UNIVERSITÀ DEGLI STUDI DI PADOVA
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione

TESI DI LAUREA MAGISTRALE IN BIOINGEGNERIA

DEVELOPMENT OF A SOFTWARE TOOL FOR ANNOTATING VASCULAR FEATURES IN IMAGES OF THE RETINAL FUNDUS

Relatore: Prof. ALFREDO RUGGERI
Correlatore: Prof. EMANUELE TRUCCO

Laureando: ILARIA PIERETTI

ANNO ACCADEMICO 2012-2013

Index

Abstract

Nowadays, the analysis of the images of the retinal fundus palys a key role in the prevention and the diagnosis of diseases of different kind, such as diabete and cerebrovascular diseases. The purpose of this thesis is to present the development and the use of a software for the manual annotation of retinal features (optic disc, fovea, junctions and widths of the blood vessels). The VAMPIRE-Annotation Tool, developed with the Matlab GUI, presents a user-friendly graphical interface, allows the user to upload previously recorded data and also saves, for each measure taken, additional information that need to be able to better characterize it. This tool is essential for the creation of ground truth for the validation of automatic algorithms. A study that aims to identify possible retinal biomarkers for Sarcopenia is reported. During this study, some annotations are performed with the VAMPIRE-Annotation Tool, thanks to which the significance of the additional information recorded for each measure, with respect to the measure itself, is analyzed.

Acknowledgements

I wish to express my gratitude to all the members of the VAMPIRE team, Manuel, Lucia, Enrico, Kris, Tom, Devanjali and Gavin, without whom this thesis would not have been possible. I want to especially thank my supervisor Prof. Manuel Trucco for having welcomed me in the team and in the CVIP group of the University of Dundee with great hospitality, and for allowing me to participate in both prestigious academic meetings and lively intercultural dinners. Deep gratitude is also due to Lucia Ballerini, my guardian angel during my six months in Scotland, whose knowledge, assistance and patience saved me in times of distress during this study. Special thanks also to Doctor Peter Wilson who has enlightened us with his knowledge, and for helping us with great patience, professionalism and enormous kindness.

Chapter 1

Introduction

The retina is the innermost membrane of the eye and it is a vascularised tissue. Its vascular network consists of arterial and venous ramifications that originate from the optic disc and progressively bifurcate into smaller branches that spreadout across the retina. This is the only place in the whole body where blood vessels are clearly visible on the surface, and therefore their visualization can be done non-invasively and in vivo [1]. This is extremely important because the vascular network of the retina is believed to be governed by physiological principles that optimize its efficiency [2] and so the monitoring of its geometry plays a key role in the diagnostics and the prevention of many diseases, not only eyepieces, but also systemic [1] and in the brain [3]. For this reason, over the past years it has been heavily invested in digital imaging systems, which are in constant and rapid evolution [1].

In order to efficiently identify in a large set of images the areas related to pathologies, an automatic system must first be able to identify the landmarks on the retinal surface:

- the optic disc, a small blind spot on the surface where the fibers of the retina leave the eye and become part of the optic nerve
- the fovea, region of the retina with maximum density of photoreceptors
- the entire vascular system, where you must be able to measure the widths and the bifurcation angles of the vessels in each point.

Once these reference points are identified you can get all those values and indices that act as biomarkers for the various diseases. An important example of biomarker is the arteriolar-to-venular diameter ratio (AVR), which is the ratio between the widths of arterioles and venules, whose variations are associated with stroke, cerebral atrophy, cognitive decline, and myocardial infarct [4]. Other examples of biomarkers are the bifurcation angles and the tortuosity of the vessels.

It is precisely in this context of identification of landmarks and computation of biomarkers that the semi-automatic software VAMPIRE was born in 2011 and it is still in continuous development. VAMPIRE offers a public and user-frendly platform, with which users can quantify optic disc, fovea, widths of the vessels and bifurcations angles of junctions in a large set of fundus camera images, in order to produce data about tortuosity, bifurcation coefficients and fractal analysis [5]. Regarding the detection of the optic disc and fovea two algorithms have recently been developed [6], based on two very simple concepts: the OD is the brighter area of the retina and with the higher concentration of blood vessels, the macula instead is the darkest area and it is located in an avascular zone. The algorithm for the quantification of the optic disc, developed by Giacchetti et al. [7], converts the image to grayscale, from which the vascular network is then identified using standard techniques of segmentation. The vascular map is then removed from the picture, on which then inpainting algorithms are applied. The Fast Radial Symmetry transform is applied to the inpainted image, generating a map of bright symmetries on which, by exploiting the gradient of brightness on four different scales, the contour of the optic disc, assumed elliptical, is found [7]. The results obtained with this algorithm are very good (although in some cases there are still errors, even in good quality images as showed in Figure 1.1) and so it was officially included into the VAMPIRE software. Regarding the fovea, an algorithm very similar has been implemented, with focus on the symmetry of dark structures instead of those bright [6]. The results in this case are not great, and then, for now, the VAMPIRE software has not yet incorporated this technique, and therefore remaining without the possibility of fovea's automatic annotation. As regards the identification of the retinal vascular network, VAMPIRE uses segmentation techniques with Soares' method [8], to which a procedure of refinement of vessels' contours obtained from vascular binary map was then added [9]. The Soares' method with this improvement produces excellent results with small images (800x600 pixels) because the training was executed precisely on this kind of databases (ie REVIEW, STARE and DRIVE), but when applied on larger images (ie 3500x2300 pixels) the technique fails in many areas as can be seen in Figure 1.2. For now, there are no better alternative techniques for mapping vessels, then this is what is currently used in VAMPIRE, and in these binary maps is where junctions are selected. Regarding widths instead, an alternative technique has been included in VAMPIRE, just as default. This method, developed by Lupascu et al. [10], is based on the construction of a parametric surface model of local cross-sectional intensities, and then using decision trees for regression to estimate width from the parameters of the best-fit surface. This

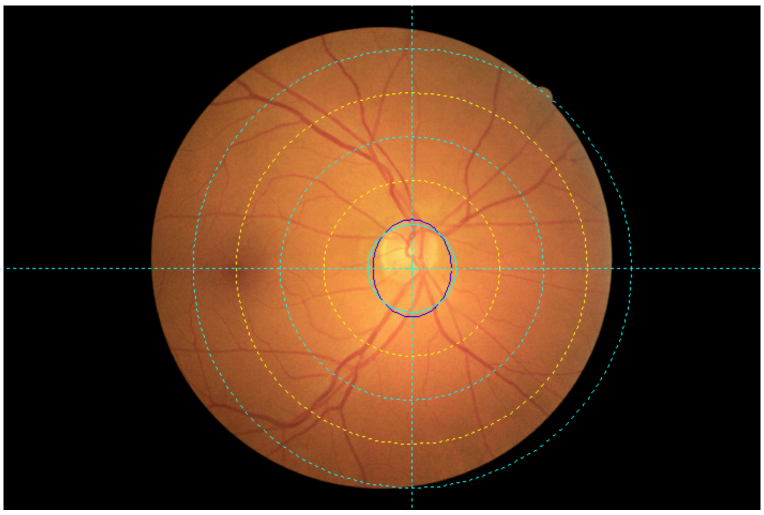


Figure 1.1: Automatic wrong detection of the optic disc

method, as it is based on techniques that do not involve training, works well for images of any size. In the VAMPIRE software the user clicks on a point inside the vascular network and on that point the width of the vessel is estimated using the method explained above. As regards the A/V classification, despite the development of new algorithms is in progress, VAMPIRE is not yet able to automatically perform this classification.

The VAMPIRE software therefore needed a manual annotation tool included in the package for the validation of these new algorithms [6] but also to take the place of the automatic software, if it does not meet cernatin needs. There were already some other manual tools developed by the team, but each of them only allowed the annotation of one or two features, the data storage was completely different between the different tools, and they were not very user-friendly and therefore used only by the members of the team or by clinicians very closed to it.

In light of this, the VAMPIRE-Annotation Tool, in addition to satisfying requirements that instead the old ones did not, it should also be supportive of VAMPIRE in what it is still incomplete (i.e. fovea, A/V classification) or when it produces errors (i.e. OD mistakenly identified, wrong segmentation). In parallel to all this, ophthalmologists of the Ninewells Hospital of Dundee have exposed to the VAMPIRE team a new vision for the treatment of data in the statistical context, to which underlies the need to attach to the data some additional information. This additional information is seen as an “adjustment” that has to be applied to the raw

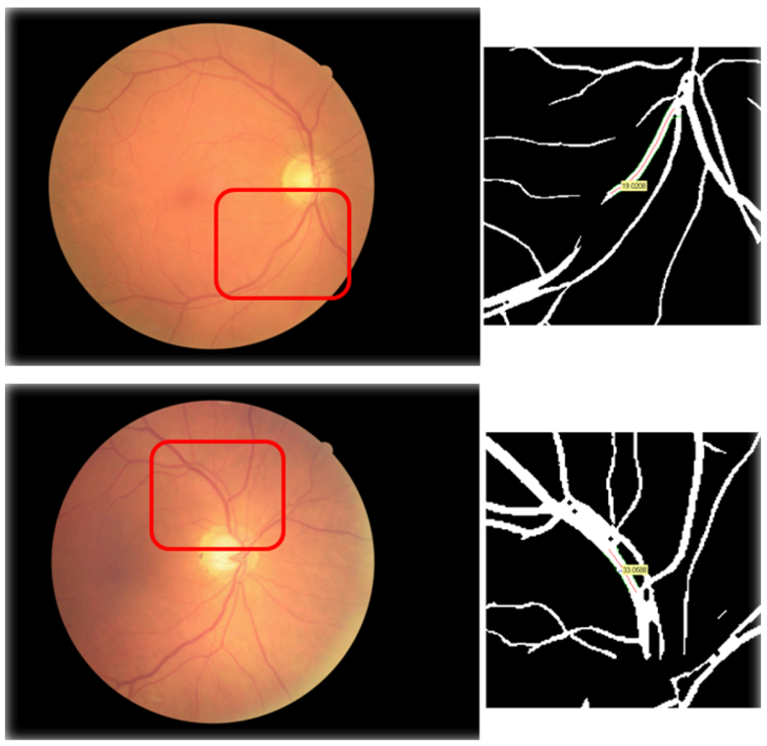


Figure 1.2: Failure of segmentation technique in some areas

measure. For this reason the tool has been designed in such a way that it is not limited only to the simply quantification of the features (location and size of OD and fovea, widths values in a point and Cartesian coordinates of that point, the values of bifurcation angles for a junction and Cartesian coordinates of the junction), but that it adds even more additional information (which are going to be exposed in Chapter 3) to characterize them in a really complete way.

In this thesis, after a brief overview of other manual tools for the analysis of images of the retina developed by other teams (Chapter 2), all the steps taken for the development of VAMPIRE-Annotation Tool are shown in Chapter 3, describing in detail all the various components with the aid of code fragments and images. Afterwards, in Chapter 4, the use of the tool in the context of the clinical study "Does the European Working Group on Sarcopenia in Older People algorithm detect all those vulnerable?" is described, ending then, in the last Chapter, with some conclusions and discussions about the work done and the future one.

Chapter 2

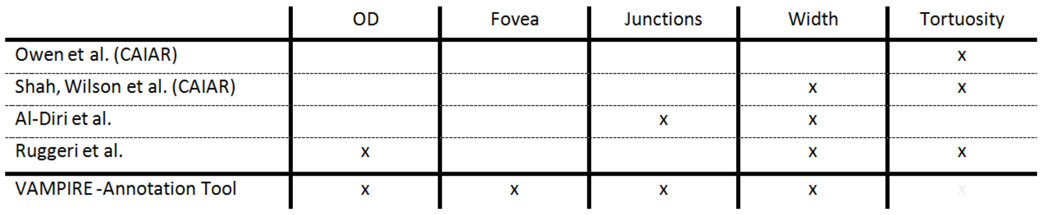
Related works

In view of the automated software, many teams make use of manual tools to validate the algorithms and to address some possible gaps. However, there are few specific and detailed publications about the manual tools used, which are only indirectly mentioned in some papers regarding validation. In order to have a reliable validation of automatic algorithms is necessary to have appropriate instruments for manual annotation [19], which can be both very complex, but in some cases also extremely simple and specific. An example of the latter case is the tool used in the study [18] for the validation of the Computer Assisted Image Analysis of the Retina (CAIAR) program, in which two observers were asked to assign a value of tortuosity from 0 to 5 to blood vessels of the 14 subjects involved in the study. Also in other studies involving CAIAR hand instruments were used: for example, in a study about the retinopathy of prematurity (ROP) performed by Shah, Wilson et al. [17] at the Division of Epidemiology and Genetics of the Institute of Ophthalmology in London, they have used the annotations of four ROP experts who used a tool for the manual measure of width and tortuosity of the vessels. Thanks to this ground truth, perform statistical comparative analysis both with CAIAR and also between the four subjects involved has been possible for the authors.

A specific tool for the geometry of the bifurcation has been implemented by Al-Diri et al. [20]. Regarding the bifurcation angles, these are obtained by first selecting the central point of the junction and then clicking at the end of the three segments of the vessels involved. Immediately after, the three segments that follow the central axis of the vessel are shown on the screen (similar to what happens in VAMPIRE-Annotation Tool). For the measure of the widths, small rectangles aligned along the vessel are used: to place a rectangle is requested to the user to select two points on the central axis of the vessel and then a point on one of the two edges of the vessel, near the central points. The rectangle is positioned according to the location of these three points. For both the junctions and the widths, the first selections can be changed and adapted to the structure of the vessel until they are believed to be correct.

The Laboratory of Biomedical Imaging of the Department of Information Engineering of the University of Padua has developed a tool for manual tracking of the retinal vessels [16]. The instrument ROPnet is a web-based tool that allows the quantification of width and tortuosity of vessels, thanks to the manual tracking of the vascular axis. The tool allows the upload of a single image at a time, on which the entire procedure is completed before going to the next. ROPnet allows the user to choose whether to display the image in the RGB or in the green channel format and to zoom in on it. During the use of the tool a online guide is always easily accessible. First, the annotation of the optic disc is required and starts clicking at the center of the OD. From here a circle with fixed center and moving boundary is formed and has to be adapted to the contours of the real OD. Later, the tool prompts you to select on the image the two extreme points of the vessel segment that has to be analyzed. A straight line passing through the two points appears. The line can be changed (keeping the ends fixed) by clicking on other points inside the vessel to make sure that it fits its particular shape. On this curve, automatic algorithms for the calculation of width and tortuosity are then applied.

As previously stated, the amount of manual tools for the annotation of retinal fundus images that are documented in the literature is poor compared to what really exists. Taking into account this fact, we try to do a little summary about the completeness of these tools with respect to the number of features that can be annotated. As you can see from the table in Figure 2.1 the general tendency is to create specific tools according to the request of a certain study. The VAMPIRE-Annotation Tool instead is designed in a different perspective, with the aim to be multifunctional and suitable for every type of study, in order to be able, independently of the context, to produce data congruent and connected also between the different features.



	OD	Fovea	Junctions	Width	Tortuosity
Owen et al. (CAIAR)					x
Shah, Wilson et al. (CAIAR)				x	x
Al-Diri et al.			x	x	
Ruggeri et al.	x			x	x
VAMPIRE -Annotation Tool	x	x	x	x	x

Figure 2.1: Features analyzed with the manual tools reported

Chapter 3

Annotation tool development

For the development of the VAMPIRE-Annotation Tool, it was decided to partially follow the "Waterfall model", convenient for small and relatively easy software projects. The "flow" of the development stages (Requirements Analysis, Software architecture and design, Implementation, Documentation) is described in this chapter. As regards the part of Testing, this has been developed thanks to the use of the tool during the study described in Chapter 4.

3.1 Requirements Analysis

Compared to the older manual tools mentioned in Chapter 2, the VAMPIRE-Annotation Tool has to allow a complete annotation of all the features (optic disc, fovea, vessels' width and bifurcation angles of the junctions), saving the related data in a consistent way between them. Being the production of ground truth one of the two main purposes of this tool, and being the ground truth the more reliable the more detected by a specialist (then a doctor), it must lodge a simple and immediate interface for the user. One of the most innovative characteristics the tool has to have is saving all the useful information for the application of what in Chapter 1 we called "adjustments" to the data, not saving only the Cartesian coordinates and the final measure as the old ones. As mentioned in the introduction, these adjustments are additional information that must be saved during the annotation. The basic information until now considered essential was:

- for OD and fovea: the size and the position expressed in Cartesian coordinates
- for widths and junctions: the Cartesian coordinates of the measuring point, measure and type of vessel (artery or vein).

