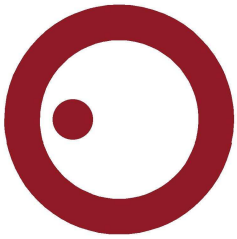


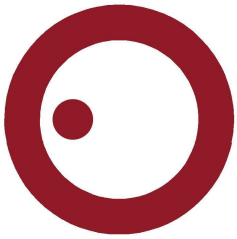
escola superior de tecnologia e gestão
instituto politécnico de leiria

Polytechnic of Leiria
School of Technology and Management
Department of Electrical Engineering
Master's in Electrical and Electronic Engineering

FRAMEWORK FOR LOW-QUALITY RETINAL MOSAICING

BRUNO REIS SILVA

Leiria, 2021, September



escola superior de tecnologia e gestão
instituto politécnico de leiria

Polytechnic of Leiria
School of Technology and Management
Department of Electrical Engineering
Master's in Electrical and Electronic Engineering

FRAMEWORK FOR LOW-QUALITY RETINAL MOSAICING

BRUNO REIS SILVA
Number: 2182734

Dissertation realized under supervision of Doctor Paulo Jorge Simões Coelho (paulo.coelho@ipleiria.pt) and Doctor António Manuel Trigueiros da Silva Cunha (acunha@utad.pt).

2021, September, Leiria

ACKNOWLEDGEMENTS

I cannot express how grateful I am for all the support and encouragement throughout my master's degree and especially in this dissertation, fortunately, I received great support.

I would like to express my very great appreciation to Doctor Paulo Jorge Simões Coelho and to Doctor António Manuel Trigueiros da Silva Cunha for their expert advice and helpful support during the development of this work.

I am very grateful to my parents, brother, and grandfather for their assistance, especially in these different times caused by the COVID-19 pandemic. Without them, this work would not have been possible.

Finally, I also want to thank my colleagues and friends from whom I have always been able to ask for their encouragement and distraction throughout the development of this dissertation.

RESUMO

Os equipamentos médicos utilizados para a captação de imagens do fundo de retina são geralmente dispendiosos. Com o desenvolvimento da tecnologia e o surgimento dos *smartphones*, novas opções de rastreio portátil têm surgido, sendo uma delas o dispositivo D-Eye. Este e outros dispositivos similares, associados a um *smartphone*, comparativamente com os equipamentos especializados, apresentam menor qualidade no vídeo captado da retina, ainda assim com qualidade suficiente para realizar um pré-rastreio médico. A partir deste, caso haja necessidade, os indivíduos poderão ser encaminhados para o rastreio médico especializado no sentido de obter um diagnóstico médico.

Esta dissertação contribui com o desenvolvimento de uma *framework*, que é uma ferramenta que permite agrupar um conjunto de métodos desenvolvidos e explorados, aplicados a vídeos de retinas com baixa qualidade. Foram definidas três áreas de intervenção: a extração das regiões pertinentes em sequências de vídeo; a criação de imagens *mosaicing*, de modo a obter uma imagem de sumário de cada vídeo de retina; desenvolvimento de uma interface gráfica para acomodar as contribuições anteriores.

Para extrair as regiões relevantes destes vídeos (a zona retinal), foram propostos dois métodos, um deles é baseado em abordagens mais clássicas de processamento de imagem como *thresholds* e Hough Circle transform. O outro, realiza a extração da localização da retina aplicando uma rede neuronal, que é um dos métodos reportados na literatura com bons desempenhos para *object detection*, a YOLOv4.

O processo de *mosaicing* foi dividido em duas etapas. Na primeira foi aplicada a rede neuronal GLAMpoints, para a extração de pontos relevantes. A partir destes são efetuadas transformações de forma a ter no mesmo referencial a sobreposição de regiões comuns das imagens. Na segunda etapa foi realizada uma suavização na transição entre imagens.

A interface gráfica foi desenvolvida para englobar todos os métodos anteriormente apresentados de modo a facilitar o acesso e utilização dos mesmos. Para além disso, outras ferramentas foram implementadas como, a comparação de resultados com *ground truth* e exportação de vídeos contendo apenas regiões de interesse.

***Palavras−chave*:** Redes Neuronais Convolucionais, Deteção de objectos, D-Eye, *mosaicing*, *Fundus*, Imagens de retina.

ABSTRACT

The medical equipment used to capture retinal fundus images is generally expensive. With the development of technology and the emergence of smartphones, new portable screening options have emerged, one of them being the D-Eye device. This and other similar devices associated with a smartphone, when compared to specialized equipment, present lower quality in the retinal video captured, yet with sufficient quality to perform a medical pre-screening. From this, if necessary, individuals can be referred for specialized screening, in order to obtain a medical diagnosis.

This dissertation contributes to the development of a framework, which is a tool that allows grouping a set of developed and explored methods, applied to low-quality retinal videos. Three areas of intervention were defined: the extraction of relevant regions in video sequences; creating mosaicing images in order to obtain a summary image of each retinal video; develop of a graphical interface to accommodate the previous contributions.

To extract the relevant regions from these videos (the retinal zone), two methods were proposed, one of them is based on more classical image processing approaches such as thresholds and Hough Circle transform. The other performs the extraction of the retinal location by applying a neural network, which is one of the methods reported in the literature with good performance for object detection, the YOLOv4.

The mosaicing process was divided into two stages; in the first stage, the GLAM-points neural network was applied to extract relevant points. From these, some transformations are carried out to have in the same referential the overlap of common regions of the images. In the second stage, a smoothing process was performed in the transition between images.

A graphical interface was developed to encompass all the above methods to facilitate access to and use of them. In addition, other features were implemented, such as comparing results with ground truth and exporting videos containing only regions of interest.

***Keywords*:** Convolutional Neural Network, Object detection, D-Eye, Mosaicing, Fundus, Retinal images.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AP	Average Precision.
BB	Bounding Box.
BoF	Bag of Freebies.
BoS	Bag of Specials.
CFP	Colored Fundus Photography.
CHT	Circle Hough Transform.
CNN	Convolutional Neural Network.
FA	Fluorescein Angiography.
FAF	Fundus Autofluorescence.
FN	False Negative.
FOV	Field of View.
FP	False Positive.
GT	Ground truth.
GUI	Graphical User Interface.
ICG	Indocyanine Green.
IoU	Intersection over Union.
LIFT	Learned invariant feature transform.
MAE	Mean Absolute Error.
mAP	mean Average Precision.

NMS	Non-maximum suppression.
NN	Neural Network.
OCT	Optical Coherence Tomography.
RoI	Region of Interest.
RPN	Region Proposal Network.
SIFT	Scale-invariant feature transform.
SSD	Single Shot MultiBox Detector.
SURF	Speeded Up Robust Features.
TN	True Negative.
TP	True Positive.
YOLO	You Only Look Once.

1

INTRODUCTION

Fundus photography is a capture of the eye's retina that ophthalmologists widely use to detect ocular diseases. The medical equipment used to capture those photographs of the retina can reach prices of tens of thousands of dollars [1]. Such professional equipment's are used to diagnose eye-related diseases produce high-quality fundus images and due to such quality, these equipment's usage is widespread for medical use.

From another perspective, the lack of mobility conditions in remote areas, the lack of means for healthcare or equipment in areas with limited economic resources, the fundus expensive prices and large dimensions (thus its lack of portability) are some drawbacks to its use and leads to the increasing trend of severity in health-related problems in such populations. Many of the eye-related diseases are degenerative, that is, they progressively aggravate the patient's clinical situation and are irreversible, leading to visual impairments, like blindness. There is the need for early action, which prevents, minimizes, or leads to the search for specialized screening and medical diagnosis, even if using devices that are not as accurate, but nevertheless may provide useful pre-screening information.

There are several commercial, portable, and low-cost devices to acquire fundus images to help and mitigate this screening access issue, namely D-Eye [2], Peek Retina [3] and iNview [4]. In particular, the D-Eye device is coupled to a smartphone and can capture retinal images with sufficient quality for pre-screening. From the resulting data will be possible to observe the need to perform a professional and more accurate screening process, having in mind that the earlier the medical diagnosis is performed, the smaller the associated degenerative impact will be, and thus the greater the associated quality of life.

1.1 OBJECTIVES

This work will contribute to the development of a tool that uses the D-Eye data and aims to contribute to verify their usefulness in the pre-screening task. It encloses

several methods to detect retinal area, methods that merge multiple images into one and to be able to present a summary image to the screener or physician. With those main goals in mind, three sub-objectives were defined:

- Detection of the retina, excluding the non-informative area, in the D-Eye images and compare the results with previous results [5];
- Explore the mosaicing techniques in images captured from devices attached with D-Eye lenses, to provide a summary image of retinal video;
- Contribute to the development of a graphical interface tool that aggregates all the previously mentioned developments, to facilitate further studies, aiming to be applied in the medical environment.

With the contribution in the development of this interface, it's possible to facilitate the medical professional's work analyzing D-Eye captures, since from a D-Eye video and applying the mosaicing technique will be possible to extract a summary image with a greater field of view of the retina.

1.2 DISSERTATION STRUCTURE

This dissertation has six chapters. The first chapter gives an overview of the work specifying the area of intervention, the motivations, and the problem to approach. Besides, presents a description of the main goals and contributions are provided.

The second chapter presents the biological related background, information related to eye anatomy, eye-related diseases, and retinal imaging in general.

The third chapter is about the technical fundamental concepts and the current state-of-the-art. This section of the dissertation presents an overview of machine learning, object detection, and mosaicing techniques. Additionally, presents the state-of-the-art methods and results in each intervention area presented.

The fourth chapter presents the methodology followed to reach the dissertation results. Here is described the pipeline to prepare the data, the methods evaluated (the two retinal detection methods and how mosaicing was implemented), and how the interface was developed.

The fifth chapter discusses the results of the several explored subjects. Firstly, is analysed the results from the proposed method and the YOLOv4 network, when compared to the previous works [5]. Secondly, is presented some visual results of mosaicing and finally is presented the developments in the interface tool.

Finally, the sixth chapter depicts the achievements and the future directions to point for further improvements and developments.

2

BACKGROUND

The current chapter provides a succinct introduction of the biological and eye-related information, that are crucial topics of this work. The first section presents a brief explanation of the human's eye anatomy, being followed in the second section by a concise presentation of some of the most common eye diseases. Finally, the third section depicts about fundus imaging, which are the types of images used as the focus in this dissertation.

2.1 EYE ANATOMY

For humans and vertebrates in general, the vision is extremely relevant as it provides another possibility to extract information from the surrounding environment. While other senses are more focused on gathering short-distance information, vision adds the possibility to acquire long-distance information. Since this dissertation focuses on retinal-based images, it is fundamental to possess solid knowledge about the eye and its constitution.

The following paragraphs provide a simplified description of the eye anatomy. Figure 1 presents a schematic of a human eye's cross-section and its anatomical features that will be described.

In order to produce an image, the light will have to travel to the eyes, and there, it is slowed down, bent, absorbed, and converted into electrical impulses [6]. When the light contacts the eye, the first layer to transpose is a transparent layer, the cornea. This thick structure of the eyeball allows light rays to go through the eye, changing its direction and making it converge inside the eye [6]. It also has a protective function against harmful ultraviolet rays and germs. The amount of light that will pass through the pupil will be regulated with the iris aperture's change, which is a thin pigmented diaphragm. The contraction and extension of the iris will lead to pupil augmentation in low light environments, and the pupil's retraction in high light environments [6], [7]. Then, the light will make its way until it reaches the lens, which is a layer responsible for getting the image focus at different distances,

and also allows the projection of the light into the retina. In the retina (a nervous layer in the internal surface of the eye), the light will be absorbed and turned into electrical impulses and then sent to the visual cortex through the optic nerve [8].

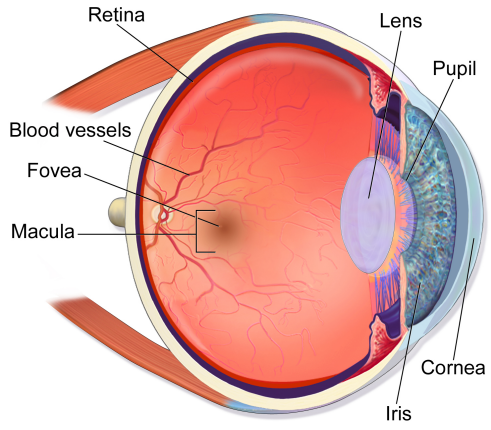


Figure 1: Sectional eye anatomy schematic - source [9].

The retina has two parts that stand out, the macula and the fovea. The macula is the retina's central area [7] and has more photo-sensible cells than the remaining retina. The fovea is the macula's central point where the light rays are focused, due to the different refractive indexes of the cornea, aqueous humor, lens, and vitreous humor [10]. The macula, fovea, and optic nerve are visible in fundus images, as depicted in Figure 2, which will be addressed in the following sections.

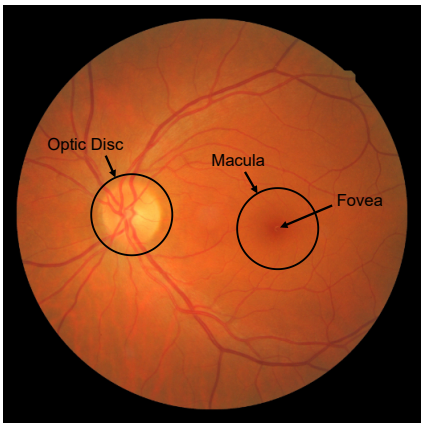


Figure 2: Photography of FIRE public dataset [11].

2.2 EYE DISEASES

Visual perception is one of the most important senses that a living being has. Thus, any chance of loss or damage would be undesirable. Like other organs, there are several eye-related diseases that affect the world perception, which can lead to impairments or lack of health in general, and thus influence well-being. Some of the most common eye-related diseases will be shortly described as follows.

Age-Related Macular Degeneration is a macula's progressive chronic disease that provokes dark patches, shadows, or distortion of the central vision [8], [12]. Aging is a major risk factor for the condition, but other factors such as genetic influences, smoking, and obesity are also risk factors. Globally, it is estimated that 196 million people have this condition worldwide [8]. Furthermore, Diabetes Mellitus leads to diabetic retinopathy. This condition causes the damage of the blood vessels, and the growth of abnormal vessels leading to vision loss [8], [13]. It is estimated that 146 million persons suffer from diabetic retinopathy worldwide [8]. Another condition is glaucoma, which is a neuropathy that causes gradual degeneration of the optic nerve. There are multiple types of glaucoma, but the most common is open-angle glaucoma that provokes peripheric vision loss, progressing to a severe loss of vision [8], [13]. Glaucoma impairment affects about 76 million persons worldwide [8]. In Figure 3 is presented a simulation of the decrease in the visual field of the patients suffering from macular degeneration (top right), diabetic retinopathy (bottom left), and glaucoma (bottom right). On the top left is shown how a healthy person's vision should be.

Figure 3: Eye diseases simulation - adapted from [14]. Healthy field vision (top left). Macular degeneration (top right). Diabetic retinopathy (bottom left). Glaucoma (bottom right).

In general, eye conditions can have diverse origins so the probability of developing them increases with both non-modifiable and modifiable risk factors. Non-modifiable risks are aging, genetics, and ethnicity that cannot be changed. On the other hand, modifiable risks can be life-style related (smoking, occupational and recreational activities) and environment-related (hygiene, access to water, etc.) [8].

Most eye diseases typically can be prevented, delayed, or mitigated with an early screening, diagnosis, and treatment. Thus, eye condition awareness and accessible health care for everybody is an important task to mitigate this problem.

2.3 RETINAL IMAGING

When monitoring the eye's health, it is possible to make different processes to accomplish medical observation, such as retinal photography.

Several kinds of equipment and diagnosis methodologies can provide distinct modes of retinal captures, with their particular qualities/advantages. Some of these diagnosing methods will be depicted in the upcoming paragraphs.

Optical Coherence Tomography (OCT) is a non-invasive technique that allows obtaining micro-scale resolution cross-section retinal images. It has the same concept as ultrasounds but uses rays of light instead of sound waves to obtain the retina's central area thickness, and volume [15], [16]. This technique doesn't need to be performed under mydriasis (artificially dilated pupil using drugs) [15]. Figure 4, shows a representation of the cross-section of the retina using OCT.

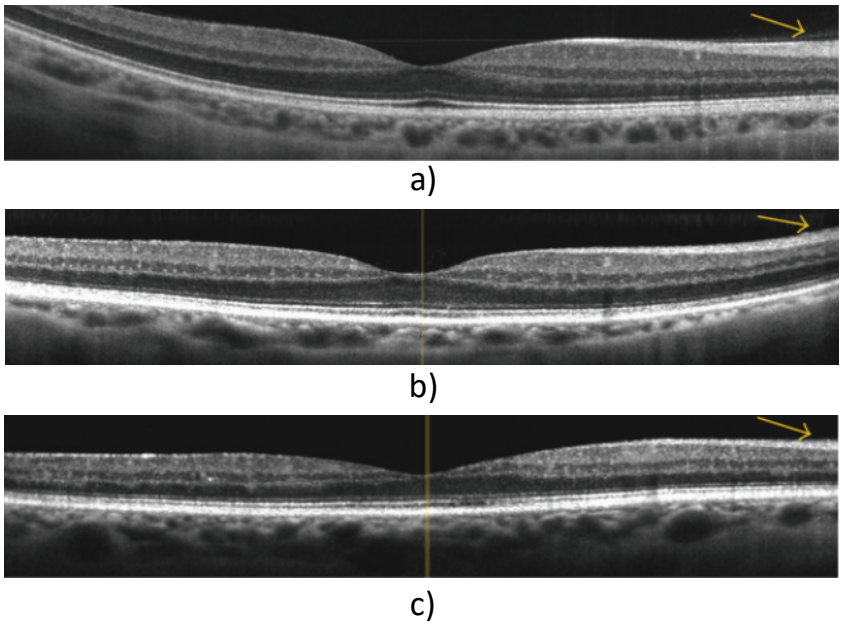


Figure 4: Optical Coherence Tomography (OCT) examples - source [17]. (a) Macula's OCT for a healthy subject; (b) Macula's OCT for a diabetic patient, without retinopathy; (c) Macula's OCT for a diabetic patient with mild non-proliferative diabetic retinopathy.

The yellow arrows presented in Figure 4 highlight the progressive thinning of the retinal nerve fiber layer that occurs when comparing OCT screening data from healthy subjects (Figure 4(a)), with diabetic subjects free from retinopathy condition (Figure 4(b)) or affected by retinopathy (Figure 4(c)).

Fundus Photography is an essential technique for diagnosing and monitoring eye health and can also be used as an educational tool [18]. This approach is painless and consists in capturing the eye's interior surface with specific cameras. There are different types of fundus photography, such as:

- Colored Fundus Photography (CFP) which are colored photographs where the retina's veins can be recognized [18]. Although not mandatory, the eyes can be previously dilated to acquire a wider area in the capture process. Typically, these types of equipment provide images from 30º to 45º Field of View (FOV) [19].
- Fundus Autofluorescence (FAF) utilizes blue-light excitation to produce an image [20]. Due to the retina surface composition, the light is reflected, creating a black and white fundus image.

Each method has its advantages over the others, as example, in Figure 5 it is possible to observe the comparison between a FAF picture (left) and a CFP picture

(right). While in the left image is possible to see details of the Robson-holder ring (the higher reflective area around fovea), in the colored fundus photography is not [20].

- Fluorescein Angiography (FA) is a technique that requires the injection of a fluorescein dye in the patient arm. This dye will rapidly travel through the body and accentuate the retinal veins [21]. Using a fundus camera is possible to record the retinal blood flow and detect conditions, if present, due to the abnormal pattern exhibited in the eye structure.
- Indocyanine Green (ICG) Angiography, is a method that uses an indocyanine Green dye to analyze the choroid's blood flow [21]. When the infrared light hits the dye, it is possible to observe the blood flow in the choroid (the layer of the eye after the retina).

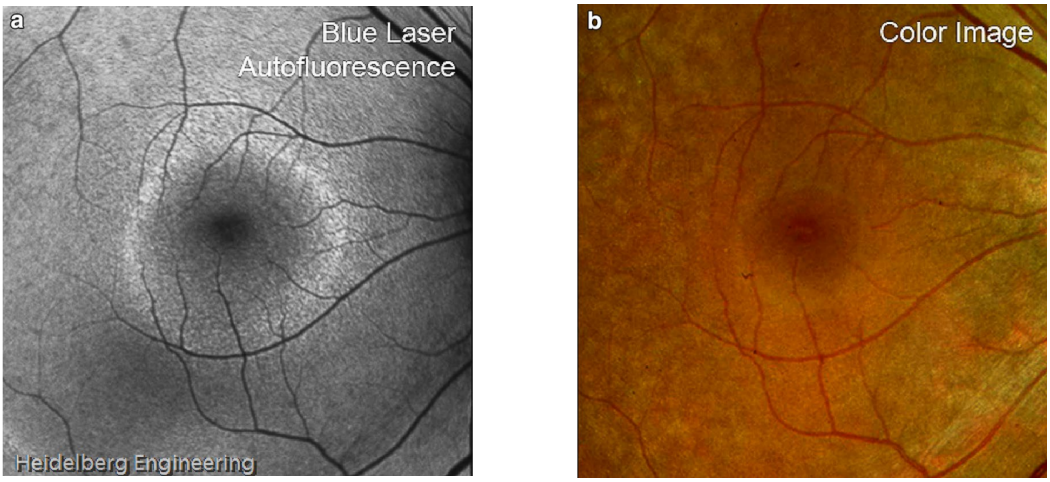


Figure 5: Comparison between Fundus Autofluorescence (left) and Colored Fundus Photography (right) - source [20].

These previously mentioned frames, with the different screening techniques, are acquired from specialized and professional equipment's. As an example, a professional fundus photography machine is presented in Figure 6.

Figure 6: Example of a TRC-50IX fundus professional equipment [22].

These pieces of equipment present as advantages the outputting of high-resolution images, great FOV, etc. However, as disadvantages, they typically present high maintenance and acquisition prices and are not portable devices [1]. Such equipment typically possesses high-resolution cameras, like TRC-50IX or Zeiss ff450plus, which can be costly, with prices that can reach tens of thousands of dollars [1].

Due to health disease risk factors, like inadequate access to nutrition, water, sanitation, and health care, poor and rural unprivileged locations tend to be locations where eye diseases have a higher incidence [8]. The required equipment, for the reasons appointed, is rare and not affordable for these unprivileged populations. Low-cost alternatives have been developed in the last few years to attempt to mitigate this problem. Some alternatives might be the iExaminer, D-Eye, Peek Retina, iNview or 20D Lens [23].

In particular, D-Eye lens [2] is presented in Figure 7 attached to smartphones for fundus capture. This is a low-cost (its price is less than $400 [8]) and portable device, which mitigates some disadvantages of professional equipment mentioned before. An opinion/preference study, made to medical students, revealed 92% use preference of D-Eye lens compared with a direct ophthalmoscope [24]. Nevertheless, against the equipment's reduced price and portability advantages, it presents the drawback of lower resolution images when compared to professional fundus machines.

Figure 7: D-Eye gadget attached to Iphone s and Iphone 5s - source [2].

This dissertation explored data acquisitions from D-Eye devices, due to the availability of a private dataset to conduct the study. An example of D-Eye captured images is presented in Figure 8 (left), and can be compared to a professional equipment image (right).

Figure 8: Photography acquired with D-Eye lens (left) and photography of FIRE public dataset [11] (right).

It is possible to observe in the figure at left that only a small portion of information (retinal area) is important for the medical diagnosis, and the remaining of the image might be disposable when comparing to the image in the right where there's a broader informative area, with better image quality

3

FUNDAMENTAL CONCEPTS AND BIBLIOGRAPHIC REVISION

This chapter presents the fundamentals and technical details related to this work, describing the relevant concepts and bibliographic reviews relevant for this work.

Firstly, the fundamentals of Machine Learning are presented, where it is explained what is a neural network and what the basics of deep learning are. In the second and third sections are presented bibliographic revisions about the main techniques applied in this dissertation: object detection and image mosaicing. To select the most relevant articles in the Google Scholar search engine, the following parameters were used:

- **keywords:** object detection, mosaicing, retinal image, deep learning, survey;
- **time interval:** 2016-2021.

