Universidade do Minho
Escola de Engenharia

André Vilas Boas da Costa

RetScan: Efficient Fovea and Optic Disc Detection in Retinographies

Outubro de 2011

Universidade do Minho
Escola de Engenharia
Departamento de Informática

André Vilas Boas da Costa

RetScan: Efficient Fovea and Optic Disc Detection in Retinographies

Dissertação de Mestrado
Mestrado de Informática

Trabalho realizado sob orientação de

Alberto José Proença

Outubro de 2011

Universidade do Minho

Declaração RepositóriUM: Dissertação de Mestrado

Nome: André Vilas Boas da Costa

~~Nº Cartão Cidadão~~ /BI: 12835048 Tel./Telem.: 938605261

Correio eletrónico: andrevilasboascosta@gmail.com

Curso: Mestrado em Informática Ano de conclusão da dissertação: 2012

Área de Especialização: Computação Paralela

Escola de Engenharia, Departamento/Centro: Departamento de Informática

TÍTULO DISSERTAÇÃO/TRABALHO DE PROJECTO:

Título em PT : ___

Título em EN : RetScan: Efficient Fovea and Optic Disc Detection in Retinographies

Orientadores Alberto José Proença

Braga/Guimarães, 31 /10 /2012

Assinatura: André Vilas Boas da Costa

Abstract

The Fovea and Optic Disc are relevant anatomical eye structures to diagnose various diseases. Its automatic detection can provide both a cost reduction when analysing large populations and improve the effectiveness of ophthalmologists and optometrists.
This dissertation describes a methodology to automatically detect these structures and analyses a, CPU only, MATLAB implementation of this methodology. RetScan is a port to a freeware environment of this methodology, its functionality and performance are evaluated and compared to the original. The results of both evaluations lead to a discussion on possible improvements in the metodology that influence the functionality and performance. The resulting improvements are implemented and integrated in RetScan. To further improve performance, a parallelization of RetScan to take advantage of a multi-core architecture or a CUDA-enabled accelerator was designed, coded and evaluated. This evaluation reveals that RetScan achieves its best throughput efficiency using a multi-core architecture only and analysing several images at once. For one image usage, using multi-core only is also the best solution, but with a small speed-up. The usage of CUDA-enabled accelerators is not recommended for this scope as the images are small and the cost of the data transfer to and from the accelerator has a severe impact on performance.

Resumo

A Fóvea e o Disco Ótico são estruturas oculares importantes quando se procura diagnosticar doenças no olho. A sua deteção automática permite reduzir o custo de um rastreio a grandes populações e também aumentar a eficácia de oftalmologistas e optometristas.

Nesta dissertação é descrita uma metodologia para detetar estas estruturas automaticamente e é analisada uma implementação em MATLAB desta metodologia. RetScan é o resultado do porte para um ambiente de desenvolvimento com ferramentas livres (open source) desta metodologia. O RetScan é avaliado quer em funcionalidade, quer em performance. Os resultados da avaliação levam a uma reflexão sobre mudanças a realizar à metodologia para melhorar os resultados em ambas as avaliações. Estas melhorias são implementadas e integradas no RetScan. Para melhorar a sua performance é também realizada um paralelização do RetScan de forma a que tire partido de uma arquitetura multi-core ou de um acelerador compatível com CUDA. Após realizar uma nova avaliação conclui-se que o RetScan atinge o seu melhor débito de dados (throughput) quando usa apenas os CPUs numa arquitetura multi-core e analisando várias imagens em paralelo. Para a análise de uma só imagem, o uso apenas de CPUs numa arquitetura multi-core também é o melhor resultado, embora tenha um ganho (speed up) reduzido. O uso de aceleradores compatíveis com CUDA não é recomendado neste âmbito pois as imagens têm um tamanho reduzido e o custo da transferência de e para estes aceleradores tem um grande impacto no tempo total.

Agradecimentos

Ao Professor Alberto Proença, pela orientação, partilha de conhecimento e constante apoio durante o desenvolvimento deste trabalho.
À Critical Health, representada na pessoa do eng. Carlos Manta Oliveira, pelo desafio lançado e por proporcionar uma oportunidade de aplicar os conhecimentos adquiridos durante este mestrado a um problema real e complexo.
Ao João Barbosa e ao Ricardo Alves, pela disponibilidade e ajuda prestada durante este processo.
Ao Dr. Franco pela bondade e disponibilidade em me ajudar a compreender as estruturas e processos fisiológicos que seriam objetivo deste trabalho.
Aos meu pais, pelo seu amor, disponibilidade e apoio incondicional na concretização dos meus objetivos. A concretização de mais uma fase académica é graças a eles, e por isso tenho que lhes agradecer por tudo.
Ao meu irmão, que é o grande responsável pelo meu interesse pela informática, e me ensinou as bases da computação desde cedo.
Finalmente, aos meus amigos, pela presença constante em todas as etapas importantes da minha vida e em que esta não é exceção.

Contents

Chapter 1

Introduction

1.1 Context

Critical Health[1] has been developing products aimed to reduce costs in healthcare by automating several stages of the healthcare service. Their products accelerate the healthcare service and provide more information for everyone involved, reducing the chance of errors.

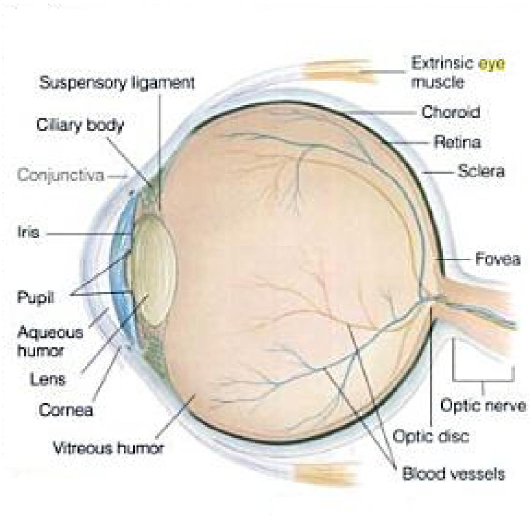


Figure 1.1. Schematic diagram of the human eye with the fovea and the optic disc on the right. Image from [3] page 220

The automatic detection of the fovea and the optic disc is included in decision

[1]Critical Health's website: http://www.critical-health.com/about_us.php

support systems based on image analysis properties from retinographies[2], namely the Retmarker family of solutions.

The fovea (*fovea centralis*) is a region of the eye, located in the centre of the macula region of the retina [3] (Fig. 1.1). The fovea is the region in the eye where the vision is formed and the closer lesions are to this area, the greater is their impact on the reduction of central vision.

The optic disc (or optic nerve head) is the beginning of optic nerve. It connects eye and brain and is also the entry point of the major blood vessels that supply the retina [3] (Fig. 1.1). It has no light sensitive photo-receptors to respond to a light stimulus and this causes a break in the visual field called the "physiological blind spot". It is the brightest region of the retinography, and this affects the performance of image processing algorithms. A detailed analysis of the retinography requires the location of the optic disc to mask it out of the image.

1.2 Objectives

The objective of this work is to analyse the existing work on automatic detection of eye structures, which includes a literature search for algorithms that can be used and a full analysis of the implemented algorithm by Pinão [6]. Furthermore this code should be ported to a programming language with available open source tools. A functional and performance analysis should be conducted and a critical analysis should be produced. Finally, an effort to parallelize the code and take advantage of multi-core architectures, CUDA-enabled GPU devices or both should be made.

1.3 Dissertation structure

In the second chapter the state of the art is debated, the methodology is described and the existing code is analysed. In the third chapter the available tools are described, the challenges overcame are reviewed, the RetScan structure is presented and its effectiveness is analysed. In the fourth chapter an critical analysis is compiled, improvements are discussed and their impact is analysed. Moreover, the parallelization strategies are discussed and analysed for multi-core environments and with a CUDA-enabled accelerator. This dissertation is finnalyzed with a critical overview of RetScan capabilities and it is outlined future work to improve them.

[2]eye fundus photographs

Chapter 2

Optic disc and fovea detection

2.1 State of the art

The detection of the optic disc and fovea has been the subject of much work, and other strategies have been used to successfully identify them. Foracchia et al. used the fact that the major blood vessels in the eye originate from the optic disc, to identify it[4], but this approach requires a version of the parabola algorithm for each data set, and thus is not reliable for large population analysis. Also this approach only identifies the centre of the optic disc but not its radius. Tobin also used the blood vessels to detect the optic disc, but in his approach he searches for the thicker blood vessels that come from within the optic disc[8]. This approach only detects the optic disc of a certain type of retinograpies and does not include every case this work is intended to analyse. Sinthanayothin et al. use the contrast between the bright optic disc and the dark red blood vessels that go through it to detect the optic disc[2]. This approach might have some difficulties detecting the optic nerve in older people because the optic nerve tends to degenerate. Huajun Ying and Jyh-Charn Liu use the dark colour of the macula and fovea and the fact that it is a vessels free zone to identify the fovea[10]. However, this approach only detects the fovea-macula region in certain types of retinograpies.

2.2 Image processing techniques

The proposed approach to do a reliable fovea and optic disc detection is based on shape detection using the Hough transform to detect the optic disc and then detect the fovea assuming that the fovea is located at two and a half optic disc radius from its centre [9]. This methodology uses numerous image processing techniques beside the Hough transform, such as image resizing, grey scaling, Sobel operator, an average and a median blur, a Gaussian gradient, thresholds, contrast enhancement, regions of interest, image dilation and erosion, image subtractions, image masking, and image quantization. These techniques have already been implemented in previous work by Pinão [6] and as been used has an effective method of detecting the optic disc and fovea automatically.

2.2.1 To detect the optic disc

Applying the Hough transform is usually a time consuming process, and a good manner in which to accelerate the transform is to reduce the number of pixels in the image. The fewer pixels it is applied to, the quicker it is. Also, applying the Hough transform to the unaltered retinography would result in the detection of the most circular shape, the full eye. A few image-processing functions must be applied to the image to ensure that the Hough transform is successful:

- **Image resize** This methodology has been refined through heuristic methods, as a consequence, several parameters are adjusted for a fixed eye size, and therefore the eye is identified and the image is resized, focusing in the eye to meet this requirements, as seen in fig 2.1. All images are resized to a fixed height while maintaining its aspect ratio.