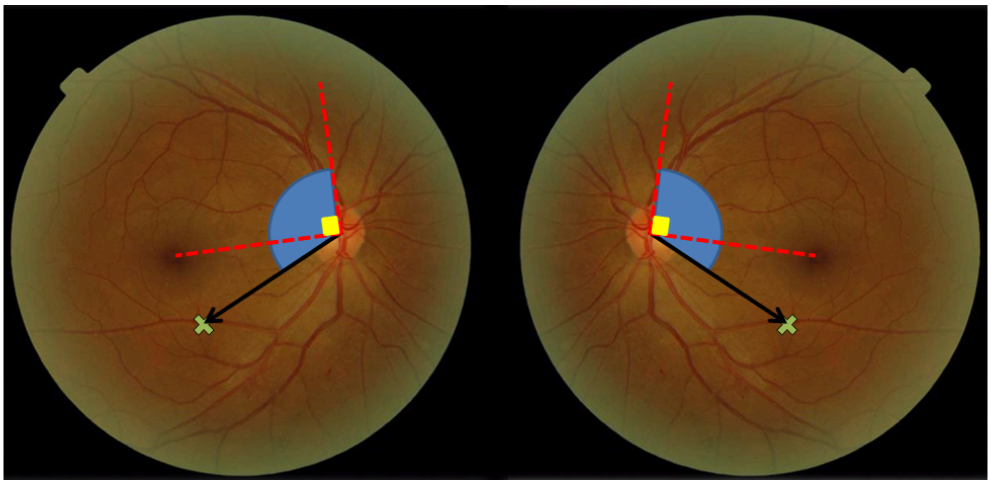


Figure 3.1: Representation of the polar coordinates of a point of interest (green cross): the black arrow represents the linear coordinate, in blue the angular coordinate with respect to the reference axis (in red)

The most important adjustment are definitely the polar coordinates of a point on the retinal surface. In fact, while those Cartesian give a pure information of position within the image, polar ones situate the point in the context of the retinal surface, and especially in relation to the features of reference. As can be seen from the Figure 3.1, the polar coordinates that are saved are special and have a pole in the optic disc. The radial distance from the OD is “classical”, while the angular one has particular characteristics: first of all it has opposite direction for the right (counterclockwise) and left eye (clockwise); this is because in nature eyes are symmetrical with respect to the central part of the face, and thanks to this choice of angular coordinate, nasal structures and temporal structures of the two eyes have congruent angles. Another important fact is that the angle is not calculated from an axis parallel to image’s contours, but from the one perpendicular to the one that connects fovea and optic disc. The polar coordinates so are not related to the image (such as Cartesian), but to the physiology of the retina of that particular patient.

Knowing that the fovea in many images is difficult to accurately locate (as in Figure 3.2, and being the fovea, as explained above, an important point of reference, save information about its visibility becomes vital to understand the quality of the annotation. If the fovea is not clearly visible, its annotation will be very rough, and therefore the polar coordinate of that point can be quite incorrect, with obvious disastrous consequences in the subsequent data analysis. During the annotation of the widths, the user is asked to specify the generation of the segment of the vessel in which the measure is performed. This is an important adjustment to the measure of width, which may strongly depend on the number of times the vassel bifurcates downstream.

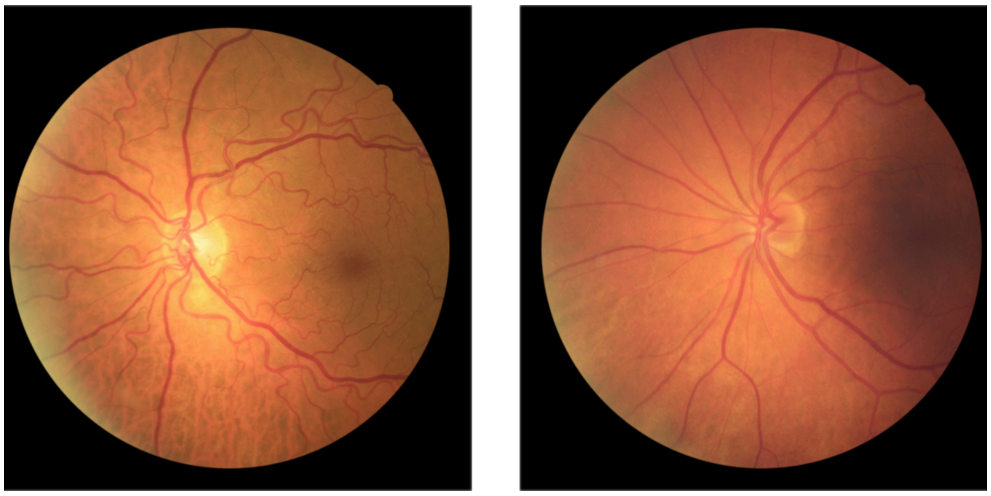


Figure 3.2: On the left side a fovea well visible; on the right side an image with the fovea clearly individuable

In addition to the functions of a manual tool, the VAMPIRE-Annotation Tool has to remain an auxiliar package for the VAMPIRE software, and thus the production of interchangeable data and comparable with automatic ones is required. In order to meet this need, the points at which the measures have to be taken must be identified thanks to coordinates referred to the same origin (for the Cartesian the upper left corner of the image, for the polar ones the optic disc) and that is why we chose to enter the automatic annotation of the OD found with the same algorithm of VAMPIRE [7] (described in Chapter 1). As an adjustment to the annotation of the optic disc, saving the annotation type (automatic or manual) it was considered util to be able, at a later time, to evaluate the quality of the annotation made. In a broad view of “data library”, we also felt the need of a tool that not only produces data from scratch, but that could also interact with data from previous annotations. So the tool has to allow the loading of old data and allow different uses that will be explained in the following paragraphs.

3.2 Software architecture and design

The tool has been designed and developed with the Graphical User Interface (GUI) of Matlab, using its basic elements such as buttons, panels, etc. [14]. The annotation mode of features is intrinsically different for each of them, so the design of different environments of interaction with the user was necessary. The basic structure of the tool has to consist of five different environments (and thus five different screens): one at the beginning where to insert user identification and planning of the work, and then other four, one for each feature. These five environments are represented by the five columns of the scheme of Figure 3.3, which shows and describes the structure of the tool.

As explained in the previous paragraph, the tool has to meet some requirements, and to ensure this happens, a precise structure of the tool itself is needed. To make the job of the user less burdensome as possible, the tool should provide an initial planning of the annotation, excluding any image in the set that could not be annotated, and immediately choosing the type of annotation. In the first part then the possibility of having an overview of all the images has to be present, so the user can already exclude those damaged or otherwise not suitable for the annotation; in this way they should not then appear again and unnecessarily every time during the annotation of the features (thus wasting time for the user). In addition, selecting the type of path for the annotation is necessary to do immediately:

- if the need is for a complete annotation, then the tool has to give only the possibility to upload the images, and after that it has to require the annotations of optic disc and fovea, and to leave optional the ones of junctions and widths.

- if the need is to continue a previous annotation using optic disc and fovea already annotated, then the tool would automatically allow not only the loading of images, but also of the old data contained in the text files, and then it has automatically to prohibit the annotations of OD and fovea, and to leave optional the ones of junctions and widths. If the purpose was to take measures in points other than those of the previous annotation, then the tool has simply to plot on screen the points of the old annotation, allowing a complete annotation of junctions and/or widths; but if the purpose is to measure angles and/or widths in the same points of the previous annotation, then the tool would automatically load the "old" points and would allow the user the only measure without adding additional points.

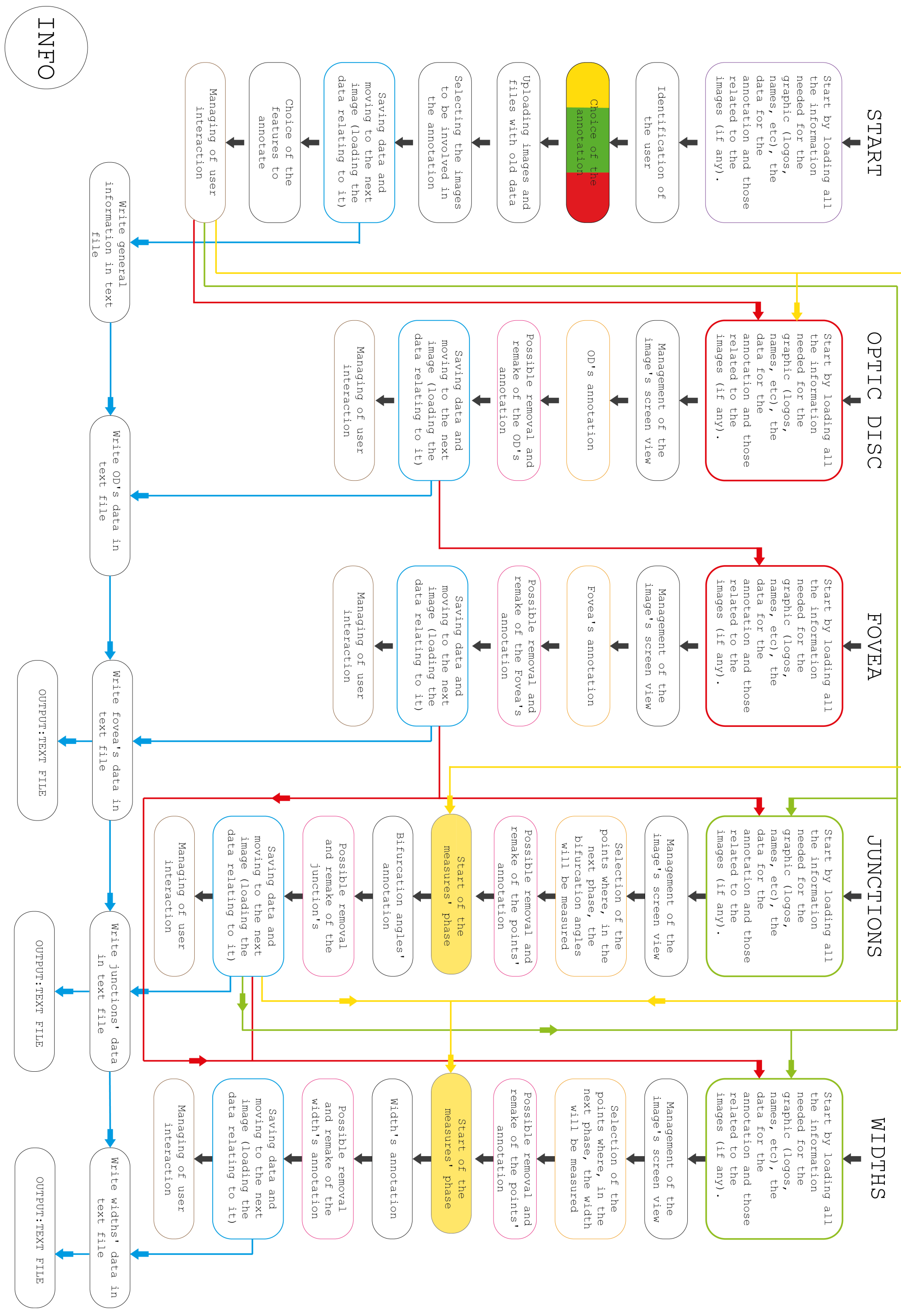

START
Start by loading all the information needed for the graphic (logos, names, etc), the data for the annotation and those related to the images (if any).
Identification of the user
Choice of the annotation
Uploading images and files with old data
Selecting the images to be involved in the annotation
Saving data and moving to the next image (loading the data relating to it)
Choice of the features to annotate
Managing of user interaction
INFO
Write general information in text file
OPTIC DISC
Start by loading all the information needed for the graphic (logos, names, etc), the data for the annotation and those related to the images (if any).
Management of the image's screen view
OD's annotation
Possible removal and remake of the OD's annotation
Saving data and moving to the next image (loading the data relating to it)
Managing of user interaction
Write OD's data in text file
FOVEA
Start by loading all the information needed for the graphic (logos, names, etc), the data for the annotation and those related to the images (if any).
Management of the image's screen view
Fovea's annotation
Possible removal and remake of the Fovea's annotation
Saving data and moving to the next image (loading the data relating to it)
Managing of user interaction
Write fovea's data in text file
OUTPUT:TEXT FILE
JUNCTIONS
Start by loading all the information needed for the graphic (logos, names, etc), the data for the annotation and those related to the images (if any).
Management of the image's screen view
Selection of the points where, in the next phase, the bifurcation angles will be measured
Possible removal and remake of the points' annotation
Start of the measures' phase
Bifurcation angles' annotation
Possible removal and remake of the junction's
Saving data and moving to the next image (loading the data relating to it)
Managing of user interaction
Write junctions' data in text file
OUTPUT:TEXT FILE
WIDTHS
Start by loading all the information needed for the graphic (logos, names, etc), the data for the annotation and those related to the images (if any).
Management of the image's screen view
Selection of the points where, in the width next phase, the width will be measured
Possible removal and remake of the points' annotation
Start of the measures' phase
Width's annotation
Possible removal and remake of the width's annotation
Saving data and moving to the next image (loading the data relating to it)
Managing of user interaction
Write widths' data in text file
OUTPUT:TEXT FILE

Figure 3.3: Scheme of the architecture of the VAMPIRE-Annotation Tool

Another property the program has to possess in order to facilitate the use, is to make permanent the order of annotation of the various features. In fact, in order to measure junctions and widths, OD and fovea has to be already annotated (by a previous user or the current one), then the tool has to create a sort of forced path the user has to follow and could not do in a different way (otherwise errors would be generated). These paths are reported in the diagram of Figure 3.3 with red, green and yellow arrows; specifically the red arrows indicate the path that the tool get the user to follow for a new annotation, the green arrows indicate the path for an annotation with OD and fovea loaded by a previous one and junctions and widths measured in new points, and finally the yellow arrows indicate the path for an annotation with OD, fovea and measuring points loaded by a previous one.

The tool has also to save the information useful for the application of the adjustments (explained in Chapter 1). In the following paragraphs the various functions used for processing this information (i.e polar coordinates, visibility of the fovea, generation of a vessel, etc.) will be described.

As regards the structure of the tool specifically, the understanding can be facilitated looking carefully the scheme of Figure 3.3, knowing that each block represents a portion of code (consisting of one or more functions) that aims to produce the same result. As one can easily see, the environment START has its own particular structure, and instead OD and fovea between them and junctions and widths between them have very similar structures, with the presence of the same blocks in the same order. Indeed, even though at first glance the OD-fovea structure and the junctions-widths one may seem totally different, as we will see later, actually this is not entirely true. The environments of OD and fovea show an initial part that loads all the necessary information for the graphics (logos, names, etc), the data for the annotation and those related to the images. In both the environments, these pieces of code are very similar, with few parts slightly different because the graphical interface and some required data are different, but essentially they are the same in substance. The second block of each environment instead represents that part of the code that allows the actual annotation of OD and fovea, and therefore is inherently different for the two features. The last four blocks represent those parts of the code related to the management of the screen display containing the image (block three), to the elimination of the annotation just performed to be able to redo it (block four), saving the data through the functions explained in the paragraph 3.3.7 and loading of the next picture (block five), and finally to the management of the activation of buttons and panels (block six). They are very similar in substance even if different in some parts, just like blocks one. Regarding the environments of junctions and widths, the first blocks are structured like those of OD and fovea, while the second ones consist in selecting of the points where you want to measure angles and widths (pay attention, the actual measure is not performed here, but in block six). The third and the fourth blocks respectively represent the code related to the menagement of the screen and to the elimination and remaking of the points just annotated and have the same structure of block four in OD and fovea's environments. The fifth blocks are special because they contain a piece of code that allows you to go to the actual measure of angles and widths, saving the selected points and moving the annotation on a small screen that is automatically zoomed in on the points. The blocks six contain the code that allows the measurement of angles and widths on the points previously annotated and are intrinsically different for the two environments. The last three blocks have the same functions as the last three of OD and fovea and are structurally similar.

3.3 Implementation of the tool

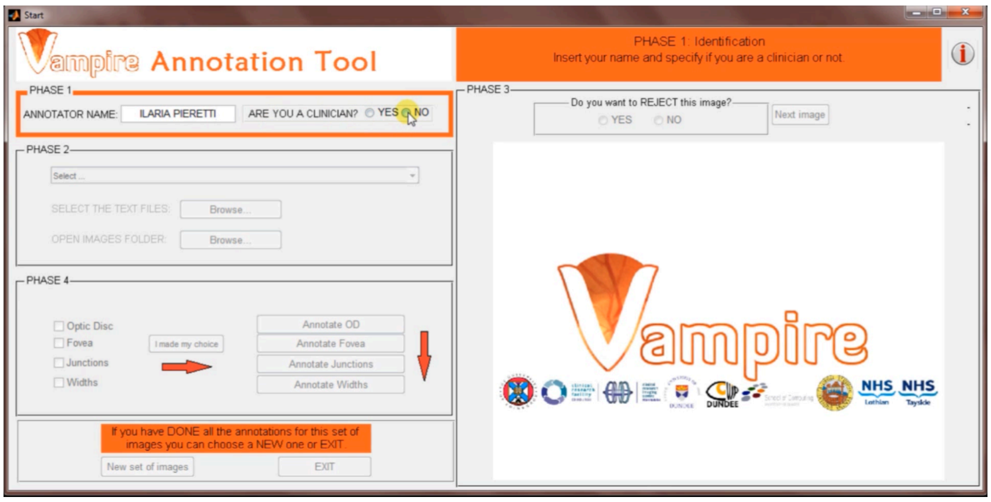
In this section the various environments of the tool will be described in detail. The related code is reported in Appendix B.