Finally, in the last section is presented a global summary, with some insights of the methods found to be more appropriate to use in this work.

3.1 MACHINE LEARNING FUNDAMENTALS

Usually, a system has input data and some process that transforms the data into output information. In classical programming approaches, the input is usually some data and rules, which are turned into answers. A different approach is used in machine learning. Instead of search for answers, the purpose of this type of algorithm is to obtain a set of rules that could lead to correct answers in a problem with similar input data [25]. Figure 9 presents the different input and outputs between the two approaches previously mentioned.

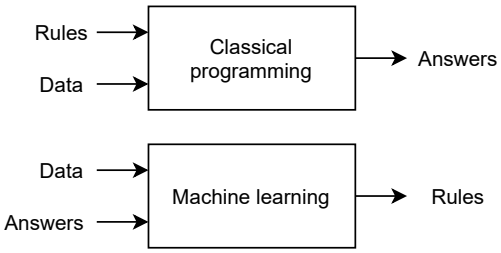


Figure 9: Classical programming and Machine Learning comparison, two different approaches to a problem - source [25].

Neural networks and deep learning are some fields of machine learning that have multiple sub-fields. These concepts have been developed in the past, although their massive use is more recent, due to the increased amount of data available, the increase of computing power, the training algorithms improvement, and the scientific research funding have been some of the reasons [26].

3.1.1 *Neural Networks*

Neural Networks (NNs) are said to be inspired by the way biological neurons send signals from one to another. In the human brain, a huge amount of neurons are connected. In a simplified way, biological neurons receive signals and send their own signal, in response to having received a sufficient number of input signals from other neurons, within a few milliseconds [26].

Neural Networks contain an input layer, an output layer, and one or more hidden layers between them. Each "neuron" of the neural network connects to another and has a weight and threshold (bias) associated, thus it can be compared to linear regression. Figure 10 represents the flow of a general neural network.

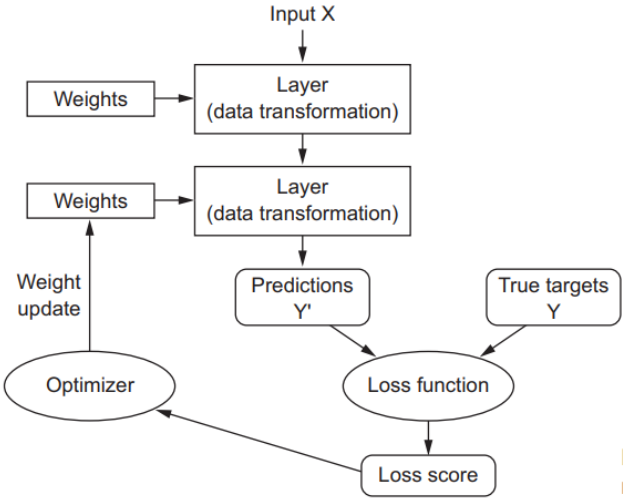


Figure 10: General neural network - source [25].

The input values are multiplied by their respective weights and then summed. After this an activation function is applied to the obtained value and, if the output surpasses the threshold value, it “fires” (or activates) the node, passing data to the next layer.

To adjust the value of the weights is measured how far the output is from the expected using a loss function. This measure is called error and can be obtained with the predictions of the neural net and the true targets from the ground truth. After this, the adjustment of weights is done with an optimizer (for example Gradient Descent), implementing a backpropagation algorithm based on the loss score. Learning means finding the correct value for the weights of the layers.

3.1.2 *Deep Learning*

The "deep" in deep learning means depth of layers in a neural network, *i. e.*, a bigger quantity of layers, which offers better performance on many problems and easier problem-solving since feature engineering is not needed like in previous machine learning techniques [25]. Feature engineering is the process of extract features from data that will be better represent a problem.

Convolutional Neural Networks (CNNs) is one example of deep learning and can be applied to many and generic problems, in particular, images. They can be used to detect patterns in images and interpret them in classification or detection problems for example. In 1998 LeCun publish an article [27] with one of the first convolutional neural networks. This net, LeNet-5, was trained with the MNIST

dataset [27] and was able to recognize handwritten digits. In 2012, CNNs took a big step and started to get very recognized as a useful tool to image processing, when "ImageNet Large Scale Visual Recognition Challenge" was won by a CNN named AlexNet [28].

As previously introduced, the Neural Network (NN) is constituted by different types of layers, such as fully connected layers, activation function layers; however, what turns a NN to a Convolutional Neural Network it is their convolutional layers and pooling layers [26]. A brief introduction to some of the layers of CNNs will be explained below.

In the convolutional layer, a filter slides over the input image detecting patterns and highlighting features, like horizontal or vertical lines that can be used to classify objects. In the first convolutional layers, the network observes the small details and features, increasing them for more complex objects in the following layers, *i. e.*, the deeper in the network, the more complex objects can be detected [26].

After each convolutional layer, typically, appears an activation function. Without this layer, the network would be like a linear classifier since convolutional layers are multiplications. The merge of multiple linear functions will create another linear function (due to their linearity), so activation functions provide non-linearity capabilities to the net [29].

The activation functions permit the network to break this linearity and enable it to solve more complex problems [26]. Figure 11 presents four types of activation functions, namely Step, Sigmoid, Tanh, and ReLU. The step function outputs 0 or 1 values. Since the function contains only flat regions, the Gradient Descent (or any other optimization algorithm for finding a local minimum of a differentiable function) cannot be applied, thus this activation function is not very used due to its incapability to converge [26]. The Sigmoid's function range is between 0 and 1. Sigmoid is not very used also, since it has flat regions that also disable gradient convergence [30]. The hyperbolic tangent (Tanh) function has output values between -1 and 1, and it is preferable to the sigmoid function since it is zero-centered, which helps speed up convergence [26], [30]. The Rectified Linear Unit (ReLU) outputs a threshold at zero for negative inputs and outputs the input values for positive inputs. Compared to the two previous activation functions, is much less expensive to compute and due to its non-saturation form, so the problems with Gradient Descent are reduced [26], [30].

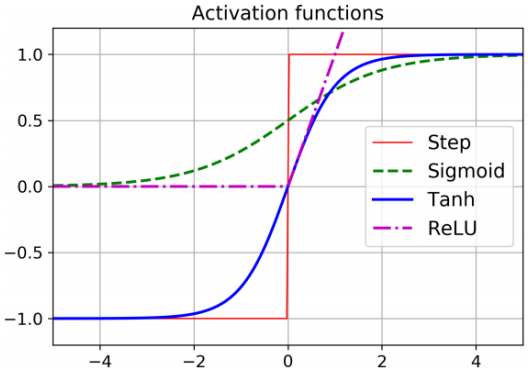


Figure 11: Some example of activation functions - source [26].

From time to time, a pooling layer is introduced between convolutional layers. These are used to reduce the spatial size of data, decrease the needed computational power and the number of parameters [26]. These layers have defined size, stride, and padding types. The most common type of pooling layer used is max-pooling. The max-pooling selects the higher value contained in the kernel to pass it for the next layer of the NN. Figure 12 shows an example of max-pooling where from a 2x2 matrix it is passed a number (the maximum value) to further layers.

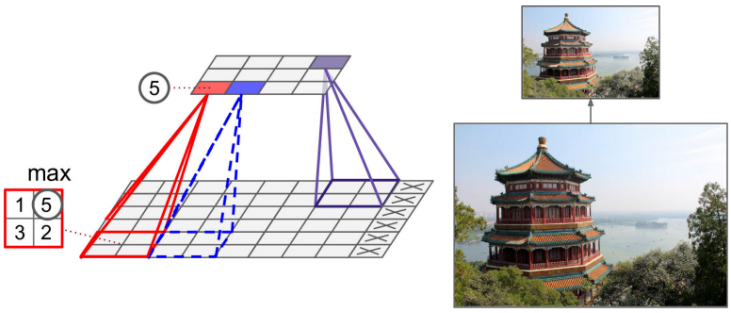


Figure 12: Max pooling layer example with 2×2 pooling kernel, stride 2 and no padding. The kernel marked with red will transform the 2x2 matrix into a single number and pass it to further layers - source [26].

The final part of the network usually is a fully connected layer. It gathers every output of the previous layers and transforms them into a single vector [31]. Many computer vision tasks can be solved using deep learning and some examples are presented in Figure 13, namely semantic segmentation, classification and location, object detection, and instance segmentation tasks.

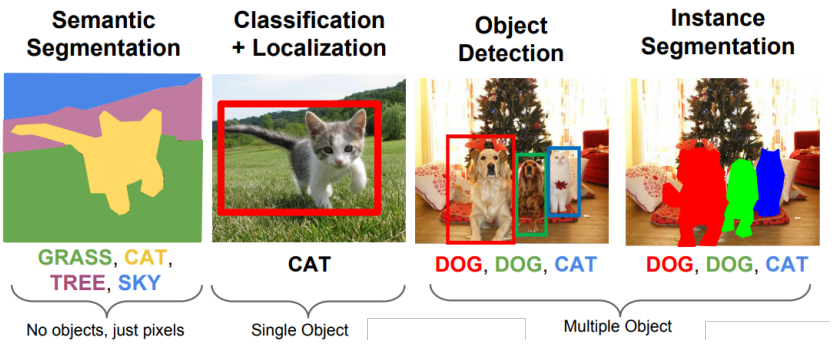


Figure 13: Some relevant computer vision tasks utilized in images - source [32].

In a simplified manner, with semantic segmentation, each individual pixel of the image will be assigned to a class, and so this method does not differentiate objects. In classification and localization tasks it is possible to localize and classify a fixed number of objects, which can be disadvantageous when the number of objects to detect is unknown. For similar tasks are used object detection methods (a similar task to classification and localization), but in this case, the number of objects to detect is flexible. The instance segmentation task is similar to the semantic segmentation task, since the object to detect is delimited by pixels, but in this case, it is possible to assign an individuality to each object.

3.2 OBJECT DETECTION

In the previous section was briefly presented some deep learning techniques used to classify and localize objects present in images or videos. Now, the deep learning object detection task will be presented in more detail.

The main goal of this technique is to determine from a set of classes which ones are present in the input image or video and their location [33]. Furthermore, this task can provide individuality to each object detected instead of assigning it to a class. For every detected object, the algorithm will provide a Bounding Box that will surround the object and identify it with a label of the target class name.

Object detection task has a wide range of applications. Some of them are:

- Autonomous driving - detecting pedestrian, traffic sign and traffic light to alert and assist the driver [34];
- Health-care - assisting doctors at detecting anomalies and diseases [35];

- Online advertisement - detecting in Google Street View publicity panels in order to replace the ads by others [36].

The deep learning object detection tasks are divided into two big groups that will be detailed in the following two subsections [37]; the two-stage detectors and one-stage detectors.

3.2.1 *Two-stage detectors*

The two-stage detectors are object detectors divided into two individual phases, the proposal region step and the classification step. The first step will propose areas of the input image where it is more probable to contain an object. In the second step, the features from the areas selected by the proposal region step are extracted to assign a classification to the found objects. Figure 14 presents the architecture which the generic two-stage detector follows. It is possible to find the region proposal stage (which estimates and proposes the regions with objects), also the Region of Interest (RoI) pooling stage (which warps the aforementioned proposal regions to a usable size, to the network use and classify objects). This kind of detector typically has higher localization and object recognition accuracy but is computationally slower when compared to one-stage detectors [33], [37], which will be described in subsection 3.2.2.

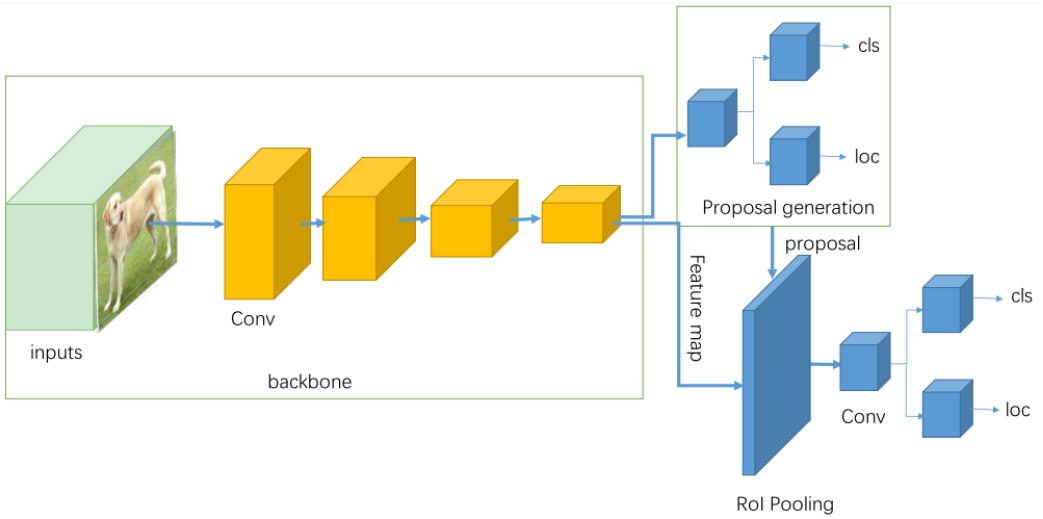


Figure 14: Object detection network: Two stage architecture - source [37].

In the following paragraphs is chronologically exposed three two-stage detectors: R-CNN [38], Fast R-CNN [39] and Faster R-CNN [40]. Figure 15 depicts a better graphical interpretation of each network that will be described.

After the rebirth of Convolutional Neural Networks in 2012 with the AlexNet [28], it didn't take long to try to apply CNNs to object detection tasks. The R-CNN [38] was the first CNN to be applied at object detection tasks [37]. This network can be divided into four main steps. In the first step, a region proposal algorithm will extract areas of the input image where category-independent objects can be potentially found [37]. This step will return 2000 Region of Interest that may contain objects. In the second step, the RoIs obtained in the previous step are cropped and warped in order to all of them have equal dimensions. Each one of these fixed-size regions are fed as input to a convolutional neural network. The third step is where the classification of the objects occurs. Finally, in the last step, the bounding boxes errors, used to surround encountered objects, are corrected.

After one year, Girshick *et al.* improved their network resulting in a new one called Fast R-CNN [39]. In this network, instead of separately process 2000 Region of Interest in the CNN, the Fast R-CNN fed the CNN with the whole image to obtain a feature map of the picture. Then before the first fully connected layer and based on selective search (like in R-CNN), features are extracted from each region with fixed length - this layer is called the Region of Interest (RoI) pooling layer. The warped regions will then be fed to fully connected layers, and their classification scores and bounding boxes offset will be predicted. In Fast R-CNN, instead of having multiple training stages, one for each task, it was created a one-stage end-to-end which enables only one training. Comparing to R-CNN, the training time of the Fast R-CNN is ten times faster. The network cannot run in real-time when in the test mode because of the time that the region proposal step takes to process the image.

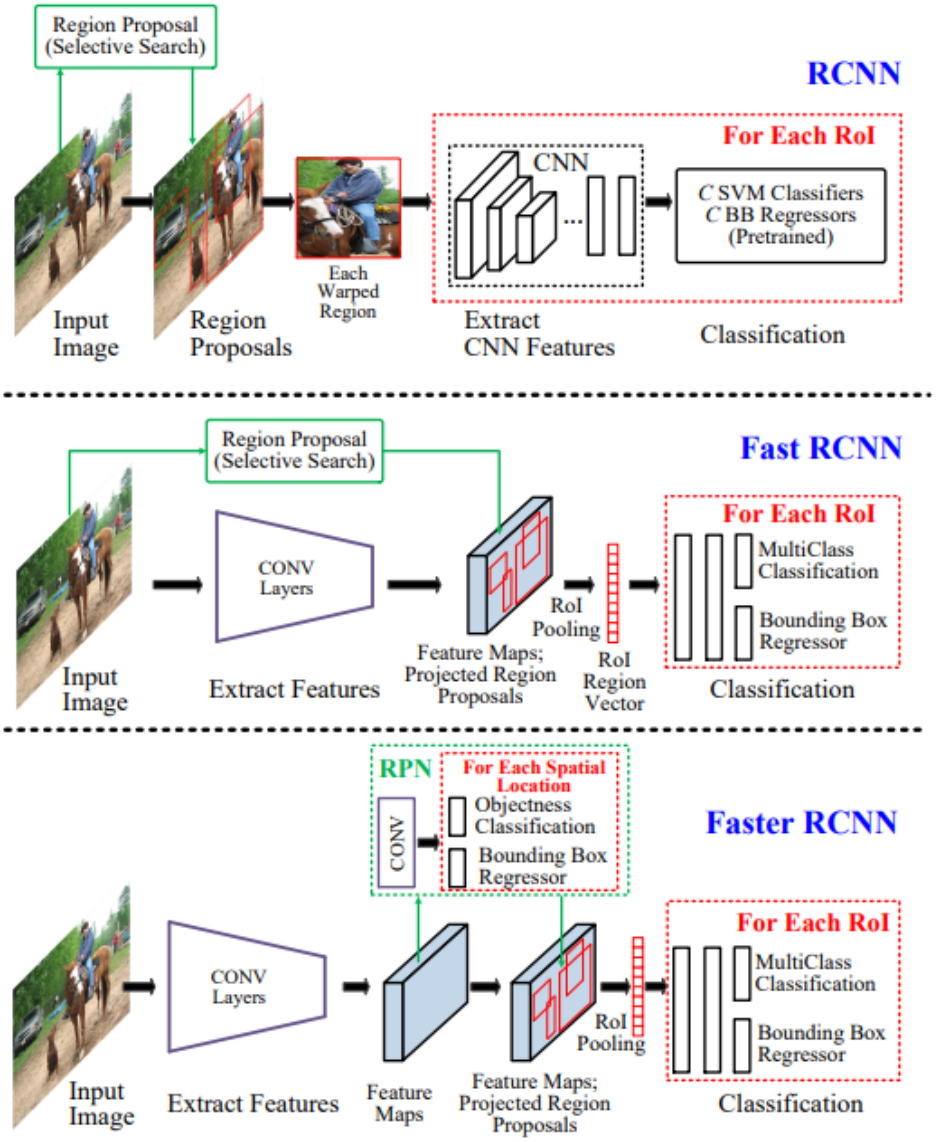


Figure 15: Two-stage networks architectures - adapted from [33].

It only took three months for Faster R-CNN [40] to be presented as an improved version of Fast R-CNN, which shows the constant change and improvement of the deep learning area. Both R-CNN and Fast R-CNN propose the RoIs with the selective search method. The Faster R-CNN proposes to change the traditional method by a Region Proposal Network (RPN) [33], a network that more accurately detect regions that may contain objects. This upgrade made possible real-time object detection.

3.2.2 *One-stage detectors*

Unlike the methods previously presented, the one-stage detectors output results in a single stage, *i. e.*, these do not have the region proposal algorithm step. Due to the fewer stages of this method, this kind of detector usually achieves higher detection speeds when compared to two-stage detectors [37] but has less accuracy detecting smaller-sized objects [33]. Figure 16 presents the architecture of a general one-stage detector.

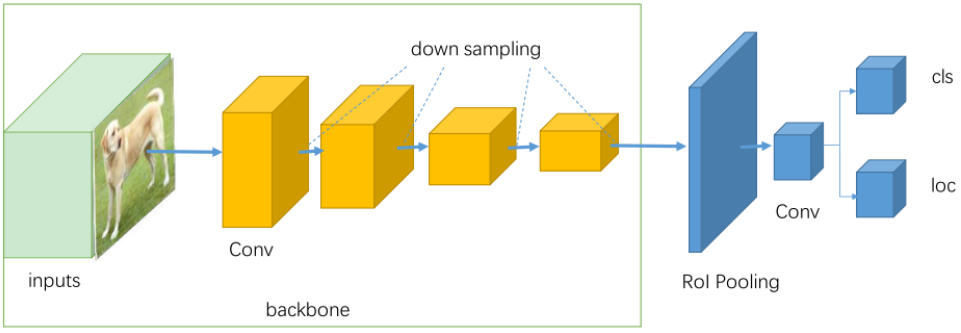


Figure 16: Object detection network: One stage architecture - source [37].

Some examples of networks will now be described, namely the Single Shot MultiBox Detector (SSD) network [41], and the improvements and evolutions of the first three You Only Look Once (YOLO networks [42]–[44], all of them one-stage detectors).

This network uses a stack of multiple scale feature maps to object detection in each one and detects different sized objects. At each feature map, anchor boxes are used (which are a set of predefined bounding boxes with a defined height and width). These are initial guesses of the object locations and are used to calculate more effectively the true box locations.

You Only Look Once (YOLO) [42] is another one-stage architecture used for object detection. It was proposed in 2015 and, just like R-CNN, had multiple versions since then that posteriorly improved its speed and accuracy.

This network divides the image into an SxS grid. In Figure 17 (left) this division is represented with S equal to 3. Every one of these cells will have B predicted Bounding Box (BB); similarly in Figure 17 (middle) each cell has two Bounding Box - B is equal to 2. The limit of each BB can cross the border of the cell as long as its center stays inside the cell. After predicting all the BB, a threshold is used to exclude poorly marked findings. This makes that only predictions with a high

confidence score are not suppressed. Then it is applied a non-max suspension to remove duplicated boxes. Figure 17 (right) shows what should be the final result.

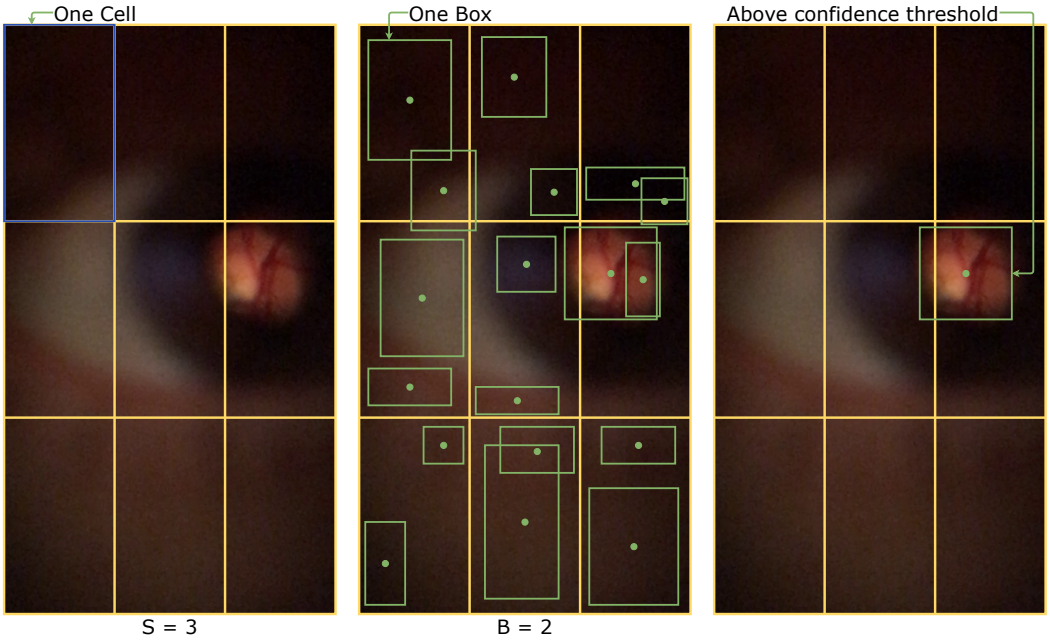


Figure 17: YOLO bounding boxes prediction: Grid division of input (left); Bounding Boxes prediction (middle); Exclusion of low confidence BB (right).

All the encoded BB will have five values in its output: one for confidence score and four numbers to define the bounding box's limits, as depicted in Figure 19. It shows the four values that limit a BB, which are the center coordinates (x and y), width and height. For each cell, the output will also contain a C number of values, that gives the detected object the probability of belonging to each particular class.

Summarily, the output will be a tensor of $S \times S \times ((x,\ y,\ h,\ w,\ pc) \times B + C)$, where:

- $S \times S$ is the number of columns/rows which the image is divided;
- **x, y** are the center coordinates of the Bounding Box;
- **h, w** are the height and width, respectively, of the Bounding Box. These values fluctuate from 0 to 1 as a ratio of the image height or width;
- **pc** is the confidence score, the probability of a BB contains an object;
- **B** is the number of Bounding Box that each cell contains;
- **C** is the number of classes that the model is trained to detect. Will return the probability of each cell contains an object.