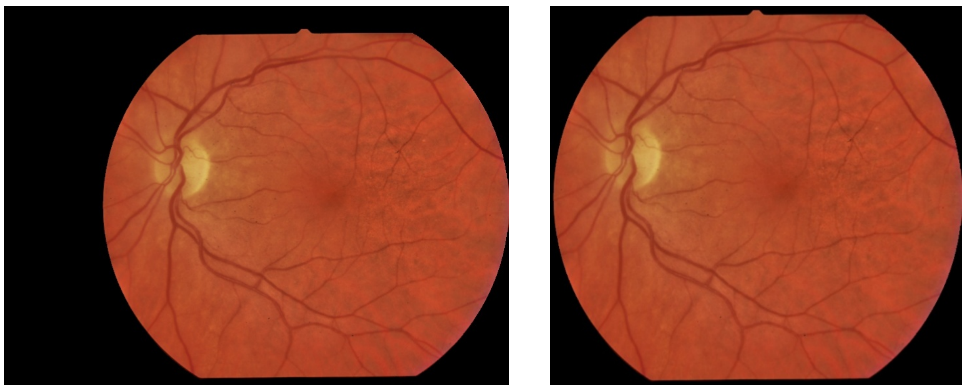


Figure 2.1. The original image (left) and its resized version (right)

- **Noise reduction** Since the eye is an irregular surface with different structures, the colour of the retinography has many fluctuations, and this can cause the Hough transform to produce erroneous results. To reduce the colour fluctuations, a blur is applied to the image (fig. 2.2 (left)). This reduces the colour fluctuations but still preserves the anatomical structures of interest, i.e, the optic disc, the fovea and the blood vessels.

- **Computation of the region of interest** The Sobel operator calculates the variance in colour intensity, and can be used to find the most likely area of the optic disc. Because the optic disc is bright and is the entrance point of the blood vessels in the eye, it usually has the most colour variations. Also, because the vessels tend to spread to the rest of the eye in a hyperbolic shape centred in the optic disc, an horizontal derivative kernel is used in the Sobel operator (fig. 2.2 (right)).

The region of interest (ROI) is the area, with 31.25% the size of the image, which has the highest sum of the absolute values of applying the Sobel operator.

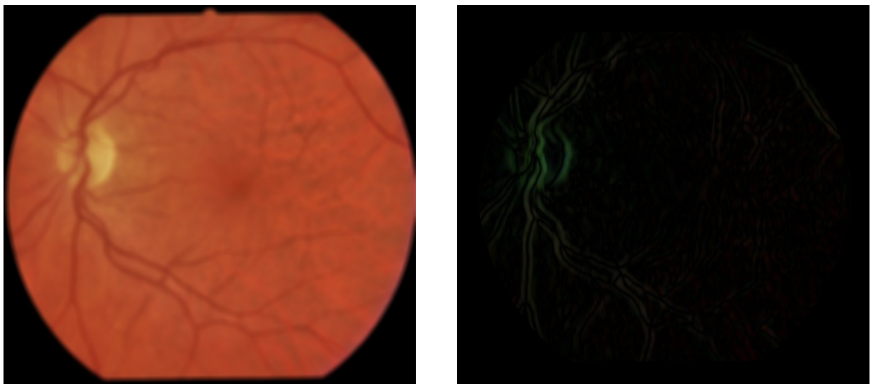


Figure 2.2. The blurred image (left), and the result of applying the Sobel operator in each colour channel (right)

- **Gauss gradient**. The Gaussian filter is a widely known filter for image processing [5]. In the existing implementation a two dimension Gaussian filter is applied, using the first order derivative of the Gaussian function to improve the contrast between the background and the foreground, thus improving the performance of the Hough transform. Two parameters define the derivative Gaussian kernel: the standard deviation which controls the width of the Gaussian function and the kernel size (fig. 2.3).
- **Vessel detection**. The structures detected by applying the Gaussian gradient are the optic disc and the blood vessels. The blood vessels are long and thin structures, and exploiting their thinness it is possible to remove them using a median filter applied to its grey scale image. By using a large number of pixels and sorting them, the blood vessels pixels will not be located near the median. This results in an image with the vessels overshadowed. Subtracting this image from the initial grey scaled image, and turning it into black and white, generates a blood vessels mask (fig. 2.4).
- **Image quantization**. By using a greyscale of the Gauss gradient image and quantizing it into 32 distinct intensity levels, selecting the levels most likely

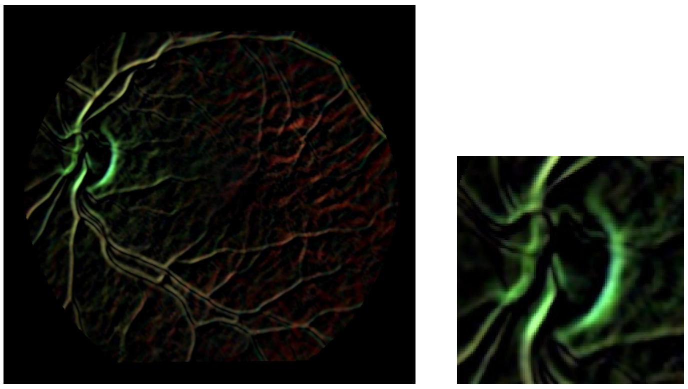


Figure 2.3. The detected gradient using a Gaussian kernel (left), and the same image focused on the earlier computed ROI (right)

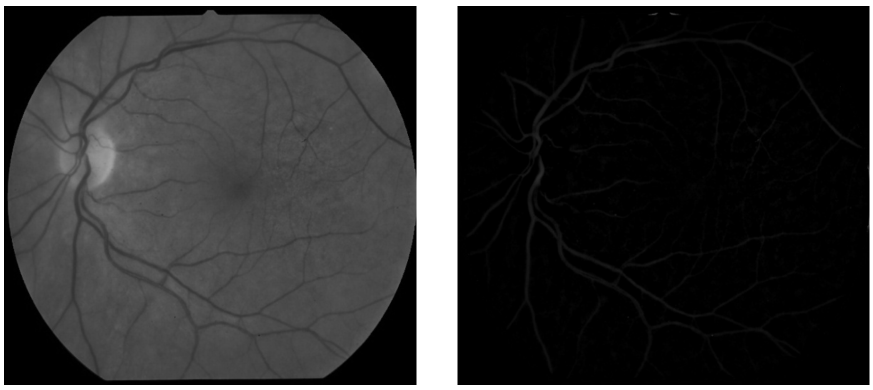


Figure 2.4. The eye grey scale (left) and the extracted vessel trace (right)

to contain the optic disc and discarding all others, can reduce the number of pixels analysed by the Hough transform without compromising its results. This image is then turned to generate a black and white image. The image passed to the Hough transform is the binarized image from the Gaussian gradient subtracted of this image and the vessel trace (fig. 2.5 (left, centre left and centre right)).

- **Hough transform**. The Hough transform is a technique that can be used to extract or isolate features of a particular shape within an image [5]. It requires that a parametric form of that shape must exist, which makes it a tool for the detection of straight lines or regular curves, such as circles or ellipses. Heuristics have shown that the optic disc is most circular shape with a 43 to 64 pixel radius (fig. 2.5 (right)).

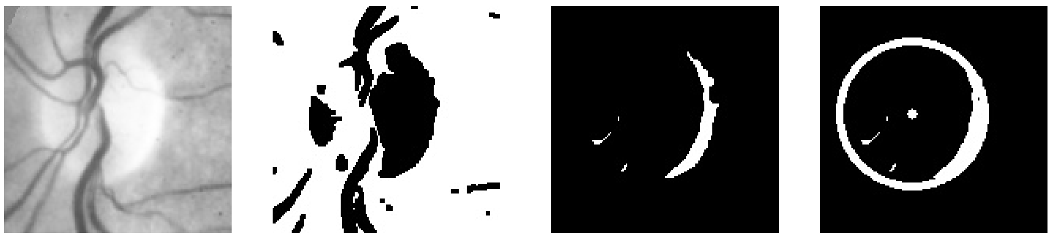


Figure 2.5. The grey scale ROI (left), the mask generated through the quantization step (centre left), the image passed to the Hough transform (centre right), and a representation of the detected circle (right).

2.2.2 To detect the fovea

The fovea is a dark point located 2.5x the radius of the optic disc from its centre[9, 3]. After successfully detecting the optic disc, a circle is defined with 2.5 times the radius of the optic disc, and centred on it. As the protocol defines, the fovea should be near the horizontal axis of the retinography[1] . Therefore a new region of interest is defined using approximately half the image height. The darkest 11x11 square in the region of interest is admitted to be the location of the fovea.

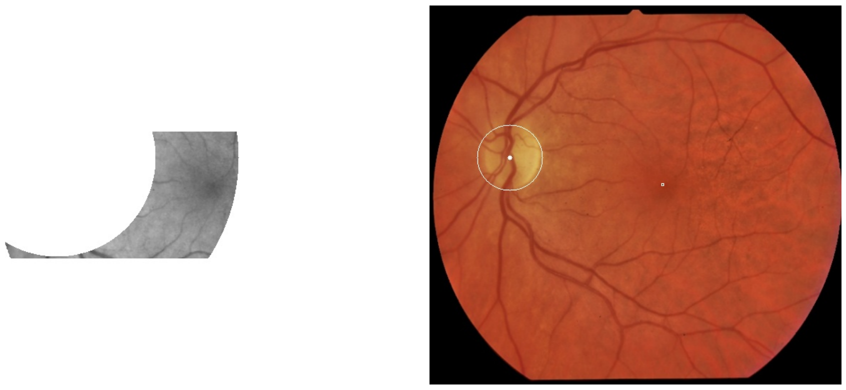


Figure 2.6. The area defined for the fovea search (left) and the final state, with the optic disc and the fovea successfully detected(right)

2.3 Original Code

The optic disc and fovea detection approach previously described was implemented in MATLAB[1], a proprietary software from MathWorks. Before attempting to create an implementation in a language with full support of open source tools, an in-depth analysis of the existing code is required.

An auxiliary tool, a profiler[2] can give insight about the structure of the code, the way it operates, and indicate the most relevant pieces of code. The profiler can provide information about the functions dependencies, the most used functions and the most computational heavy functions.

2.3.1 Code analysis

The approach described previously has a strict division of steps, and actions taken within each step, and the call graph reflects this (Fig. 2.7). The steps include few recursive calls, which makes this call graph a flat tree, most functions are aligned side by side and are called in succession by the main function having low code reuse.