3.3.1 Start menu

The first environment of the tool is made up of four sections. In the first section the user is asked to enter his identifying information (Figure 3.4). The name has to be written inside the edit-text box [14] controlled by the functions `Name_CreateFcn()` and `Name_Callback()`, which respectively manage the appearance of the box and save the text entered by the user.

Than the user is asked to specify if he belongs or not to the category of clinicians. It is considered important to save this information as an attached of the output data because the clinical knowledge possessed by clinicians may affect their annotation. This information is managed by a panel of buttons, controlled by the function `uipanelClinician()`, which saves the strings 'Clinician' and 'Not a clinician' (depending on whether the user selected 'YES' or 'NO'), that will then be printed on the output file.

In the second section the user starts his annotation path. First he is asked to indicate which is going to be his annotation, that could be a whole new one, the continuation of a previous one with the addition of new junctions and/or widths, and the importation of points previously annotated on which measure again bifurcation angles and widths. The user can choose one of these three options using a pop-up menu (Figure 3.5) controlled by the functions `popupmenu_Callback()` and `popupmenu_CreateFcn()`. Depending on which selection the user made, the function `popupmenu_Callback()`, thanks to the use of `set()`, activates or not the push-buttons that will be used to upload images and text files.



PHASE 1: Identification
Insert your name and specify if you are a clinician or not

ANNOTATOR NAME: ILARIA PIERETTI

ARE YOU A CLINICIAN? YES NO

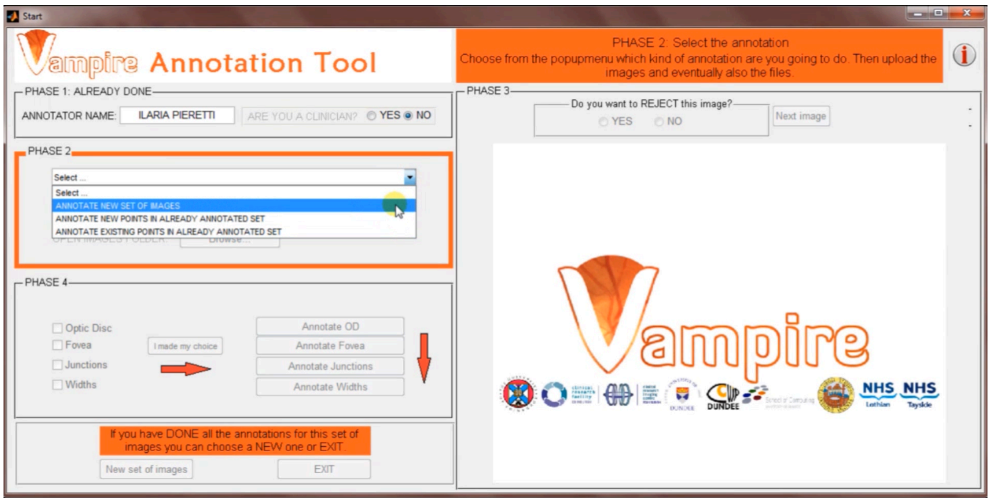
Do you want to REJECT this image? YES NO

If you have DONE all the annotations for this set of images you can choose a NEW one or EXIT

Figure 3.4: User's identification phase

After this, the user has to select the folder that contains the set of images he wants to annotate and eventually the text files where the data of previous annotation are saved. This operation is done through the use of two push-buttons (Figure 3.6) controlled by the functions `openFolderButton()` and `openFileButton()`.

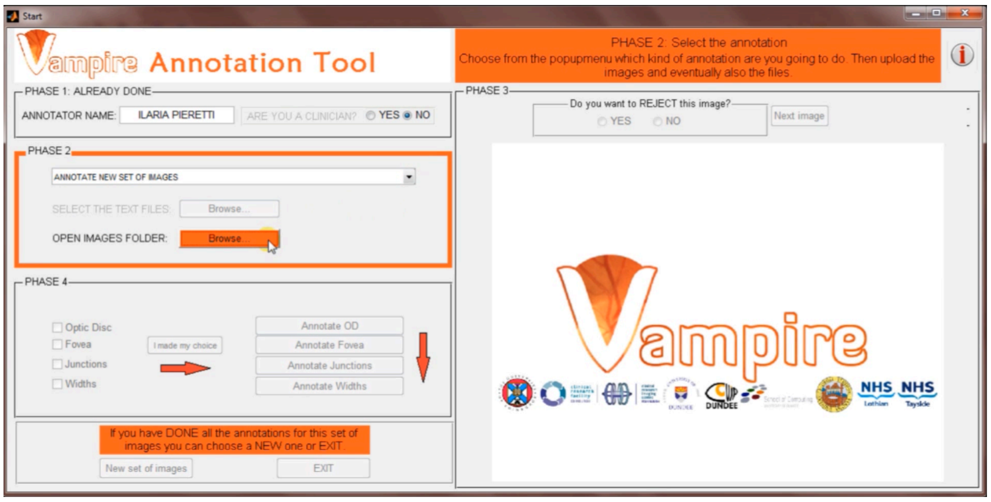
When the user presses the button to choose the set of images, a window, where the user has to find and select the folder containing the set, will appear (Figure 3.7) thanks to the command `uigetdir()`. When the folder is selected, the images contained are saved in the structure (handles) and the folder "RESULTS" is created by `mkdir()`; right there the text files with the outputs, one for each image, are produced (`fopen()`). Then all the retinal images are read and saved in the structure (`imread()`), and the first one is plotted on the screen


PHASE 2: Select the annotation
Choose from the popupmenu which kind of annotation are you going to do. Then upload the images and eventually also the files.

ANNOTATOR NAME: ILARIA PIERETTI

ARE YOU A CLINICIAN? YES [✓ NO]

Figure 3.5: Process of choosing the annotation type


PHASE 2: Select the annotation
Choose from the popupmenu which kind of annotation are you going to do. Then upload the images and eventually also the files.

ANNOTATOR NAME: ILARIA PIERETTI

ARE YOU A CLINICIAN? YES [✓ NO]

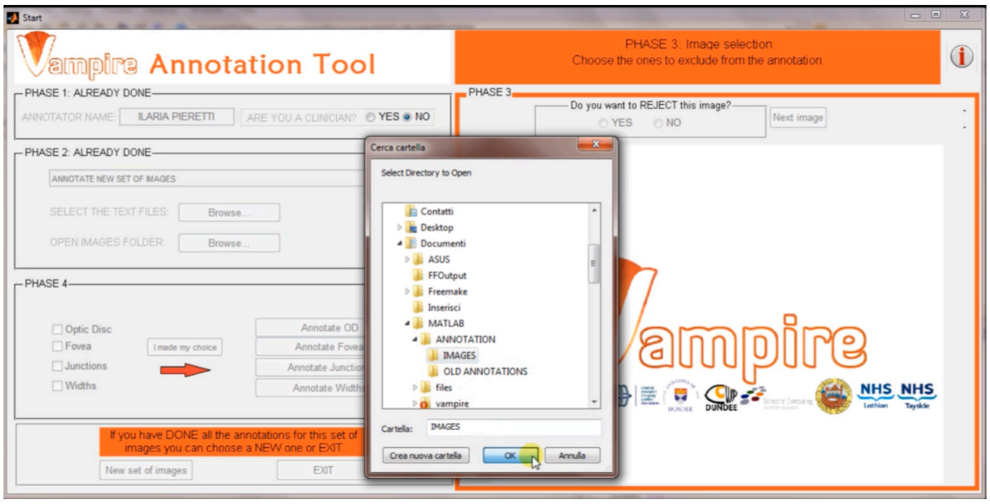
ANNOTATE NEW SET OF IMAGES

Figure 3.6: Loading images and data

(`imshow()`).

In the same way a window appears when the user presses the button to select the text files with the output data of an old annotation (`uigetfile()`). He can select all the files at once, or select one at a time, and their path is stored (in the variable 'FileName') by the program in order to be recovered in the later stages.

In the third section the user has an overview of all the images contained in the folder previously selected (Figure 3.8). With this overview the user can assess the quality of the images of the set and decide which ones to include or reject from his annotation. The user can express his choice selecting one of the two radio-buttons "Yes" or "No", regulated by the functions `YesProcess()` and `NoProcess()`.



PHASE 3: Image selection
Choose the ones to exclude from the annotation.

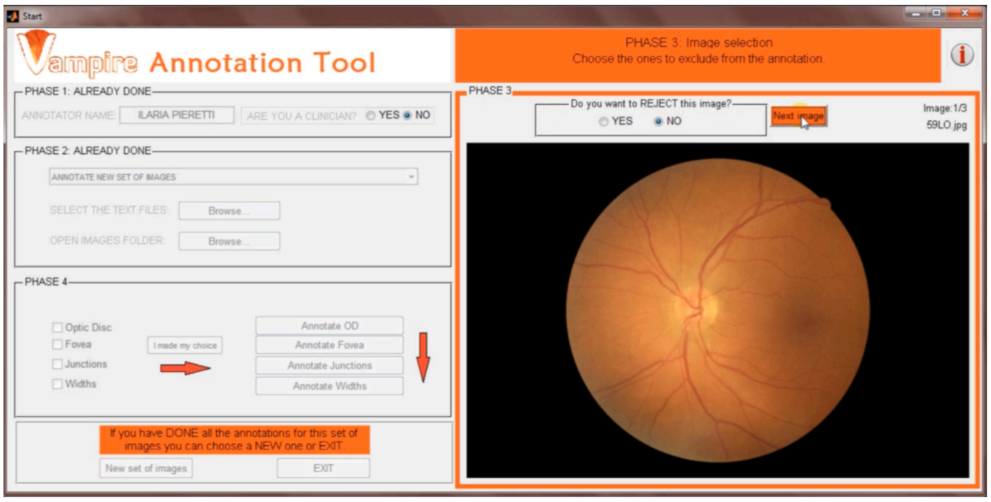
ANNOTATOR NAME: ILARIA PIERETTI
ARE YOU A CLINICIAN? YES [✓ NO]

If you have DONE all the annotations for this set of images you can choose a NEW one or EXIT

Cartella: IMAGES

Figure 3.7: Search for the folder of the images to be annotated

These two functions respectively save the strings "PROCESSED" and "NOT PROCESSED", that will then be printed in the output text file. After making the choice, the user presses the push-button "Next Image", which activates the function `NextImage()`, which, despite the long code, simply writes in the text file all the information so far obtained (thanks to the function `write_GeneralInformations()` described in paragraph 3.3.7), plots the next image and resets the processing panel. If the user has decided to upload data from previous annotations, at this moment of the program, they are stored in a structure in order to be better managed later thanks to the function `readTextFile()` (described in paragraph 3.3.8).


PHASE 3: Image selection
Choose the ones to exclude from the annotation.

ANNOTATOR NAME: ILARIA PIERETTI
ARE YOU A CLINICIAN? YES [✓ NO]

Do you want to REJECT this image? YES [✓ NO]

Image:1/3
59LO.jpg

If you have DONE all the annotations for this set of images you can choose a NEW one or EXIT

Figure 3.8: Overview of images and selection

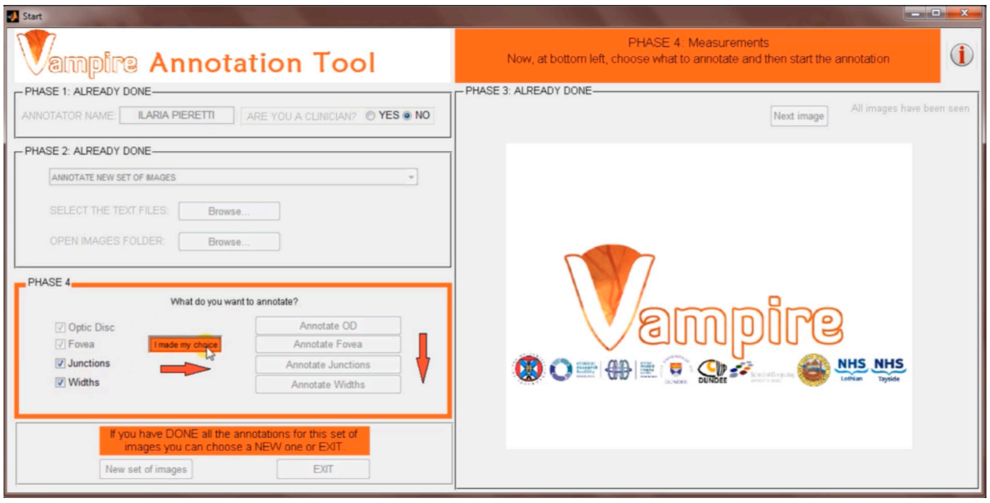
When the overview of images is finished, the actual annotation can finally begin. In the fourth phase, the user has to select from a menu, consisting of four check-boxes, what features he wants to annotate (Figure 3.9). As already mentioned above, depending on whether he decided to do a completely new annotation or continue a previous one, the boxes of optic disc and fovea will already be automatically selected or deselected (and the user can not modify them). Those of junctions and widths will instead let be checkable. All this is handled by the function `ChooseAnnotationsButton()`, thanks to `get()` that reads the status of the check-boxes.

For each selected check-box, a push-button is activated (Figure 3.10), which, once pressed, will bring the user in the annotation section of the corresponding feature. The push-buttons are activated thanks to the function `MenuAnnotationSelection()`.

3.3.2 Optic Disc Annotation Tool

This section is executed only if the user began the annotation of a new set of images ever annotated before, otherwise the OD for these images is uploaded from the output data of a previous annotation. In this section, as in the next ones, only the images not rejected in the initial overview are displayed (and so accessible to be annotated).

The first block consists of the function `AnnotateOD_OutputFcn()` containing the lines of code


Vampire Annotation Tool

PHASE 4: Measurements
Now, at bottom left, choose what to annotate and then start the annotation

PHASE 1: ALREADY DONE
ANNOTATOR NAME: ILARIA PIERETTI
ARE YOU A CLINICIAN? YES [✓ NO]

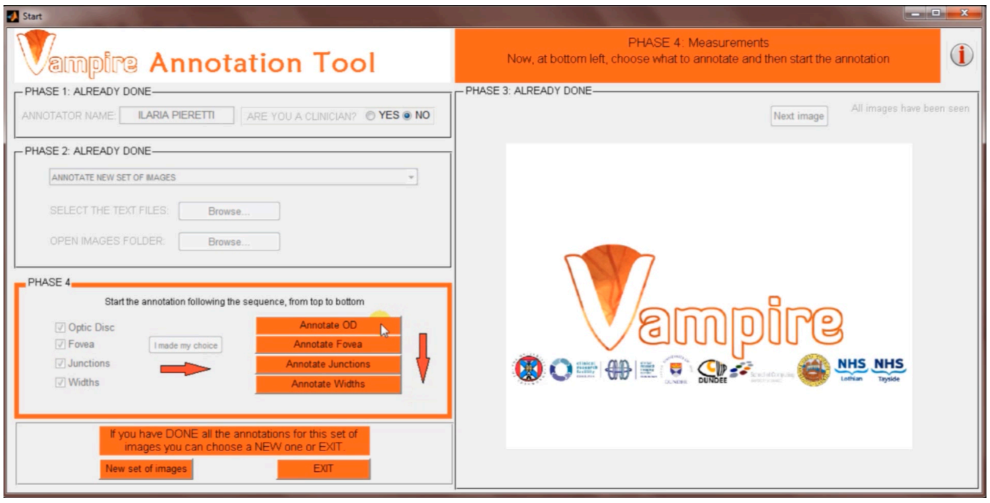
PHASE 2: ALREADY DONE
ANNOTATE NEW SET OF IMAGES
SELECT THE TEXT FILES:
OPEN IMAGES FOLDER:

PHASE 3: ALREADY DONE
All images have been seen

PHASE 4
What do you want to annotate?
[✓ Optic Disc] [✓ Fovea] [✓ Junctions] [✓ Widths]

If you have DONE all the annotations for this set of images you can choose a NEW one or EXIT

Figure 3.9: The check-menu of the features


Vampire Annotation Tool

PHASE 4: Measurements
Now, at bottom left, choose what to annotate and then start the annotation

PHASE 1: ALREADY DONE
ANNOTATOR NAME: ILARIA PIERETTI
ARE YOU A CLINICIAN? YES [✓ NO]

PHASE 2: ALREADY DONE
ANNOTATE NEW SET OF IMAGES
SELECT THE TEXT FILES:
OPEN IMAGES FOLDER:

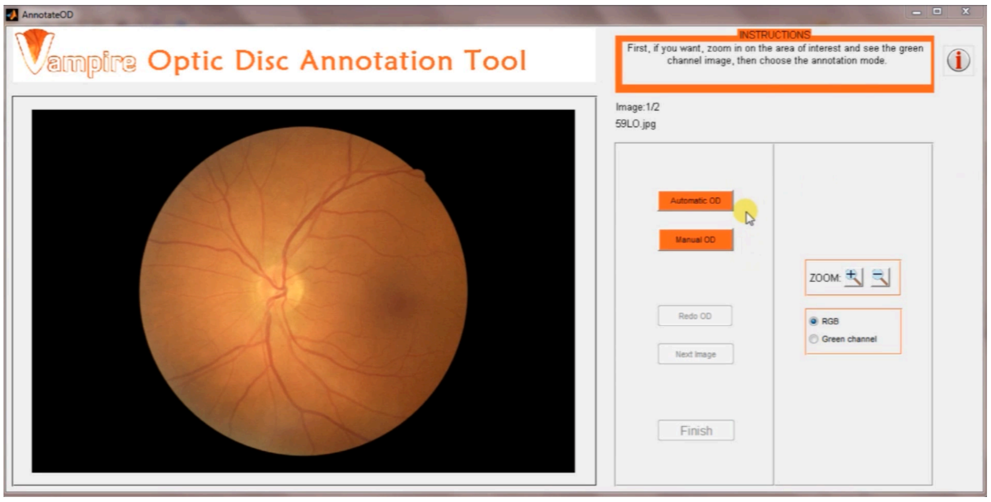
PHASE 3: ALREADY DONE
All images have been seen

PHASE 4
Start the annotation following the sequence, from top to bottom
[✓ Optic Disc] [✓ Fovea] [✓ Junctions] [✓ Widths]

If you have DONE all the annotations for this set of images you can choose a NEW one or EXIT

Figure 3.10: The buttons that bring to the screens of the different features

that set the initialization data for the graphics (eg. `imread('logo')`) and for the annotation, and load the images to be processed. In addition, at the end of the function, the first image to be processed is loaded on the screen and the buttons of interaction with the user are set (thanks to `setButtonStateFor_ChoseAnnotation()`) so that the annotation can begin.