Figure 18 shows the architectural differences between YOLOv1 and SSD. While the first only use one feature map to extract the object locations, the second decreases the feature map in size progressively and extracts the object locations in each one.

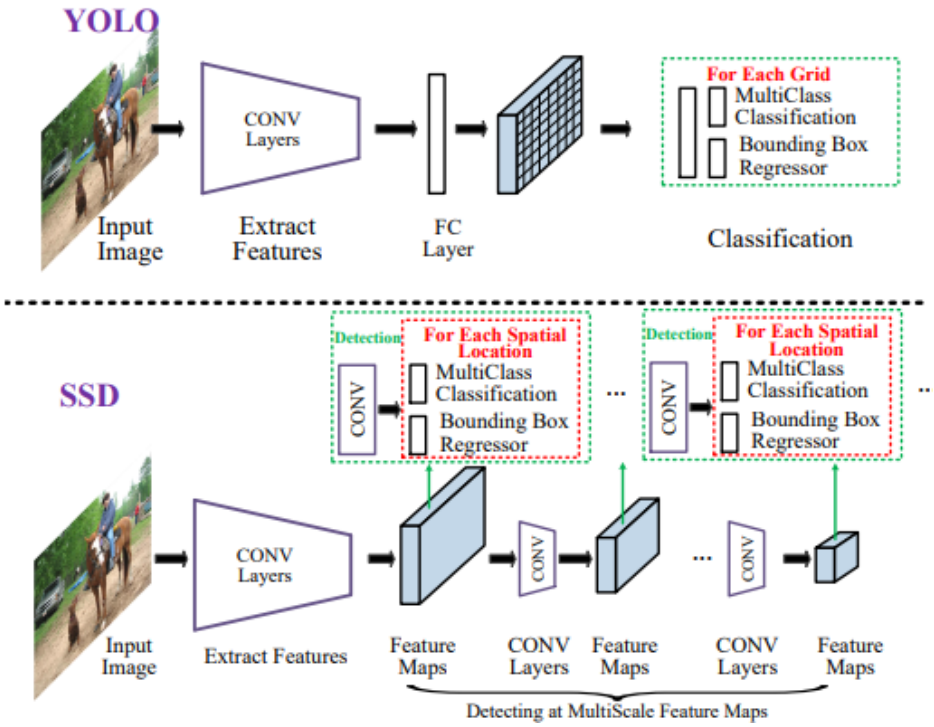


Figure 18: Architecture comparison between YOLO [42] and SSD [41] - adapted from [33].

Briefly, the first version of YOLO [42] has some limitations like high localization error and could only detect SxS number of objects, one for each cell. The second version of YOLO [43] brought solutions to most of these problems, some of the improvements implemented were:

The adding of a **Batch Normalization** [45] layer ahead of each convolutional layer. This implementation brought more than 2% improvement in mean Average Precision (mAP) and increased the speed. In the first YOLO version, it was possible to detect only one object per cell. In the second version, the authors allowed the network to detect more than one object per cell using anchor boxes. With this method, for each BB the network predicts the probability of existing an object (named as classification score or objectness). Thus, for B bounding boxes is possible to detect B objects. Due to this, the output will be a tensor of $S \times S \times (B \times (x,\ y,\ h,\ w,\ pc,\ C))$. In YOLOv2 was implemented **Darknet-19** [43] as a new classification model used as a backbone. It has 19 convolutional layers and 5 max-pooling layers.

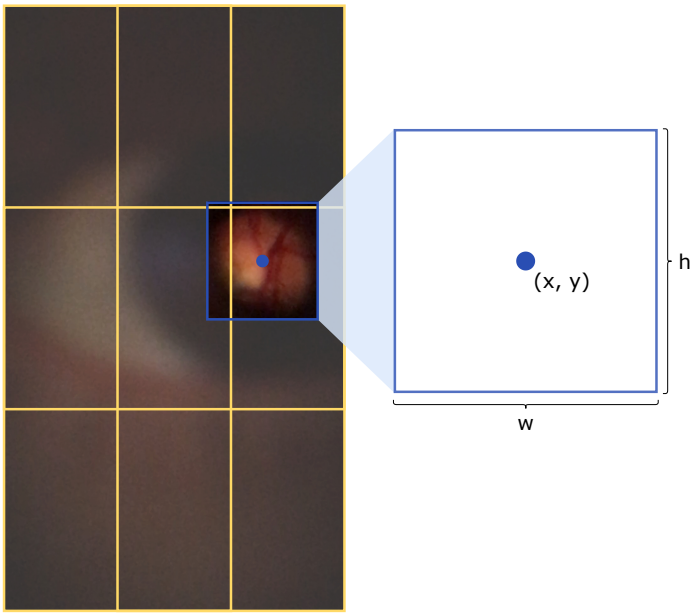


Figure 19: Bounding Box detected by YOLO schematic - adapted from [46]

In the third version of YOLO [44], more improvements were performed, making the network more extensive and accurate. The Darknet-19 of YOLOv2 was replaced by the **Darknet-53** [44], an alternative that brings robustness. This network has 53 convolutional layers, making it more complex.

To have a comparison between all the presented networks, is presented in Table 1 with their highlights and disadvantages.

Table 1: Comparison between deep learning object detectors of two-stage and one-stage architecture: source [33], [34], [37].

Networks	Highlights	Disadvantages
R-CNN [38]	Kick-start of object detectors using deep learning; High performance in comparison to the previous state-of-the-art;	Multistage training; Can't provide real-time detection; Low accuracy in comparison to current state of the art;
Fast R-CNN [39]	First to implement single-stage training; Improvement of R-CNN;	Can't provide real-time detection; Low accuracy in comparison to the current state-of-the-art;
Faster R-CNN [40]	Improvement of Fast R-CNN and turned it to fully CNN	No real-time detection; Low accuracy in comparison to current state-of-the-art;
YOLO [42]	First one-stage architecture; Can provide real-time detection;	The accuracy obtained was worst than the time state-of-the-art; Difficulty detecting small objects;
SSD [41]	Better performance than the first version of YOLO;	Difficulty detecting small objects;
YOLOv2 [43]	Faster and more accurate than its first version;	Difficulty detecting small objects; Low accuracy in comparison to the current state-of-the-art;
YOLOv3 [44]	Better performance than the previous two versions of the YOLO family; Better at small object detection;	Slower than the current state-of-the-art, YOLOv4, described in chapter 4;

3.2.3 *Retinal images application*

In this subsection, it will be analyzed in greater detail some works developed to apply the object detection task to Glaucoma.

In Zengin *et al.* [5], the segmentation of the veins from fundus captured through a smartphone, coupled to a D-Eye device is made. For this purpose, a system was created that can be divided into two blocks, the first in which the location of the position of the retina in the images is detected, and the second in which the image is segmented. To make the first part of the system, the Faster-RCNN object detector was used.

There are also reports on the application of this network as a classification tool. In article [47], the Faster-RCNN network was used to classify colored fundus images the presence or absence of Glaucoma in colored fundus images.

In Thanh *et al.* [48], something similar is done, in which glaucoma detection is verified on colored fundus images. However, instead of using the Faster-RCNN network, the YOLOv3 neural network is used. This enabled the development of a real-time glaucoma screening which was given the name of EyeDr.

3.3 MOSAICING

Mosaicing is a technique used to merge multiple images into a single one [49]. This method is typically divided into two steps, image registration and image blending. Although there is a vast literature about both steps, image registration is considered the fundamental step [50], since without image alignment it is impossible to proceed to the second step. Due to this particularity of mosaicing a greater focus will be applied to image registration description, and image blending will be only briefly described.

3.3.1 *Image Registration*

The first step of mosaicing is image registration. In this step, from two images, the algorithm transforms one to match similarities between them in the same referential. Before the emergence of convolutional neural networks in 2012, image registration was done with traditional feature-based approaches. Typically consisted of finding keypoints (similarities) between near consecutive frames and then describing each

of them using, for example, Scale-invariant feature transform (SIFT) or other methods. Independently of the approach, after detecting and describing each point, the matching between both images was done to obtain the correct correspondence of points. Based on the correspondences, a warp is performed in one of the images to have equal proportions of objects and be placed in the same coordinate system. It is very relevant to extract quality keypoints from the images to produce helpful and meaningful mosaicing, *i. e.*, find enough correspondence points in the input images to merge.

The technique can be classified as unimodal or multimodal, it depends on the number of imaging sources used to perform the merge. For example, if only fundus photography images were used, the method would be considered as uni-modal. However, when fundus and tomography are used simultaneously to obtain more information, the method becomes multi-modal. The transformation of images can be rigid, if exists only rotation and translation transformations, or deformable, if the scale and shear transformations are involved.

In this subsection, are presented and compared some image registration works, which were selected using the following parameters:

- using the keywords "cnn", "image registration", "mosaicing", "retinal images" and "fundus" at databases like Scopus and search engine Google Scholar;
- using the references of the paper GLAMpoints [51] and surveys [52], [53].

The following paragraphs present five networks that perform detection and description of interest points. They can be used jointly with techniques like SIFT or Speeded Up Robust Features (SURF) to produce an image registration algorithm.

DeTone *et al.* [54], present two CNN's, MagicPoint for interest point detection and MagicWarp for homography estimation between two images. These nets were trained in synthetic data. In [55], DeTone *et al.* presents an enhancement of the MagicPoint network and named it SuperPoint. This net can be used to make detection and description of interest points in an image. The net picks a full-sized image and introduces it in an encoder to reduce its spatial size. This resized image is then given to an "Interest Point Decoder" where the "point-ness" of each pixel is calculated and to a "Descriptor Decoder" that gives descriptors of the points as output. The improvement of the net was necessary since the previous net, MagicPoint, performed well at detecting points on synthetic images but had many failures detecting interest points location on natural images. So, DeTone *et al.* added a technique they named Homographic Adaptation that does the transformation to the input image and uses them to train the net.

In those articles were shown interesting ways of pre-train a network if exists a lack of dataset, thus, they used synthetic images for it. Although this characteristic can be advantageous for natural images if the purpose of the work is to process medical images, it may not be the best solution since this type of image has specific features like smooth transactions and textures, which is not observed in a synthetic dataset.

Learned invariant feature transform (LIFT) [56], is a new network architecture that joins three features in one. This work presents a solution that can detect, estimate orientation, and make feature descriptions of interest points, while most of the other works focus only on one or two of these features. Each of these features is a different network. In opposition to SuperPoint, LIFT uses patches of images as input instead of full-sized imaged because only a few parts of the image contain points of interest. The detector is responsible for creating a score map from the input patch (this is the first step and the last feature to be trained). After detecting the point, its orientation is estimated, based on the net presented in [57]. Finally, the point is described based on [58], being the first feature to be trained.

LF-Net [59], like LIFT, can detect, describe and estimate the orientation of interest points. Commonly, to train feature detection networks is used hand-crafted detectors, like SIFT, to find keypoints in the dataset. The authors of this paper [59] preferred to use pairs of images for which is know the depth map. These depth maps are created using laser scanners or shape-from-structure algorithms. Since this type of dataset is difficult to find or obtain for retinal images, this can be considered a disadvantage.

Key.Net[60] is an interest point detector that combines traditional methods and learned CNN features. This combination intends to bring more stability to point detection. To train it was applied transformation on images of ImageNet ILSVRC 2012 dataset [60]. The detector Key.Net was combined with the descriptor HardNet [61] for comparison of matching score and got a better score than LIFT, SuperPoint, and LF-Net previously described. The disadvantage of this net is that it can only perform detection of keypoints, thus, it must be combined with other features to do the image registration complete process.

GLAMpoints [51] is a keypoint detector and uses root-SIFT to do the description of the interest points. An advantage of this network is that it uses unsupervised learning. To do this, a transform is applied for each image of the dataset, obtaining two images and a matrix of homography between them as ground truth. The authors point out as a problem of other detectors the enormous density of keypoints

increasing the false matches, and claim good results when compared to other nets like SuperPoints, LIFT, and LF-Net, which is considered an advantage. Unfortunately, the lack of a public dataset for results comparison and the lack of a metric to know if the output of the net is a good result are points that must be improved from this paper.

A summary of these networks, is presented in Table 2. To perform a complete image registration step, these networks would need an additional algorithm to warp the images based on the detected and described keypoints. GLAMpoints, for example, applies the OpenCV function *findHomography()* to perform the image warping.

Table 2: Comparison between convolutional neural network of features detectors.

Networks	Highlights	Disadvantages
SuperPoint [55]	Detection and description of keypoints in a single network;	Key.Net and GLAMpoints show increased results;
LIFT [56]	Combines in one tool detection, orientation estimation, and feature description;	Key.Net and GLAMpoints show increased results;
LF-Net [59]	Can detect and describe keypoints in one tool;	Key.Net and GLAMpoints show increased results; Needs depth maps as ground truth;
Key.Net [60]	Good performance concerning other detectors; No need of ground truth, transformations are applied to the training sets;	Only focus on detection, needs to be combined with a descriptor;
GLAMpoints [51]	No need for dataset ground truth, transformations are applied to the training sets;Non-Max-Suppression is applied to reduce the density of detected points;	Dataset used for network evaluation is private; Description of the points is made with root-SIFT;

3.3.2 *Image Blending*

After image registration, the overlap of the transformed images needs further attention, due to different intensities of colors or irregularities in the transition between images, the borders where the images meet will need to be smoothed. This step is known as image blending or stitching and can be divided into four classes: feathering-based, pyramid-based, gradient-base, and optimal seam-based blending [50], [62].

The feathering-based method creates one weighted mask for each image. In the overlap area, these masks change their value having lower or higher weight if closer or further to the image to merge [50], [62]. The multiplication between the image and the correspondent mask leads to a blur section in the resulting image. Figure 20 presents a feathering-based method example, where the image A and B are the input images. Image C and D are the generated weighted masks based on distance. Finally, image E is the resulting smoother image.

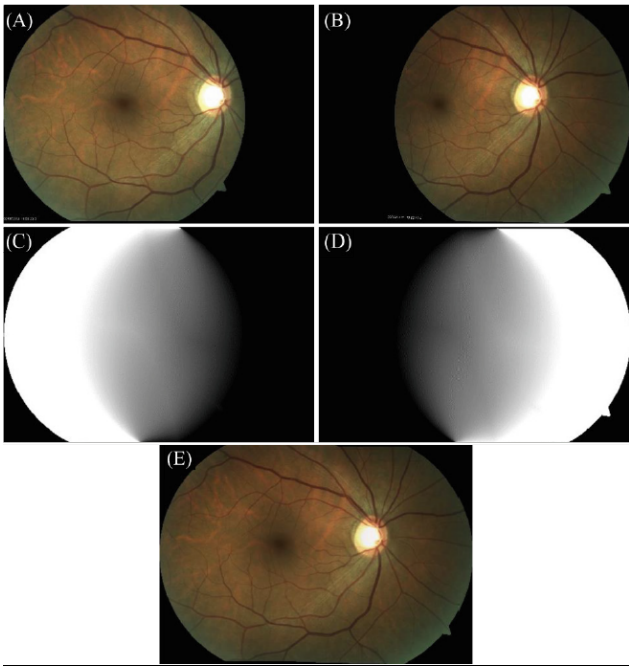


Figure 20: Feathering-based method - source [62].

The pyramid-based method uses Laplacian and Gaussian pyramids of the input images and a Gaussian mask pyramid to get a final result with smooth border transitions [50], [62]. To create the Gaussian pyramids, the input images are firstly blurred (using a Gaussian kernel) and then spatially down-sampled, typically, by order of two. When creating the Gaussian pyramids, the input is blurred to keep

low-frequency information and remove the high-frequency one (edges). To create the Laplacian pyramid is used the Gaussian one. To get a Laplacian of level k, the k-1 level of the Gaussian pyramid needs to be up-sampled and subtracted from the Gaussian level k. The Gaussian mask pyramid is obtained by the same way done for the input images; initially blurring the input and then down-sampling by a factor of two. To obtain the combined Laplacian pyramid of the two inputs is calculated using equation 1. To obtain the final image, each level of the combined Laplacian pyramid is added to the original Gaussian-resized images.

$$CombinedLaplacian_k = Mask_k \times LaplacianA_k + (1 - Mask_k) \times LaplacianB_k \tag{1}$$

The gradient-based method mixes the gradient of the images to obtain a final smoother result [50]. At [63], Levin *et al.* presents two examples of gradient-based blending. The first one aims to create a result minimizing the cost function that measures the gradient difference of the input images. The second approach aims to stitch the derivative input images.

The optimal seam-based blending method crops both images in a path considering the minor changes possible between them [50], [64]. Figure 21 presents, as an example, how an optimal seam-based algorithm would produce a final result between two input images.

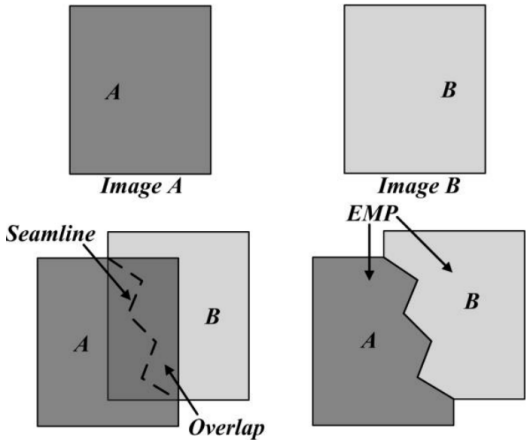


Figure 21: Optimal seam-based method - source [64].

Table 3, presented in [50], shows the advantages and disadvantages of the four types of image blending methods presented previously.

Table 3: Comparison between image blending techniques: source [50].

Category	Highlights	Disadvantages
Feathering-based	Fast and a good performer under exposure differences.	The output often suffers from blur and ghosting effect.
Pyramid-based	Good in preventing blur and edge duplication.	Suffers from double contouring and ghosting when registration error is significant.
Gradient-based	Output visually more appealing than other methods.	The high computation required and registration error must be small for good performance.
Opt. seam-based	Good in dealing with moving objects and parallax.	Transition is obvious when there are exposure differences.

3.3.3 *Retinal images application*

This subsection will present some works in which mosaicing was applied to retinal images. The paper [65] proposes a framework to obtain super-resolved mosaicing from low-resolution images captured by a smartphone. This method firstly selects appropriate views for the mosaicing. From each view is generated one super-resolved image merging multiple images. Then is generated a mosaicing using each of the super-resolved views.

When the fundus are captured, the spherical structure of the retinal surface becomes planar and may causes distortions in the retinal representation. In [66] the merge of the multiple images includes radial distortion correction, in order to obtain a mosaicing with smother transitions between images.

Although not applies specifically to retinal images, Bano *et al.* [67] presents a mosaicing of medical images that can be reproduced with fundus images. In this article, an existing network is presented, but with a training in which a controlled data augmentation is performed, since in the medical images only a small rotation and translation takes place.

3.4 SUMMARY

This work aims to make use of deep learning to maximize the usability of D-Eye captures, and as have been seen in this chapter, there are several ways to approach a problem.

The first objective of this dissertation is to make detection of the retina in the D-Eye private dataset of [5]. Deep learning is not always the optimal solution to every problem, since sometimes other procedures can be faster and less computationally expensive. This way, two methods will be implemented: one with a classical programming approach and another with the current state-of-the-art object detector YOLOv4 network. The YOLOv4 was the chosen network since, compared to all the other networks analyzed, this was the one that stood out the most.

The second objective is to obtain a mosaicing result from the retinas of the D-Eye dataset. As has been analyzed in this chapter this method can be divided into two steps: image registration and image blending. Image registration matches the similarities of multiple images in the same referential. Machine learning was used in this step of mosaicing instead of classical approaches since, from the literature, is expected to have better results in images with fewer key points like medical images. Furthermore, GLAMpoints (the chosen network) uses unsupervised learning, which may be an advantage in a dataset like the aforementioned D-Eye private dataset [5] that is not annotated for mosaicing. For image blending, the second step of mosaicing will be used the feathering-based method, which creates a weighted mask for each image to merge them.

4

METHODOLOGY

This chapter presents the methodology used to implement the three main objectives of this dissertation, which are the development of object detection and mosaicing techniques and, integrating these in a Graphical User Interface (GUI). Figure 22 presents an overview of how each of these methods was implemented and how they were divided into sections to be more easily explained.

As depicted Figure 22, two different methods were developed for retinal detection applied to D-Eye-based images. The first method is based on a classical image processing approach and during this dissertation, it will be named as "Proposed method". The second method for object detection was implemented using a state-of-the-art network, the YOLOv4.

For the mosaicing task were developed two approaches as well. In the first approach (that will be named "Fine-Tuned model") were made a fine-tune of the original GLAMpoint model [51], trained and made available by Truong *et al.* To perform this task, it was used only the retinal area of each frame from the D-Eye dataset (DS1). In the second approach (that will be named "Original model") it was used the original GLAMpoints model to warp cropped images from the D-Eye dataset and to perform the mosaicing task from the videos.

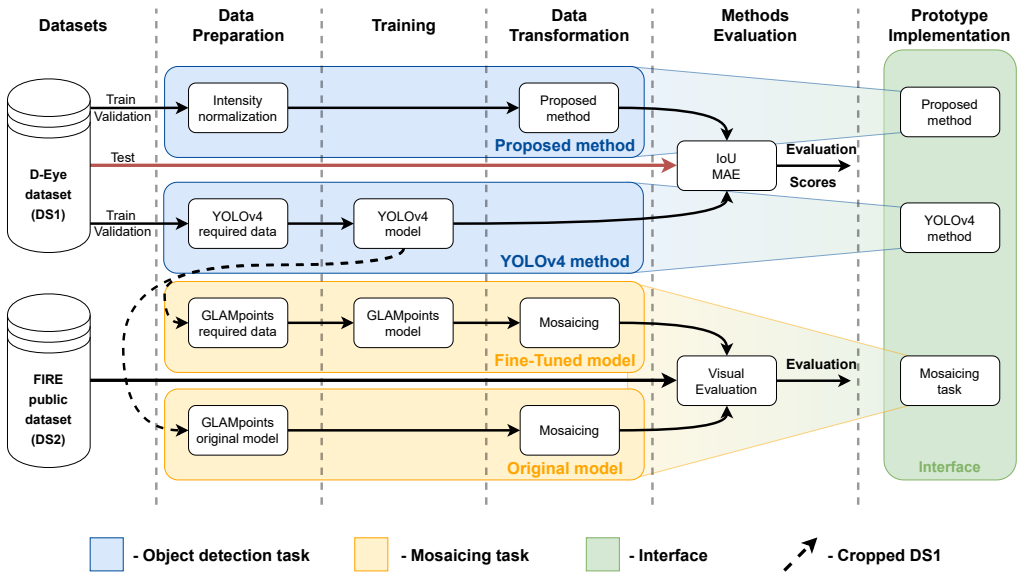


Figure 22: Methodology pipeline.

After introducing these two tasks, is presented the GUI and all of the developed features. This chapter contains six sections, according to Figure 22. The first section describes the two retinal image datasets used. From the second to the fifth sections is presented the Object detection and Mosaicing techniques, according to the vertical subdivisions presented in Figure 22. The sixth section depicts the interface and each feature of the GUI.

4.1 DATASETS

Two retinal datasets were used in the development of this work. One is private, the D-Eye dataset [5], and the other is public, the FIRE dataset [11].

The D-Eye dataset will be named DS1 throughout this document, containing 26 low-resolution retinal videos of the optic disc with a resolution of 1920x1080 pixels, and it is annotated with the localization of the retina visible area. The dataset is divided into folders as stated in Table 4. The DS1 was used in the object detection and mosaicing algorithms analyzed in this work for training and evaluation.

The FIRE dataset [11] will be named DS2 throughout this document, which has 134 image pairs with 45º of FOV and a resolution of 2912x2912. These pairs of images are divided into three categories taking into account their anatomic changes and their level of overlapping. The S category has 71 image pairs with increased overlap and small anatomical changes between images. The P category is constituted

Table 4: D-Eye dataset division in train, validation, and test sets - adapted from [5].

	Resolution (pixels)	Train	Validation	Test
DS1	1920x1080	18 videos; 3881 images	3 videos; 776 images	5 videos; 1375 images

with 49 pairs and the images have reduced overlap when compared to the S category, and keep small changes between images. The last 14 pairs are in the A category. They have increased overlap and significant anatomical changes between images. Table 5 summarizes the previously mentioned category separation.