There is a set up, the *od_Test* function, that gets the data ready for the main function. The main function, *detectODandFoveaV2* receives each image and starts by resizing it, calling the respective function, and then calls a function for each of the steps described in the approach. This function call graph (Fig. 2.7) has two features that deserve special notice, the fact that the functions *vesselsVicinity* and *vesselMask* are not called by any other function, which seems to indicate that it was part of deprecated version of the code, and has become obsolete. The *contrast_enhancement* is the only function that is used in different image processing techniques, and the function most susceptible of causing bugs if changed.

2.3.2 Code profile

The MATLAB profiler was used to get statistic of the code in a test where 180 retinographies were processed in a laptop with an Intel i7-2670QM processor (quad-core with Hyper-threading) @2.2 GHz with 32KB instruction cache, 32KB data cache and 256KB of L2 cache *per* core, a shared 6MB L3 cache and 8GB DDR3 RAM. It is running Windows 7 Home Premium and MATLAB 7.11.0.584.

The results are shown in figure 2.8.

Analysing the total execution times it comes as no surprise that the function that prepares the data for execution and then calls the main function and the main

[1]MATLAB: http://www.mathworks.com/products/matlab/

[2]MATLAB has an integrated profiler

function itself have the most total execution time. In fact, the "set up" function execution time is equal to the execution time of the program because every line of code in the program is part of that function, or one of its children.
Furthermore, analysing the total execution times shows that the auxiliary step *clean_boundary* is the function that takes up most of the execution time. This result was not an expected result, but is justified, this function prepares images for the Sobel operator and the Gaussian gradient and has a high total execution time because it is called several times during the execution of the algorithm.
The functions *circle_detection* and *houghcircles* are both related to the execution of the Hough transform, which was expected to be the step with the biggest workload. When analysing the self execution time it becomes apparent that the application of the Hough transform and finding the region of interest are the two techniques that take the most time, as *houghcircles* and *od_detection_quare* are the functions that implement them. The built-in MATLAB functions that appear in both tables are by-products of the execution of *houghcircles* and *clean_boundary.*
This analysis confirms that the Hough transform is indeed a heavy segment of the implementation, but shows that finding the region of interest cannot be neglected. The interesting and surprising result is that the auxiliary function *clean_boundary* is the function with the most total execution time (apart from the two functions that include everything) and should be revised to improve performance.

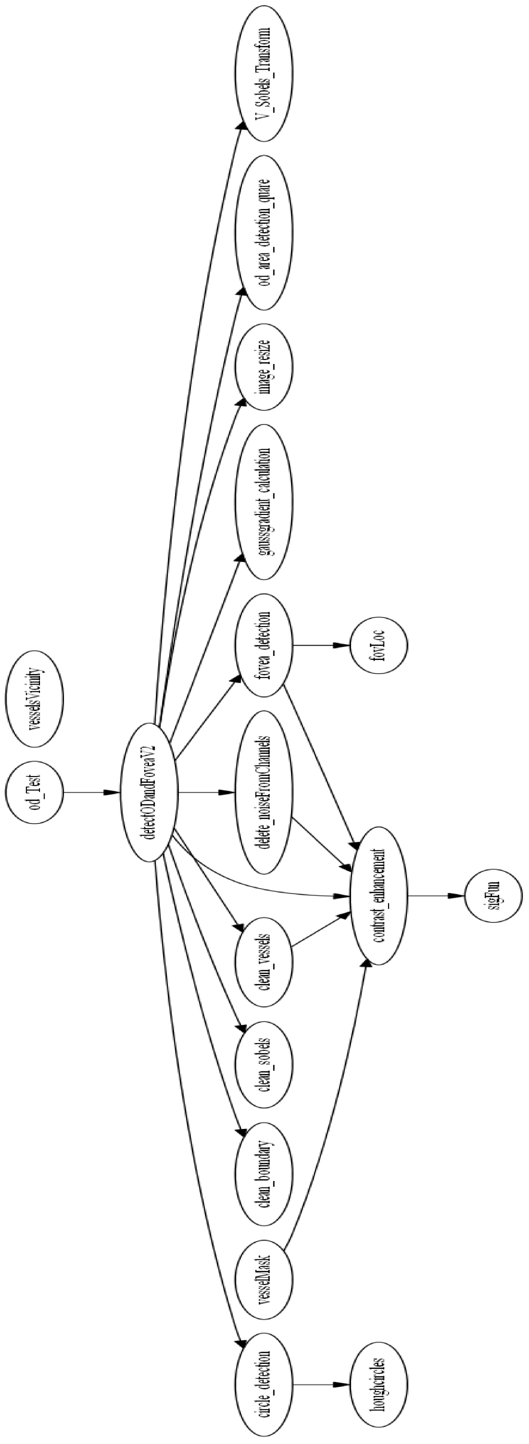
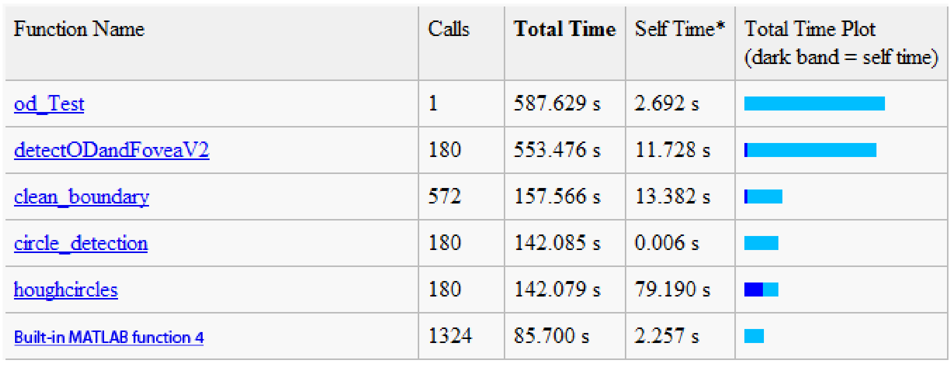
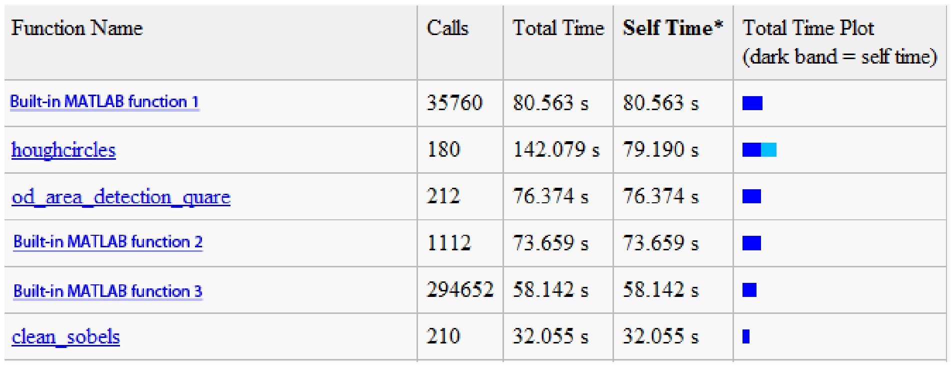


Figure 2.7. The call graph.



Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
od_Test	1	587.629 s	2.692 s	▬
detectODandFoveaV2	180	553.476 s	11.728 s	▬
clean_boundary	572	157.566 s	13.382 s	▬
circle_detection	180	142.085 s	0.006 s	▬
houghcircles	180	142.079 s	79.190 s	▬
Built-in MATLAB function 4	1324	85.700 s	2.257 s	▬



Function Name	Calls	Total Time	**Self Time***	Total Time Plot (dark band = self time)
Built-in MATLAB function 1	35760	80.563 s	80.563 s	▬
houghcircles	180	142.079 s	79.190 s	▬
od_area_detection_quare	212	76.374 s	76.374 s	▬
Built-in MATLAB function 2	1112	73.659 s	73.659 s	▬
Built-in MATLAB function 3	294652	58.142 s	58.142 s	▬
clean_sobels	210	32.055 s	32.055 s	▬

Figure 2.8. Runtime statistics for 6 functions with the highest total execution time (top) and with the highest self execution time (bottom). *Self time is the time spent in a function excluding the time spent in its child functions and also includes the profiling overhead.

Chapter 3

Porting to a freeware environment

3.1 Available environments

The original code was developed in MATLAB, a platform developed by MathWorks, for numerical computation, visualization and programming. MATLAB has a proprietary high-level programming language following a imperative programming paradigm.

The C programming language is arguably the most influential programming language in the world. It is still one of the most used languages and most of the major programming languages have been influenced by it, e.g., C#, Java, PHP, Perl and Python. The C Language has a wide range of open source tools and libraries available, that can be used freely and safely. Most of these tools and libraries have been developed for several years and are still maintained. Due to widespread use and support, there is hardly any risk of any of these tools being abandoned in the foreseeable future.

MATLAB has support for image processing, with several functions in its kernel for colour manipulation and the most common image processing techniques. There are a few open source image processing libraries available for the C language such as the OpenCV[1], VXL[2], IVT[3] and VIGRA[4]. Of these open source projects, OpenCV is the most matured project, with more functionalities and stable multi-platform versions.

[1]OpenCV: **Open** Source **C**omputer **V**ision; http://opencv.org/about.html

[2]VXL: Vision-something-Libraries; http://vxl.sourceforge.net/

[3]IVT: **I**ntegrating **V**ision **T**oolkit; http://ivt.sourceforge.net/

[4]VIGRA: **Vi**sion with **G**ene**r**ic **A**lgorithms; http://hci.iwr.uni-heidelberg.de/vigra/

3.1.1 MATLAB vs C/C++

The MATLAB programming language has a syntax similar to C, with no direct memory access and no control of how the data is stored. The original code uses the RGB colour model[5] which is stored, conceptually, in a tri-dimensional matrix. MATLAB has simple matrix manipulation operators and together with the image processing functions its kernel possesses it was a viable tool for the original code. The MATLAB kernel has a massive number of functions and functionalities supported and as such, it has become a heavy computational load and is a herculean task to squeeze performance out of it.

The C language allows for low level memory access and manipulation, and allows for greater control of the data, as the programmer knows exactly what and how everything is stored. The C language has no native support for image processing, but its functionality can be extended via the use of libraries.