First, if you want, zoom in on the area of interest and see the green channel image, then choose the annotation mode.

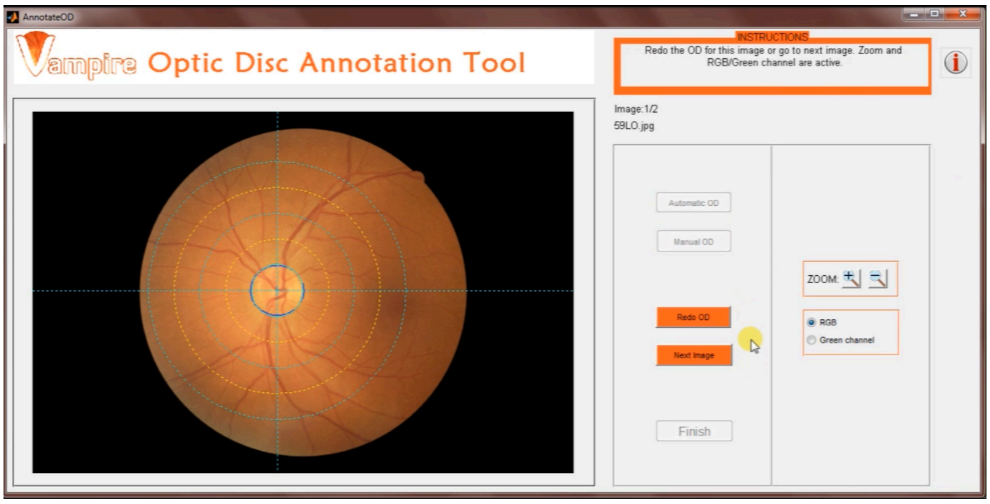
Image:1/2
59LO.jpg

Figure 3.11: Choice of automatic or manual annotation of the OD

Two modes of annotation of the OD are available for the user (block two): a manual one and an automatic one that is the VAMPIRE's one [7] (explained in Chapter 2). The choice to insert an automatic part (which is the only one in the whole tool) in a manual tool, arises from the need to have a fixed reference point (the OD center) that has to be the same one calculated with the VAMPIRE automatic software, so that the same point in a certain image has the same polar coordinate referred to the OD. This means that the measures for example of the width of the vessel in that point made with VAMPIRE and the one made with this manual tool can be comparable in terms of position in the whole vasculature.

The automatic detection starts by pressing the push-button "Automatic OD" (Figure 3.11) regulated by the function `autolocateButton()`. The VAMPIRE's function `automaticODlocation()` calculates the OD, and then the function `drawCircles()` plots the circle that represents the OD and other four respectively distant 0.5ODD, 1ODD, 1.5ODD and 2ODD (ODD=diameter of the OD) from the border of the OD (Figure 3.12).

The manual annotation begins instead by pressing the push-button "Manual OD" managed



Redo the OD for this image or go to next image. Zoom and RGB/Green channel are active.

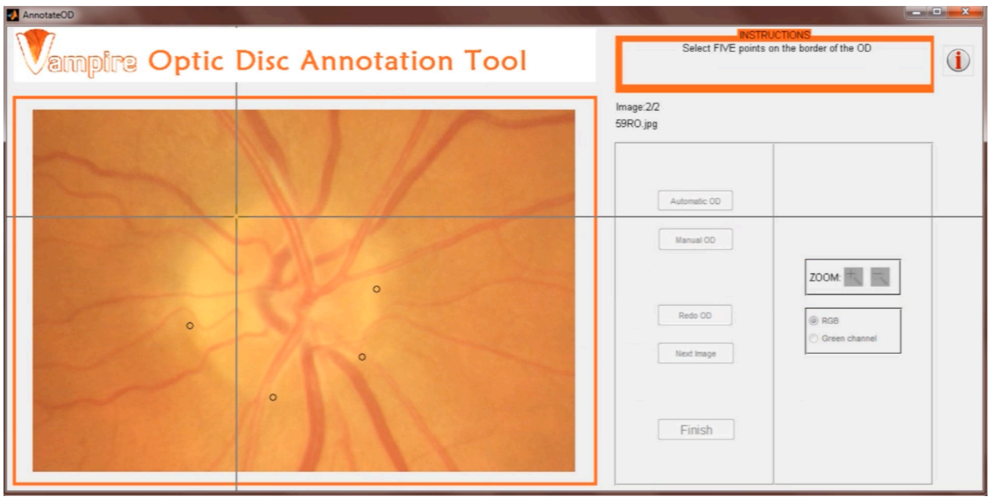
Figure 3.12: The OD calculated and plotted with the circles that delimit the zones

by the function `locateButton()`. The user is asked to take five points on the OD's border (Figure 3.13). The command that detects the user's input and allows to save it is `ginput()` that memorizes the Cartesian coordinates of the point selected. When all the points on the border are taken, the function `fitellip()` calculates the ellipse that fits on the input points and then the function `drawellip()` plots it. Then the function `fitellipse()` returns the OD center and its two radius. With this information the function `drawCircles()` plots the OD and the circles that delimit the zones around it.

Once the Optic Disc (manual or automatic) is calculated and plotted, the tool allows the user to zoom in and/or to see the green channel image, so that he can better evaluate the quality of the computed OD and then decide whether to do it again or go to the next image (this part of code is represented by the third block of OD's column in the scheme of Figure 3.3). The functions that manage the image's screen view are `zoomOUT()` and `zoomIN()`, which control the activation of two push-bottons respectively for the decrease and the increase of the zoom on the image, and `uipanelC_G()` which controls a panel consisting of two radio-buttons that allow the visualization of the image in RGB format or in green channel format (Figure 3.14).

If the user decides to redo the OD (block four of OD's column in the scheme of Figure 3.3) he has to press the push-button "Redo OD" that produces the run of the function `clearButton()`

in which all the data related to the wrong OD are deleted and the image replotted.



Select FIVE points on the border of the OD

Figure 3.13: Manual selection of the five points on the border of the OD

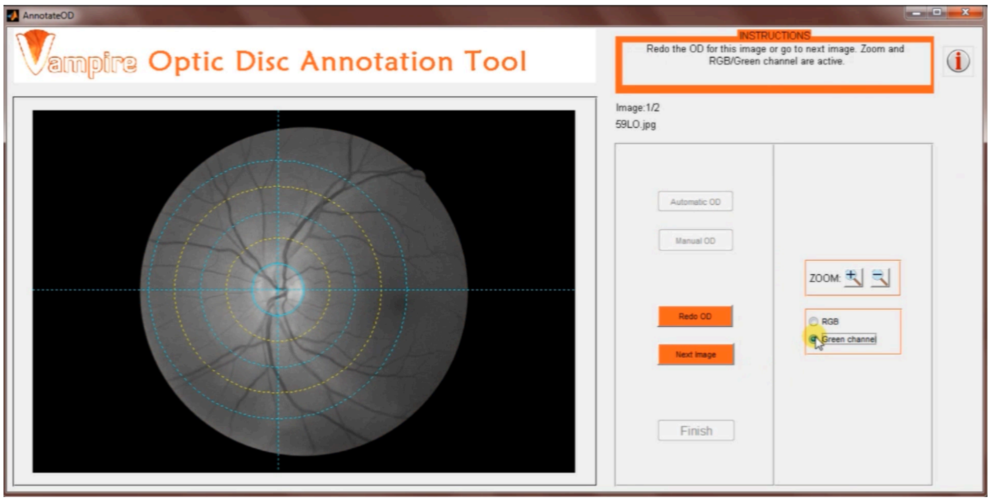
When the OD is believed to be accurate, the user can go to the next one pressing the push-button "Next image". Before plotting the image, the tool saves all the data for the current image in the text file (block five). The data are saved with the use of the function `write_OD()` which is described together with all the other functions with this aim in paragraph 3.3.7.

If the current image was not the last one, the user follows the same protocol used for the annotation of the OD of the previous image. If it was instead the last one, the push-button "Finish" runs the function `finish()` (also included in block five) that closes the "Optic Disc Annotation Tool" screen and takes the user to the START menu.

The last block represents all those functions that regulate the activation of buttons and panels, which are each time called in the code. In Appendix B just one is reported as an example because they are all structurally similar.

3.3.3 Fovea Annotation Tool

As for the section of the OD, even that of the fovea is executed only if the user has chosen a set of images not previously annotated, otherwise the data of the fovea are loaded directly from the output file of another annotation. The first block in this case represents two functions,



INSTRUCTIONS
Redo the OD for this image or go to next image. Zoom and RGB/Green channel are active.

Image:1/2
59LO.jpg

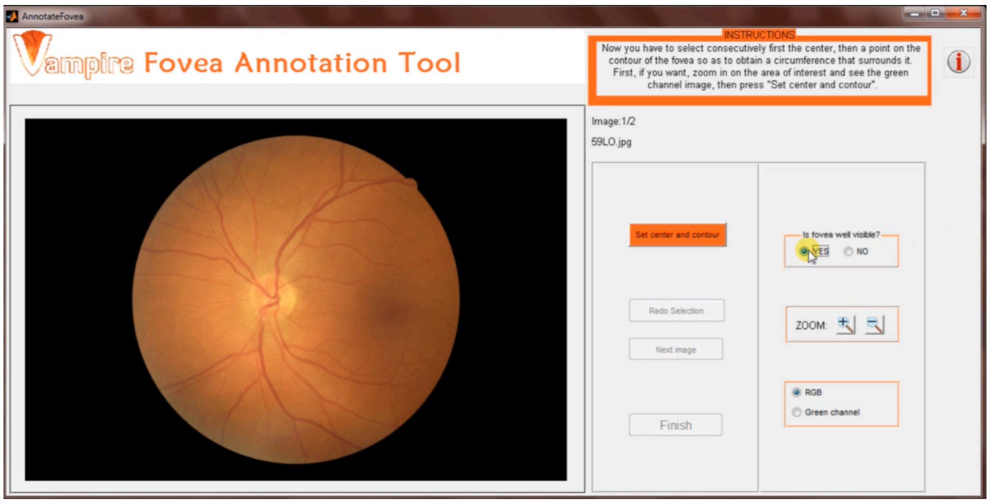
Figure 3.14: The green channel view of the image

one similar to the one of the first block of the OD (here called `AnnotateFovea_OutputFcn()`), and another one called `uipanelFovea_SelectionChangeFcn()`. This function activates a panel (Figure 3.15) that controls two radio-buttons with which the user is forced to specify whether the fovea in the current image is clearly visible or not, essential information for the reasons explained in paragraph 3.1.

After specifying the visibility of the fovea, the user has to do its outright annotation, by pressing the push-button “Set center and contour” (Figure 3.16) that activates the function `setCenterContour()`.

After pressing the button, he has to select consecutively two points, the first in the center of the fovea and the other one on the contour (Figure 3.17). This selection is managed by the function `getUserInput()` that saves the Cartesian coordinates of the points thanks to `ginput()` and plots them on the image.

As already repeated several times before, the tool also saves the polar coordinates of each point (therefore also of fovea’s center and contour), and it is done thanks to the function `polar_coordinates()`. This function requires eight input values: the Cartesian coordinates of the point, the size of the image (height and width), the Cartesian coordinates of OD and fovea. The polar coordinates with pole in the OD (angular distance with respect to both the vertical axis and to the one perpendicular to the OD-fovea axis) and those with pole in the



INSTRUCTIONS

Now you have to select consecutively first the center, then a point on the contour of the fovea so as to obtain a circumference that surrounds it. First, if you want, zoom in on the area of interest and see the green channel image, then press "Set center and contour".

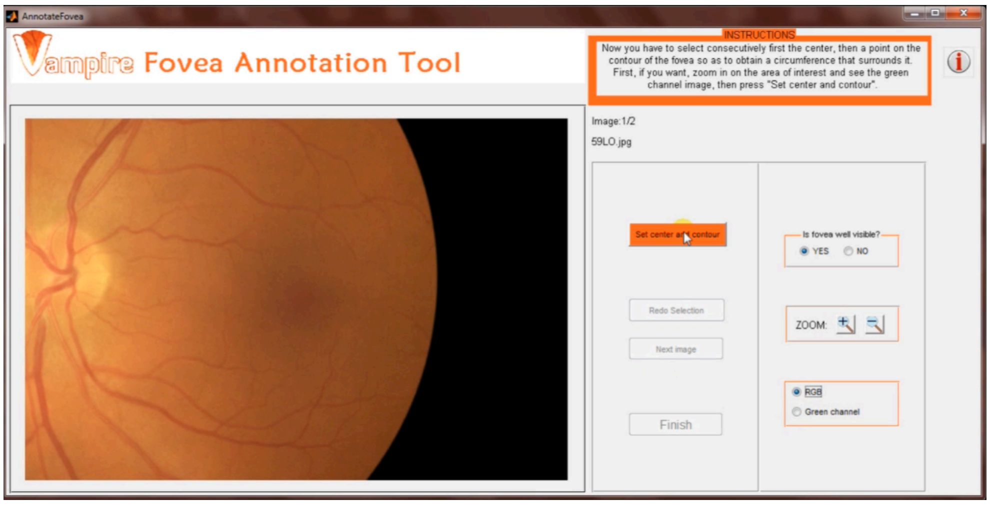
Image:1/2

59LO.jpg

Is fovea well visible? [✓ YES] NO

[✓ RGB] Green channel

Figure 3.15: Specification panel of the visibility of the fovea



INSTRUCTIONS

Now you have to select consecutively first the center, then a point on the contour of the fovea so as to obtain a circumference that surrounds it. First, if you want, zoom in on the area of interest and see the green channel image, then press "Set center and contour".