Table 5: FIRE public dataset division in categories.

	Resolution (pixels)	S category	P category	A category
DS2	2912x2912	71 pairs	49 pairs	14 pairs

The DS2 was applied to in the mosaicing algorithm for the visual evaluation of the GLAMpoints models. Since this model was not used for any NN model training, the folder division into train, validation, and test was not necessary with this dataset.

4.2 DATA PREPARATION

The data preparation will be the same for this work as the *mise en place* is for culinary since here all the required data will be organized and prepared before starting to cook, *i. e.*, start to implement the developed methods.

4.2.1 *Object detection*

The two techniques implemented for retinal localization detection are quite different from each other, resulting in, different requirements in terms of data preparation for each one. For the object detection task, it was only used the dataset DS1.

4.2.1.1 *Proposed method*

Due to the non-constant color intensity of the DS1 images, the data preparation of the proposed method is to normalize the intensity of all the used images. This

process has the objective of preparing and transforming the data for a better overall result. Based on [62] approach, this problem's correction was made by performing the normalization of all the datasets image mean intensity. The normalization of each image is describe with the Equation 2. To perform the normalization task, firstly, is calculated the mean intensity of each analyzed image, I_{Xmean}. Then, the lowest mean of pixel intensities is found, I_{min}. That value is divided by each image's mean pixel intensity, obtaining a list of ratio values. Each of the ratio values is then multiplied by its correspondent image (Fig_X in the equation), obtaining an normalized image (Fig_{Xnorm}). This process changes the intensity of all images to the same value in every frame. This task can take a while to complete if a substantial amount of images are being processed, nevertheless bringing better results, as will be presented in chapter 5.

$$Fig_{Xnorm} = \frac{I_{min}}{I_{Xmean}} Fig_X \tag{2}$$

4.2.1.2 *YOLOv4 network*

Unlike the proposed method, neural networks are robust to image intensity changes. If the model is well trained, the expected results will be typically better than those based on image transformations. For a custom-trained model, some specific data preparation is needed and will framework-dependent.

To train the YOLOv4 model it was applied the network implemented with the Darknet framework [68], which is required, a labeled dataset and three files - one *.cfg* file, one *.data* file, and one *.names* file, whose content and usefulness will be described a little further down in this chapter.

The dataset DS2 needs to be divided into three folders (train, validation, and test) like the DS1 is. Each image inside train and validation folders should have a correspondent annotation *.txt* file with the same name as the image. The annotation files have five numbers, one for the object's class and another four for the bounding boxes that surround the object. If the image has multiple objects in it, each line will correspond to one annotation. This type of annotation is called YOLO annotation and has the following format:

<class id> <X> <Y> <W> <H>

Where:

- **class id** is the label index of the class to be annotated.

- **X** is the ratio between the X center coordinate of the bounding box and the width of the used image.
- **Y** is the ratio between the Y center coordinate of the bounding box and the height of the used image.
- **W** is the ratio between the width of the bounding box and the width of the used image.
- **H** is the ratio between the height of the bounding box and the height of the used image.

The D-Eye dataset has every annotation in a single excel file and it uses the annotation format Tensorflow Object Detection CSV, where each line represents one retina location with the x top left corner, y top left corner, x bottom right corner, and y bottom right corner coordinates of the bounding box. This way, to be able to apply the D-Eye dataset to YOLO, each line of the excel annotation file must be converted to a *.txt* file matching the YOLO annotations format.

For the YOLO network to correctly work, the original configuration (*.cfg*) file, provided by the author, needs to be updated. All the changes were done based on AlexeyAB [69] recommendations. Firstly, the batch parameter was defined as 64 and the subdivisions as 16. Width and height values were defined with the standard value of 416, a standard used value [69]. The remaining changes were done based on the number of object classes necessary to detect. In this case, it is intended to find only one object, the retina, so the classes variable was defined as 1. In the three convolutional layers section, before the YOLO layers, the max_batches, steps, and filters values were changed. The max_batches value is defined by multiplying the number of classes by 2000, but this value must never be less than 6000. This way, it was defined as 6000. The steps parameter has two values, the lower must be equivalent to 80% of max_batches and higher must be equivalent to 90% of max_batches. Then, they were defined as 4800 and 5400. The filters value is obtained by multiplying the number of classes plus five by three. The value of filters for one class is 18 [69].

The *.names* file has the names of the classes, and each line is a class name. In this specific case was only used one class and was named eye.

The *.data* file indicates the number of classes, in this case, one class, and the important paths, such as the one for the train folder, the validation folder, the *.names* file, and the folder where the model will be saved.

The version of YOLOv4 implemented in TensorFlow at this present moment is not reproducing the full performance of the Darknet framework [68]. It is possible to convert the Darknet's trained weights to use in other frameworks like Tensorflow. To achieve this, the YOLO authors recommend training it in Darknet and then converting the *.weights* to TensorFlow or TFlite with the GitHub available converter algorithm [70].

4.2.2 *Mosaicing*

The mosaicing task will use the two datasets, DS1 and DS2, to implement the methodology described in this chapter. To fine-tune the GLAMpoints original model, it was used only the retinal area of the DS1 images. To identify and crop the retinal areas from all the images of DS1, the YOLOv4 network was used. The images were cropped and padding was added to all the images have the same spatial size of 400x400 pixels, keeping the retina's original size and scale (corresponds to the Cropped DS1 in Figure 22). Figure 23 presents an example of which information was considered important from D-Eye images to perform the mosaicing task.

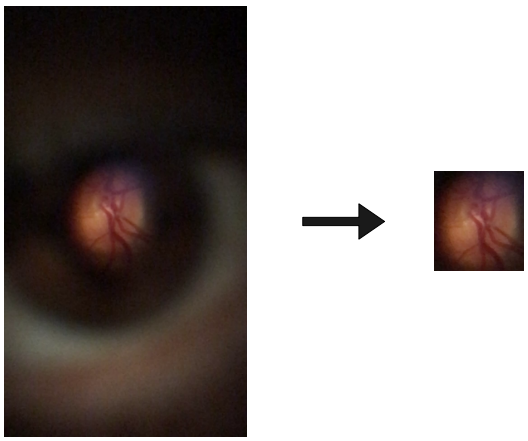


Figure 23: Example of how the D-Eye images were cropped. At left is presented a D-Eye image and at right is presented the result after data preparation.

The dataset DS2 was resized to 15% of its original size, similarly to what was done in the Truong *et al.* [51] work, for increased computational performance. As result, all the images were resized to the size of 436x436 pixels. This last dataset was only used for the visual evaluation of the mosaicing models.

4.3 TRAINING

This section explains how the network models training was performed. Two models were trained, one of the YOLOv4 network, for the object detection task, and the other of the GLAMpoints network, for keypoint detection to use in a mosaicing technique.

In the fourth version, YOLO was improved to a more dense network. It was used two main approaches to obtain better accuracy results: Bag of Freebies (BoF) and Bag of Specials (BoS) [68]. BoF is a method that increases the training cost to get a better training strategy. Data augmentation, applying transformations to the training images, is an example of BoF. In YOLOv4, BoF was applied to the Backbone and the Detector of the network. BoS are plugin modules and post-processing modules whose application increases the accuracy of the network. Like BoF, BoS was applied to the Backbone and the Detector of the network.

Google Colab and Google drive were the tools used to train the network. The image dataset, ground truths, and configuration files were all uploaded to Google Drive. Then was installed on Colab the OpenCV, CUDA, CUDNN, and Darknet. The YOLOv4 model was trained over the COCO dataset's pre-trained weights, which can predict 80 classes [71].

After the training in Google Colab, the models were automatically saved to Google Drive and obtained the following chart that shows the average loss versus iterations, Figure 24. As can be seen, Figure 25 after the 1000 iterations reached the 100% mAP. The training of the model was stopped after 3300 iterations.

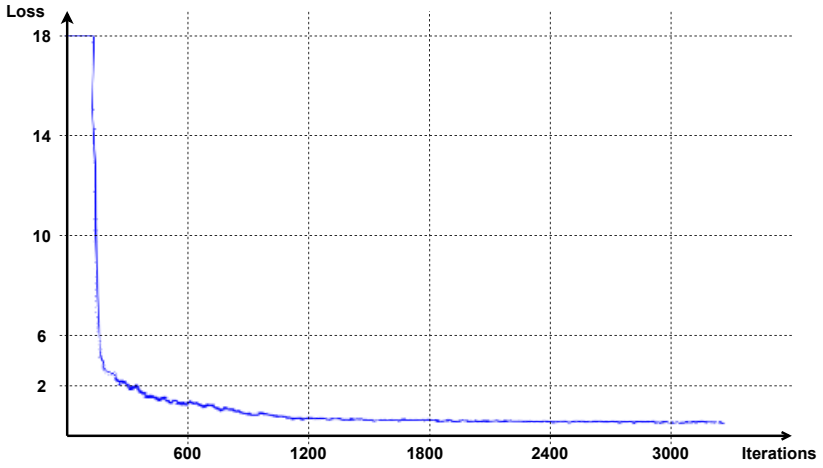


Figure 24: Chart of loss from YOLOv4 model training.

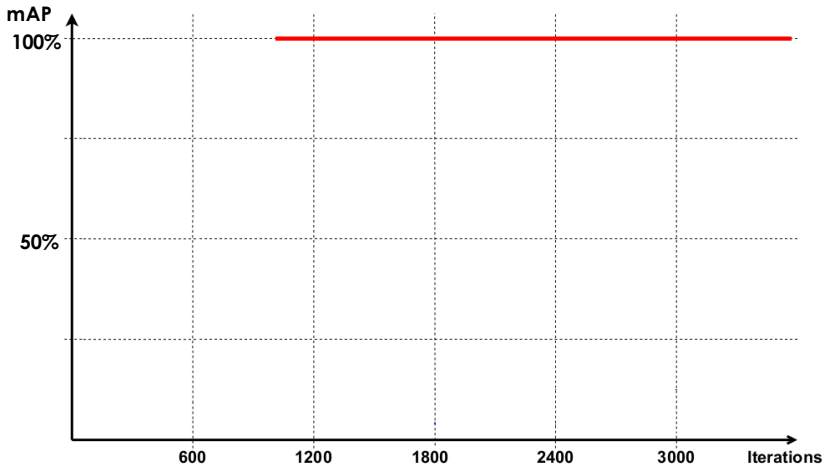


Figure 25: Chart of mAP from YOLOv4 model training.

In the Mosaicing task, like what happened with the YOLOv4 network, the GLAMpoints network was trained using the Google Colab. The original model provided by Truong *et al.* was fine-tuned with the retinal areas of the DS1. To obtain the retinal area, the YOLOv4 previously trained model was used to get the BB coordinates that encapsulate the retina. Then those coordinates were used to crop each one of the DS1 images, creating a new dataset from DS1 only with the low-resolution retinal area (Cropped DS1 in Figure 22).

The network was trained multiple times and for training results comparison was used the tool WandB [72], which allows the tracking of the training data results. As D-Eye images have fewer frame-to-frame transformations, it won't be necessary to perform high demanding training (with high transformation values). This way, several training sessions were carried out with controlled data augmentation, activating and deactivating the parameters presented in Table 6. Other parameters were kept constant like for example the image size of 256x256 pixels, the learning rate of 0.001, and the epochs number of 14.

With the reduction in the use of data augmentation parameters, there is less transformation in the images and it will be easier to perform the homography in the training process. This is reflected in the loss graphs shown in Figure 26 where training with fewer transformations presents a greater convergence than those using more transformations.

Table 6: Multiple parameters variations in GLAMpoints network training.

Name	Use green channel	Use rotation	Use scaling	Use perspective	Use shearing
Train 1	no	yes	yes	yes	yes
Train 2	no	yes	yes	yes	yes
Train 3	no	no	yes	yes	yes
Train 4	no	no	no	yes	yes
Train 5	no	no	no	no	yes
Train 6	no	no	yes	yes	no
Train 7	yes	no	no	yes	yes

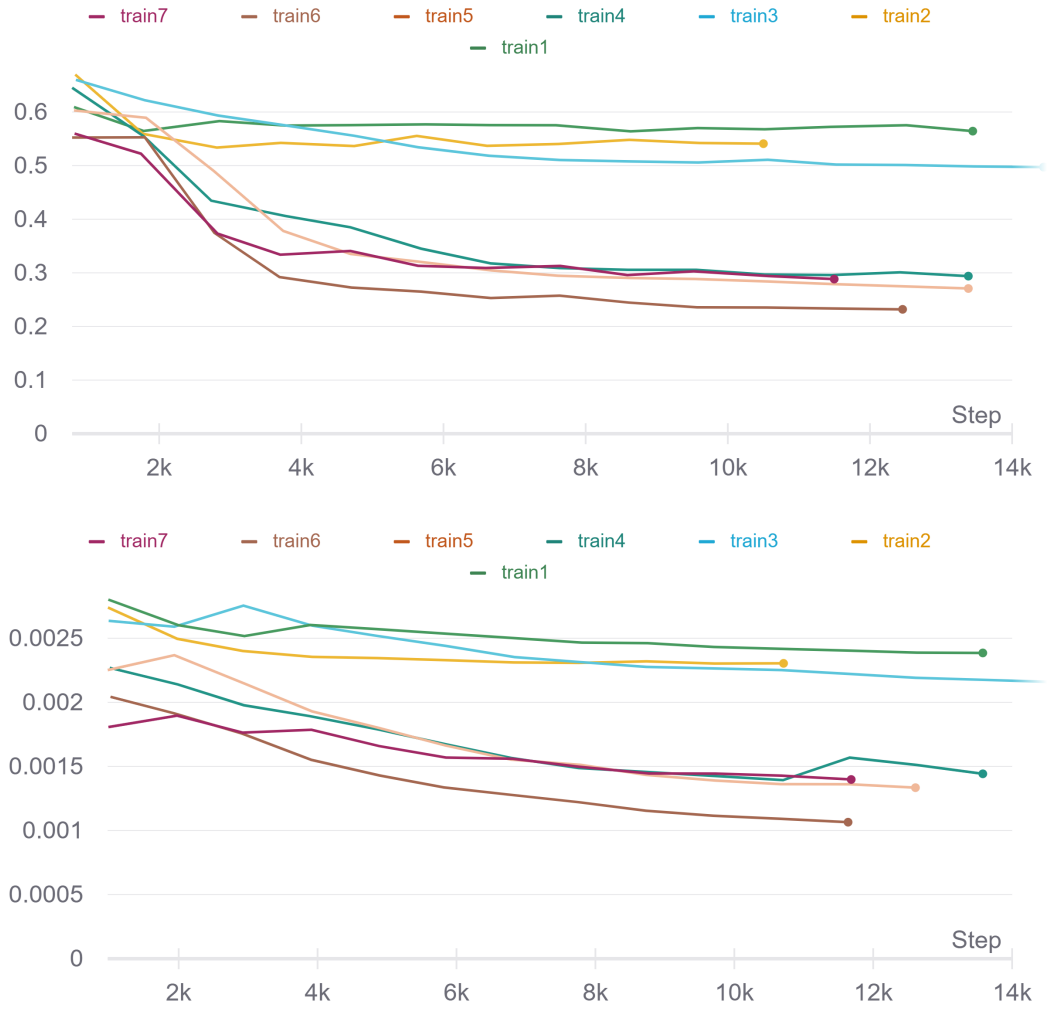


Figure 26: Train loss graphic (top) and validation loss graphic (bottom).

4.4 DATA TRANSFORMATION

This stage of the methodology is reserved to explain the additional transformations of the data before the evaluation step. After this step is expected that the object detection task outputs the retinal location and that the mosaicing task to have an image.

4.4.1 *Object detection*

In the data transformation step of the Proposed method, the objective is to extract the location of the retina from the normalized images of the DS1. To extract the retina location and since the retina has a circular shape in the images, it was decided to apply the Circle Hough Transform (CHT) [73] algorithm to the images.

The CHT finds circles centers and radius from edged (contour) images like left image in Figure 27. For each black pixel on the contour image, the algorithm will draw a circle with a defined radius in a matrix named the accumulator. In Figure 27, in the middle and on the right pictures are shown the results of this task for two different radii, 20 and 25 pixels. The agglomeration of circles over each other in the accumulator will lead to the highlight of the center of a circle in the original edged image if the radius corresponds to the correct value, like Figure 27 (left) for a radius of 20 pixels and Figure 27 (right) for a radius of 25 pixels. Different sized circles will need different accumulators to be detected (since they have a different radius).

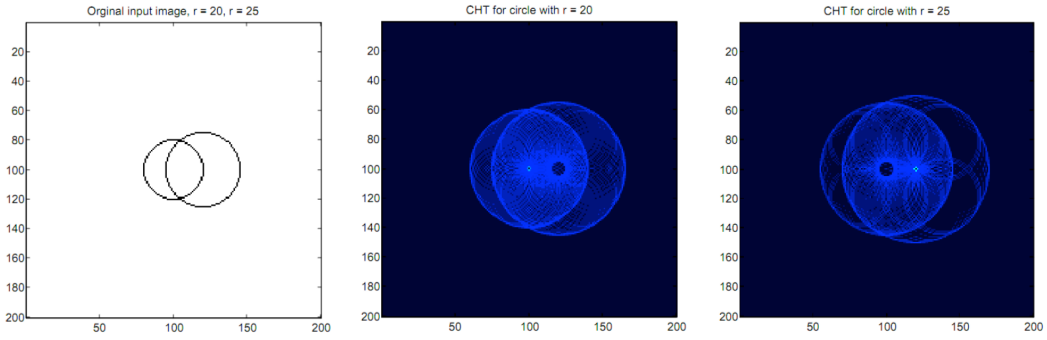


Figure 27: An example showing the accumulator of the CHT for two different ratios - source [74]. The input image (left) obtained an accumulator result of the middle if the radius is 20 pixels and the right image result if the radius is 25 pixels.

This way, in order to find the location of the retina with the CHT, is needed to transform the DS1 images into contour images with highlighted retina area.

In the first step, the extraction of the red channel in the RGB color space was performed, with the intention to reduce information. The image is then blurred in the second step before applying the threshold to the image in order to have an image with smother edges and later get better results after apply CHT. The Otsu method [75] was chosen as the threshold tool in the third step since it is an adaptive threshold. To be adaptive means that a different threshold value will be automatically calculated depending on the image profile. The result of this step is a binary image, as presented in Figure 28 (middle). Afterward, the binary image is transformed into contours in step four. In step five, the contours are transformed into double contours for better performance of the next step. With the double contour it will be more easy for CHT algorithm to later find circles. The generated image is binary and has a white background and black double lines in it. The lines in the image should be marking the retina region with a well-defined circle and the other shapes should be discarded. Something similar to Figure 28 (right).

Figure 28: Example of an image where the Proposed method was applied (left). Result of the Otsu threshold (middle). Double contour after adaptive threshold (right).

As mentioned, since the retina is a circle, it is possible to apply a Circle Hough Transform [76] and proceed with the location's extraction. CHT output the coordinates of the center of the detected circle and its radius. These circle values are transformed into a square enclosing the circle inside.

Due to the the shapes present in some images, sometimes it is possible to find more than one circle. To solve multiple circle detections, it is first searched for one image where only one detection occurs. This position will be used in frames where

multiple detections happen, choosing the location closest to that value. Although this is a rather rough approximation, for this particular application the test results were good.

The implemented steps can be summarized as follows:

1. Extract of the red channel of the original image;
2. Obtain the blurred image of the red channel;
3. Apply Otsu threshold in the blurred image obtaining a binary image;
4. Generate an image with contours through a morphological transformation;
5. Double the contours of the image with an adaptive threshold;
6. Apply Circle Hough Transform to find coordinates of the retina.

4.4.2 *Mosaicing*

As previously mentioned in Chapter 3, mosaicing is a technique that can be divided into two steps: image registration, where images keypoints are found and images are warped, and image blending, where image borders are smoothed.

The first step of this mosaicing algorithm is to normalize the intensities of all the images. This step is done for a smoother transaction between image borders in the final result. After this, the image registration step is implemented, which uses the network GLAMpoints [51] to obtain keypoint correspondences between two images. In this network, the keypoints description is done using SIFT. These two images will be identified as "base image" that is used as reference and "following image" that will be warped in correspondence to the similarities between images. As result, the two images that have similarities will match in the same referential, and the image registration step is done.

The next step is image blending. For this, it stage was implemented the featuring-based method as presented by Melo *et al.* [62], which creates two weighted masks to achieve a smoother look to the transition between images. Firstly is found the interception between both of the images and, for each pixel inside of the overlapping region, is applied Equations 3 and 4.

$$w1(\text{x,y}) = \begin{cases} 0 & \text{if } m1(\text{x,y}) = 0 \\ \frac{d2}{d1+d2} & \text{if } m1(\text{x,y}) = 1 \bigwedge R(\text{x,y}) = 1 \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

$$w2(\text{x,y}) = \begin{cases} 0 & \text{if } m2(\text{x,y}) = 0 \\ \frac{d1}{d1+d2} & \text{if } m2(\text{x,y}) = 1 \bigwedge R(\text{x,y}) = 1 \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

$$I(\text{x,y}) = w1(\text{x,y}) * I1(\text{x,y}) + w2(\text{x,y}) * I2(\text{x,y}) \tag{5}$$

These equations describe the Euclidean distance between each pixel and the closest pixel of that image which is outside the overlapping region. After applying this to all the interception pixels is applied the Equation 5, which multiplies the weighted masks to their correspondent warped image to obtain the final image with smooth transitions, named as “result image”.

Figure 29 presents the overall scheme that has been described in the previous paragraphs to implement the mosaicing.

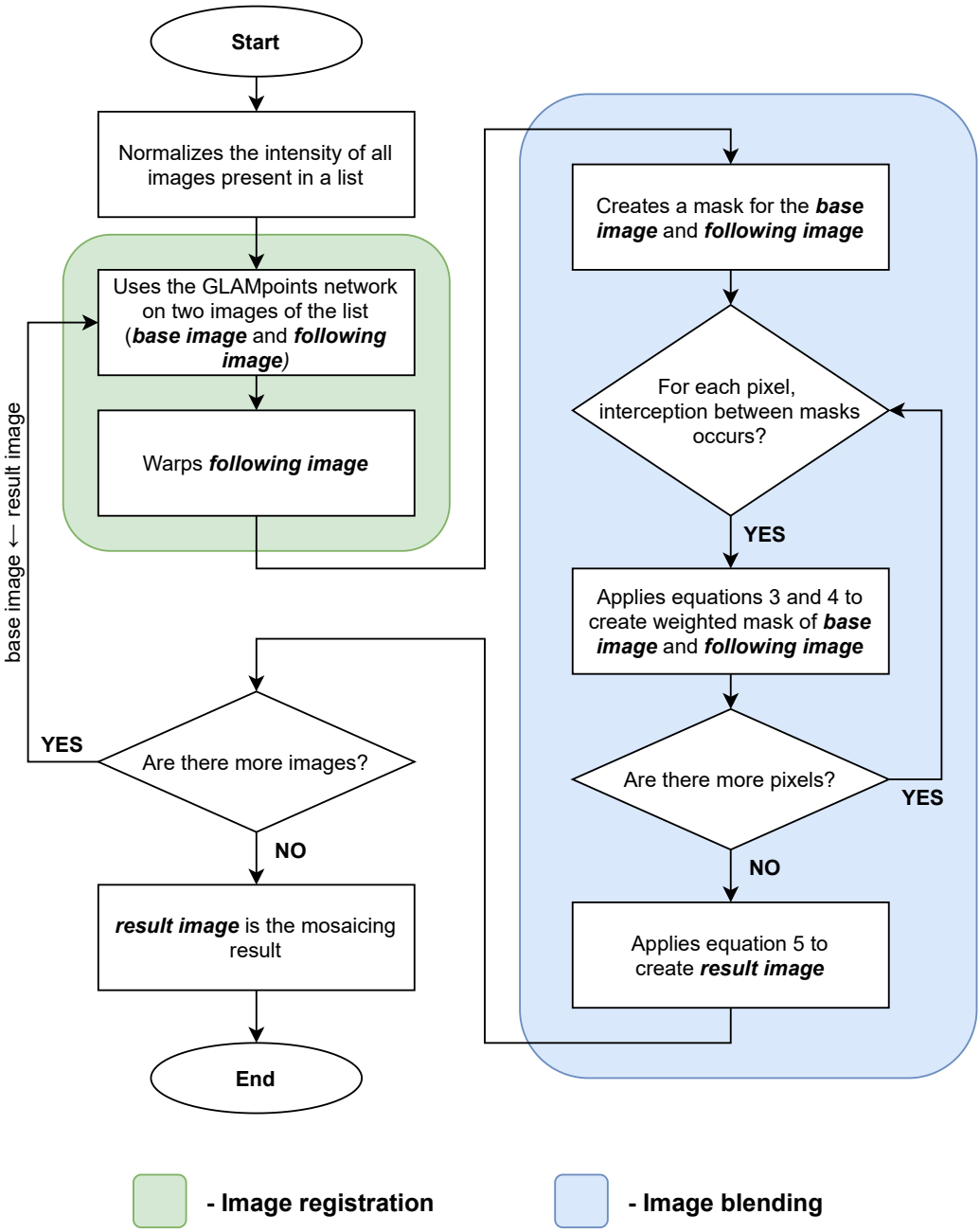


Figure 29: Image mosaicing flowchart.