3.1.2 OpenCV

The OpenCV library offers real-time computer-vision support. Released in 1999 by Intel, it is currently supported by Willow Garage[5] and Itseez[6]. It is in version 2.4.2[7] and has support for Windows, Linux, MacOS, Android and iOS.

The OpenCV library also has a CUDA implementation of some of its most used functions. However this project[8] is still in its early stages and so it only has a limited number of functions.

With a reliable CPU implementation and a (small) CUDA implementation of image processing functionalities, combined with the fact that is open source and freely distributed, it makes OpenCV a good choice for library support in the context of this port to a freeware environment.

3.2 Challenges

During the implementation of this port some challenges emerged that had to be solved. These challenges were caused by various factors, namely the insufficient documentation, the specificity of functions required and heuristic parameters.

[5]Willow Garage Website: http://www.willowgarage.com/pages/about-us

[6]Itseez Website: http://Itseez.com/index.php?page=about_us

[7]At the time of the writing there is a Release Candidate for 2.4.3

[8]OpenCV_GPU: http://opencv.willowgarage.com/wiki/OpenCV_GPU

Documentation

To help with the interpretation and understanding of the original code, the [7] document was made available. Despite prividing a good description of the general algorithm, it is not a good description of the original code, as the code was further developed internally by Critical-Health. Some functions in the code are not included in this documentation and others have changed some or all of their criterion. Due to these obvious discrepancies between the documentation and the original code, and the fact that no information about the changes made to the work of Pinão were available, the documentation could not be trusted. There was a need for a full analysis and an in-depth look at each line of the code to find its true functionality, if it was introduced after the documentation was produced, and if not, why had it been changed. This process was time consuming due to the number of lines of code and the fact that some of the used functions and operators required a deeper understanding of their internal structure. The analysis of the code identified these changes:

- Non-documented functions:
  - **“clean_boundary”**; one of the surprising results of the code profile is the weight of an undocumented auxiliary function called “clean_boundary” (Fig. 2.8), which removes the border of the eye from the image by creating a black & white mask of connected components in the red channel, and applying a strong erode to them. This function is called several times during the execution of the original code, and since the position of the eye never changes, in this port the mask is only generated once at the beginning of the execution, the mask is stored and used when necessary.
  - **“contrast_enhancement”**; a contrast enhancement function is also used several times during the execution, which is not documented either.
- Implementation that does not follow the documentation:
  - The use of the **Sobel operator**, and the computation of the optic disc ROI is very different from what the documentation states; the original code has the Sobel operator running from within a cycle that, from what is discernible, executes only once. This is probably a remnant of the noise reduction cycle described in the documentation. Instead, after the smoothing, the Sobel operator is used in each colour channel. For the computation of the ROI, instead of a sum of all values within a region, a count of pixels that have non zero values, but with red or blue values bellow 40 and a green value above 25.

  - The **vessel trace mask** also has significantly different usage than the described; the documentation says it applies the vessel mask in the sobel operator, but that is not the case. Also, the documentation states that the vessel subtraction is done after the applying the Gauss gradient, but in fact it is applied after the quantization but before the small structure clean up described in it.

Function specificity

OpenCV provides a wide variety of image processing functionalities, but some of the image processing techniques used in this methodology are so specific for this usage that cannot use library support. Some functions can still use library support within themselves using some form of tweaking or refactoring, but these functions could not:

- **clean_sobels**: this function is a special case of a threshold, this is a three channel conditional threshold. A traditional threshold uses only one colour channel images, and decides if the colour of a pixel changes depending on its value. This uses a 3 channel image and decides the colour of the pixel depending on different values in each channel a conditional clause relating the three channels.
- **contrast_enhancement**: This function is a contrast enhancement function that uses a sigmoid function to redistribute the colour values in an uniform manner throughout the value range.
- **bwQuantize**: this is can be considered a double threshold, as it is what it does, but in reality is a fused version of many functions. This function quantizes the image, discards the irrelevant levels, and binarizes the image.
- **minChannel**: currently OpenCV does not support the fusing of channels using a pixel by pixel comparison.
- **foveamask**: the fovea ROI mask is an area enclosed by two circumferences centred in the optic disc and has to be generated pixel by pixel.

Heuristic parameters

In the original code, several parameters were calculated using heuristic methods. Despite being a valid method to improve the quality of the results, this can be troublesome when designing a new solution.

MATLAB stores the image in matrix form which allows the representation of colour values in floating point notation, and these values are only rounded when the

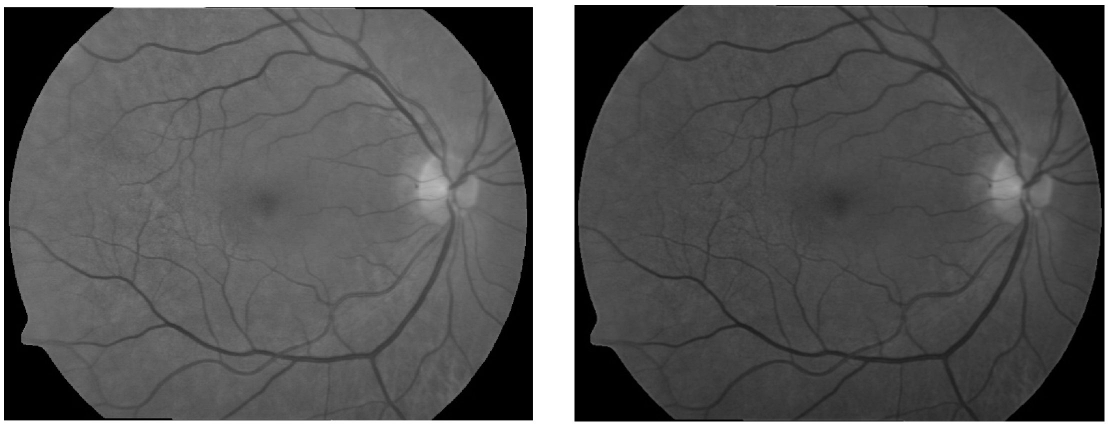


Figure 3.1. A greyscale using MATLAB (left) and a darker greyscale created using OpenCV from the same image (right).

image is written. OpenCV uses a single byte to represent each colour value which can only represent 256 integer numbers (0-255). Considering this, it is necessary to round the values after every operation. In addition, some colour transformations might not be implemented using the exact same kernel values, which can produce slightly different results. Observing figure 3.1, it is possible to compare the results of applying a greyscale to the same image in MATLAB and in OpenCV, which demonstrates how the differences between the software(resources) can be problematic when using heuristically calculated parameters.

3.3 RetScan

RetScan is the port to the C language of the original MATLAB code using the support of OpenCV. The functionality remains the same, and the only difference is the new approach to the clean boundary problem. Instead of calculating and removing the eye boundary before applying the Sobel operator and the Gaussian gradient, a mask is produced when resizing the eye and is applied before them.

3.3.1 Overview

RetScan starts by loading the image to the OpenCV basic image structure, using the RGB colour model (the alpha channel is ignored). As the background of the image is mostly black with colour values, to easily identify the eye, the three channels are fused by summing their colour values. This will maintain the darkness of the background, but will brighten the eye, which allows for an easy eye detection using

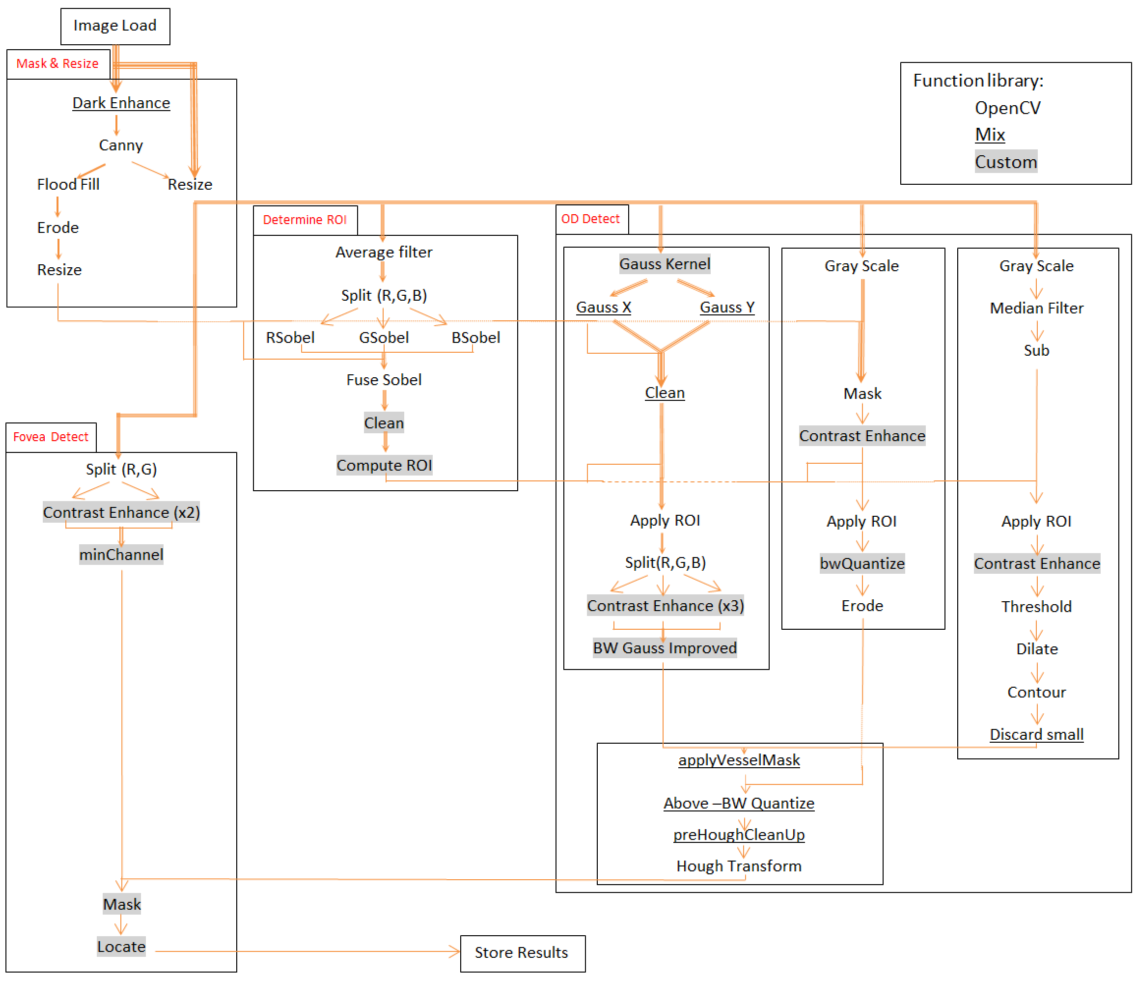


Figure 3.2. A diagram of the RetScan functionalities; each arrow represents data dependencies and its width the amount of data required.

the Canny edge detection filter. After identifying the eye, the image is resized to a standard size, cropping most of the background (fig.3.3).

