[y,x] == 0:
                            tn += 1
                        else:
                            tp += 1
                    else:
                        if self.img[y,x] == 0:
                            fn += 1
                        else:
                            fp += 1

        self.pm.accuracy = (tp+tn) / (tp+tn+fn+fp)
        self.pm.precision = tp/(tp+fp)
        self.pm.recall = tp/(tp+fn)
        self.pm.f1score =
2*(self.pm.recall*self.pm.precision)/(self.pm.recall+self.pm.precision)
        self.pm.specifity = tn/(tn+fp)

    def write_img(self, stepName, writeMerged=False):
```

```
        outputPath = p.join(r"imgout",f"{self.imgId}")

        if not p.isdir(outputPath):
            mkdir(outputPath)

        outputPath = p.join(outputPath, f"{stepName}.jpg")

        if writeMerged:
            sts = cv2.imwrite(outputPath, self.mergedImg)
        else:
            sts = cv2.imwrite(outputPath, self.img)

        if not sts:
            print("Error while writing: " + outputPath)
```

APPENDIX 3: Content of the results.py file

```
import xlsxwriter

class perfMetrics:
    accuracy = 0
    precision = 0
    recall = 0
    f1score = 0
    specifity = 0

    def __iadd__(self, pm):
        # Operator overloading for clean metric calculation in main

        self.accuracy += pm.accuracy
        self.precision += pm.precision
        self.recall += pm.recall
        self.f1score += pm.f1score
        self.specifity += pm.specifity

        return self

    def avg(self, nOfImg):
        # For average metric calculation and display

        self.accuracy /= nOfImg
        self.precision /= nOfImg
        self.recall /= nOfImg
        self.f1score /= nOfImg
        self.specifity /= nOfImg

        self.print_metric("*Average")

    def print_metric(self, metricId):
        print('{} metrics -> Accuracy: {:2.3%}\t Precision: {:2.3%}\t
Recall: {:2.3%}\t F1Score: {:2.3%}\t Specifity: {:2.3%}'\
            .format(metricId,
                        self.accuracy,
                            self.precision,
                                self.recall,
                                    self.f1score,
                                        self.specifity))

class resultTable:
    workbook = xlsxwriter.Workbook('imgout/results.xlsx')
```

```
    # displaying 4 fraction digits is sufficient
    number_format = workbook.add_format({'num_format': '0.0000'})
    ws = workbook.add_worksheet()
    ws.write_row('A1', ('Image ID', 'Accuracy', 'Precision','Recall',
'F1-Score', 'Specifity'))

    # each image result printed to a corresponding row
    def insert_line(self, pm, rowIdx, firstCol):

        self.ws.write_row('A'+str(rowIdx),
                          (firstCol,
pm.accuracy,pm.precision,pm.recall,pm.f1score,pm.specifity))

    def close(self):
        # self.ws.autofit()
        self.workbook.close()
```

APPENDIX 4: Profiling result for program execution

13933 function calls (13812 primitive calls) in 28.750 seconds

Ordered by: cumulative time

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
20	13.296	0.665	13.296	0.665	retinalImageProcessing.py:181(calculate_match)
20	7.427	0.371	7.465	0.373	retinalImageProcessing.py:130(apply_mask)
20	7.219	0.361	7.219	0.361	retinalImageProcessing.py:172(merge_and_display)
160	0.529	0.003	0.529	0.003	{imwrite}
140	0.001	0.000	0.508	0.004	retinalImageProcessing.py:207(write_img)
20	0.000	0.000	0.195	0.010	retinalImageProcessing.py:17(__init__)
60	0.191	0.003	0.191	0.003	{imread}
1	0.000	0.000	0.018	0.018	results.py:55(close)
1	0.000	0.000	0.018	0.018	xlsxwriter\workbook.py:309(close)
1	0.000	0.000	0.018	0.018	xlsxwriter\workbook.py:657(_store_workbook)
1	0.000	0.000	0.012	0.012	xlsxwriter\packager.py:132(_create_package)
20	0.000	0.000	0.012	0.001	retinalImageProcessing.py:149(clahe)
20	0.011	0.001	0.011	0.001	{method 'apply' of 'cv2.CLAHE' objects}
20	0.000	0.000	0.010	0.000	retinalImageProcessing.py:46(extract_channel)
20	0.010	0.000	0.010	0.000	{split}
154	0.009	0.000	0.009	0.000	{built-in method nt.stat}
140	0.000	0.000	0.009	0.000	<frozen genericpath>:39(isdir)
20	0.000	0.000	0.008	0.000	retinalImageProcessing.py:74(threshold)
20	0.008	0.000	0.008	0.000	{adaptiveThreshold}
21	0.000	0.000	0.007	0.000	results.py:32(print_metric)
21	0.007	0.000	0.007	0.000	{built-in method builtins.print}
10	0.000	0.000	0.006	0.001	xlsxwriter\packager.py:160(_filename)
20	0.000	0.000	0.005	0.000	retinalImageProcessing.py:58(smoothen)
10	0.000	0.000	0.005	0.001	tempfile.py:305(mkstemp)
20	0.005	0.000	0.005	0.000	{GaussianBlur}
391	0.002	0.000	0.005	0.000	<frozen ntpath>:103(join)
1	0.000	0.000	0.005	0.005	xlsxwriter\packager.py:369(_write_content_types_file)
21	0.000	0.000	0.004	0.000	results.py:50(insert_line)
20	0.000	0.000	0.004	0.000	retinalImageProcessing.py:99(morph_operator)
21	0.001	0.000	0.004	0.000	xlsxwriter\worksheet.py:81(cell_wrapper)
20	0.004	0.000	0.004	0.000	{erode}
20	0.004	0.000	0.004	0.000	{morphologyEx}
10	0.000	0.000	0.003	0.000	tempfile.py:114(_sanitize_params)
10	0.000	0.000	0.003	0.000	tempfile.py:297(gettempdir)
10	0.000	0.000	0.003	0.000	tempfile.py:285(_gettempdir)
1	0.000	0.000	0.003	0.003	tempfile.py:183(_get_default_tempdir)
40	0.003	0.000	0.003	0.000	{threshold}
21	0.000	0.000	0.003	0.000	xlsxwriter\worksheet.py:1406(write_row)
126	0.000	0.000	0.003	0.000	xlsxwriter\worksheet.py:462(_write)