4.5 EVALUATION

This section explains how the object detection methods were evaluated. In mosaicing, a quantitative evaluation was not implemented. For this method, visual evaluation was done.

4.5.1 *Object detection*

Metrics are a way to measure the performance of a process. Without metrics evaluation, it is difficult to affirm that a model surpasses others since quantitative appraisal does not exist. The D-Eye dataset includes BB location ground truth and, since the Proposed method and YOLOv4 return BBs, it is possible to apply metrics and present the comparative results of the methods. For reliable results, two metrics were used in this dissertation: Mean Absolute Error (MAE) [77] and Intersection over Union (IoU) [78].

The Mean Absolute Error is a simple calculation. Firstly, from the ground-truth value is subtracted the predicted value, resulting in a predicted error. Then, from this predicted error is calculated its magnitude, turning it into an absolute value. This procedure is done four times per image, one for each axis of the bounding box's coordinates. Finally, the mean value between the four absolute values is calculated, resulting in that image's MAE value, as presented in Equation 6. MAE results will be proportionately weighted since it is a linear operation. The resulting score of this metric can go from zero to infinite. Lower MAE represents better model performance. The average value of all the contributions is considered the global MAE performance of the model.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{6}$$

Unfortunately, the MAE value alone will not be enough to prove that one model is outputting good results. Figure 30 is demonstrating one example where the bounding box of the Figure in the right is poorly annotated when compared to the Figure in the left (yellow square). Figure 30 (left), is present all the retina (red square) and in the other case, the retina was cropped. Even though the MAE is lower (wrongly indicates better performance) in Figure 30 (right), the implementation of additional metrics will prevent the wrong analysis of the MAE results.

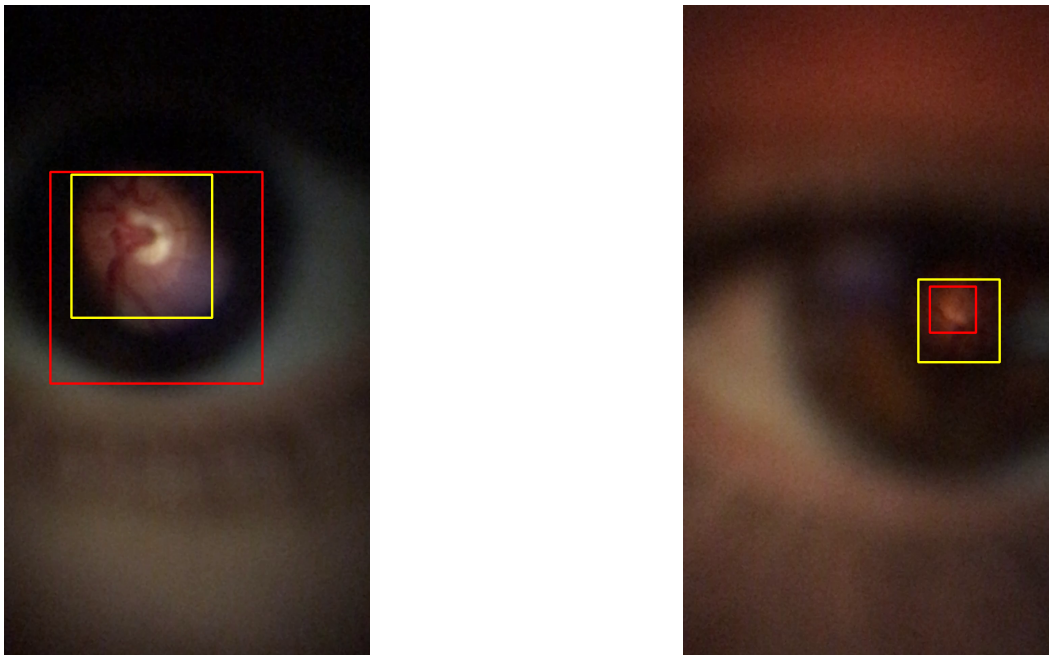


Figure 30: Comparison between MAE and IoU metric results: MAE = 103.0 and IoU = 0.45 (left). MAE = 53.0 and IoU = 0.32 (right).

The other selected metric was Intersection over Union. For two finite sample sets A and B, this metric calculates a ratio dividing the area of overlap of the samples and the area of the union of both samples. The IoU result value varies between zero and one and indicates the total BB area percentage that overlaps. Higher values will represent better performance.

Equation 7 calculates the IoU for each image. The average value of all the values is considered the global IoU performance of the model. As can be seen in Figure 30, this metric attributes the higher value (better performance) to the best image, Figure 30 on the left side.

$$IoU = \frac{A \cap B}{A \cup B} \tag{7}$$

In short, the two metrics complete each other. While the IoU says how right the prediction is, the MAE gives an idea about the prediction deviation's size.

4.6 INTERFACE

The study of the networks and metrics previously presented led to the development of a friendly Graphical User Interface (GUI). The main objective of this GUI is

to facilitate the implementation of the previously presented techniques, aiming to medical usage in the future. Also, has been integrated developed with additional capabilities, such as dataset annotation, GT, and cropped frames video generation. This interface was developed in Python using the framework TKinter [79].

5

RESULTS AND COMPARISON

This chapter presents the obtained results after following the methodology of the previous chapter 4. The chapter is divided into two sections, one to present the results of the object detection task and another for the mosaicing results. In those sections will be seen:

- The influence of the Proposed method steps in the overall results;
- An analytical and visual comparison between the Proposed method and YOLOv4 results, integrated into the developed interface;
- The mosaicing results in the cropped DS1 dataset and resized DS2 and a comparison between the results from the original model and fine-tuned model.

5.1 OBJECT DETECTION

During the development of the Proposed method, all the steps were confirmed to achieve a better overall result for the algorithm.

In the first step of the proposed method, the extraction of one color channel from the image is done; the intention is to reduce data processing, removing non-relevant colorspace channels. Appendix A shows the channel comparison of multiple color spaces in different images. Since the objective was to apply the CHT to the images, it was defined that the best channel options would be those in which the retina was better defined in a circular shape and with the same dimensions. After a visual evaluation, it was concluded that the different channels that could have a good performance would be the R channel from the RGB color space, the Cr channel of the YCrCb color space, and V from the HSV color space. From all the color spaces, the red channel of RGB color space was the chosen one, since this channel was the one that, when applied to Otsu's threshold, exhibit better performance detecting the retina. This can be verified in Table 7, where a comparison between predicted location and ground truth is made. In this Table is possible to verify that the R channel in comparison with the other channels it has lower MAE and higher IoU values.

Table 7: Comparison of the proposed method results with different channels applied.

	MAE	**MAE standard deviation**	**IoU**	**IoU standard deviation**
R channel of RGB	27,5	29,29	0,75	0,13
Cr channel of YCrCb	54,89	99,97	0,69	0,23
V channel of HSV	33,65	35,67	0,72	0,15

The image is then blurred in the second step before applying the threshold to the image in order to obtain an image with smoother edges and later have better results applying the CHT. To prove that the blur would improve the retina's detection, a comparison of its use (with and without blur) was made. When the detected location was compared with the GT, it was concluded that this step increments the global algorithm's success since was obtained a lower MAE and higher IoU when blur is used, as shown in Table 8. The results were better with the blur step since that way the retina region has a more circular outline with fewer bumps.

Table 8: Comparison of the proposed method results with and without image blur step.

	MAE	**MAE standard deviation**	**IoU**	**IoU standard deviation**
With blur	27,5	29,29	0,75	0,13
Without blur	100,89	135,14	0,56	0,31

After optimizing all the steps of the Proposed method, the overall results, the trained model of YOLOv4 with DS1, and the Zengin *et al.* [5] retinal detection task results were compared.

Based on visual analysis and the metic IoU values, three ranges were defined for image classification: Successful class is defined for images with an IoU results greater than 0,8. For IoU result between 0,6 and 0,8 (included), is given the Acceptable class. Images with less than or equal to 0,6 are attributed to the Failed class. Figure 31, Figure 32, and Figure 33 present multiple D-Eye examples with bounding boxes marking RoI, where yellow boxes are the ground truth, red boxes are the Proposed method, purple boxes are the Zengin *et al.* [5] annotations, and blue boxes the YOLOv4 network outputs. The first column of images shows Successful class

examples, in the second column Acceptable class examples, and the Failed class examples are displayed in the last column.

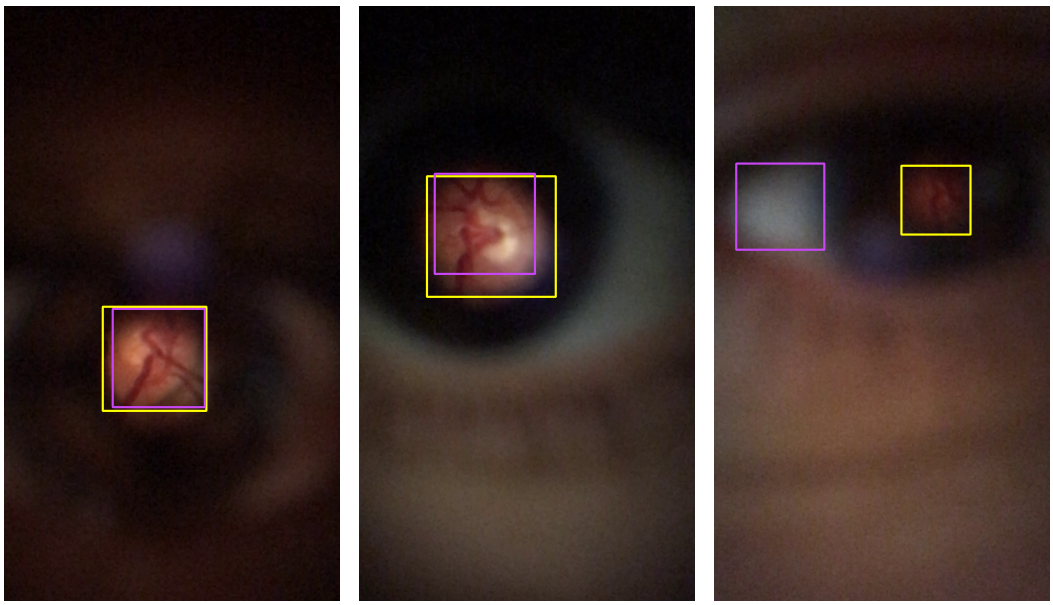


Figure 31: Example of bounding boxes visual results of [5]. Successful classification (left). Acceptable classification (middle). Failed classification (right).

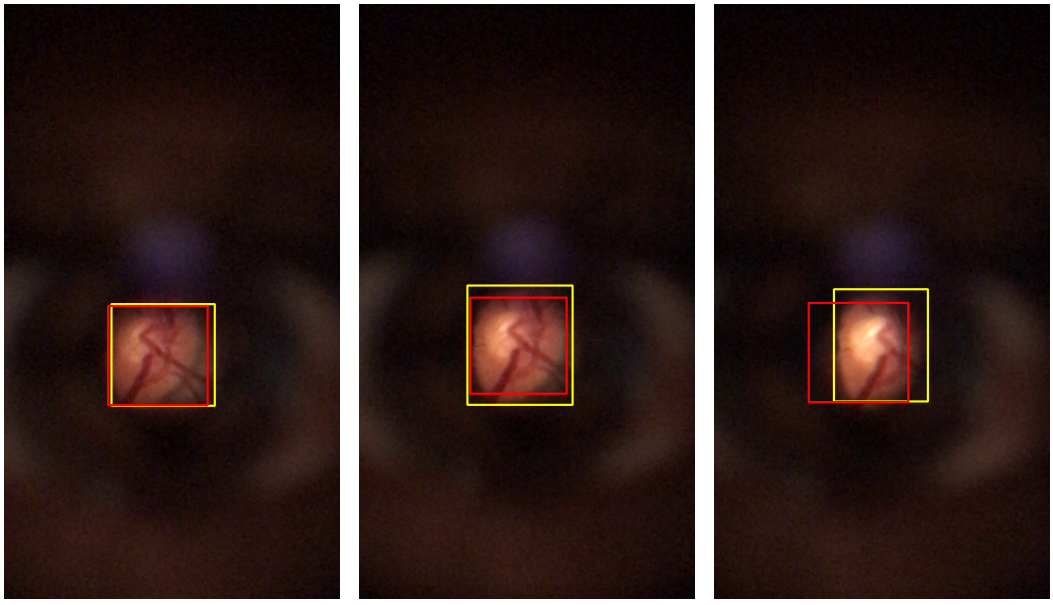


Figure 32: Example of bounding boxes visual results of proposed method. Successful classification (left). Acceptable classification (middle). Failed classification (right).

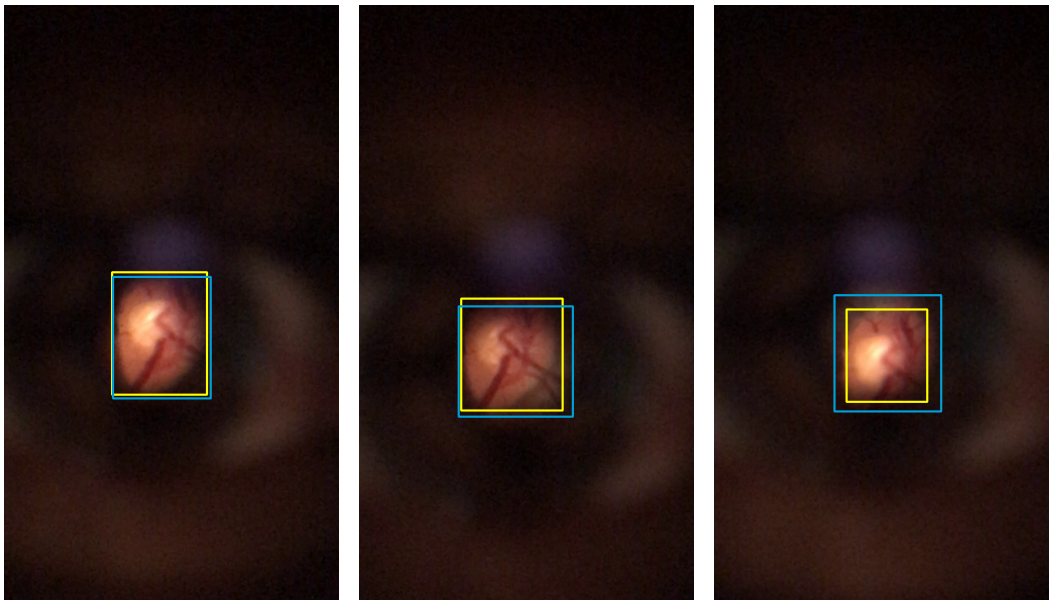


Figure 33: Example of bounding boxes visual results of YOLOv4. Successful classification (left). Acceptable classification (middle). Failed classification (right).

Based on those class divisions, Table 9 was compiled showing the number of images, the mean MAE, and mean IoU for each classification class, being the best results highlighted in bold. Since the dataset is the same (DS1), a direct comparison of these results can be performed to the ones presented in Zengin *et al.* [5].

Table 9: Comparison between methods, for Successful, Acceptable and Failed class results.

	SUCESSFUL (IoU >0,8)				**ACCEPTABLE** (0,6 <IoU ≤ 0,8)				**FAILED** (IoU ≤ 0,6)			
	Frequency		**MAE**	**IoU**	**Frequency**		**MAE**	**IoU**	**Frequency**		**MAE**	**IoU**
	Absolute	**Relative**			**Absolute**	**Relative**			**Absolute**	**Relative**		
Zengin *et al.* [5]	615	44,73%	12,84	0,85	544	39,56%	26,69	0,73	216	15,71%	80,27	0,48
Proposed method	562	40,87%	13,52	0,85	**627**	**45,60%**	26,24	0,73	186	13,53%	73,99	0,48
YOLOv4	**1075**	**78,18%**	**11,05**	**0,88**	299	21,75%	**25,73**	**0,75**	**1**	**0,07%**	**40,25**	**0,60**

Considering the overall success as the merge of both Successful and Acceptable classes (IoU > 0,6), for Zengin *et al.* [5] can be observed that 84,3% were successfully classified, although only being 44,7% from Successful class. With a slightly lower performance, there are the results associated with the Proposed method, where 40,9% of the images are classified as Successful and 45,6% as Acceptable. In this case, the cumulative frequency of these two indicators is 86,5%, which slightly improves Zengin *et al.* results [5]. On the other hand, from the YOLOv4 network application, it is verified that it presents the best performance, since only one image is classified as Failed, and as such, 99,9% of the dataset is classified as Successful or Acceptable. It should also be noted that from this 99,9%, around 78,2% are classified as Success, which stands out concerning the other works in comparison, regarding the location of the retinal area. Considering the metrics that assess the quality of the identification of the retinal location, it is observed that for the YOLOv4 network, the best classifications are obtained in terms of MAE, with a value of 11,05 for Successful class (instead of 12,84 for [5] and 13,52 for the Proposed method), for the Acceptable class the value of 25,73 (instead of 26,69 for [5] and 26,24 for the Proposed method) and for the Failed class the values of 80,27 and 73,99 for [5] and for Proposed method respectively). Similarly, the YOLOv4 network showed an improvement for all classifications considered, with an average IoU value of 0,88 for the Successful class, instead of the 0,85 that the other methods present. For the Acceptable class, the behavior is similar, with 0,75 (instead of 0,73 for [5] and Proposed method) and 0,60 (instead of 0,48) for the Failed class.

Table 10 presents the global results for each method. In this table, it is possible to observe that the Proposed method results are very similar to the results obtained by [5], however, the Proposed method had a substantially better standard deviation value for MAE. Taking into account the YOLOv4 network, it can be seen that it stands out in an improvement, both in the mean values of MAE and IoU and in the standard deviation values, also corroborating the analysis carried out in Table 9.

Table 10: Overall results (average and standard deviation) for methods to be compared.

		Mean	**Standard deviation**
Zengin *et al.* [5]	**MAE**	28,91	47,04
	IoU	0,75	0,14
Proposed method	**MAE**	27,5	29,29
	IoU	0,75	0,13
YOLOv4	**MAE**	**14,26**	**7,92**
	IoU	**0,85**	**0,07**

As summary, both the Proposed method and YOLOv4 would bring an improved global result to the work of [5], and as expected due to the nature of YOLOv4 as detector network, that it stands out as the best approach to follow.

5.2 MOSAICING

Two different approaches were used for the mosaicing which only differ in the GLAMpoints model used. The first is a fine-tuned model applying the DS1 cropped dataset and the second was used the original model trained by Truong *et al.* [51].

Mosaicing can be divided into image registration and image blending. Figure 34 (left) presents the results for an image registration step between three DS2 [11] images. This image was obtained applying the image registration implementation to the original GLAMpoints model. To eliminate the transitions between optic discs, the image blending step was applied, resulting in Figure 34 (right). The latter presents a smoother transition when aggregating all the contributions.

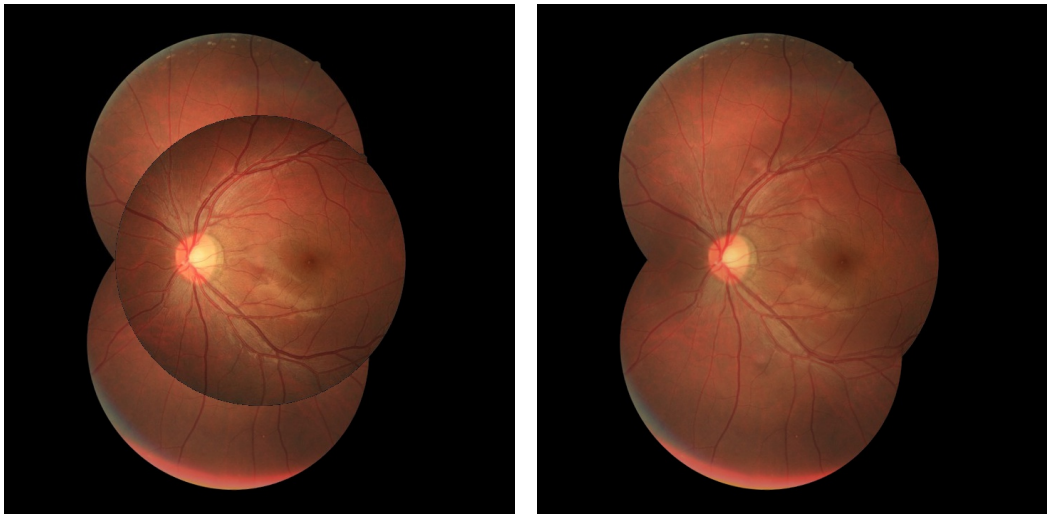


Figure 34: Mosaicing result obtained with the original model of GLAMpoints applied to three images of DS2. Image registration result (left). Image blending result (right).

Figure 35 presents the mosaicing result executed with the fine-tuned model, applied to the same three images of the DS2 dataset previously refereed. It is possible to observe that the fine-tuned model made the results quite worse than the result of Figure 34 (right). The fine-tuned model, using the cropped DS1 dataset, made adjustments in the weights of the model to be able to detect keypoints in

images with lower quality, although losing its ability to correctly detect keypoints in higher quality images like DS2.

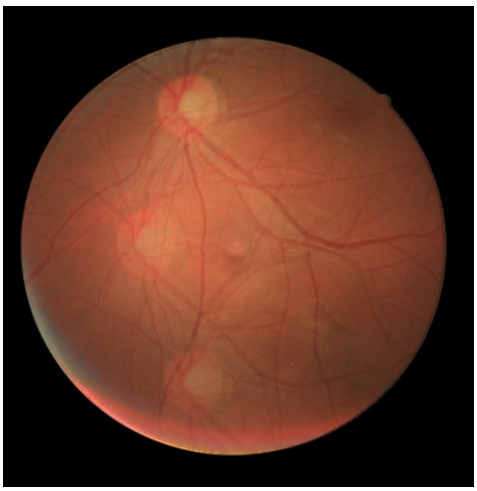


Figure 35: Mosaicing result obtained with the fine-tuned model of GLAMpoints applied to three images of DS2.

Figure 36 presents the comparison between mosaicing using the original and the fine-tuned models in DS1 cropped images. As can be seen, the original model got better visual results than the fine-tuned. The original GLAMpoints model could easily find keypoints in DS2 images, with the fine-tuning of the original model, it was intended to adapt the weights to find keypoints in cropped DS1. As can be observed the set of training images provided may not have been sufficient to adapt the model to find relevant keypoint in D-Eye images, resulting in worsen results.