The result of applying the Canny is also used to create the mask used for the boundary removal. A flood fill starting from the centre of the image paints all the eye white, and an erode is applied to remove the boundaries from the mask. Afterwards, this mask is resized to fit the standard size image (fig.3.3).

To compute the optic disc ROI the image is blurred and the Sobel operator is applied separately to each channel. This image is then fused and prepared using the conditional three channel threshold described previously (section 3.2) and the boundaryless eye mask. The ROI is admitted to be the region with the highest number of remaining pixels(fig. 3.3).

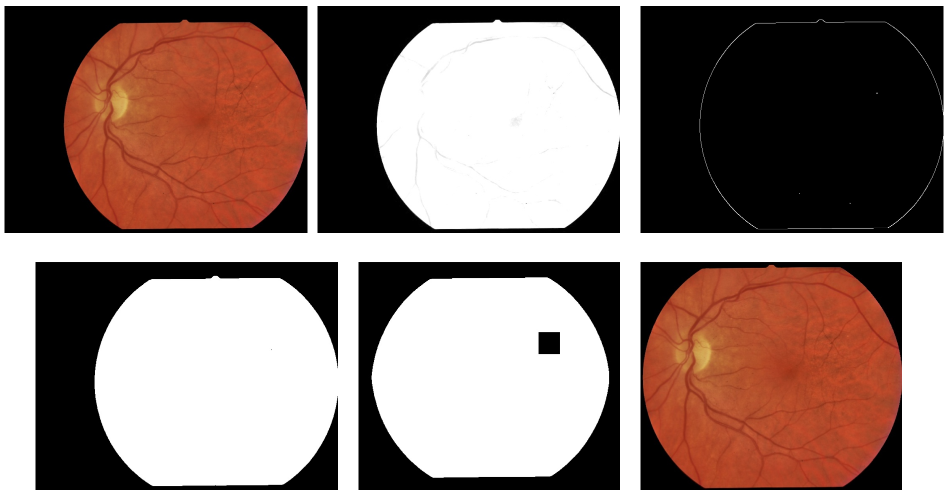


Figure 3.3. An overview of the mask generation and eye identification process. The original image (top left), the sum of its channels (top centre), the Canny detection (top right), the flood fill result (bottom left), the mask after an erosion (bottom centre) and the resized image(bottom right).

The Gaussian kernel is created and used in the horizontal and vertical axis of the image. The results are fused and the boundaries are removed using the eye mask. The image is then reduced to the ROI calculated earlier and binarized after enhancing the contrast of each colour channel independently.

To quantize the image, the grey scale of the resized image is calculated, and the eye boundaries are removed. Afterwards, the colour contrast is enhanced to ensure a good colour redistribution and the image is restricted to the region of interest. Finally, the image is quantized and binarized and eroded to reduce the size of the detected structures.

The vessel mask is computed using the grey scale of the resized image and applying a median filter. This new image does not include the vessels and is subtracted from the grey scale. The result of this subtraction is a faint vessel trace, which is reduced to relevant ROI and its quality is improved by enhancing its contrast. Since other colour variations are also present, a threshold is used to select the vessel trace. This trace is dilated, to ensure the entire vessel is included in this mask. A search for connected components is conducted and small structures are discarded.

The vessel mask is used to remove the blood vessels from the binarized Gaussian gradient image, and the binarized image from the quantization is used to remove other structures, leaving the contours of the optic disc. The Hough transform is then used to detect the circular shape of the disc.

The first step to locate the fovea is to discard the blue channel and enhance the contrast of the red and green channel independently. A single channel image is formed by fusing the channels selecting the lower colour value in a pixel-by-pixel comparison. Using the location of the optic disc and its radius, the fovea ROI can be calculated and the fovea is detected by finding the darkest area in the ROI.

3.3.2 Experimental Validation

The first version of RetScan (or RetScan 1.0) was tested with 200 different retinographies but the overall results are below expectations. The success rate is only 56%, which is far below the 90% efficiency of the original code. A significant amount of failures is due to detecting the optic disc slightly out of its position. This is caused because the filters applied to the Gauss gradient, the vessel mask and the mask generated through quantization, eliminate part of the optic disc, thus the detected circle is either smaller than the optic disc, or is detected out of place (fig. 3.4). The generation of the vessel mask and the mask created by the quantization step errors are due to the heuristic parameters used.

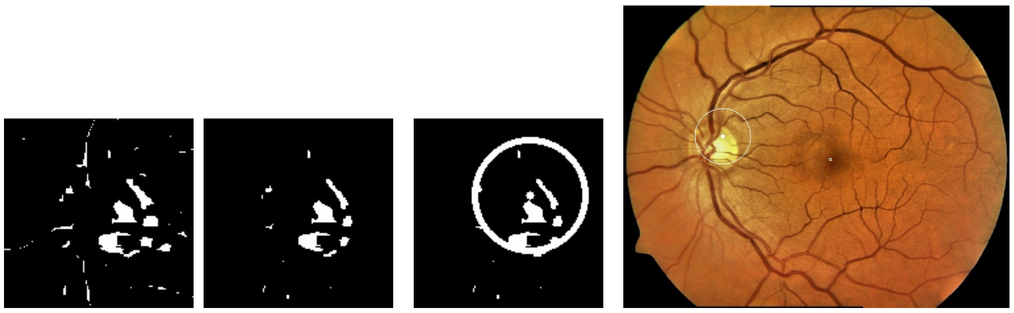


Figure 3.4. An example of an image where the elimination of a significant part of the optic disc causes a wrong detection.

The fovea is always correctly detected when the optic disc is also detected, and is still correctly detected when the optic is detected out of its place.

Assuming that the failed detections can be eliminated through a parameter recalibration, its other failures are due to a erroneous computation of the optic disc ROI. In these cases, the difference in colour between the optic disc and its surroundings is small and the Sobel operator does not differentiate the optic disc from the rest of the eye (fig. 3.5). This is a limitation of this methodology, as the problem comes from the assumption that the optic disc is always brighter and differentiable from the rest of the eye. This limitation is also present in the results of the MATLAB implementation.

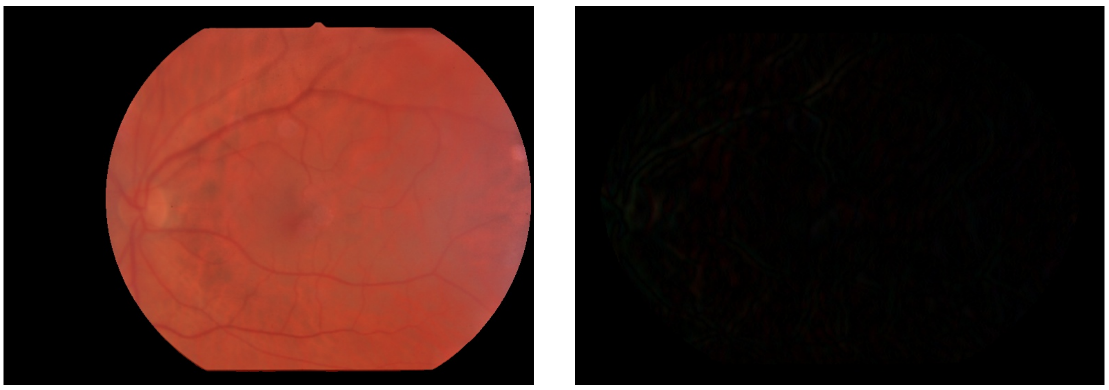


Figure 3.5. An example of an image where the optic disc is not identifiable using the Sobel operator. Original on the left and the result on the right

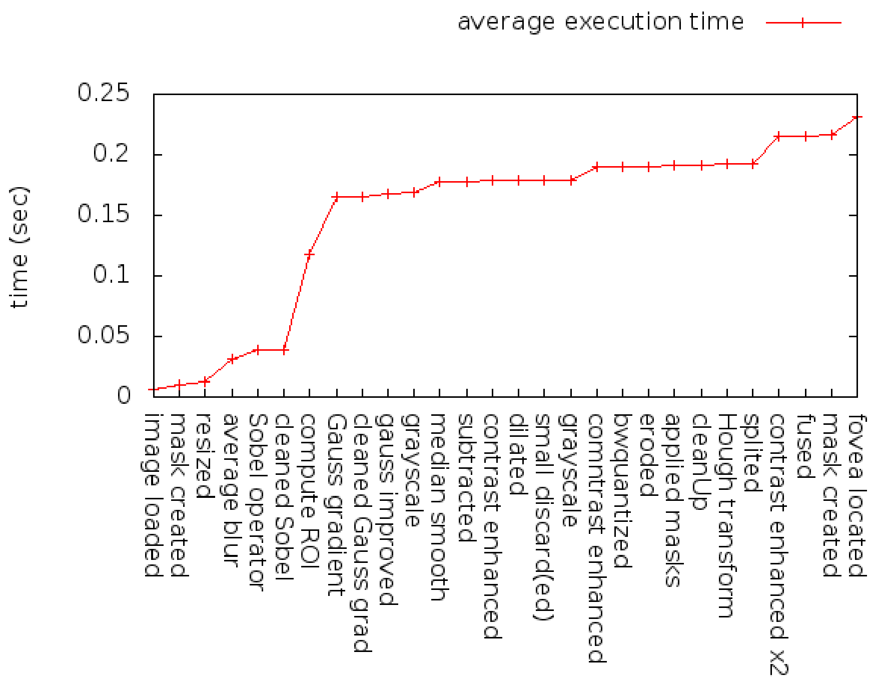


Figure 3.6. The average execution time of RetScan 1.0, and which functions consume more time.

3.3.3 Performance Analysis

The original MATLAB code, set a benchmark of 3.3 seconds per image and can process a sample of 180 images in roughly 10 minutes (section 2.3.2). RetScan significantly improved performance compared to the original code. Each image takes, on average, 228 milliseconds to process, and can process the full data sample, of 200 images, in under 46 seconds (fig. 3.6). RetScan is 14 times faster than its original code.