Image:1/2

59LO.jpg

Is fovea well visible? [✓ YES] NO

[✓ RGB] Green channel

Figure 3.16: Start of fovea's annotation

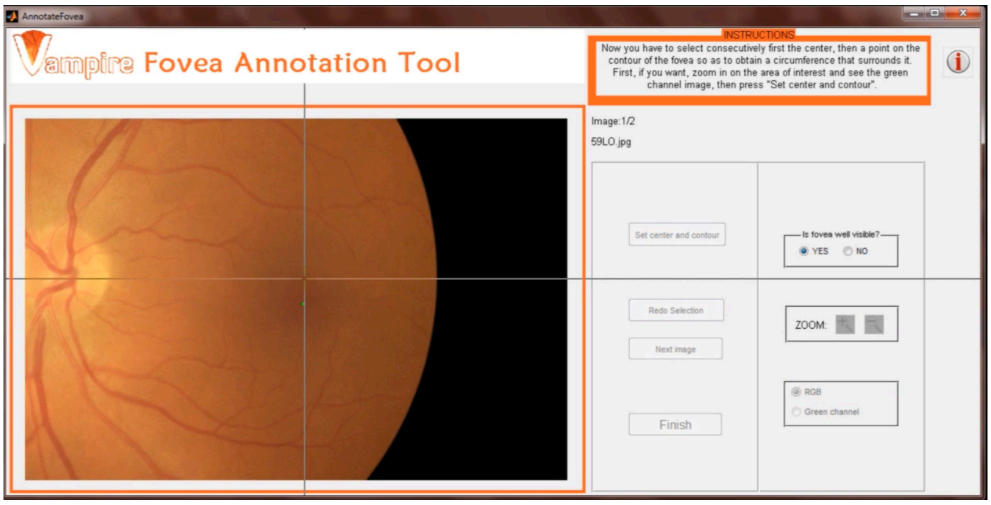

INSTRUCTIONS
Now you have to select consecutively first the center, then a point on the contour of the fovea so as to obtain a circumference that surrounds it. First, if you want, zoom in on the area of interest and see the green channel image, then press "Set center and contour".

Image:1/2
59LO.jpg

Is fovea well visible? [✓ YES] NO

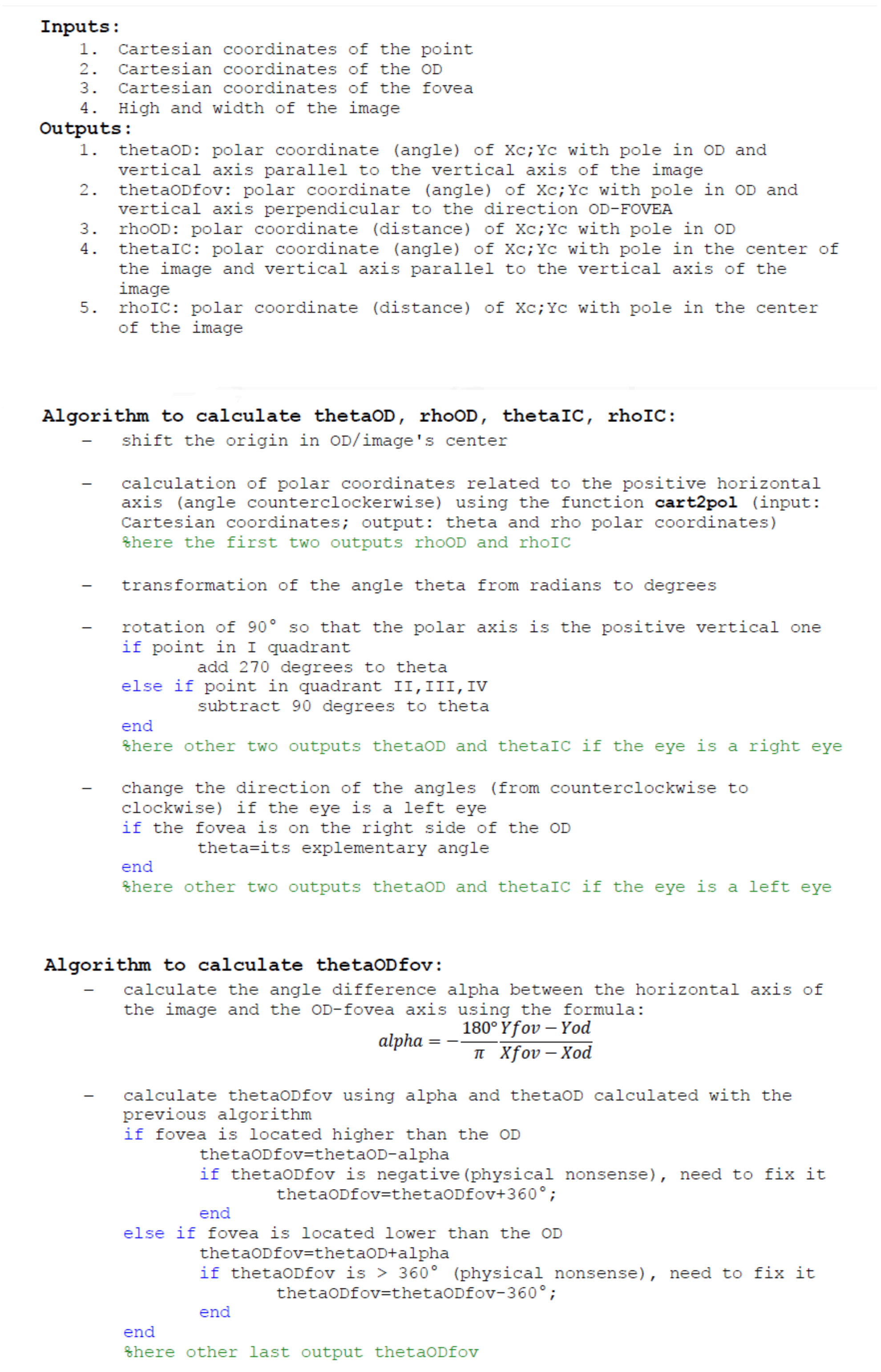
Figure 3.17: Selection of fovea's center and contour

center of the image are given as outputs. Please refer to paragraph 3.1 for the explanation of these parameters in detail. The algorithm used for the calculation of the polar coordinates from Cartesian ones is shown in Figure 3.18.

At this stage the tool calculates the radius of the fovea and runs the function `drawcirclefovea()` which calculates the circle with center in the center of the fovea and radius just calculated, and then plots this circle to help the user evaluating if the selection is well taken or not (Figure 3.19).

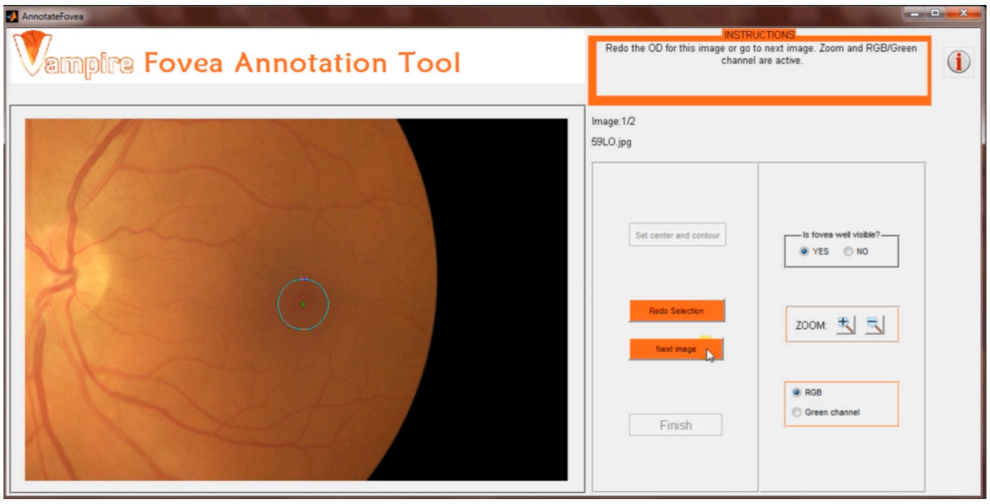
As in the OD section the user can now use the zoom panel and/or switch the RGB image in the green channel one for better evaluate his selection (block three). If he is not satisfy of the annotation he toke, he is allowed to redo the selection of the two points pressing the push-button "Redo selection" (block four) . The function `RemoveButton()` regulates this operation deleting the values of center and contour of the fovea of the current image and then plotting again the image without the incorrect fovea.

If instead the user believes the fovea annotated matches with the real one, he can go to the next image by pressing the push-button "Next image". The function `nextImageButton()` (block five) ensures that the data of the fovea are saved in the text file (function `write_Fovea()` described in paragraph 3.3.7), and then plots the next image if the current one is not the last one, otherwise the function `finish()` (also block five) runs, taking the user to the main menu.



```
Inputs:
     1.  Cartesian coordinates of the point
     2.  Cartesian coordinates of the OD
     3.  Cartesian coordinates of the fovea
     4.  High and width of the image
Outputs:
     1.  thetaOD: polar coordinate (angle) of Xc;Yc with pole in OD and
         vertical axis parallel to the vertical axis of the image
     2.  thetaODfov: polar coordinate (angle) of Xc;Yc with pole in OD and
         vertical axis perpendicular to the direction OD-FOVEA
     3.  rhoOD: polar coordinate (distance) of Xc;Yc with pole in OD
     4.  thetaIC: polar coordinate (angle) of Xc;Yc with pole in the center of
         the image and vertical axis parallel to the vertical axis of the
         image
     5.  rhoIC: polar coordinate (distance) of Xc;Yc with pole in the center
         of the image


Algorithm to calculate thetaOD, rhoOD, thetaIC, rhoIC:
     -   shift the origin in OD/image's center

     -   calculation of polar coordinates related to the positive horizontal
         axis (angle counterclockerwise) using the function cart2pol (input:
         Cartesian coordinates; output: theta and rho polar coordinates)
         %here the first two outputs rhoOD and rhoIC

     -   transformation of the angle theta from radians to degrees

     -   rotation of 90° so that the polar axis is the positive vertical one
         if point in I quadrant
                add 270 degrees to theta
         else if point in quadrant II,III,IV
                subtract 90 degrees to theta
         end
         %here other two outputs thetaOD and thetaIC if the eye is a right eye

     -   change the direction of the angles (from counterclockwise to
         clockwise) if the eye is a left eye
         if the fovea is on the right side of the OD
                theta=its explementary angle
         end
         %here other two outputs thetaOD and thetaIC if the eye is a left eye


Algorithm to calculate thetaODfov:
     -   calculate the angle difference alpha between the horizontal axis of
         the image and the OD-fovea axis using the formula:
                         $alpha = -\frac{180°}{\pi}\frac{Yfov - Yod}{Xfov - Xod}$

     -   calculate thetaODfov using alpha and thetaOD calculated with the
         previous algorithm
         if fovea is located higher than the OD
                thetaODfov=thetaOD-alpha
                if thetaODfov is negative(physical nonsense), need to fix it
                       thetaODfov=thetaODfov+360°;
                end
         else if fovea is located lower than the OD
                thetaODfov=thetaOD+alpha
                if thetaODfov is > 360° (physical nonsense), need to fix it
                       thetaODfov=thetaODfov-360°;
                end
         end
         %here other last output thetaODfov
```

Figure 3.18: Algorithm for calculating of polar coordinates



INSTRUCTIONS
Redo the OD for this image or go to next image. Zoom and RGB/Green channel are active.

Image:1/2
59LO.jpg

Is fovea well visible? [✓ YES] NO

[✓ RGB] Green channel

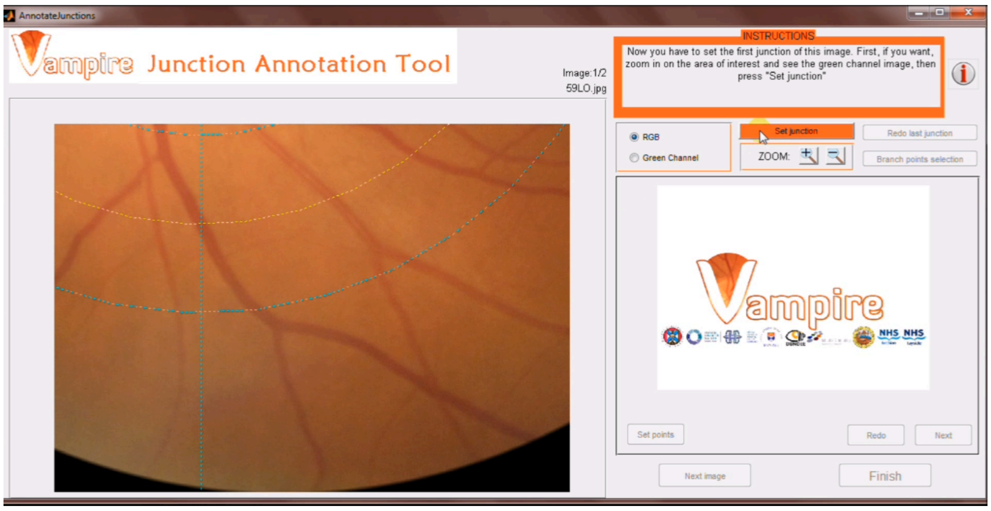
Figure 3.19: The circle which delimits the fovea as selected by the user

Even in the environment of the fovea the functions of block six, those who manage the activation of the graphical components, can be found at the end. The code of these functions is not reported in Appendix B as it is substantially the same as the one of the OD.

3.3.4 Junctions Annotation Tool

The annotation section of the junctions can be divided into two big parts. The first one consists of the annotation of the points where the user wants to perform the measurement of the bifurcation angles, the second one is the execution of their actual measure.

The first part is shown schematically by the first four blocks in the fourth column of Figure 3.3, which have a structure very similar to those of OD and fovea: the first block consists of the code that loads all the necessary information, the second one handles the annotation of the points in a big screen where the entire image can be seen, a third block for the management of zoom and color of the image, and the fourth one that is involved in changing and/or removing the annotated points. In Appendix B, only the code of the functions of block two is shown because the other three are very similar to those fully described in the OD's section. The annotation of the points is managed by the function `setJunctionButton()` that runs when the user presses the push-button "Set junction" (Figure 3.20).



Now you have to set the first junction of this image. First, if you want, zoom in on the area of interest and see the green channel image, then press "Set junction"

Figure 3.20: Start of junctions' annotation

First, the function `getUserInputJunction()` is called: it allows the user to select a point in the retinal image, memorizing its Cartesian coordinates. One of the essential things the tool should allow is the rescue on the type of vessel (vein or artery) in which each point is selected. This is possible thanks to the use of the Shift key: if the user presses simultaneously the Shift key while clicking on the image, then the point will appear blue indicating a vein (Figure 3.21); if instead the selection is made with a simple click without the Shift key pressed, then the point (plotted in red) belongs to an artery. To manage all this, the property "SelectionType" is used, which reads the state of a key (in this case Shift), which can be then extrapolated thanks to `get()`. Finally, the polar coordinates of the selected point are calculated and stored, using the function `polar_coordinates()` explained in the previous paragraph.

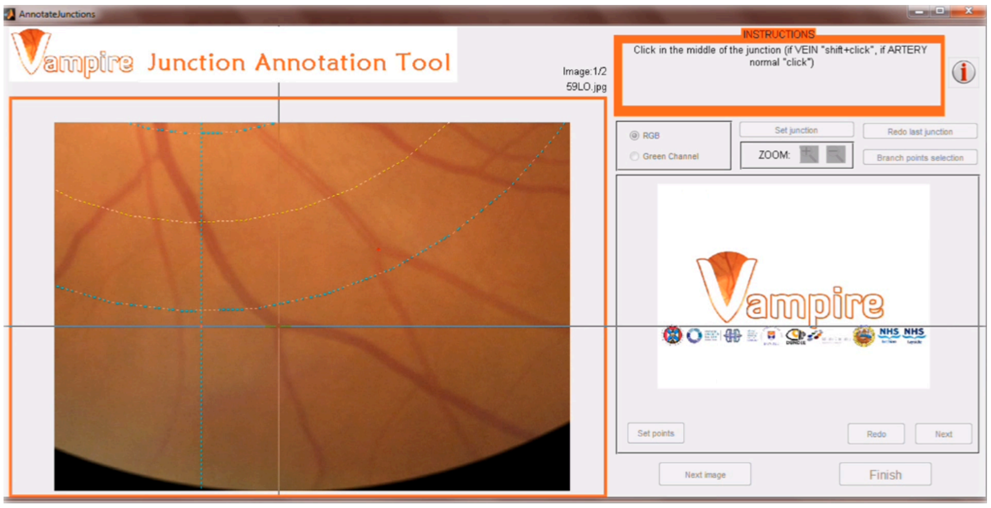
After the annotation of the points on big picture, the function `setVesselsPointsButton()` allows the passage to the second part. This function is run by the user pressing the push-button "Branch points selection", thus indicating the end of the first phase and the need to move to the second one. During the second phase all the points just annotated are loaded one at a time on a small screen zoomed in on the current one, in order to allow a more accurate measure the angles. In order not to lose the general view, however, at the same time in the main screen you can see all the points, and a yellow box appears around the one in which the annotation is in progress (Figure 3.22). As you can see in Figure 3.22, the function `setVesselsPointsButton()`

activates the small screen and the buttons related to it, loads the image, plots the first point annotated and makes an automatic zoom in on the area having as its center the point itself. This feature also contains the code to plot the yellow box on the main screen.