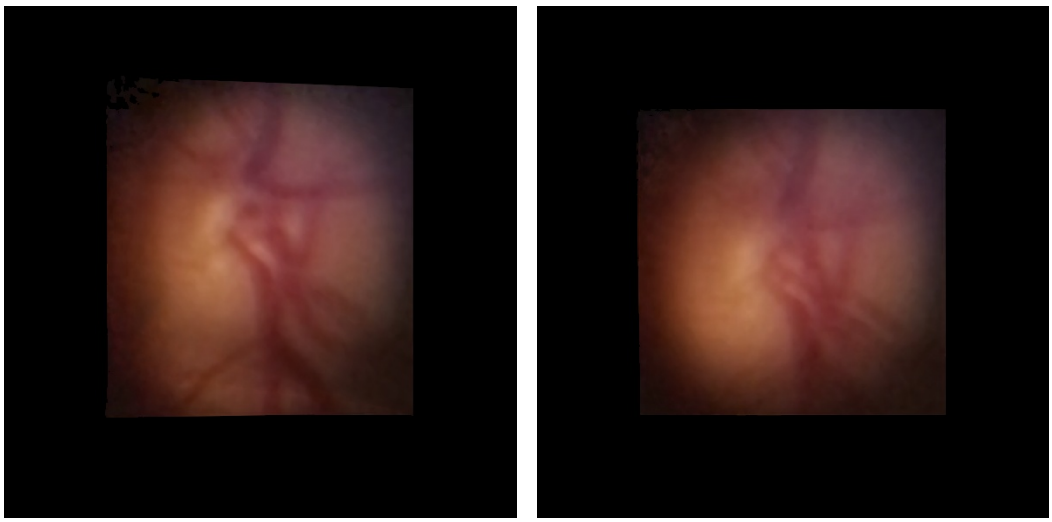


Figure 36: Comparison between mosaicing of the DS1 cropped images: Using the original model (left) and using the fine-tuned model (right).

5.3 INTERFACE

With the GUI the user can manually label each image or select one of the automatic annotations made by methods that detect the retina or the object that is being annotated. Figure 37 presents the initial page of the GUI developed.

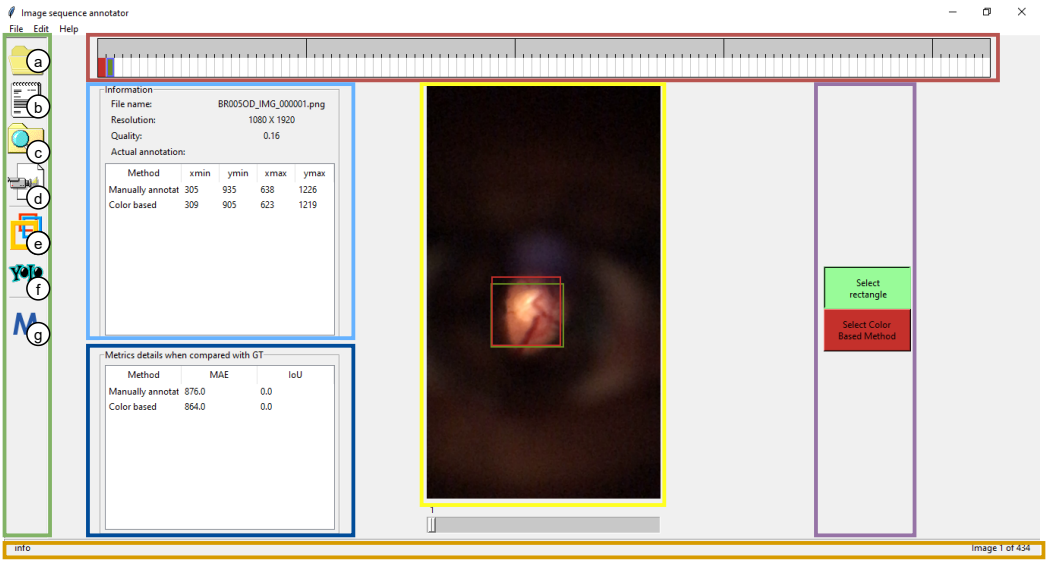


Figure 37: Initial page of the interface developed.

On top of the window is implemented a sequential bar, marked with a red rectangle. This bar indicates the selected image of the dataset and the type of annotation selected by the user to define the RoI. The user can select the RoI from multiple ways: manual selection, Proposed method annotations, YOLOv4 method annotations, or from ground truth CSV files. Different annotation origins will paint the bar with different colors. On the right side of the window are presented the buttons that allow selecting the origin of the annotations. In Figure 37 the area where these buttons are located is marked with a purple rectangle.

On the center is presented the image that the user is labeling, marked with a yellow rectangle in Figure 37. To manually annotate an image, the user must press the mouse's left button, drag, and release over the region of interest. The result of this is presented in Figure 38 (left). If a correction of the marked regions is required, it is possible to edit the marked rectangle by clicking inside it and then dragging one of the circles that appear in the windows. Figure 38 (right) presents the visual aspect of the edition mode. To select one automatic annotation, firstly, the method needs to be activated. This is explained how in the following subsections.

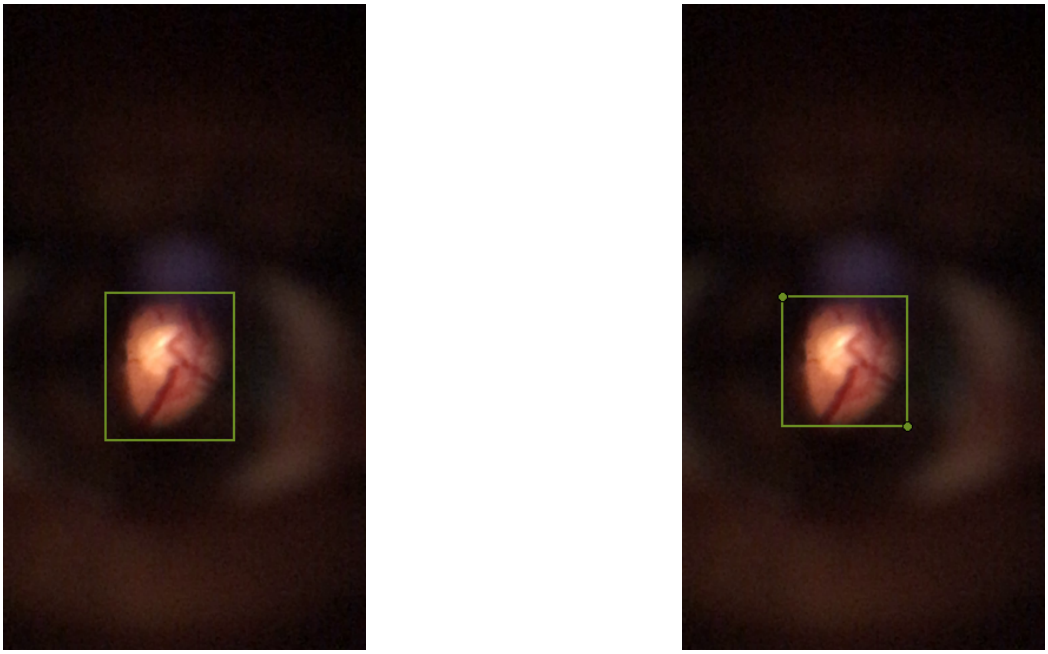


Figure 38: Manual selection tool: Default aspect (left) and Edit mode (right).

On the left side of the image being labeled, is presented two frames. The top one, marked with a cyan rectangle in Figure 37, gives the user details about the image, such as the name, resolution, quality, and actual annotations coordinates. The bottom frame, marked with a blue rectangle in Figure 37, presents a comparison between one imported GT and the annotation methods used. A better explanation of how to import annotation files is present in the following subsection 5.3.3.

The status bar, present at the bottom of the windows and marked with an orange rectangle in Figure 37 presents relevant information about the GUI state, like the number of the image in the sequence being annotated or information about widgets in the window.

On the left most side of the windows is a toolbar marked with a green rectangle in Figure 37 that provides the used quick access to all tools. The description of each one of the icons and its functionalities is made in the following sections.

5.3.1 *Open images tool*

The first icon marked with an a) in Figure 37 is used to open the image dataset folder that the user plans to label. After selecting the dataset, the first image is displayed to the user to be able to manually annotate it or by selecting any other method to detect the RoI automatically. After the RoI size and location are well

defined, the user can set it as the final annotation of that image. To do so, there's the need to click over the button on the right side of the window, and a new image will be displayed.

5.3.2 *Export annotation*

After defining the RoI in all the dataset images, the user may want to export these annotations. This functionality is accessible by clicking in the b) tool of the toolbar in Figure 37. It will open the window presented in Figure 39 (left). It is possible to choose between three types of annotation formats: YOLO Darknet, YOLOv4 PyTorch, and Tensorflow Object Detection. The first format, YOLO Darknet, provides one text file for each image. Every text file has one line for each annotation with five values. The first value is for the annotation class. The remaining four values are for the bounding box annotation coordinates given in the YOLO annotation format described in this chapter's previous section. The format YOLOv4 PyTorch creates two text files. One of them presents the name of the classes used for annotation. The other file has all the annotations of the dataset. In this last document, each line corresponds to one image. This way, multiple annotations in the same image are marked in a single line. The first value is the name of the image file. The next five values are the coordinates in the TensorFlow format of the BB and the class number. These values repeat depending on the number of annotations presented in the image. Finally, the Tensorflow Object Detection format creates one CSV file. Each line of the CSV file corresponds to one annotation. This way, multiple lines can correspond to only one image if multiple objects were marked, unlike the previous case. Each line has information about the image's file name, width, and height of the image, class of the annotation, and BB coordinates in TensorFlow format of the RoI.

5.3.3 *Import annotation*

The interface gives the user the option to import a ground truth in the Tensorflow Object Detection format. It is accessible by clicking in the second tab of the windows opened by the b) option of the toolbar in Figure 37 or pressing the button of the blue rectangle in Figure 37. The visual aspect of the import tab is presented in Figure 39 (right). The windows for import annotation have three widgets. Two of them are entry widgets. Their function is to define the colors to represent the

ground truth over the image graphically. Furthermore, the checkbox activates the ground truth functionalities. When the checkbox is active, the annotation BB will be presented over the dataset's image and given to the user the possibility of choosing between one of the BB presented on the screen. Furthermore, it will be activated a feature that calculates the MAE and IoU metrics between the active methods and the ground truth.

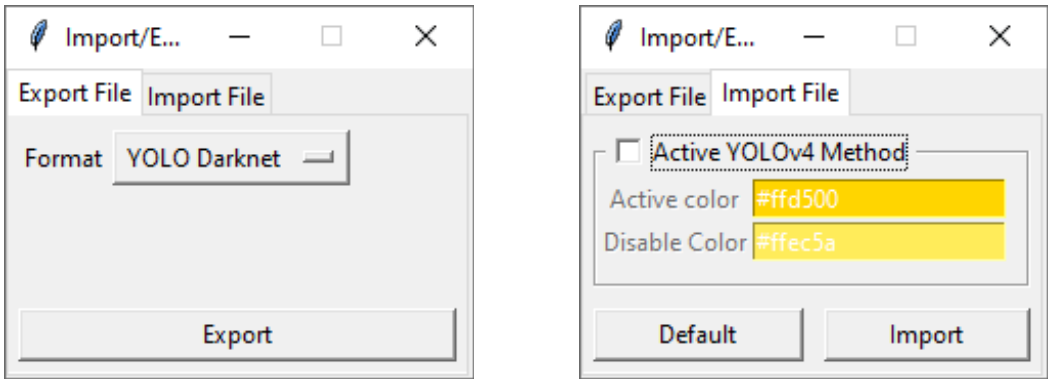


Figure 39: Windows presenting the export (left) and import (right) annotations options.

5.3.4 *Search tool*

The search tool enables the user to export individual annotations based on the obtained metrics. The user needs to select the images of interest from the presented table and then press the export button. When this tool is activated, a window will be presented with four tabs. The first tab presents the metrics of the annotations that the user in the main windows selected. The second, third, and fourth tabs have the metrics obtained with the coordinates obtained by the manual, color-based, and YOLOv4 methods, respectively. Another option of this tool is to export a CSV file with information about the mean and standard deviation of the metrics and the individual image metrics. To do it, the user needs to press the export report button. This tool is marked with c) in the toolbar in Figure 37.

5.3.5 *Export tool for video and frames*

Beyond the annotation coordinates exportation, it is possible to export the video or frames of the images cropped area using the d) option in the toolbar displayed in Figure 37. Figure 40 presents the appearance of the window. It is possible to observe multiple exporting configurations. The first option inquires the user for

the image resizing or padding. If he chooses the padding one, then the image will maintain its aspect ratio but will fill the remaining space with the users defined color. If the resize option is selected, then the image will expand until it fits the width and height also defined by the user. It is possible to configure the number of frames per second on the video.

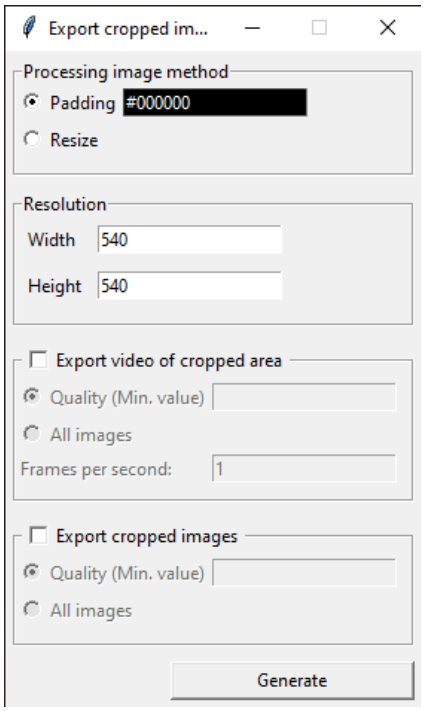


Figure 40: Export video and frames tool.

5.3.6 *Proposed method tool*

It is possible to activate the Proposed method for detection, using option e) in the toolbar in Figure 37. The user can configure the method, but a default button can set the standard values that were found to work better for the D-Eye dataset. Those values were found as previously presented in section 5.1. When the method is activated, a rectangle will appear over the image marking the RoI. A button on the right side of the window will allow the user to set that rectangle as the final annotation for that image. Figure 41 presents the appearance of the windows.

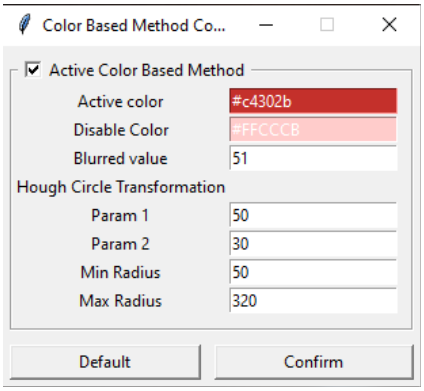


Figure 41: Proposed method tool.

5.3.7 *YOLOv4 method tool*

To annotate the images using the YOLOv4 method, the user needs to activate the YOLOv4 method, using the f) tool of the toolbar in Figure 37. It is allowed for the user to use its own model or use one previously trained. Figure 42 (left) presents the appearance of the windows that activate the YOLOv4 labeling method. Like the previous model, activating this model will add a button to the right side of the window that allows to set the RoI as final annotation and define RoIs in all images defined with the YOLOv4 method. The user can train its custom model too, accessing the second tab, presented in Figure 42 (middle), and configuring the parameters. The train is done using the Darknet framework since it is the one that nowadays gives better results; however, it is possible to convert the model to other frameworks using the third tab, windows in Figure 42 at the right.

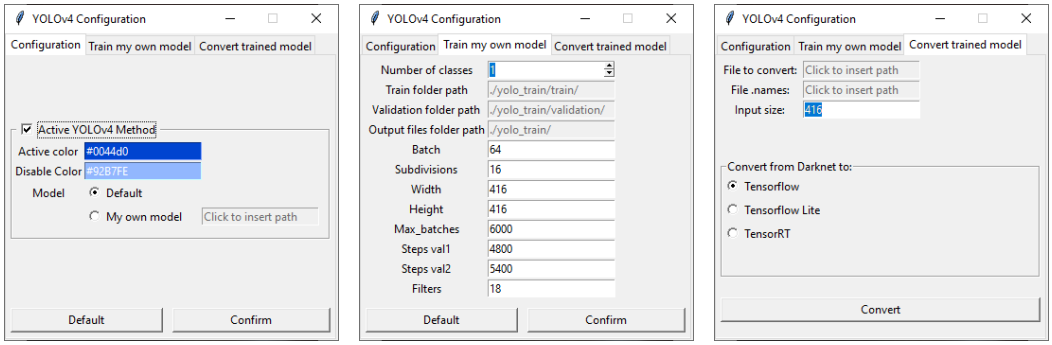


Figure 42: YOLOv4 method window: Activate the YOLOv4 method tab (left). Train custom model tab (middle). Convert model tab (right).

5.3.8 *Mosaicing tool*

The mosaicing tool is accessible in the icon marked with g) in the toolbar of Figure 37. After pressing that option, the window of Figure 43 (left) will appear. The interface asks for a GLAMpoints model, one path to save the results, and annotated images to create the mosaicing. The annotated images that the algorithm will use are the ones defined as "selected annotation" in the GUI initial page. The example presented in Figure 43 (left) only has two images since only two were defined as "selected annotation".

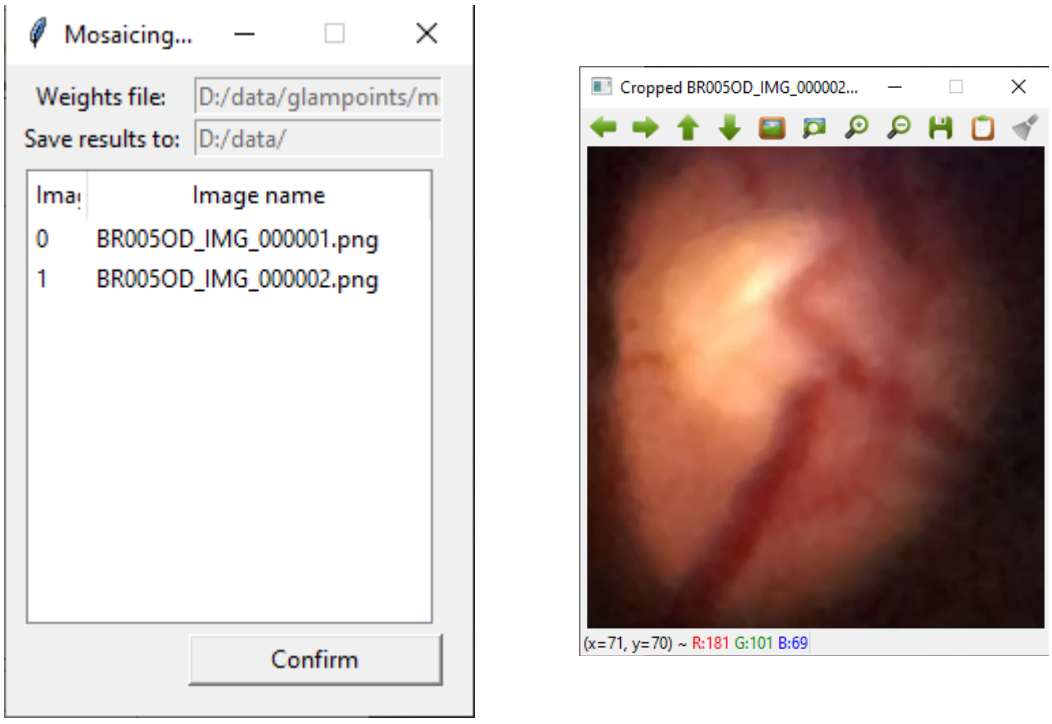


Figure 43: Mosaicing tool: Initial window (left). Preview of the cropped image to use as central mosaicing image (right).

Is possible to define one image as the central mosaicing image pressing with the mouse left button over the image name. If a double click is done over the image name, the cropped retinal area of that image will be displayed to the user, Figure 43 (right).

6

CONCLUSION AND FUTURE WORK

The equipment to capture retinal fundus images are normally expensive and have lack of portability. The development of technology brought new alternatives for screening that can be attached to smartphones, being one example the D-Eye device. When compared to specialized equipment, the D-Eye present lower quality in the captured retinal video, yet is enough to perform a medical pre-screening. This work contributes to the development of a framework with tools developed to extract useful information from the low-quality D-Eye retinal videos. This chapter has the objective to summarize the contributions made with this dissertation and address some points that require attention and would be interesting to be analyzed and developed in the future.

6.1 CONCLUSIONS

The main objective of this work is to contribute in the development of a retinal framework. With the idea of achieving this goal, three sub-objectives were proposed. The first sub-objective was use object detection methods to extract D-Eye relevant information, *i. e.*, retinal location. This is an important task since a big area of D-Eye images has no relevant information, being the retinal (relevant information) only a small portion of the image. The second sub-objective was apply mosaicing technique to D-Eye images, *i. e.*, merge multiple retinal images to create a single and more informative image. This way, from the D-Eye captured videos, would be possible to obtain a single summary image. The third sub-objective was add the developed methods and add other features to the Graphical User Interface to make them more accessible to the user.

In the Object detection task, two different methods were implemented with success for retinal region detection in D-Eye images. The first was purely developed based on a classical programming approach using Circle Hough Transform. The second was done implementing CNNs, more exactly training a model of YOLOv4 with the D-Eye dataset [5]. When comparing the performance of the two implemented methods of this dissertation with the object detection implemented on [5], it is

noticeable an improvement in the region proposals. Zengin *et al.* [5] presented results with an IoU of 0,75 and a standard deviation of 0,14. The Proposed method presented a slight improvement reducing the standard deviation to 0,13. Better results were obtained with the trained YOLOv4 model, having an IoU of 0,85 and a standard deviation of 0,07. To facilitate the development of future works using YOLOv4, some documentation explaining how to install the Darknet framework (framework where YOLOv4 was implemented) was also produced and is available as in Appendix B. Furthermore, a Jupiter notebook file is available which facilitates and simplifies to new users the training of the network.

In the mosaicing, the process was divided into two stages: image registration and image blending. In image registration, keypoints were found using the CNN GLAMpoints and then described with SIFT descriptor. After the warp of the images to have the same coordinate system, image blending was applied to smooth the transition between images. In this dissertation, this step was applied with a feature-based method. The implementation of mosaicing was considered a success but needs model improvement for D-Eye images since the one used was obtained from the Truong *et al.* [51], due to lack of improvements in the results of the fine-tuned model trained, in which were applied the cropped retinal area images of D-Eye dataset. To simplify the training of GLAMpoints models to use in future works, a Jupiter notebook file is available with this dissertation. This file contains all the necessary steps to conclude a model train with success in the frameworks TensorFlow and Pytorch.

In order to group and simplify the usage of the previously described methods, an interface was developed. The Graphical User Interface was developed with Python using Tkinter and can be utilized for manual or automatic labeling of the retinal images using the Proposed or YOLOv4 method and the obtained coordinates can be exported in multiple formats. Is possible to use the interface to make a labeling comparison between manual, Proposed method, or YOLOv4 and ground truth coordinate files and export the results. The found or manually selected coordinates of the retina can also be used to export cropped images or videos with adjustable parameters like padding, color padding, video frame rate, and resolution. Furthermore, is possible to modify the parameters of the Proposed method, use the own trained YOLOv4 model at the automatic labeling, convert Darknet YOLOv4 models to other frameworks, and merge multiple retinal images into a single one based on a GLAMpoint model.

The contributions implemented in the interface provided important features to work, in a simpler way, with the relevant information of the D-Eye images. With

the developed tools it is now easier to annotate D-Eye datasets, calculate metrics, compare results and even generate retinal image mosaicing.

6.2 FUTURE WORK

In this dissertation, the wide variety of tools development left space for their improvement, this way, much work can still be done. Some examples are, since the importation of GT can only be in Tensorflow format, the implementation of different formats would be valuable. The YOLOv4 tool has room for improvement, namely in the training section that can only train Darknet models and it would be interesting to implement the train to run in the background, this way the interface could be used while the model is being trained. It would be interesting to develop an approach to add and remove metrics to compare the Ground truth and selection, similar to what is already done with the Proposed and YOLOv4 methods. Although GLAMpoint models were trained, the expected results were not reached when applied to D-Eye images. The train of GLAMpoint model that works well with the D-Eye images would be interesting in the future. Furthermore, mosaicing has no quantitative evaluation metric for their results; it would be compelling to develop an evaluation metric to quantify the model's performance.

Overall, the use of the proposed GUI and methods allowed to extract better and more useful information from the D-Eye images which can be lead to pre-screening of diseases if analysed by a doctor.

BIBLIOGRAPHY

[1] K. Jin, H. Lu, Z. Su, C. Cheng, J. Ye, and D. Qian, "Telemedicine screening of retinal diseases with a handheld portable non-mydriatic fundus camera", *BMC ophthalmology*, vol. 17, no. 1, pp. 1–7, 2017.