Profiling the process of a single image, a function stands out, as the heaviest function, the "Compute_ROI". This function is responsible for approximately 40% of the time consumption of RetScan, and further performance improvement efforts should focus on this function. The "contrast_enhancement" function is responsible for approximately 15% of the time, but unlike the "Compute_ROI", its relative weight is due to the high number of function calls (7 per image). Apart from these two functions, no other function has a significant weight on the execution time on its own.

Chapter 4

RetScan revisited

4.1 Critical analysis

RetScan is a freeware version of the original code and has no limitations associated with a proprietary product. RetScan can be used and distributed without cost, and because a C compilers exists for nearly any hardware, its deployment is only dependent on the limitations of the OpenCV library, which currently is restricted to Windows, Linux, MacOS, iOS and Android. RetScan is still not ready for deployment, because of its optic disc detection capabilities are still unreliable, 50.5% change of a successful detection is still far from the required. The RetScan requires an adjustment in its parameters, in order to be a reliable optic disc and fovea detection utility. With a successful calibration of its parameters, RetScan would be a good replacement of its original code, as its performance is indisputably better. RetScan is fourteen times faster than the original code, retaining the same functionality.

4.2 Improvements

Assuming that a calibrated RetScan, with its parameters perfectly adjusted to the new functions of OpenCV and its slightly different colour treatment, there are still areas where the methodology can be improved. The original code has roughly a 90% success rate, but that success rate can still be improved. Also, the pre-Hough transform preparation of the image seems too complicated with some techniques redoing, or destroying some of the work done by others, and a simpler approach might achieve a similar result with a lighter computational load.

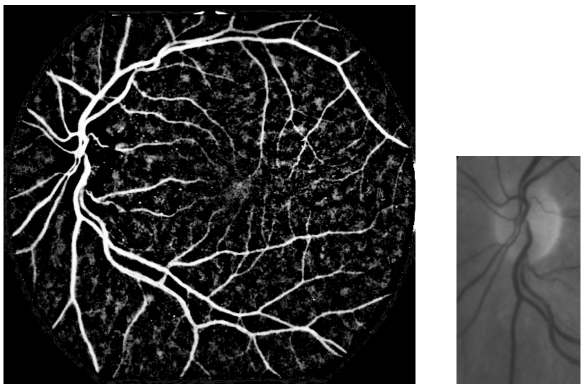


Figure 4.1. The enhanced vessel extraction image (left), and the computed ROI (right)

4.2.1 Functional improvements

In the original code, most of this methodology failures to detect the optic disc, and consequently the fovea, are due to the miscalculation of the optic disc ROI. These are the cases where the “clean_sobels” either eliminates too many pixels or too few.

The blood vessels enter the eye from a single point (optic disc) and tend to spread in an hyperbolic manner, in such a way that the fovea will be located in a area without any blood vessels. This behaviour displayed by the blood vessels can be used to determine the optic disc, as shown by Forracchia et al.[4], Tobin[8] and Sinthanayothin et al.[2]. All these works use the behaviour of the blood vessels as a way to detect the optic disc, but they use several steps with some computational load to achieve it. In RetScan a lightweight method to pinpoint the approximate location of the optic disc might be more appropriate, and to that affect, the vessel trace can be used.

Since the hyperbolic shape of the blood vessels, the blood vessels will be more concentrated near the vertical axis passing through the optic disc, which allows for an horizontal localization of the optic disc. Retinography protocol dictates that the optic disc should be located in the horizontal axis of the image[1], which allows for a vertical localization.

The vessel trace can be produced using the greyscale image of the eye, applying a large median filter and subtracting it from the original image. This vessel trace has mostly dark tones, therefore using a contrast enhancement function improves its quality. The horizontal position can be determined using a rectangular shape, where the vertical size is larger than the horizontal one. Despite the fact that the protocol dictates that the optic disc should be in the horizontal axis, the optic disc position in the retinographies is not always compliant, but seldom strays more than a quarter of the eye size distance.

Impact analysis

This improvement provides a perfect optic disc ROI determination for the 200 retinographies analysed, completely solving the issue of bad ROI, compromising the results.

4.2.2 Performance improvements

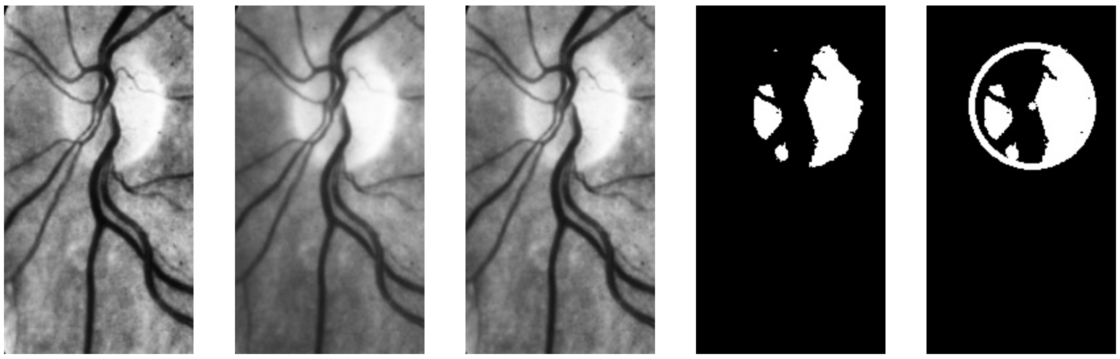


Figure 4.2. The red (left) and green (centre left) colour channels of the ROI image, the fused channels image (centre), the result of the threshold (centre right) and the detected circle (right).

The optic disc is the brightest spot in the retinography, and therefore should also be the spot with the highest channel colour contributions. Discarding pixels with low luminosity in the optic disc ROI, and keeping the brighter ones, allows for a quick way to reduce the number of possible locations for the optic disc.

This selection is calculated using only the red and green channels in the ROI. The blue channel is discarded because the optic disc is mostly yellow, but in some images it can present itself as mostly red. Because of this colour fluctuations of the optic disc, both colour channels have their contrasts enhanced. A binarized image is produced by selecting the pixels that have a combined colour contribution (red + green) above 200.

Impact analysis

This approach has less computational load than the original, thus improves the performance of RetScan, but also increases the number of pixels that are analysed by the Hough transform. This generates more detected circles, and in some cases, because of blood vessel interference, a smaller (wrong) circle is selected.

To avoid this problem, a separate circle selection criteria was introduced. This criteria is the number of (white) pixels included in each detected circle. This removes

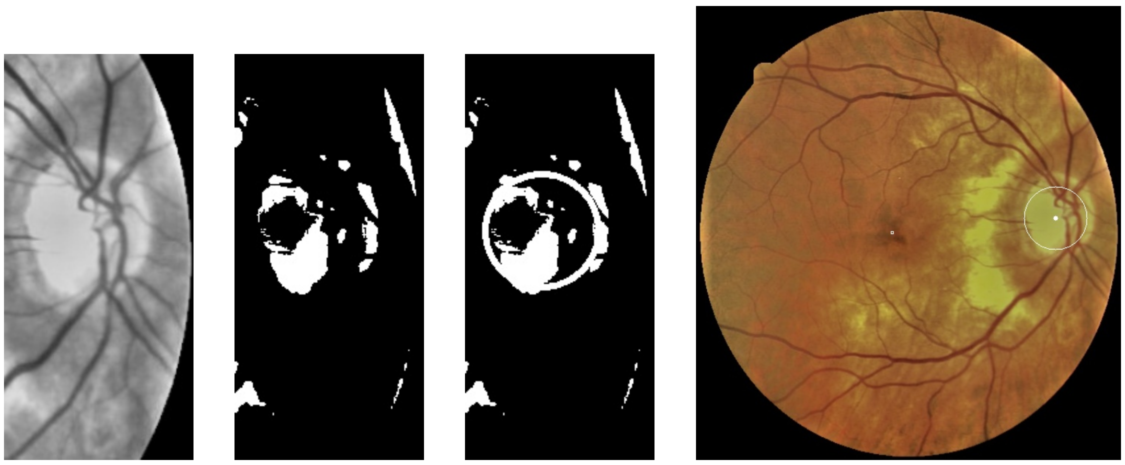


Figure 4.3. An example of an inaccurate detection due to the interference of blood vessels: the fused channel image, the result of thresholding, the detected circle and the detected circle in the eye (from left to right).

the problem of selecting a smaller circle, but introduces other errors, as pixels from outside the optic that were not removed in the first step can form a circle big enough to include all pixels in the image. This problem occurs less frequently, but a different criteria could be more effective.

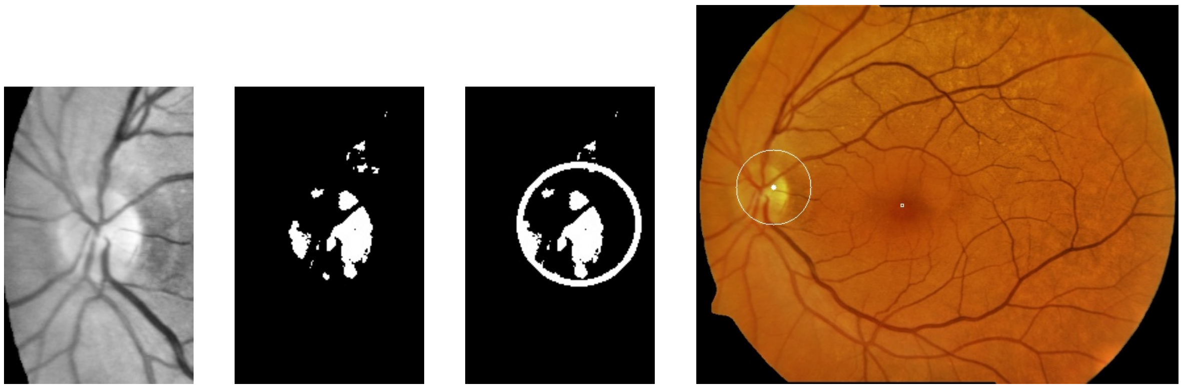


Figure 4.4. An example of the number of pixels in a circle can lead to a erroneous detection: the fused channel image, the result of thresholding, the detected circle and the detected circle in the eye (from left to right).

Despite adding a circle selection criteria after the Hough transform, this methodology still has a better performance than the original one. In fact, this new methodology can process two hundred retinographies in 22.2 seconds, an average of 111 milliseconds per image.