The function `PointsForNextJunctionButton()` is the one that manages the measure of bifurcation angles , and when it runs the tool has already loaded the current point on which to perform the measure. This function immediately calls the function `getUserInputPoints()` that allows the user the selection of three points (by saving their Cartesian coordinates), which must be taken within the three vessels involved in the junction chosen (as centrally as possible), as shown in Figure 3.22. After the selection of the first point in the mother vessel, the function `drawcirclefovea()` (described in the previous paragraph) plots a circle centered in the junction and with radius equal to the distance between the center and the point in the vein mother. Thanks to this expedient, the user is facilitated in the selection of the two points in the children vessels, thus increasing the accuracy. Subsequently, inside `PointsForNextJunctionButton()`, the function `calculateAngles()` is invoked, whom plots the three segments along the mother vessel and the children ones (Figure 3.23), and automatically calculates the value of the three angles of bifurcation taking into account the fact that the user may have made the selection of the points either clockwise or counterclockwise.

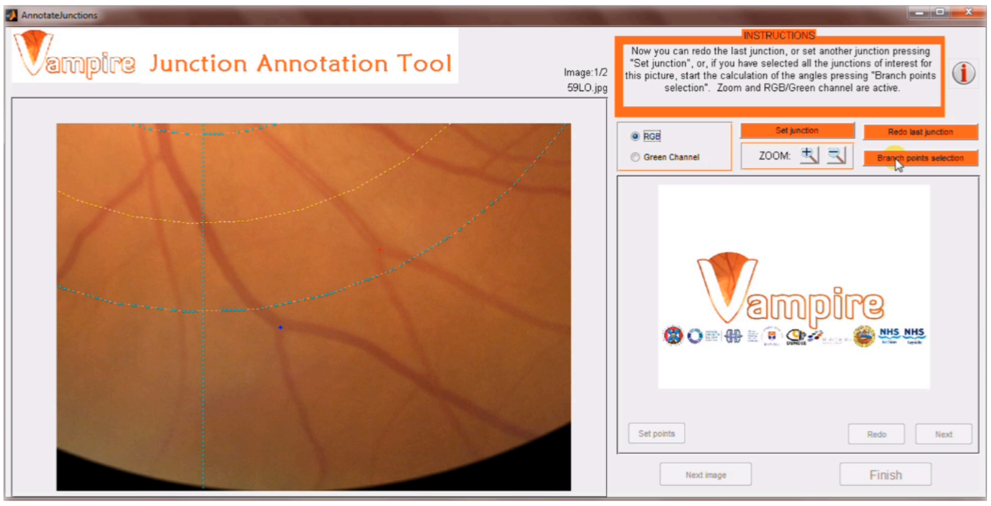
After making the selection of the three points, the user can choose whether to continue the annotation with next junction pressing the push-button "Next", or to redo the selection just made pressing the push-button "Redo". These two buttons respectively run the functions `nextJunction()` and `removeLastVesselsPointsButton()`, of whom the code will not be reported. The function `nextJunction()`partially coincides with the function `PointsForNextJunct ionButton()`, but obviously without the part of the initialization of the small screen, and so having only the code that plots the yellow box on the main screen, and those that automatically zoom and plot the new point in the small screen . The function `removeLastVesselsPointsButton()` is basically very similar to all the "Remove" functions previously seen and described.

When the points for the current image are finished, the user goes to the next one pressing the push-button "Next Image", which controls the function `nextImageButton()`, very similar to the functions with the same task in the environments of OD and fovea. If the images are finished, the function `finish()`, regulated by the push-button "Finish", is always the one that returns the user to the START menu. Even in the environment of the junctions the last block consists of all those functions that regulate the activation of the graphical buttons.



INSTRUCTIONS

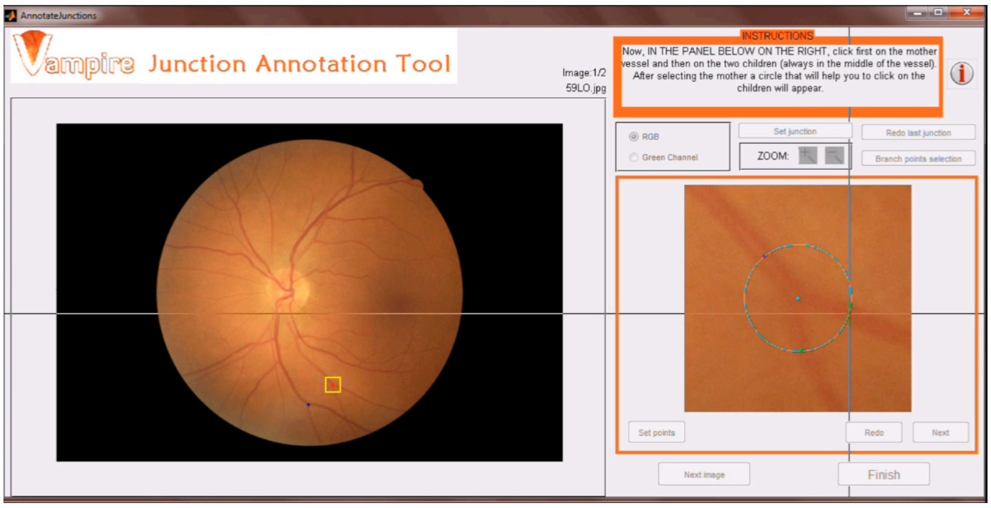
Click in the middle of the junction (if VEIN "shift+click", if ARTERY normal "click")



INSTRUCTIONS

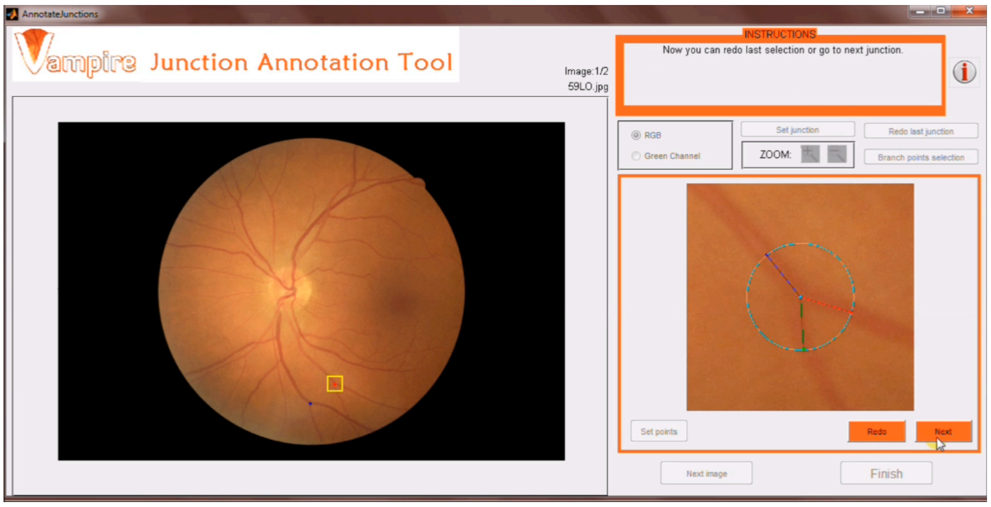
Now you can redo the last junction, or set another junction pressing "Set junction", or, if you have selected all the junctions of interest for this picture, start the calculation of the angles pressing "Branch points selection". Zoom and RGB/Green channel are active.

Figure 3.21: Selection of a junction by the user, indicating the vessel of belonging as a vein



Now, IN THE PANEL BELOW ON THE RIGHT, click first on the mother vessel and then on the two children (always in the middle of the vessel). After selecting the mother a circle that will help you to click on the children will appear.

Figure 3.22: On the right side the small screen where to select the points in the branches, and on the left side, at the same time, the whole image in which the current junction is indicated by the yellow box.



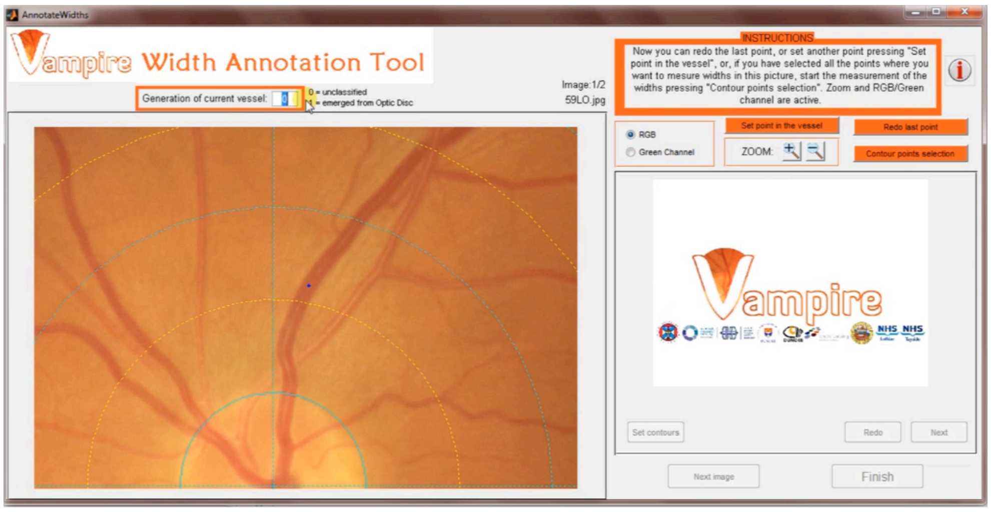
Now you can redo last selection or go to next junction.

Figure 3.23: The three segments on which the angles of bifurcation are calculated

3.3.5 Widths Annotation Tool

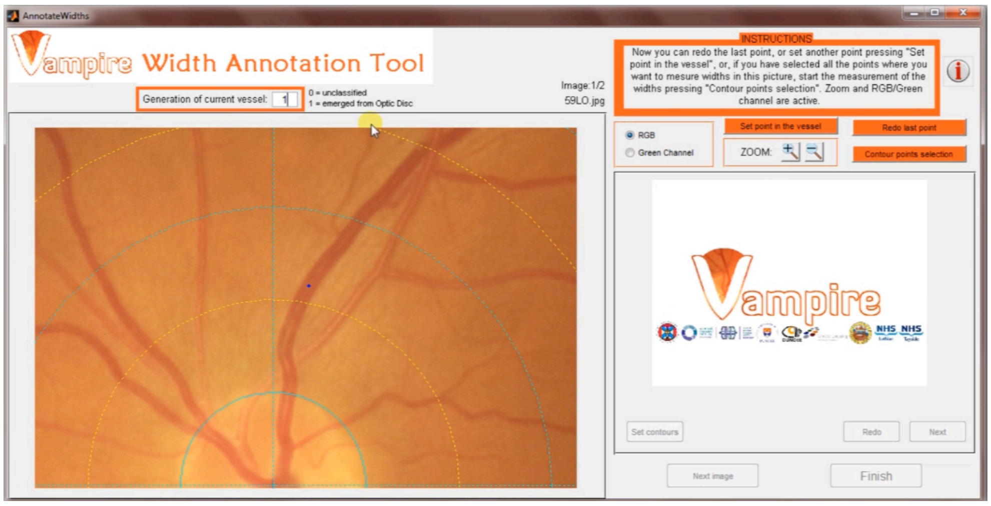
As regards the environment of the widths, its structure is extremely similar to the one of the junctions just seen. The code of the first big phase, which has the aim to select only the measuring points (and not to do the actual measure), coincides largely with the corresponding one in the junctions environment, and therefore it will not be reported again. The only great difference between these two environments in the first phase consists in the fact that, for the widths, an additional information is required: the generation of the vessel. After the selection of each point, the user, if desired, could indicate the generation of the vessel in which the point has been selected. This is done by a panel containing an edit-text box (Figure 3.24) where the user can enter a numeric value that indicates the generation. The edit-text is managed by two functions, `Generation_CreateFcn()` and `generation_Callback ()`, which respectively handle the graphics of the text-edit box and its contents. The number entered by the user is stored in a variable thanks to the function `get()`, considering as default (generation not specified) the number zero.

Even the second phase is very similar to the one of the junctions and has the same execution modes: yellow box on the main screen indicating the current point, measurement made in small screen with zoom fixed on that point, etc. What obviously changes is the function that handles the actual measure (`ContoursForNextPointButton()`) which, at the beginning, calls the function `GetUserInputContours()`. This allows the user to select in the small screen two points, one on each contour of the vessel close to the point selected in the first phase (Figure 3.25). After selecting the first point on one contour, the function `GetUserInputContours()` plots a line passing through the central point and the one just clicked (Figure 3.25), so that the point on the second contour is taken with precision on the direction of the other two. This direction is that perpendicular to the one passing through the center of the vessel, which is automatically plotted on the screen, in red if the vessel is an artery, in blue if it is a vein. Then the function `ContoursForNextPointButton()` then calls `calculateWidths()` which calculates the simple linear distance between the two points just selected on the contours of the vessel, that is its width.



INSTRUCTIONS

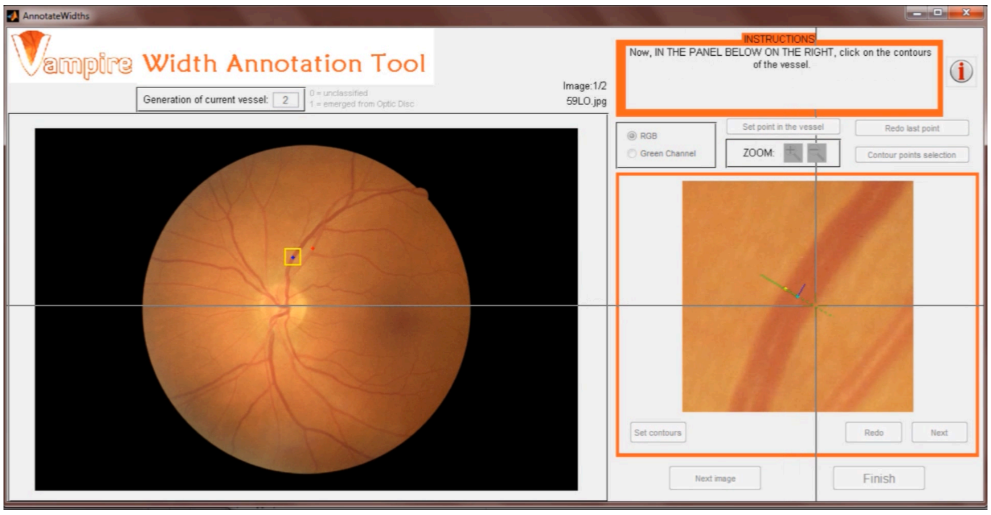
Now you can redo the last point, or set another point pressing "Set point in the vessel", or, if you have selected all the points where you want to mesure widths in this picture, start the measurement of the widths pressing "Contour points selection". Zoom and RGB/Green channel are active.



INSTRUCTIONS

Now you can redo the last point, or set another point pressing "Set point in the vessel", or, if you have selected all the points where you want to mesure widths in this picture, start the measurement of the widths pressing "Contour points selection". Zoom and RGB/Green channel are active.

Figure 3.24: Input from the user of the generation of the vessel to which the newly selected point belongs



Now, IN THE PANEL BELOW ON THE RIGHT, click on the contours of the vessel.

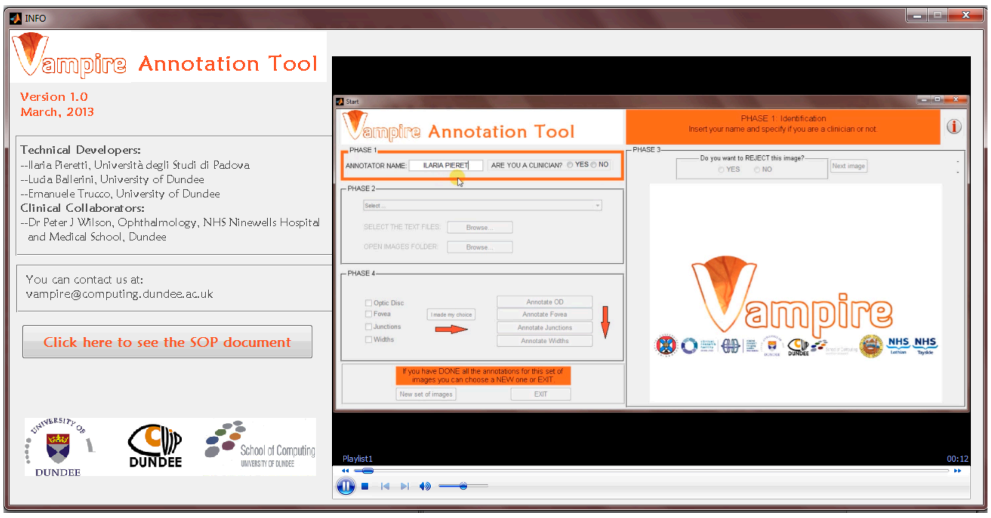
Figure 3.25: Selection of the two points on the vessel's contours

3.3.6 Info environment

Within each screen of the tool, in the upper right corner you can find a push button with the shape of an "i" (Figure 3.26) with which the program opens a new screen that you can see in Figure 3.27. Within this screen you can find information about the software, but also the links to the ".pdf file" containing the Standard Operating Procedure (SOP) and, on the right side, a video with a demo of the program. In this way the user, at any time during the use of the tool, can easily access a sort of "Help".