[2] A. Russo, F. Morescalchi, C. Costagliola, L. Delcassi, and F. Semeraro, "A novel device to exploit the smartphone camera for fundus photography", *Journal of ophthalmology*, vol. 2015, 2015.

[3] R. N. Maamari, J. D. Keenan, D. A. Fletcher, and T. P. Margolis, "A mobile phone-based retinal camera for portable wide field imaging", *British Journal of Ophthalmology*, vol. 98, no. 4, pp. 438–441, 2014.

[4] *Inview®*, https://www.volk.com/collections/diagnostic-imaging/products/inview-for-iphone-6-6s.html, (Accessed on 26/02/2021).

[5] H. Zengin, J. Camara, P. Coelho, J. M. Rodrigues, and A. Cunha, "Low-resolution retinal image vessel segmentation", in *International Conference on Human-Computer Interaction*, Springer, 2020, pp. 611–627.

[6] J. Zhu, E. Zhang, and K. Del Rio-Tsonis, "Eye anatomy", *eLS*, 2012.

[7] M. F. Mafee, A. Karimi, J. Shah, M. Rapoport, and S. A. Ansari, "Anatomy and pathology of the eye: Role of mr imaging and ct", *Radiologic Clinics*, vol. 44, no. 1, pp. 135–157, 2006.

[8] W. H. Organization *et al.*, "World report on vision", 2019.

[9] B. Medical, "Medical gallery of blausen medical 2014", *WikiJournal of Medicine*, vol. 1, no. 2, pp. 1–79, 2014.

[10] H. J. Kaplan, "Anatomy and function of the eye", in *Immune Response and the Eye*, vol. 92, Karger Publishers, 2007, pp. 4–10.

[11] C. Hernandez-Matas, X. Zabulis, A. Triantafyllou, P. Anyfanti, S. Douma, and A. A. Argyros, "Fire: Fundus image registration dataset", *Journal for Modeling in Ophthalmology*, vol. 1, no. 4, pp. 16–28, 2017.

[12] L. S. Lim, P. Mitchell, J. M. Seddon, F. G. Holz, and T. Y. Wong, "Age-related macular degeneration", *The Lancet*, vol. 379, no. 9827, pp. 1728–1738, 2012.

[13] M. D. Abràmoff, M. K. Garvin, and M. Sonka, "Retinal imaging and image analysis", *IEEE reviews in biomedical engineering*, vol. 3, pp. 169–208, 2010.

[14] *Eye condition simulator*, https://www.eyesiteonwellness.com/eye-diseases/, (Accessed on 26/02/2021).

[15] J. G. Fujimoto, C. Pitris, S. A. Boppart, and M. E. Brezinski, "Optical coherence tomography: An emerging technology for biomedical imaging and optical biopsy", *Neoplasia*, vol. 2, no. 1-2, pp. 9–25, 2000.

[16] J. G. Fujimoto, "Optical coherence tomography: Principles and applications", vol. 31, no. 10, pp. 635–642, 2003.

[17] S. Vujosevic and E. Midena, "Retinal layers changes in human preclinical and early clinical diabetic retinopathy support early retinal neuronal and müller cells alterations", *Journal of diabetes research*, vol. 2013, 2013.

[18] V. Sarao, D. Veritti, E. Borrelli, S. V. R. Sadda, E. Poletti, and P. Lanzetta, "A comparison between a white led confocal imaging system and a conventional flash fundus camera using chromaticity analysis", *BMC ophthalmology*, vol. 19, no. 1, p. 231, 2019.

[19] A. B. Jain, V. J. Prakash, and M. Bhende, "Techniques of fundus imaging", *MEDICAL & VISION RESEARCH FOUNDATIONS*, vol. 33, no. 2, p. 100, 2015.

[20] M. Yung, M. A. Klufas, and D. Sarraf, "Clinical applications of fundus autofluorescence in retinal disease", *International journal of retina and vitreous*, vol. 2, no. 1, pp. 1–25, 2016.

[21] T. E. De Carlo, A. Romano, N. K. Waheed, and J. S. Duker, "A review of optical coherence tomography angiography (octa)", *International journal of retina and vitreous*, vol. 1, no. 1, p. 5, 2015.

[22] *Topcon trc-50ix fundus camera*, https://sky-optic.com/catalog/topcon-trc-50ix-fundus-camera/, (Accessed on 22/09/2021).

[23] M. Karakaya and R. E. Hacisoftaoglu, "Comparison of smartphone-based retinal imaging systems for diabetic retinopathy detection using deep learning", *BMC bioinformatics*, vol. 21, no. 4, pp. 1–18, 2020.

[24] A. R. Wu, S. Fouzdar-Jain, and D. W. Suh, "Comparison study of funduscopic examination using a smartphone-based digital ophthalmoscope and the direct ophthalmoscope", *Journal of pediatric ophthalmology and strabismus*, vol. 55, no. 3, pp. 201–206, 2018.

[25] F. Chollet, *Deep learning with Python*. Simon and Schuster, 2017.

[26] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[27] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[29] U. of Toronto, "Csc 321 winter 2018: Introduction to neural networks and machine learning", *CSC 321*, 2018.

[30] F.-F. Li, J. Johnson, and S. Yeung, *Lecture 6: Training neural networks, part i*, https://cs231n.github.io/neural-networks-1/, Apr. 2017.

[31] F.-F. Li, J. Johnson, and S. Yeung, *Lecture 5: Convolutional neural networks*, https://cs231n.github.io/convolutional-networks/, Apr. 2017.

[32] F.-F. Li, J. Johnson, and S. Yeung, *Lecture 12: Detection and segmentation*, http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf, May 2017.

[33] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey", *International journal of computer vision*, vol. 128, no. 2, pp. 261–318, 2020.

[34] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey", *arXiv preprint arXiv:1905.05055*, 2019.

[35] S. K. Pal, A. Pramanik, J. Maiti, and P. Mitra, "Deep learning in multi-object detection and tracking: State of the art", *Applied Intelligence*, pp. 1–30, 2021.

[36] Á. Morera, Á. Sánchez, A. B. Moreno, Á. D. Sappa, and J. F. Vélez, "Ssd vs. yolo for detection of outdoor urban advertising panels under multiple variabilities", *Sensors*, vol. 20, no. 16, p. 4587, 2020.

[37] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A survey of deep learning-based object detection", *IEEE Access*, vol. 7, pp. 128 837–128 868, 2019.

[38] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[39] R. Girshick, "Fast r-cnn", in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[40] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks", *arXiv preprint arXiv:1506.01497*, 2015.

[41] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector", in *European conference on computer vision*, Springer, 2016, pp. 21–37.

[42] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[43] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[44] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement", *arXiv preprint arXiv:1804.02767*, 2018.

[45] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", in *International conference on machine learning*, PMLR, 2015, pp. 448–456.

[46] C. Wakamiya, *Object detection with yolo*, https://datax.berkeley.edu/wp-content/uploads/2020/09/slides-m330-YOLO-object-detection.pdf, 2020.

[47] S. Ajitha and M. Judy, "Faster r-cnn classification for the recognition of glaucoma", in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1706, 2020, p. 012 170.

[48] T. H. P. Thanh, T. P. T. Thuy, T. N. Hieu, and M. S. Nguyen, "A real-time classification of glaucoma from retinal fundus images using ai technology", in *2020 International Conference on Advanced Computing and Applications (ACOMP)*, IEEE, 2020, pp. 114–121.

[49] N. Patton, T. M. Aslam, T. MacGillivray, I. J. Deary, B. Dhillon, R. H. Eikelboom, K. Yogesan, and I. J. Constable, "Retinal image analysis: Concepts, applications and potential", *Progress in retinal and eye research*, vol. 25, no. 1, pp. 99–127, 2006.

[50] D. Ghosh and N. Kaabouch, "A survey on image mosaicing techniques", *Journal of Visual Communication and Image Representation*, vol. 34, pp. 1–11, 2016.

[51] P. Truong, S. Apostolopoulos, A. Mosinska, S. Stucky, C. Ciller, and S. D. Zanet, "Glampoints: Greedily learned accurate match points", in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 10 732–10 741.

[52] G. Haskins, U. Kruger, and P. Yan, "Deep learning in medical image registration: A survey", *Machine Vision and Applications*, vol. 31, no. 1, p. 8, 2020.

[53] Y. Fu, Y. Lei, T. Wang, W. J. Curran, T. Liu, and X. Yang, "Deep learning in medical image registration: A review", *Physics in Medicine & Biology*, 2020.

[54] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Toward geometric deep slam", *arXiv preprint arXiv:1707.07410*, 2017.

[55] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 224–236.

[56] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "Lift: Learned invariant feature transform", in *European Conference on Computer Vision*, Springer, 2016, pp. 467–483.

[57] K. Moo Yi, Y. Verdie, P. Fua, and V. Lepetit, "Learning to assign orientations to feature points", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 107–116.

[58] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors", in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 118–126.

[59] Y. Ono, E. Trulls, P. Fua, and K. M. Yi, "Lf-net: Learning local features from images", in *Advances in neural information processing systems*, 2018, pp. 6234–6244.

[60] A. B. Laguna, E. Riba, D. Ponsa, and K. Mikolajczyk, "Key. net: Keypoint detection by handcrafted and learned cnn filters", *arXiv preprint arXiv:1904.00889*, 2019.

[61] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas, "Working hard to know your neighbor's margins: Local descriptor learning loss", in *Advances in Neural Information Processing Systems*, 2017, pp. 4826–4837.

[62] T. Melo, A. M. Mendonça, and A. Campilho, "Creation of retinal mosaics for diabetic retinopathy screening: A comparative study", in *International Conference Image Analysis and Recognition*, Springer, 2018, pp. 669–678.

[63] A. Levin, A. Zomet, S. Peleg, and Y. Weiss, "Seamless image stitching in the gradient domain", in *European Conference on Computer Vision*, Springer, 2004, pp. 377–389.

[64] J. Pan and M. Wang, "A seam-line optimized method based on difference image and gradient image", in *2011 19th International Conference on Geoinformatics*, IEEE, 2011, pp. 1–6.

[65] T. Köhler, A. Heinrich, A. Maier, J. Hornegger, and R. P. Tornow, "Super-resolved retinal image mosaicing", in *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, IEEE, 2016, pp. 1063–1067.

[66] S. Lee, M. D. Abràmoff, and J. M. Reinhardt, "Retinal image mosaicing using the radial distortion correction model", in *Medical Imaging 2008: Image Processing*, International Society for Optics and Photonics, vol. 6914, 2008, p. 691 435.

[67] S. Bano, F. Vasconcelos, M. T. Amo, G. Dwyer, C. Gruijthuijsen, J. Deprest, S. Ourselin, E. Vander Poorten, T. Vercauteren, and D. Stoyanov, "Deep sequential mosaicking of fetoscopic videos", in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2019, pp. 311–319.

[68] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection", *arXiv preprint arXiv:2004.10934*, 2020.

[69] AlexeyAB, *Darknet yolov4/scaled-yolov4/yolo - how to train to detect your custom objects*, https://github.com/AlexeyAB/darknet#how-to-train-to-detect-your-custom-objects, (Accessed on 20/09/2021).

[70] hunglc007, *Convert yolo v4 .weights tensorflow, tensorrt and tflite*, https://github.com/hunglc007/tensorflow-yolov4-tflite, (Accessed on 20/08/2021).

[71] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context", in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[72] *Wandb*, http://www.wandb.ai, (Accessed on 20/09/2021).

[73] H. Yuen, J. Princen, J. Illingworth, and J. Kittler, "Comparative study of hough transform methods for circle finding", *Image and vision computing*, vol. 8, no. 1, pp. 71–77, 1990.

[74] A. Riquelme, J. Bouchard, A. Desbiens, and R. Villar, "Bubble detection in flotation columns based on circular hough transform", *World Min. Congr.*, vol. 1, pp. 2–10, Jan. 2013.

[75] N. Otsu, "A threshold selection method from gray-level histograms", *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[76] C. Kimme, D. Ballard, and J. Sklansky, "Finding circles by an array of accumulators", *Communications of the ACM*, vol. 18, no. 2, pp. 120–122, 1975.

[77] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance", *Climate research*, vol. 30, no. 1, pp. 79–82, 2005.

[78] D. Zhou, J. Fang, X. Song, C. Guan, J. Yin, Y. Dai, and R. Yang, "Iou loss for 2d/3d object detection", in *2019 International Conference on 3D Vision (3DV)*, IEEE, 2019, pp. 85–94.

[79] F. Lundh, "An introduction to tkinter", *www.pythonware.com/library/tkinter/introduction/index.htm*, 1999.

APPENDIX

APPENDIX A

In this appendix is presented the visual comparison between multiple image channels. The main point taken into account when choosing the images from the D-Eye dataset was the diversity of features, *i. e.*, the images should be the most different possible for one to another for a fair comparison. To obtain the black and white image, the Otsu threshold was applied. The channels from the several color spaces under analysis, which visually appeared to have better characteristics for detecting the retina have been marked with a red border.

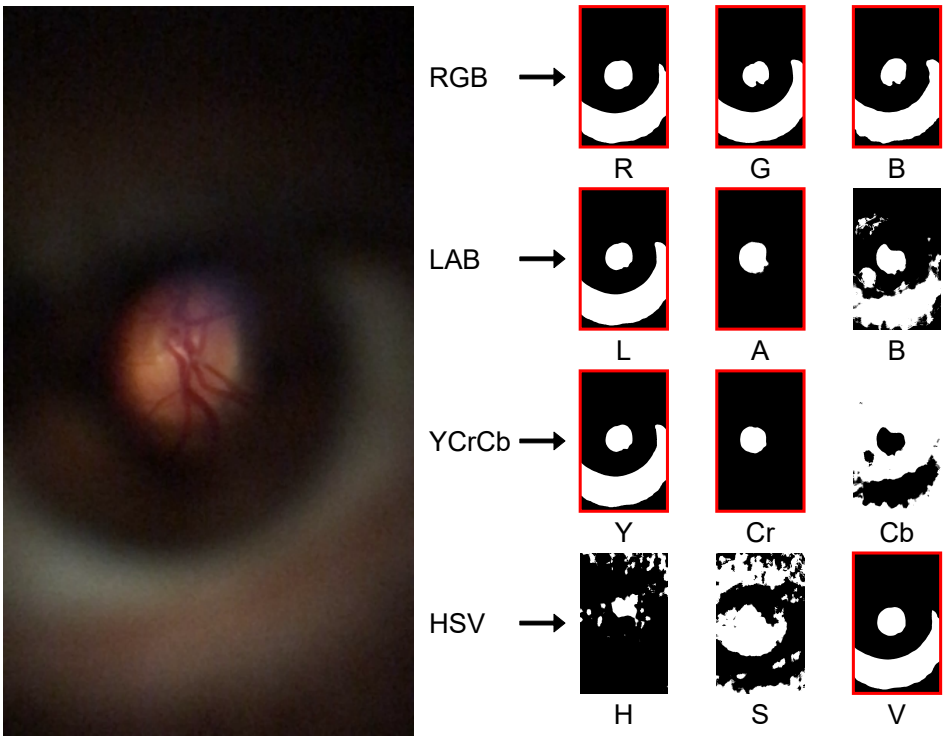


Figure 44: Example #1 of the Otsu threshold applied to multiple color space channels.

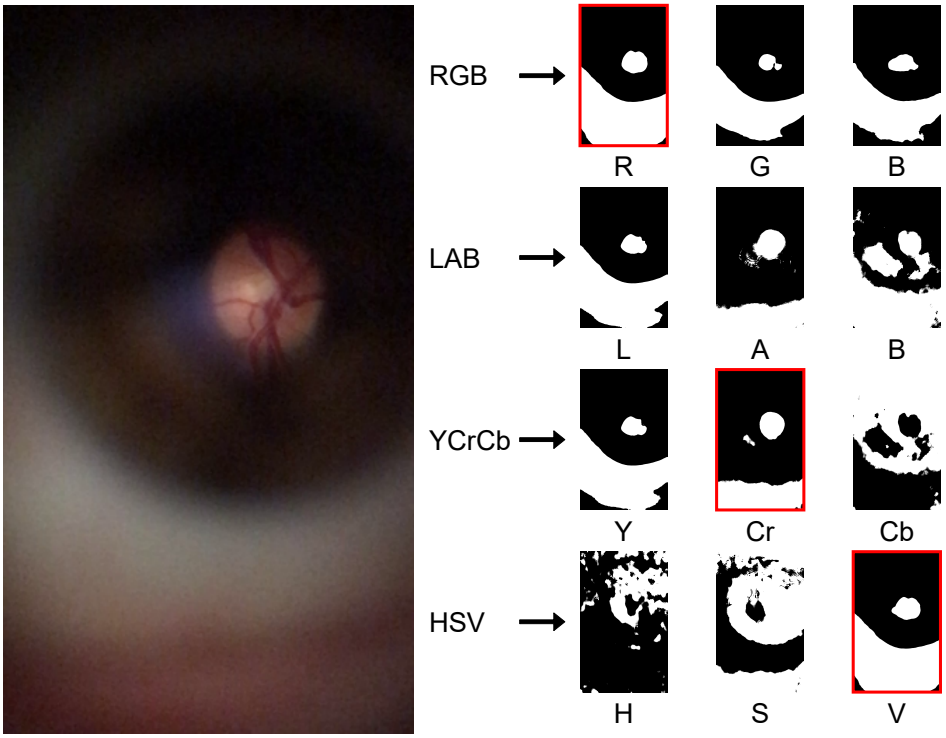


Figure 45: Example #2 of the Otsu threshold applied to multiple color space channels.

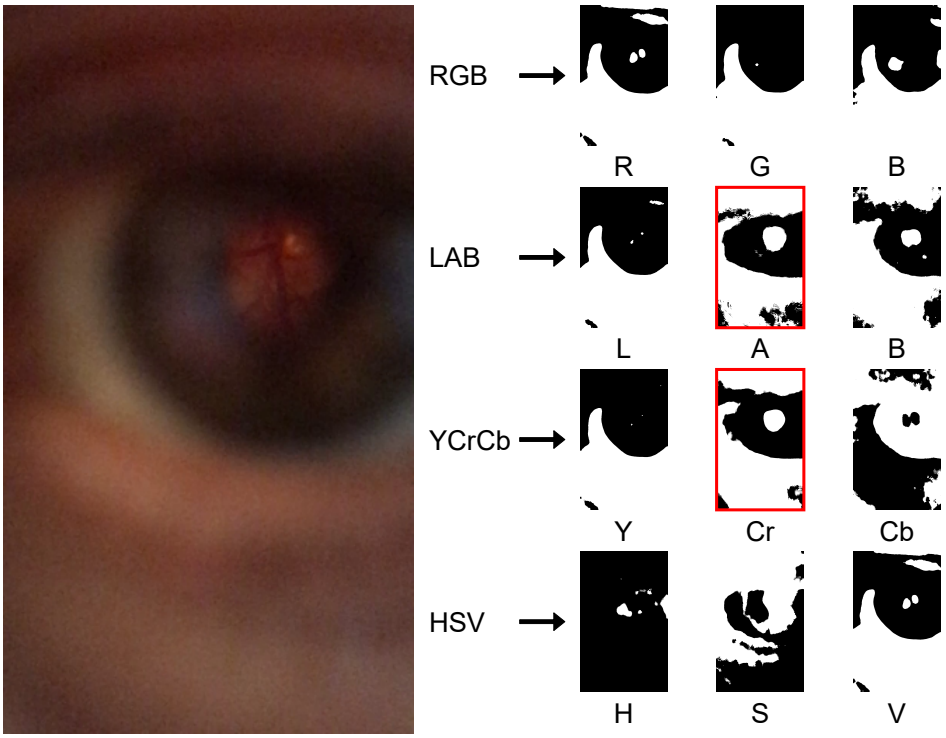


Figure 46: Example #3 of the Otsu threshold applied to multiple color space channels.

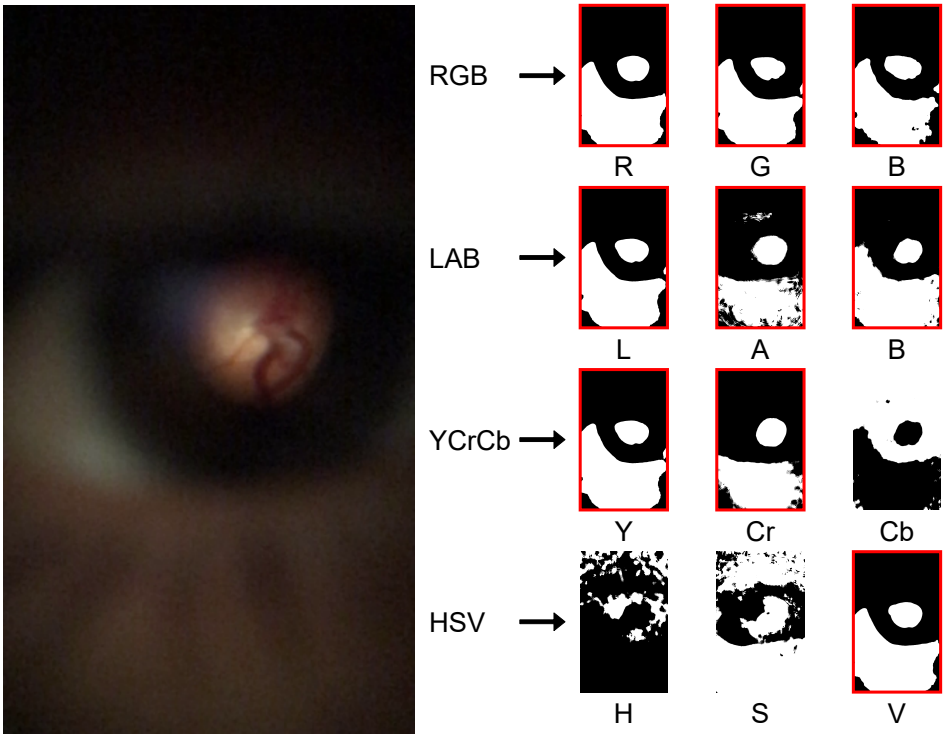


Figure 47: Example #4 of the Otsu threshold applied to multiple color space channels.

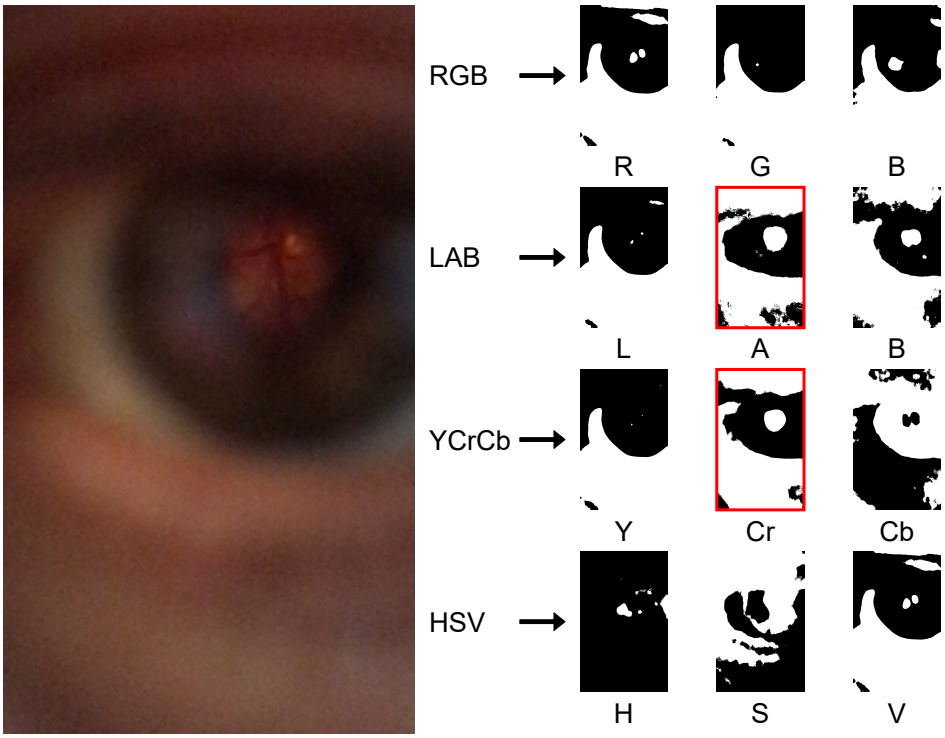


Figure 48: Example #5 of the Otsu threshold applied to multiple color space channels.