This improved version of RetScan, or RetScan 2.0, also has an improved detection success rate compared to the first implementation. While the first version had a 50.5% success rate, this version has a 71% success rate for the analysed images, which despite being a significant improvement from the first version is still below the MATLAB version.

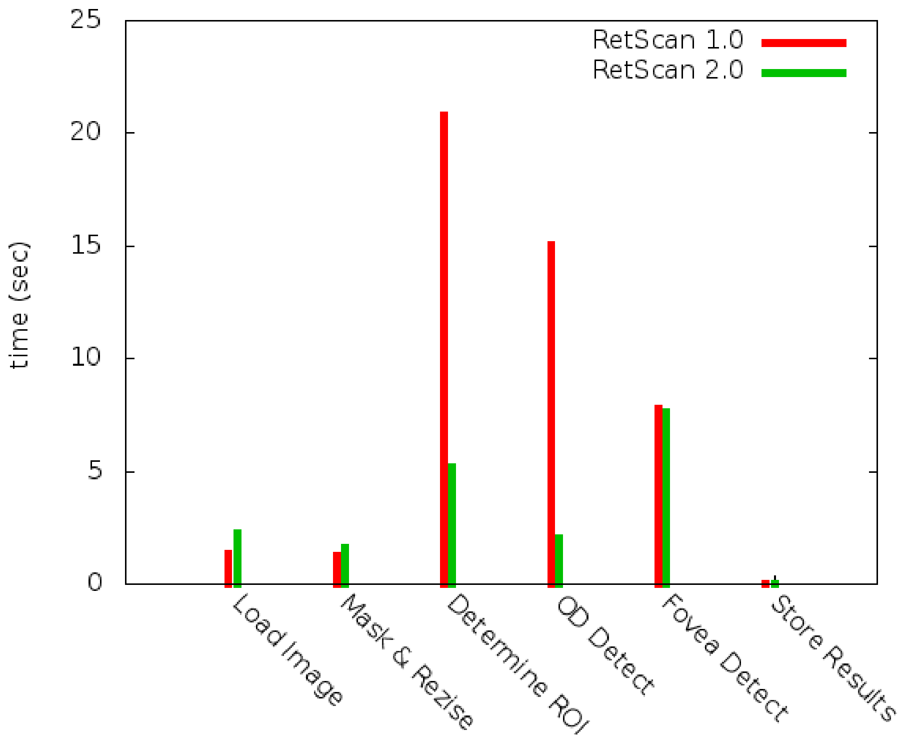


Figure 4.5. A comparison between the average executing times of each main functional block in RetScan v1.0 and 2.0

RetScan 2.0 is much faster than its first version, and fig. 4.5 shows the impact of the changes made to the methodology.

4.3 Parallelizing the code

In recent years, the CPU architecture has evolved from a single processing unit (core), to several independent cores integrated in a single chip. Before the introduction of the multi-core architecture, each new generation of CPU's were faster than previous, but since then speed of the processing units has become stable, the focus has been in increasing the number of these processing units. To take advantage of the current generation of processors, every program must move from the sequential execution model towards the multi-thread environment.

A Graphics Processing Unit (GPU), as the name suggests, is an integrated circuit specialized in rendering graphics. It became popular in the mid-nineties with the appearance of 3D games and most computers currently have a dedicated card. Fuelled by an increasingly relevant video game industry, the GPUs have grown in computational power and currently a high-end graphics card has a peak performance well above of any CPU. Originally, their functionality were mainly texture mapping and rendering polygons, but later became able to do geometric transformations. Programming for GPU's started with the introduction of shaders, which allowed for custom texture, lighting and geometric transformations to be used. Today, GPUs are open to general purpose computing, with programming languages such as CUDA, OpenCL[1].

To efficiently parallelize code, one must consider the two criterion to judge performance: time-to-solution and throughput. Time-to-solution is the traditional form of performance analysis. It is the time it takes the code to produce results of a single input; therefore, the lower the latency the better. Throughput is how many results can be produced within a certain time frame; thus, higher throughput means better performance. RetScan will be deployed in screening centres and in software aimed at supporting ophthalmologists, optometrists and opticians in their analysis of individual patients. In the first case, a large number of retinographies are analysed and classified. In this case, high throughput is paramount, RetScan should be focused in analysing a high number of retinographies and producing a constant flow of results. In the context of analysing the retinographies of a single patient, time-to-solution gains more importance as diagnosis decisions should be fast.

4.3.1 In a multi-core environment

A multi-core environment is a system with several independent processing units (cores) and its memory uses the shared memory model. In this memory model care must be taken regarding cache coherence: if a core updates a value in its cache, the other cores are aware and able to obtain its new value immediately. This model can have a UMA or NUMA memory access. There are several tools available to implement code in a multi-core environment, such as OpenMP [2], TBB [3] and

[1]OpenCL: **Open C**omputing **L**anguage; http://www.khronos.org/opencl/

[2]OpenMP: Open Multiprocessing API http://openmp.org/wp/

[3]TBB:Intel Threading Building Blocks;http://threadingbuildingblocks.org/

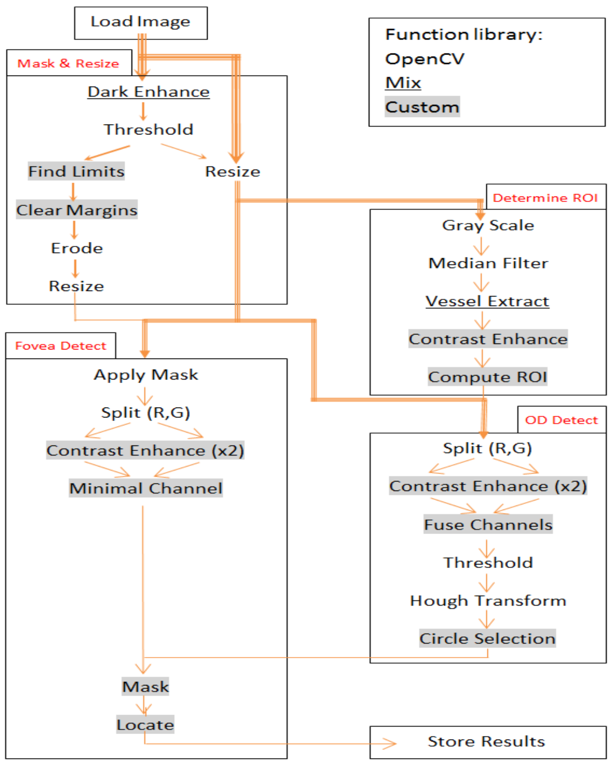


Figure 4.6. A diagram of RetScan 2.0 functionalities. Each arrow represents data dependencies and its width the amount of data required.

Pthreads [4].

OpenMP is an API that offers support for multi-platform shared memory multi-processing programming in C/C++ and Fortran, and it consists in a set of compiler directives, environment variables and a few library routines. OpenMP is easy to use due to its compiler directives, most of the parallelism is implicit and requires little changes to the sequential code to explore shared memory parallelism.

Optimizing throughput

Since the analysis of each image is an isolated event, i.e. the results of analysing one image does not influence the results of another, the analysis of several images is an embarrassingly parallel problem. These problems are the easiest to maximize their throughput, and the best solution is to treat every image as an isolated action

[4]Pthreads: POSIX Threads API; The standard, POSIX.1c, Threads extensions (IEEE Std 1003.1c-1995)

and distribute them between the available processing units. Furthermore, because the process of detecting the optic disc and the fovea has a stable behaviour with all images, and the available processing units are homogeneous, the images can be statically distributed between the cores evenly available without it being detrimental to performance. However, the image distributed is not explicitly defined because the tools used in this work have schedulers that can dynamically adapt to changes in the load balance between cores in an efficient manner.

A different machine was used to run the parellelization tests, this machine has two Intel Xeons E5645 (hexa-core processor with hyper-threading) @ 2.4 GHz with 32Kb of Instruction cache, 32Kb of data caches and 256Kb of L2 cache per core, a shared 12 Mb L3 cache and 12 GB DDR3 RAM. The machine also has a NVidia GTX 580 graphics card. It is running Fedora 15 with 2.6.43 64 bits linux kernel, has GCC 4.6.3 and CUDA toolkit 5.0 beta.

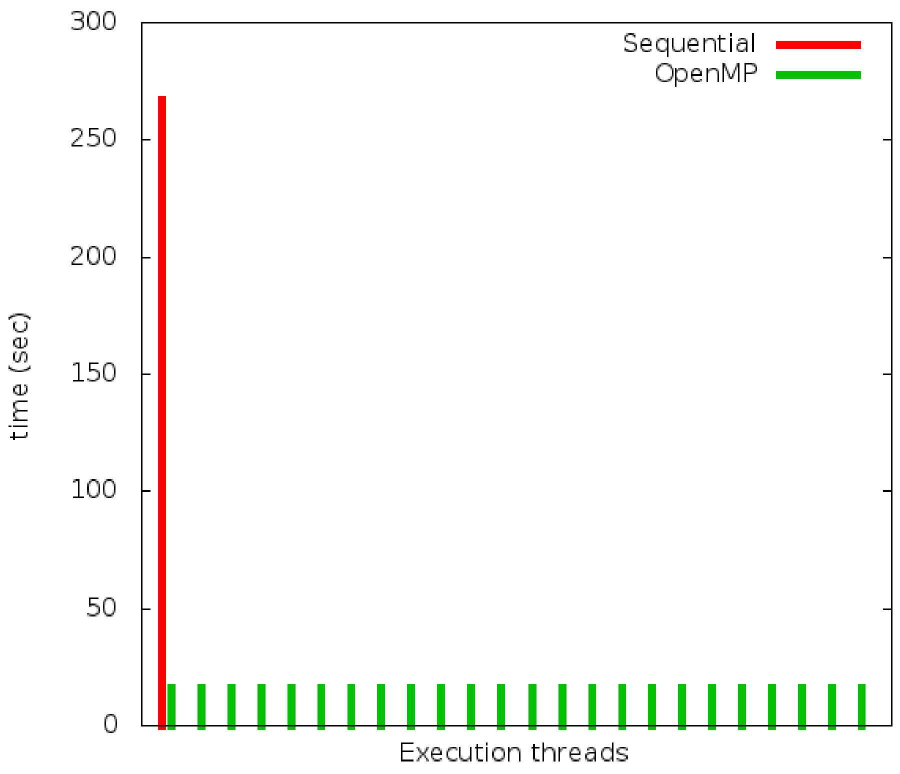


Figure 4.7. A comparison between the sequential version and the OpenMP version optimized for throughput

The OpenMP version completes the process of 2000 images under 17 seconds, which corresponds to a speedup of 16 compared to the sequential version. Despite these good results, the performance is still hindered by the concurrency control of writing the results to a single file.