With regard to the part of the code, the video with the demo appears directly on the screen thanks to the command `actxcontrol()` which creates an ActiveX control, or rather opens the link to a program, which in this case is one for video playback. The command `actxcontrol()` returns a handle `h` to the control, and the video is played thanks to the use of `h.URL` and `h.controls.play`. For the SOP file instead the procedure is very simple: the user, pressing a push button, automatically opens the ".pdf file" thanks to the command `open()`.

Figure 3.26: The logo on the button that brings to the screen containing information and help



INFO

Vampire Annotation Tool

Version 1.0
March, 2013

Technical Developers:
--Ilaria Pieretti, Università degli Studi di Padova
--Lucia Ballerini, University of Dundee
--Emanuele Trucco, University of Dundee
Clinical Collaborators:
--Dr Peter J Wilson, Ophthalmology, NHS Ninewells Hospital
and Medical School, Dundee

You can contact us at:
vampire@computing.dundee.ac.uk

Click here to see the SOP document

Start

PHASE 1: Identification
Insert your name and specify if you are a clinician or not.

PHASE 1
ANNOTATOR NAME: ILARIA PIERET
ARE YOU A CLINICIAN? YES NO

PHASE 3
Do you want to REJECT this image? YES NO

If you have DONE all the annotations for this set of images you can choose a NEW one or EXIT.

Figure 3.27: The screen containing information about the program, a link to the SOP document and, on the right side, the video with the demo

3.3.7 Functions to write outputs

The Vampire Annotation Tool produces as output, for each image belonging to the annotation, a text file containing all the information recorded and the measures performed during the annotation. The operation of writing a text file involves three instructions:

1. Create a new file or open an existing one: `fileID=fopen(fileName, permission)`. The function `fopen()` wants as input the name of the file to create/open, and produces as output a numeric variable that represents the identifier of this file. The string permission describes the type of access you want for the current file: read, write, append or update. In this specific code 'a+' is always used, because it allows to open or create a new file for reading and writing on it and, every time it is re-opened, append data to its end.

2. Write the file: `fprintf(fileID, formats, data)`. The function `fprintf()` writes in the text file identified by fileID the information contained in the variable data. What are called formats, consist in a string delimited by single quotation marks, inside which are the conversion specifications (always preceded by a "%" that indicate the formats of the input data) and some literal text to print (in the specific case of this code it are the data labels.). The available formats are various, but in the code of this program only the follows have been used: %s (string), %c

(single character), %.1f (floating-point number with one decimal place), %u (unsigned integer in base ten). Only one of the escape characters was used: \n (new line).

3. Close the file: fclose(fileID).

In the first line of the text file is always displayed if the image has been excluded or not from the annotation. Specifically, this information is recorded printing "PROCESSED" or "NOT PROCESSED" in the text file. In the second line, information about the image (its name), the user who made the annotation (the name and if he belongs to the category of clinician or not), and the data that uniquely identify the record itself (date and time in which was made) are always present. The function `write_GeneralInformations()` wants as inputs the identifier of the text file (fileID), the variable that contains the string "PROCESSED" or "NOT PROCESSED", the names of the image and the annotator, the string "CLINICIAN" or "NOT CLINICIAN", the variable containing the information of date and time.

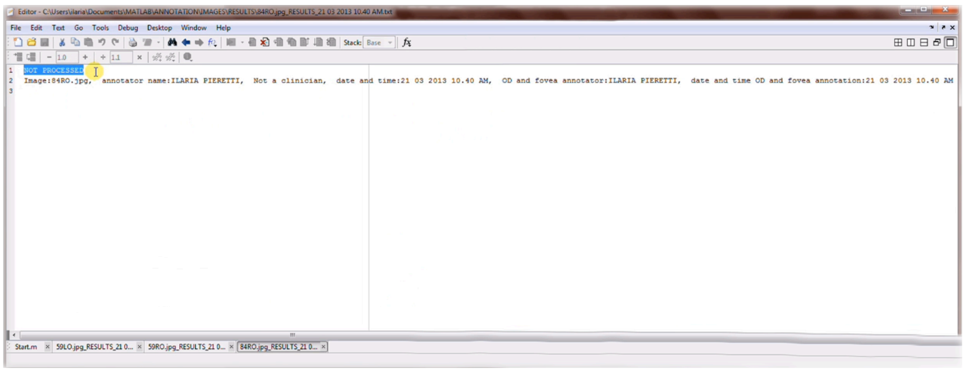
The third line always contains the data related to the Optic Disc: Cartesian coordinates of its center, the two radii and the coefficient theta of the ellipse that fits the OD, and these are also the inputs of the function `write_OD()` that writes this line.

In the fourth line you can always found all the information about the Fovea, written by the function `write_Fovea()` that requires as inputs Cartesian and polar coordinates of its center, the radius and its visibility.

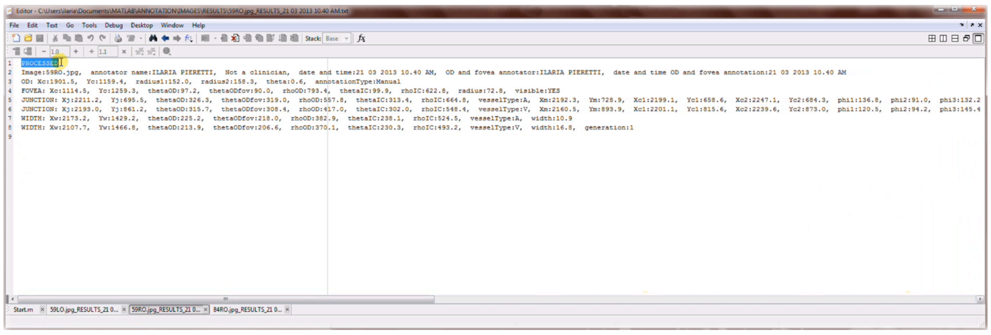
From fifth line onwards you can find information about junctions and widths (depending on the case the number of junctions and widths could range from zero to many). Lines concerning the junctions are written by the function `write_Junctions()` that requires as inputs: Cartesian and polar coordinates of the point in the center of the junction, the type of the vessel (vein or artery) where the junction is, Cartesian coordinates of the three points belonging to the three vessels involved in the junction and the three bifurcation angles. For widths instead the function `write_Widths_Generation()` that writes the text line has as input the Cartesian and polar coordinates of the point within the vessel where you have measured the width, the type of the vessel (vein or artery) where the point is, the width and the generation of this vessel.

For the widths was written a further function (`write_Widths()`) to write text lines when the generation was not indicated by the user.

In Figure 3.28 you can see two examples of the text file the VAMPIRE-Annotation Tool produces, one for an unprocessed image, and one for an image on which all the features have been annotated.


Editor - C:\Users\ilaria\Documents\MATLAB\ANNOTATION\IMAGES\RESULTS\84RO.jpg_RESULTS_21 03 2013 10.40 AM.txt
File Edit Text Go Tools Debug Desktop Window Help
Start.m
59LO.jpg_RESULTS_21 0...
59RO.jpg_RESULTS_21 0...
84RO.jpg_RESULTS_21 0...

```
NOT PROCESSED
Image:84RO.jpg,  annotator name:ILARIA PIERETTI,  Not a clinician,  date and time:21 03 2013 10.40 AM,  OD and fovea annotator:ILARIA PIERETTI,  date and time OD and fovea annotation:21 03 2013 10.40 AM
```


Editor - C:\Users\ilaria\Documents\MATLAB\ANNOTATION\IMAGES\RESULTS\59RO.jpg_RESULTS_21 03 2013 10.40 AM.txt
File Edit Text Go Tools Debug Desktop Window Help
Start.m
59LO.jpg_RESULTS_21 0...
59RO.jpg_RESULTS_21 0...
84RO.jpg_RESULTS_21 0...

```
PROCESSED
Image:59RO.jpg,  annotator name:ILARIA PIERETTI,  Not a clinician,  date and time:21 03 2013 10.40 AM,  OD and fovea annotator:ILARIA PIERETTI,  date and time OD and fovea annotation:21 03 2013 10.40 AM
OD: Xc:1901.5,  Yc:1159.4,  radius1:152.0,  radius2:158.3,  theta:0.6,  annotationType:Manual
FOVEA: Xc:1114.5,  Yc:1259.3,  thetaOD:97.2,  thetaODfov:90.0,  rhoOD:793.4,  thetaIC:99.9,  rhoIC:622.8,  radius:72.8,  visible:YES
JUNCTION: Xj:2211.2,  Yj:695.5,  thetaOD:326.3,  thetaODfov:319.0,  rhoOD:557.8,  thetaIC:313.4,  rhoIC:664.8,  vesselType:A,  Xm:2192.3,  Ym:728.9,  Xc1:2199.1,  Yc1:658.6,  Xc2:2247.1,  Yc2:684.3,  phi1:136.8,  phi2:91.0,  phi3:132.2
JUNCTION: Xj:2193.0,  Yj:861.2,  thetaOD:315.7,  thetaODfov:308.4,  rhoOD:417.0,  thetaIC:302.0,  rhoIC:548.4,  vesselType:V,  Xm:2160.5,  Ym:893.9,  Xc1:2201.1,  Yc1:815.6,  Xc2:2239.6,  Yc2:873.0,  phi1:120.5,  phi2:94.2,  phi3:145.4
WIDTH: Xw:2173.2,  Yw:1429.2,  thetaOD:225.2,  thetaODfov:218.0,  rhoOD:382.9,  thetaIC:238.1,  rhoIC:524.5,  vesselType:A,  width:10.9
WIDTH: Xw:2107.7,  Yw:1466.8,  thetaOD:213.9,  thetaODfov:206.6,  rhoOD:370.1,  thetaIC:230.3,  rhoIC:493.2,  vesselType:V,  width:16.8,  generation:1
```

Figure 3.28: The output text files with the data of the annotation printed

3.3.8 Functions to read the data

In order to get data from the text files and bring them into a Matlab's structure, the function `readTextFile()`, which reads the outputs created by the tool, is used. The syntax for saving data is always the same: "label" + ":" + "data" + ",". Between label and data type there is a one to one relationship, i.e. to each type of data corresponds one and only one label, and vice versa. To read the data, and "recognize" this one-to-one correspondence is then exploited: the label is read and then immediately proceeds the corresponding data. The procedure is always the same for any data, then only a single example is shown in Appendix B.

Thanks to the command `regexp()`, the position of the label within the row is extracted. The label, if it exists in that row, is found and its length is stored. The position of the string containing the data is then within a range whose lower limit is the end of the label after the ":" (then: start of the label + label's length + 1) and the upper limit is the character before the next comma, whose position is found always with `regexp()`.

3.4 Documentation

With the idea of distributing the program also to clinicians not directly related to the VAMPIRE team, we saw the need to produce a documentation as complete as possible. For this reason we have created both a text file containing the Standard Operating Procedure and a video with a demo. The text file describes in detail, with the aid of figures, which procedure the user must follow, explaining carefully step by step what to do (you can find the file in Appendix A). The video shows an example of utilization of the program, trying to reproduce a wide range of use cases. Both of these documents are delivered to clinicians together with the package of the tool, and the user can also directly connect to them through the environment "INFO" described above.

Chapter 4

Application: the study about Sarcopenia

As explained in Chapter 1, by monitoring the retinal vascular network, prevent or diagnose certain particular diseases is now possible. Precisely with this regard, Deepa et al. have decided to examine retinal biomarkers for Sarcopenia in the context of the study "Does the European Working Group on Sarcopenia in Older People algorithm detect all those vulnerable?" [11]. Sarcopenia is that condition associated with the loss of muscle mass and function, and the authors in this study wanted to determine, with the use of standard methods and involving 75 elderly patients (>65 years), the effectiveness of the algorithm developed by Cruz-Jentoft AJ et al. [12] to screen and identify people affected by this disease. In parallel, retinal scans were

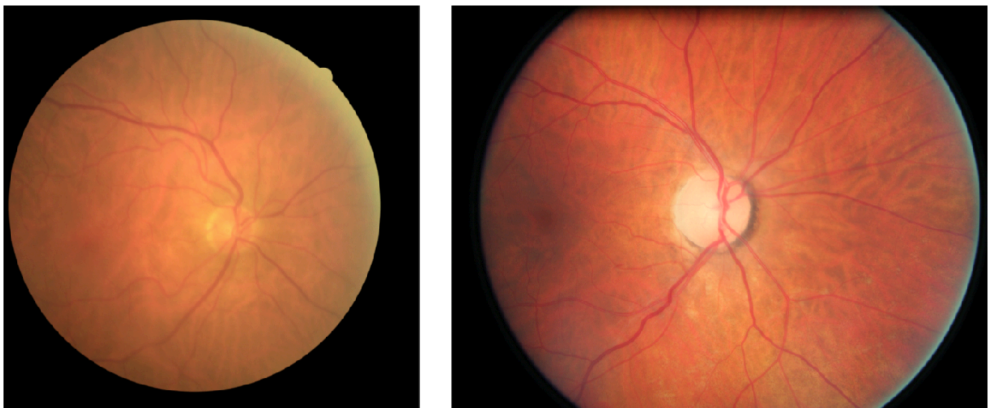


Figure 4.1: On the left side a blurred image and on the right one an image with a wrong scale

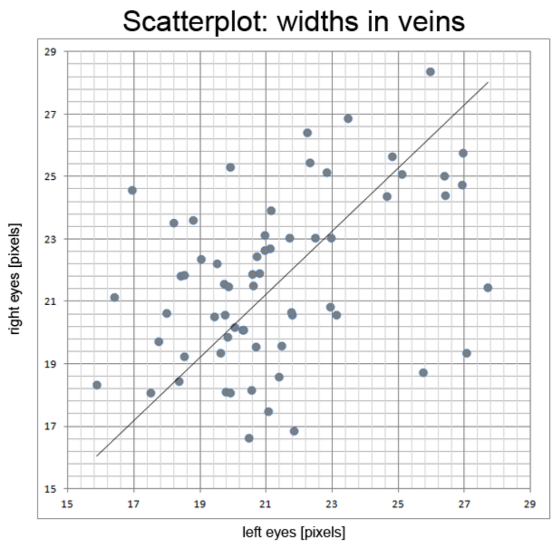
done in these patients (selected so that no one had visual impairemets) in order to evaluate also potential retinal biomarkers for Sarcopenia. These fundus camera images of the 75 subjects were examined and 12 of them, of which two examples can be seen in the Figure 4.1, were excluded from the analysis because too damaged or collected with a wrong scale factor. We decided to focus on vessels' width and AVR, because they were considered the most significant values in the first stage. From the images of the 63 patients not excluded, the widths of the vessels and the AVR were automatically measured with the previous version of the VAMPIRE software (the one based on the binary map, resulting from the segmentation with Soares' algorithm [8],[9]). The measuring points for the calculation of AVR were recorded following the same protocol used by Knudtson et al. in [13]. The results obtained, visible in the scatter plot of Figure 4.2 and in the Bland-Altman plot of Figure 4.3, show a low correlation (Pearson's coefficient) between the two eyes of the same patient especially for vessels' width (0.42 for veins and 0.61 for arteries), which instead according to the literature should be around 0.74 for veins and 0.71 for arteries [15]. For the AVR instead the correlation, equal to 0.47, is very close to that reported in the literature (0.49) [15]. In the first analysis we have assumed three possible causes for this low correlation obtained:

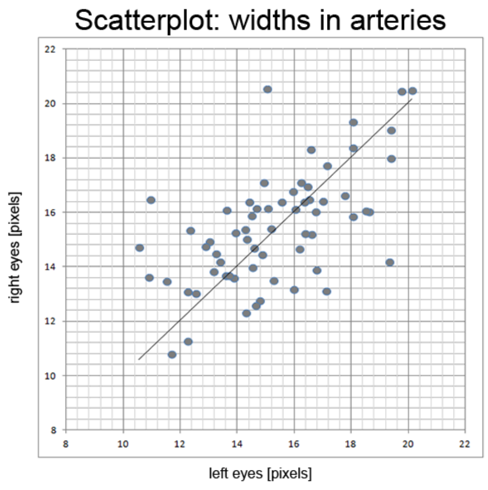
1. because of the bad segmentation, some vascular areas have not been mapped and then the measuring points were chosen "forcedly" in certain areas, but that do not always correspond to the best ones in terms of position
2. the information about the measure was too "poor" and needed what we called "adjustments".