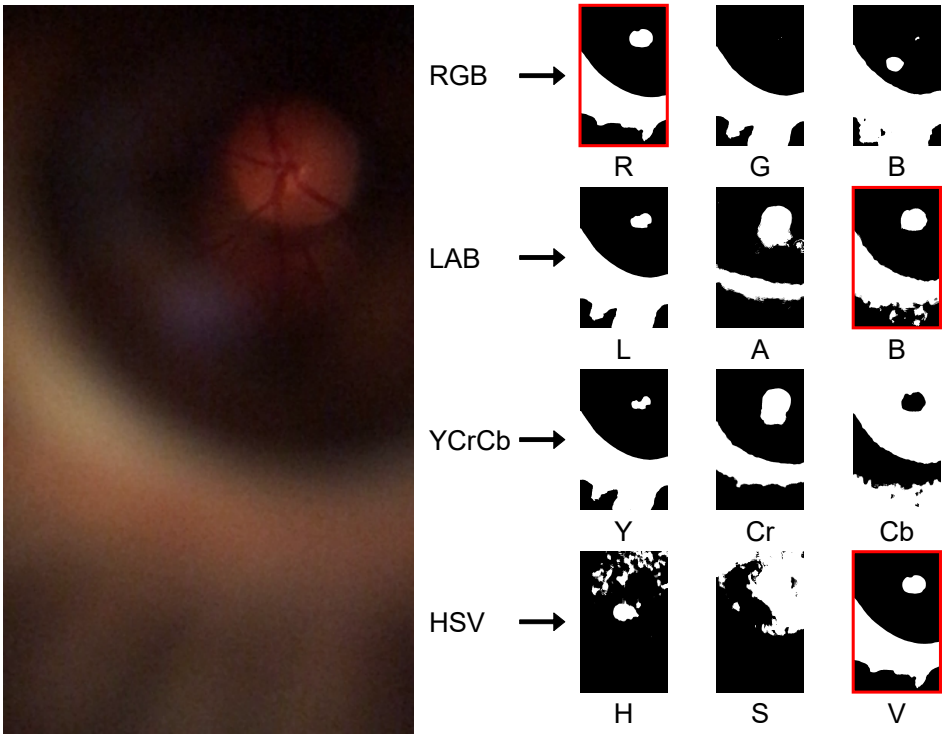


Figure 49: Example #6 of the Otsu threshold applied to multiple color space channels.

B

APPENDIX B

Installing Darknet on Windows 10

Bruno Reis Silva, 2182734

Master's in Electronic Engineering

Polytechnic of Leiria

Installing CUDA and cuDNN

1) Download Visual Studio 2019 Community.
2) Install it.
3) Open Visual Studio Installer and install "Desktop development with C++".

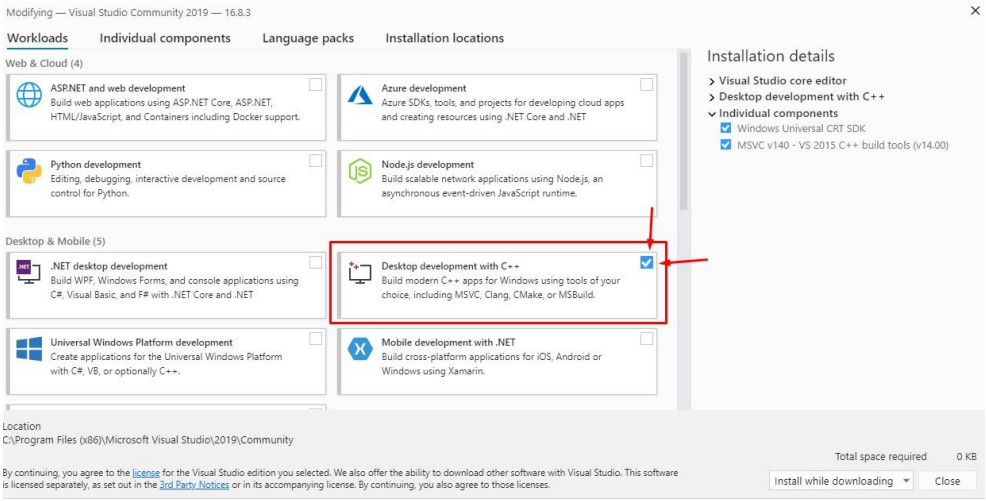


Figure 1 - Install "Desktop development with C++".

4) Check if you have a GPU on your PC. For this access you Device Manager with right click on Start Windows Button and choose "Device Manager" option.

Figure 2 - Device Manager option.

5) Go to “Display adaptors” and if you have an NVIDIA it means that you have a GPU. In Figure 3) case, the computer has NVIDIA GeForce 940M.

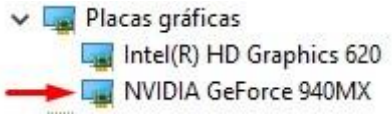
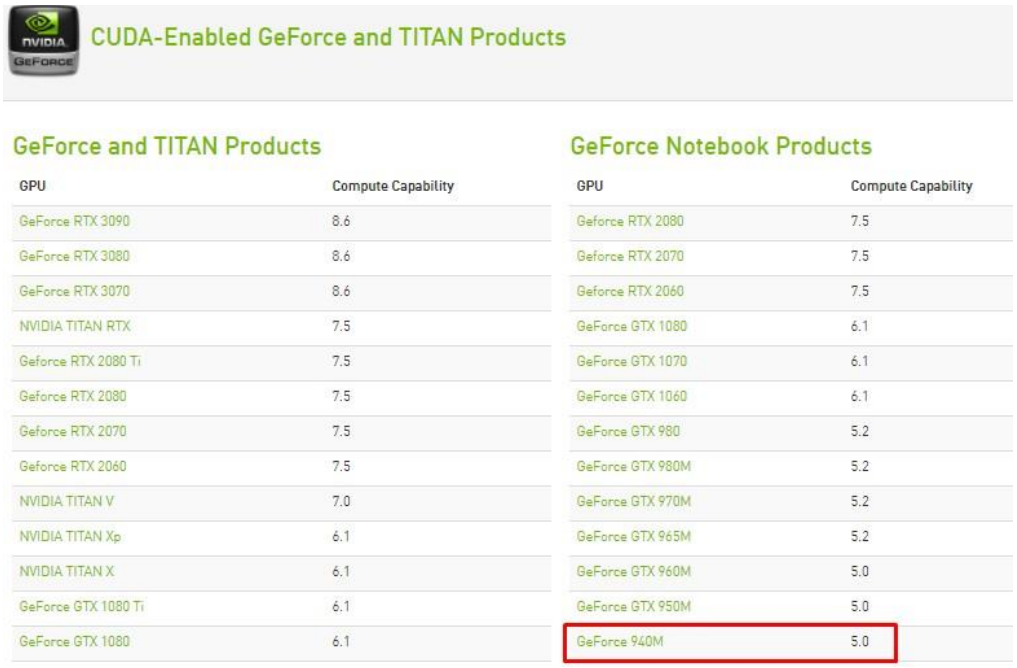


Figure 3 - See what GPU you have.

6) Go to this NVIDIA site and check the GPU is compatibility version. In this case, is version 5.0.



CUDA-Enabled GeForce and TITAN Products

GeForce and TITAN Products

GPU	Compute Capability
GeForce RTX 3090	8.6
GeForce RTX 3080	8.6
GeForce RTX 3070	8.6
NVIDIA TITAN RTX	7.5
Geforce RTX 2080 Ti	7.5
Geforce RTX 2080	7.5
Geforce RTX 2070	7.5
Geforce RTX 2060	7.5
NVIDIA TITAN V	7.0
NVIDIA TITAN Xp	6.1
NVIDIA TITAN X	6.1
GeForce GTX 1080 Ti	6.1
GeForce GTX 1080	6.1

GeForce Notebook Products

GPU	Compute Capability
Geforce RTX 2080	7.5
Geforce RTX 2070	7.5
Geforce RTX 2060	7.5
GeForce GTX 1080	6.1
GeForce GTX 1070	6.1
GeForce GTX 1060	6.1
GeForce GTX 980	5.2
GeForce GTX 980M	5.2
GeForce GTX 970M	5.2
GeForce GTX 965M	5.2
GeForce GTX 960M	5.0
GeForce GTX 950M	5.0
GeForce 940M	5.0

Figure 4 - CUDA.

7) Go to NVIDIA site and download CUDA toolkit version 10.1 or higher. You need to create an account for this.
8) Go to CUDA Toolkit Documentation and follow the Installation Guide.
9) Download and install NVIDIA cuDNN.

Installing OpenCV

1) Download OpenCV Sources. It will download a zip file.
2) Extract the zip to a folder named “OpenCV”.
3) Download the corresponding version of the OpenCV contrib from GitHub. To do this, select the tag that has the saved version of your OpenCV source (Figure 5). After this download the zip (Figure 6).

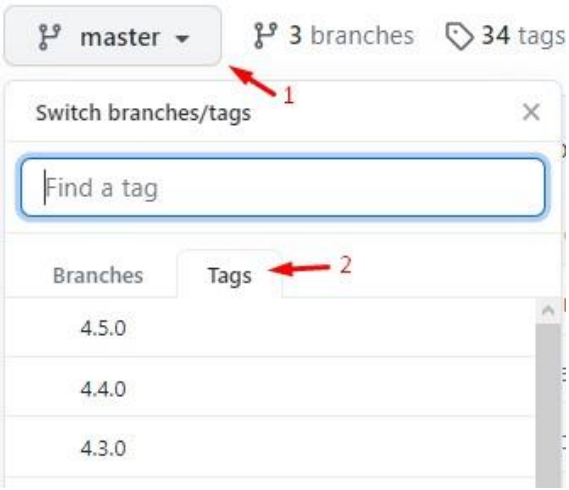


Figure 5 - Select the correct tag.

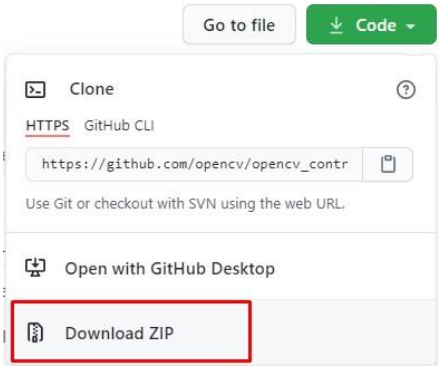


Figure 6 - Download zip.

4) Extract the contrib zip to the folder named “OpenCV”.
5) Create a folder named “build” in the “OpenCV” folder. At this moment you should have something like the following Figure 7.

Figure 7 - "OpenCV" folder.

6) Download CMake version >= 3.8.

Figure 8 - CMake download.

7) Install CMake. Add CMake to the system PATH for all users.

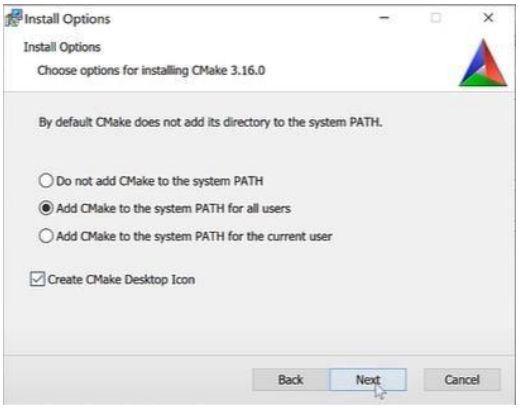
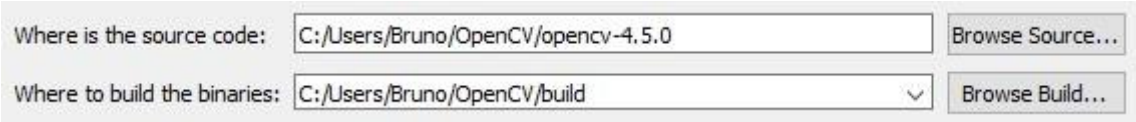


Figure 9 - Installing CMake.

8) Open CMake.
9) Click on the “Browse Source” button and select the OpenCV base folder of your “OpenCV” folder. Click on the “Browse Build” button and select the “build” folder that you created on the “OpenCV” folder.



10) Click on the “Configure” button and configure like in Figure 10. Then click “Finish”.

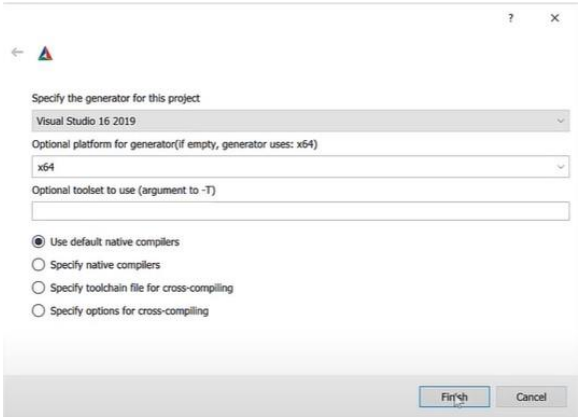


Figure 10 - Configure.

11) Red lines will appear on the screen, like in Figure 11. Scroll and find the “OPENCV_EXTRA_MODULES_PATH”.

Figure 11 - Red lines.

12) Insert there the path of the folder “modules” that is inside of your “opencv_contrib”.

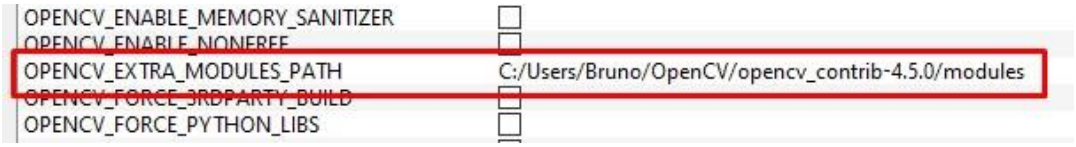


Figure 12 - Path to the "modules" folder.

13) Enable the “WITH_CUDA” box.

Figure 13 - Enable box.

14) Click on the “Configure” button.
15) In “CUDA_ARCH_BIN” delete the version numbers that are lower than your version that was found in Figure 4. If you have version 5.2 delete all below 5.0 and maintain 5.0, if you are 7.5 delete version below 7.0 and maintain 7.0.

Figure 14 - CUDA_ARCH_BIN.

16) Click on the “Configure” button. If red lines appear again then click on the “Configure” button again.
17) Click on the “Generate” button.
18) Open the “build” folder that is inside the “OpenCV” folder.
19) Double click on the file “OpenCV.sln”. Visual Studio should open.
20) On the top bar, turn “Debug” to “Release”, like Figure 15.

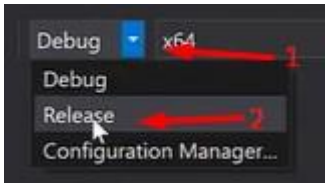


Figure 15 - Turn to "Release".

21) Expand “CMake Targets” (Figure 16), then right click on “ALL_Build” and then “Build” (Figure 17). If everything goes well you should have no errors.

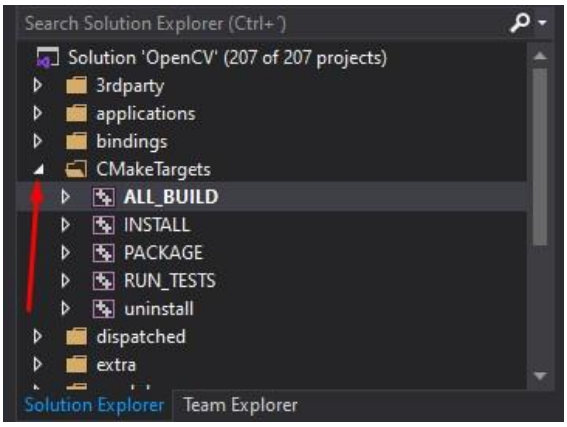


Figure 16 - Expand "CMake Targets".

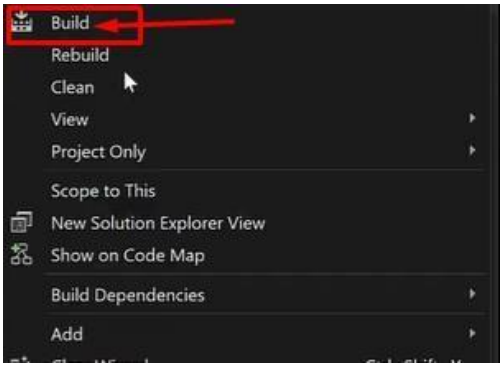


Figure 17 - Choose "Build".

22) Right click on “INSTALL” and then “Build”.
23) To see if it was installed correctly open the CMD and type the following commands:

>>python

>>import cv2

>>print(cv2.__version__)

24) If cmd output OpenCV version, then it was correctly installed.

Installing darknet framework

1) Clone darknet repository from GitHub.

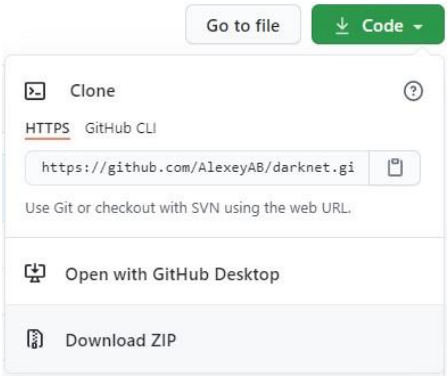


Figure 18 - Clone darknet GitHub repository.

2) Download yolov4.weights and put the file inside the darknet root folder.
3) Edit Makefile like the following image.

```
1  GPU=1
2  CUDNN=1
3  CUDNN_HALF=0
4  OPENCV=1
5  AVX=0
6  OPENMP=0
```

Figure 19 - Makefile editon.

4) Open CMake. Click on the “Browse Source” button and select the downloaded darknet base folder. Click on the “Browse Build” button and select the “build” inside the darknet folder.

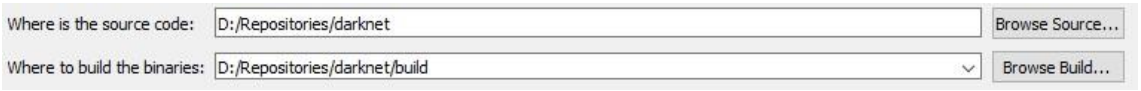


Figure 20 - CMake selections.

5) Click on the “Configure” button and configure the windows like the following Figure 21. Then click Finish.

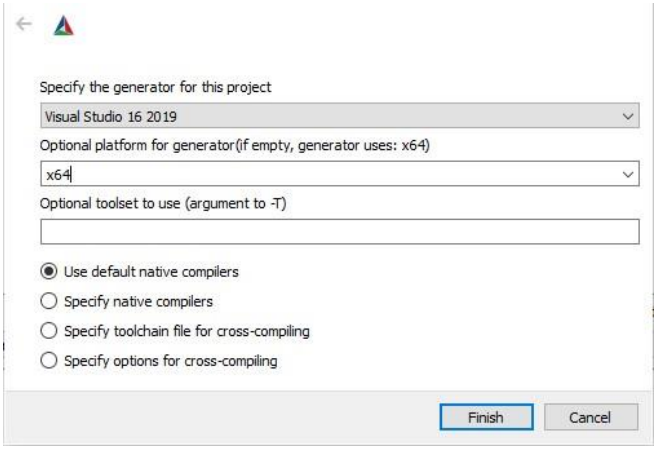


Figure 21 - Configure CMake.

6) Check the “Advanced” box.

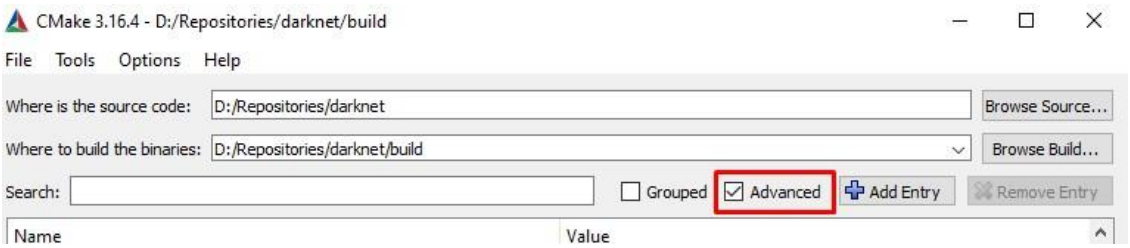


Figure 22 - Advanced box.

7) If “CMAKE_CUDA_COMPILER” appears as NOT FOUND then you have CUDA badly installed. Try to re-install. It should automatically fill the space with something similar to Figure 23.

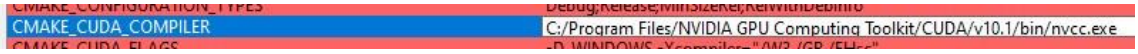


Figure 23 - CMAKE_CUDA_COMPILER.

8) In OpenCV_DIR select the path to build folder inside OpenCV folder.

Figure 24 - OpenCV_DIR.

9) Enable the “ENABLE_CUDA”, “ENABLE_CUDNN” and “ENABLE_OPENCV” boxes. Disable “ENABLE_CUDNN_HALF”.

Figure 25 - Enable boxes.

10) Click on “Configure”.
11) Click on “Generate”.
12) Open darknet folder. Then open the “build” folder that is inside.
13) Double click on “Darknet.sln” file. It will open Visual Studio. 14) Change version to “Release”.

Figure 26 - Change from "Debug" to "Release".

15) Right-click on “ALL_BUILD”, then click on “Build”.

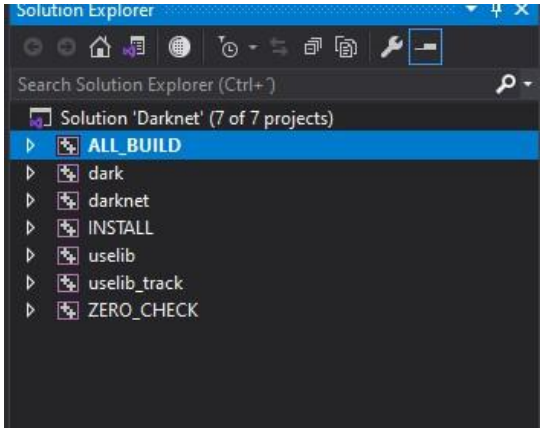


Figure 27 - Right click on "ALL_BUILD" then "Build".

16) Right-click on “INSTALL”, then click on “Build”.
17) Open cmd in the darknet root. Execute the following line:

>>darknet.exe detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights data/person.jpg

18) If any error related to darklib.dll and uselib.exe then:
    - Copy darklib.dll and uselib.exe from darknet/Release folder to the darknet/ root folder.
    - Also copy pthreadGC2.dll and pthreadVC2.dll from darknet\3rdparty\pthreads\bin\ folder to the darknet/ root folder.
19) If any error related to .dll occur then go to OpenCV\build\bin\Release\ and copy the dll file needed to the darknet root folder.
20) The result should be Figure 28.

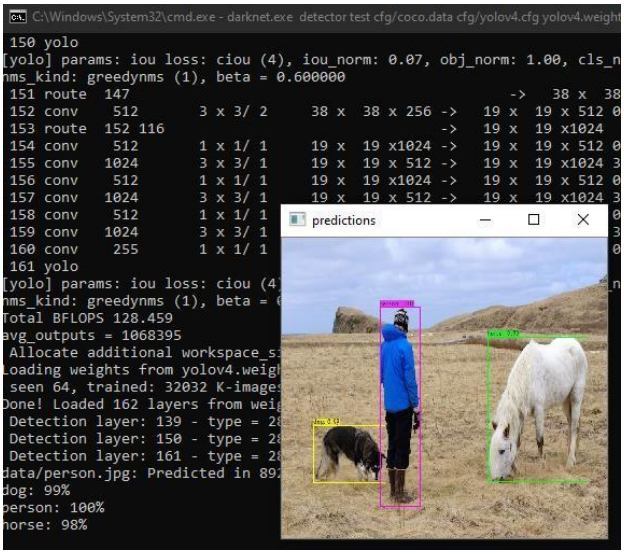


Figure 28 - Result of the cmd line.