Reducing the latency

RetScan 2.0 has a methodology with reduced inherent parallelism, at any given point in the execution, no more than 2 functions can be run in parallel due to data dependencies, which limits the approach of reducing latency by distributing the functionalities between the different cores. The other alternative is to parallelize its functionalities.

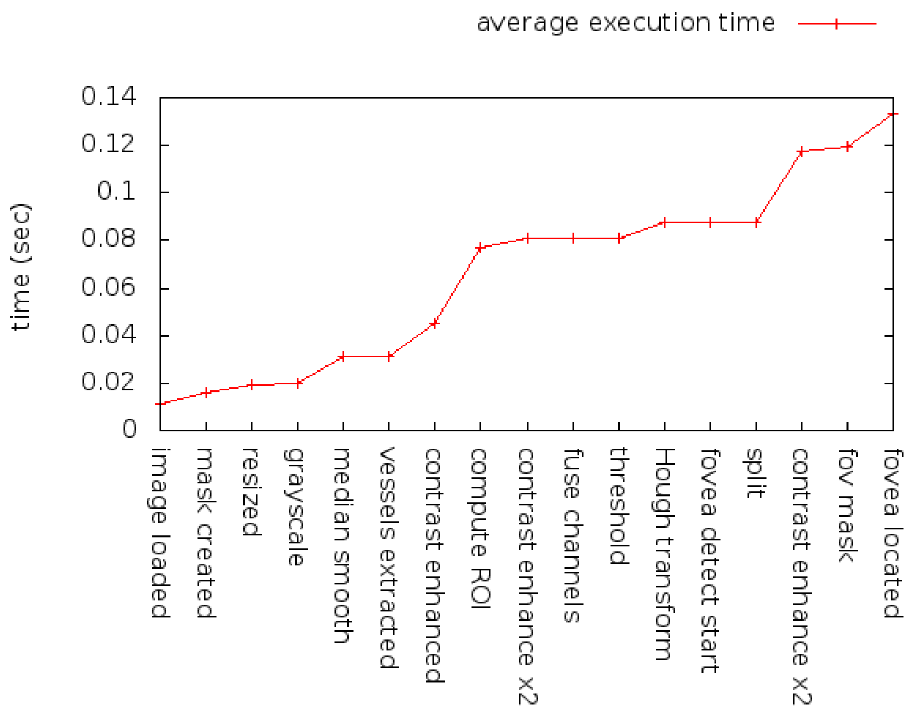


Figure 4.8. The execution time of RetScan 2.0 functionalities

The code profile indicates that the heaviest functionality in RetScan is "Compute ROI", which is responsible for 23% of the execution time (fig. 4.8), should be the first candidate for parallelization. "Compute ROI" is the last step in determining which region of the eye is most likely to contain the optic disc. Its process basically sums the values of pixels within each region and then finds the region with the greatest sum. The parallelization of this function was implemented in two stages, the sums of each region are parallelized as if they are were completely independent, i.e., each sum is considered an embarrassingly parallel operation and distributed by the available threads, afterwards the greatest sum is found using the reduction primitives available by the tools.

#Images	Sequential	OpenMP
1	0.141	0,107
200	26,506	20,674
500	66,504	51,611
1000	132,785	103,116
1500	199,474	154,713
2000	265,773	206,296

Table 4.1. A performance comparison between the sequential version and the OpenMP version. Time is measured in seconds

In table 4.1 are the results of processing 1, 200, 500, 1000, 1500 and 2000 images using the sequential and the OpenMP version of the “Compute ROI” function. The theoretical maximum speedup obtained by parallelizing the “Compute ROI” function, which is responsible for 23% of the execution time, assuming a perfect parallelization (without communication costs and reducing the funstions running time to 0), is 1.3, as Amdahl’s Law states. The OpenMP version has a speedup of 1.28, which is a small improvement in latency, but near the theoretical maximum speedup. Despite achieving a good speedup, this version has a low efficiency because the computational resources increased 24 times but the speedup is only of 1.28.

4.3.2 With a CUDA-enabled accelerator

Compute Unified Device Architecture or CUDA is a parallel computing architecture model developed by Nvidia. This new model has its own programming language called C for CUDA, which is a version of the C language with some extensions and restrictions. C for CUDA is a language developed to facilitate general purpose computing in GPUs (GPGPU). These devices are more powerful than CPU’s, and excel when executing large, regular and embarrassingly parallel problems.

The GPUs are connected to the motherboard through PCIe[5] slots, and as a consequence, the memory transfers between CPU and GPU usually having a big latency but offering a good data throughput. This limitation should be considered when programming for these machines, by avoiding small data transfers that suffer from the high latency but do not take advance of the high throughput connection.

[5]PCI Express: Peripheral Component Interconnect Express

The OpenCV library has a small port to CUDA-enabled devices, which does not include all the functions required to port the RetScan methodology to these accelerators, such as GPUs. Due to time restrictions only the “Compute ROI” function was implemented in this environment. This limits the parallelization effort to a latency reduction approach, and therefore the function calls to the GPU between different images are not streamed. “Compute ROI”, uses a single channel image of the eye, which is roughly 300Kb. This is a small data transfer and the high latency penalty should be a major factor in the performance.

#Images	Sequential	OpenMP	Cuda
1	0.141	0,107	4,671
200	26,506	20,674	25,124
500	66,504	51,611	55,824
1000	132,785	103,116	107,274
1500	199,474	154,713	158,407
2000	265,773	206,296	210,793

Table 4.2. A performance comparison between the sequential version, the OpenMP and the Cuda version. Time is measured in seconds

The OpenCV GPU port has great initialization cost, and the first function call of each run takes approximately 4.5 seconds (Table 4.2). Excluding this initialization cost and the first image, this version also achieves a speedup of 1.28 compared to the sequential version. These results are surprising, as the data transfers were expected to severely hinder the performance of this version.

When the images are distributed through the cores using the GPU version of this function the results are similar, from which we can conclude that the GPU is still not operating at the maximum of its capacities even when 24 threads are using it.

Chapter 5

Conclusion

5.1 Critical Overview

The lack of information about the changes made to the original code after the documented work of Pinão, was the first obstacle to overcome. This lack of information forced a complete code check of the supplied code to verify what was, and was not changed. The slight differences in implementation of image processing techniques between MATLAB and OpenCV also forced some extensive debugging. This methodology applies several image filter in succession and an image could behave very differently in the pixel selection steps calibrated with heuristic parameters. RetScan does offer the functionalities the original code provided, with a greatly increased performance, and is in agreement with the established goals. The RetScan (1.0 and 2.0 version) has a low detection success rate and is still not ready for deployment; also the heuristic parameters need to be revised and tweaked, and a different circle selection criteria is needed.

The parallelization effort in a multi-core environment has returned good results, the throughput oriented version has a good speedup, and the results of parallelizing the "Compute ROI" is close to the theoretical maximum speedup. Despite the good parallelization of "Compute ROI" the overall performance of only parallelizing that function is hindered by the fact that the function has a a small weight in the execution time. The GPU version of the the "Compute ROI" also achieved a good speed up, which was an unexpected result due to the relative small size of the images and the cost of memory transfers to the GPU.

5.2 Future Work

This dissertation completed all the goals it aimed, but there are still areas where RetScan can, and need, to improve.

The detection success rate still requires work before it is ready for real world cases. Fine tuning the heuristic parameters used and a better circle selection criteria can be applied to improve the results in both version of the methodology. A circle selection criteria that besides counting the number of pixels also looks at the number pixels that form the circle boundary might be a good improvement. The parallelization effort focused on optimizing the throughput can benefit from a reducing the number of synchronization points. In its current version every thread writes the results in the same file, having separate files for each thread and only fusing them after all images have been processed reduces the number of times when concurrency control is required. In the effort to reduce latency, extending the number of parallelized functions, can improve its performance even further. The OpenCV GPU is a library with high initialization costs, this penalty can be minimized if the GPU computational load increases by coding more functions that use the GPU.

Bibliography

[1] Centers for Disease Control and Prevention. *Fundus Photography for Health Technicians Manual*, 1989. http://www.cdc.gov/nchs/data/nhanes/nhanes3/cdrom/nchs/manuals/fundus.pdf.

[2] Helen L Cook Chanjira Sinthanayothin, James F Boyce and Thomas H Williamson. Automated localisation of the optic disc, fovea, and retinal blood vessels from digital colour fundus images. *Br J Ophthalmol*, 83(8):902–910, 1999.

[3] Daniel D. Chiras. *Human Biology*. Jones & Bartlett Learning, 7th edition, 2011.

[4] M. Foracchia, E. Grisan, and A. Ruggeri. Detection of optic disc in retinal images by means of a geometrical model of vessel structure. *Medical Imaging, IEEE Transactions on*, 23(10):1189 –1195, 2004.

[5] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2001.

[6] José Pinão and Carlos Manta Oliveira. Fovea and optic disc detection in retinal images with visible lesions. In LuisM. Camarinha-Matos, Ehsan Shahamatnia, and Gonçalo Nunes, editors, *Technological Innovation for Value Creation*, volume 372 of *IFIP Advances in Information and Communication Technology*, pages 543–552. Springer Berlin Heidelberg, 2012.

[7] José Manuel Neves Pinão. Fovea and optic disk detection and key performance indicators process automation. Master's thesis, University of Coimbra, Palácio dos Grilos, Rua da Ilha, 3000-214 Coimbra, Portugal, 2011.

[8] Kenneth W. Tobin. Detection of anatomic structures in human retinal imagery. *Medical Imaging, IEEE Transactions on*, 26(12), 2007.

[9] T. D. Williams and J.M. Wilkinson. Position of the fovea centralis with respect to the optic nerve head. *Optometry and vision science : official publication of the American Academy of Optometry*, 69(5):369–377, 1992.

[10] Huajun Ying and Jyh-Charn Liu. Automated localization of macula-fovea area on retina images using blood vessel network topology. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 650 –653, march 2010.