Even if the first hypothesis was very plausible and definitely, at least in part, responsible for the bad results, we realized that a complete description of the data was required to be able to correlate data truly corresponding between them. And this correspondence could be well evaluated only through further key information. Precisely at this stage of the study the VAMPIRE-Annotation Tool has been planned and implemented, and, when finished, used to extract new data from the images of the study about Sarcopenia.

There have been two different techniques of data collection with the VAMPIRE-Annotation Tool:

- one completely manual
- one consisting of a phase of manual annotation of the measuring points (using the VAMPIRE-Annotation Tool and then saving also all the adjustments) and a second phase where the algorithm in [10] was used for automatic measure of the widths in those points.





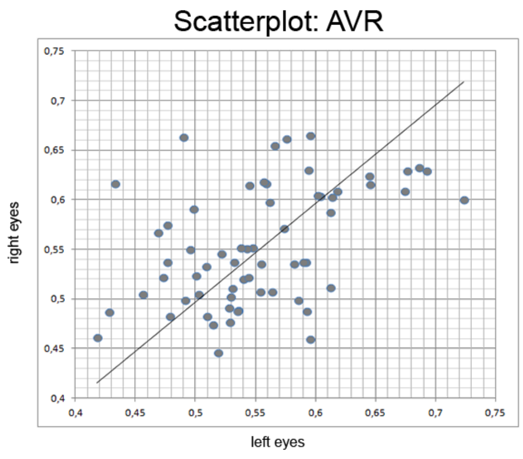


Figure 4.2: From the top: scatter plot of widths' values in veins (left vs right eyes), widths' values in arteries and AVR, elaborated with the automatic algorithms in VAMPIRE

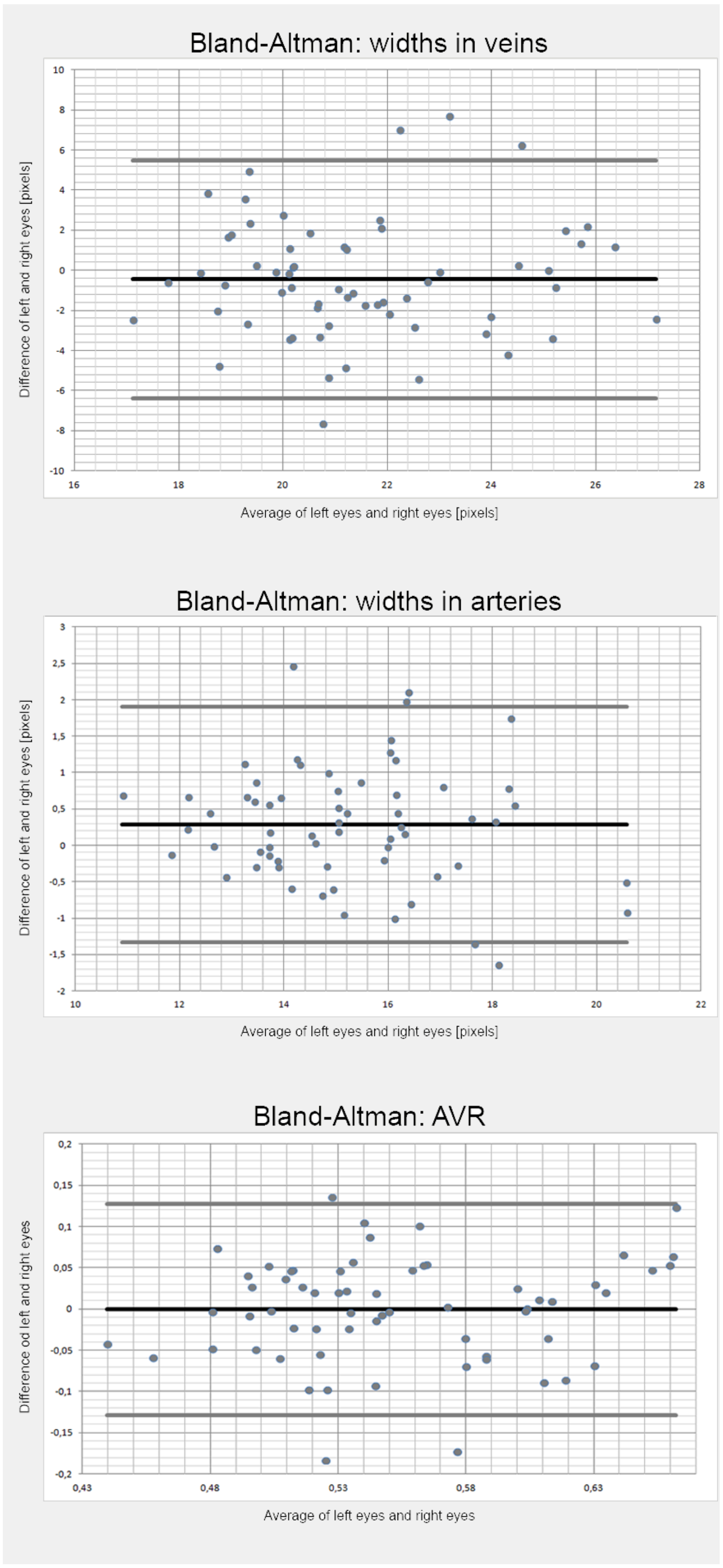
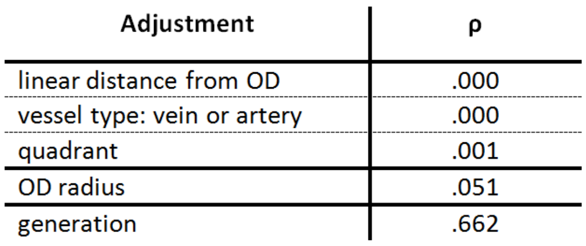


Figure 4.3: From the top: Bland-Altman plot of widths' values in veins, widths' values in arteries and AVR, elaborated with the automatic algorithms in VAMPIRE

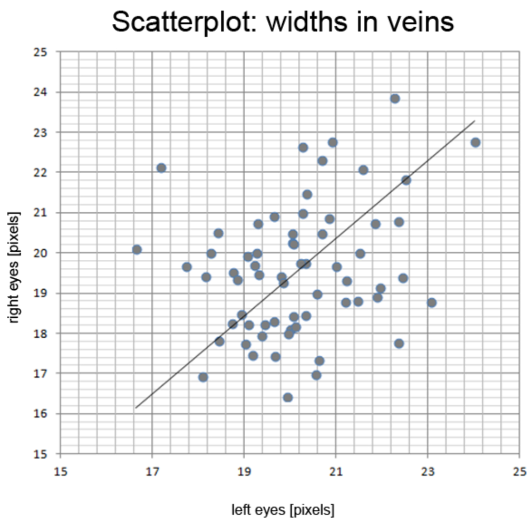
Following will be described only the data obtained with the second method as the results of the two techniques were very similar. First of all, a simple operation of correlation was performed on the data, without the use of the adjustments. Predictably a poor correlation was again obtained for the widths in both veins and arteries and for the AVR (0.26, 0.46 and 0.17 respectively) as can be seen in Figures 4.5 and 4.6. This result was another confirm of what we thought: the main problem to be solved was that of higher characterization of the data. Then operations of "simple" correlation have been performed between the width value in a point and each adjustment (linear distance from OD that is the polar radial coord., vessel type, generation of the vessel, quadrant with respect to OD-fovea system that is related to the polar angular coordinate, OD radius) in order to evaluate which of these are geometrically and clinically significant. The results of these correlations are shown in figure 4.4.

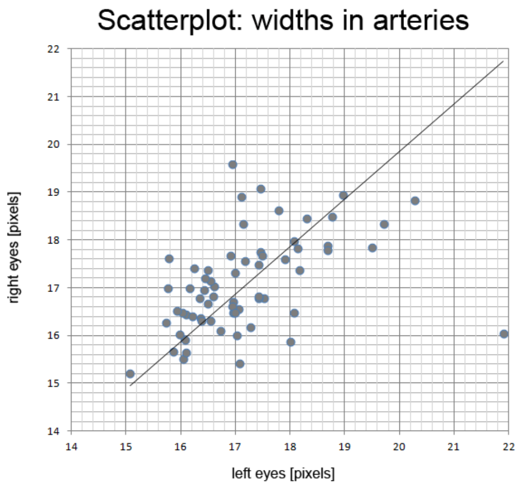
Taking into account the fact that the data set is very small (about 1400 measuring points) and that therefore results must be interpreted in this perspective, the linear distance from OD, the vessel type and the quadrant can be considered highly significant variables because they show a level of significance $\rho \leq 0.001$, the OD radius can be considered almost significant, and the generation instead not significant. The first reflections lead to say that as regards the type of the vessel, it is believed to be key information since many years in the literature, and also the distance from the optic disc is intuitively easy to understand (vessels become smaller with increasing distance from the OD). A positive innovation instead is the quadrant of belonging. This can be explained from the clinical point of view by introducing the concept of drainage and knowing that the macula is the most metabolically active area of the retina: this should correspond, at the vascular level, in vessels branching out from the side of the fovea having greater caliber than those branching off from the opposite side. Surprisingly little significance in this dataset is instead the generation, which apparently has little effect on the magnitude of the width. The phase of study of the correlation between the two eyes using adjustments' information is still in progress and therefore no definitive result can be reported, and also it is considered appropriate, before drawing any conclusion, to apply this idea even at larger dataset. Without any doubt, however, regardless of the results that will be obtained with this technique, the need to find a way of characterizing in detail the measures is certain.



Adjustment	p
linear distance from OD	.000
vessel type: vein or artery	.000
quadrant	.001
OD radius	.051
generation	.662

Figure 4.4: Level of significance of each adjustment





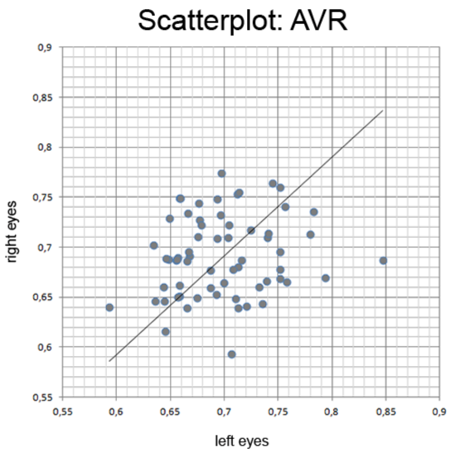


Figure 4.5: From the top: scatter plot of widths' values in veins (left vs right eyes), widths' values in arteries and AVR, elaborated with the VAMPIRE-Annotation Tool

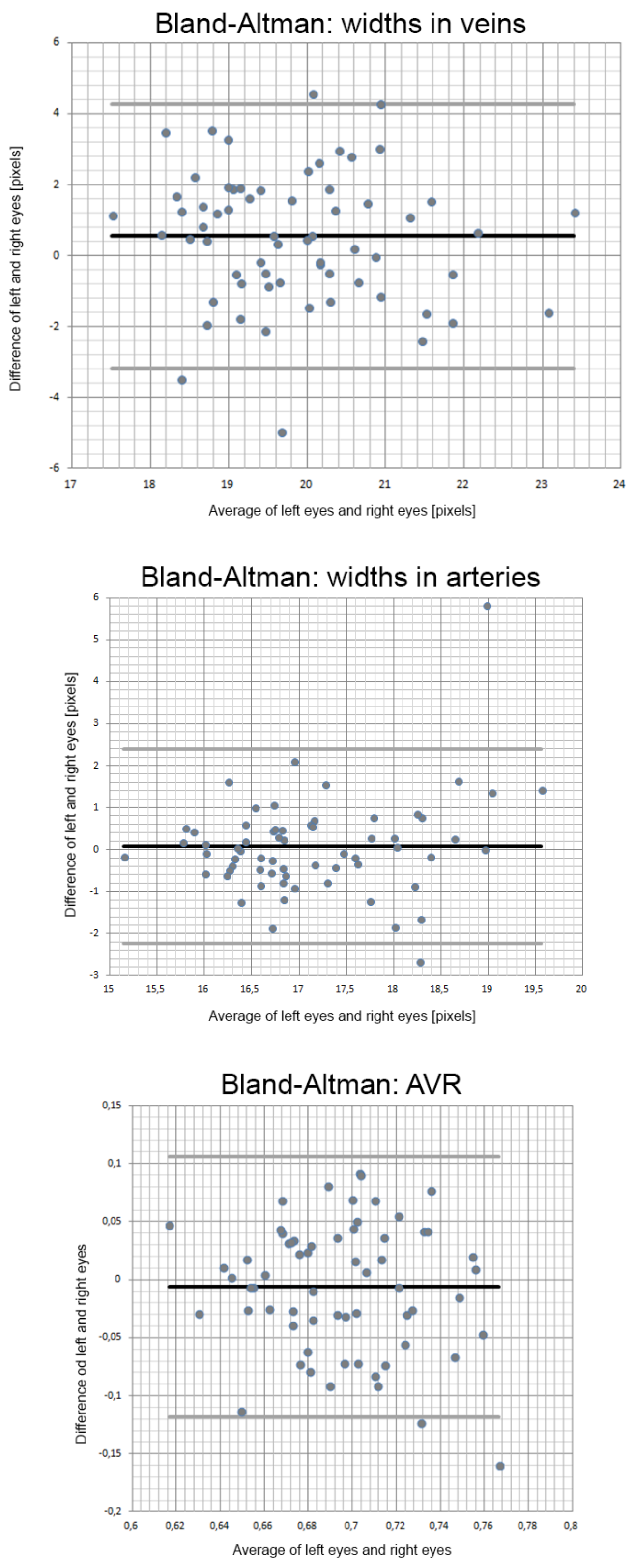


Figure 4.6: From the top: Bland-Altman plot of widths' values in veins, widths' values in arteries and AVR, elaborated with the VAMPIRE-Annotation Tool

Chapter 5

Conclusions

This thesis, after a first overview about the manual tools reported in the literature, proposed the development of a manual annotation tool for the features of the retina in fundus camera images, to be included in the package of the VAMPIRE software as a tool for the production of ground truth for the validation of the automatic algorithms, but also as a support to the software itself when it fails. Subsequently, the protocol used in a study whose purpose was the analysis of retinal biomarkers for Sarcopenia, has been reported.

The VAMPIRE-Annotation Tool has three important characteristics: ease of use thanks to a user-friendly graphical, completeness and flexibility of use, deep characterization of the measures. The first two allow the use of the tool also externally to the team that developed it. In fact, thanks to its immediacy it can be used by clinicians, and thanks to the possibility of both annotate all the features and choose the path most appropriate for the purpose, it can also be used in very different studies. The characterization of the measures instead allows a new view on the data analysis and, thanks also to the ongoing study described in chapter 4, we can select all the information considered more or less significant to a certain measure. In the future we will try to save even more information that we consider important (distance along the vessel between a point and the optic disc, distortion due to both the sphericity of the retina and the image itself, etc.), although this will require more automatic techniques, and we will go on to analyze their level of significance, as done in the context of the study about Sarcopenia.

About the data storage, some decisions have been taken with the intention, at the international community level, to create common, accessible, and representative datasets [21]; all this in a vision of a future library "data-centered" and no longer dependent instead on the task or the measuring tool used. Because there is still no general agreement on the structure and content of these datasets, it was decided to use as a support for the output data simple text files, fast and easy both to write and to read, but above all without any restriction about the operating system used.

We can say that the manual tool, with regard to both the adjustments and the way of saving/reading data, has been a forerunner compared to the main software. The team VAMPIRE in the next months will focus on adapting the automatic software to what are the innovations introduced by the VAMPIRE-Annotation Tool.

Appendix A

Document name: **SOP_VAMPIRE_ANNOTATION_TOOL**

Title: **VAMPIRE-Annotation Tool**

Author: **Ilaria Pieretti**

Version: **1.0**

1.0 Background:

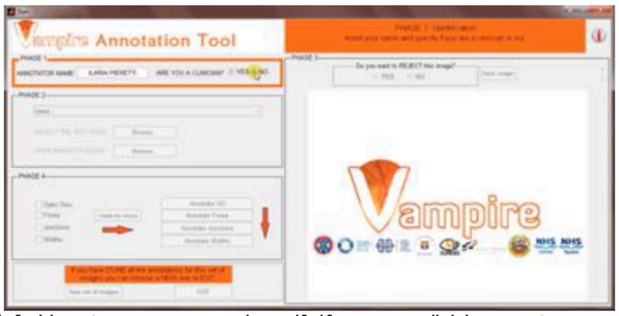
VAMPIRE (Vascular Assessment and Measurement Platform for Images of the Retina) is an international collaboration developing a software suit for efficient semi-automatic analysis of digital retinal images in clinical research. The project is coordinated by two leading groups, Dundee and Edinburgh. The VAMPIRE suite includes interfaces for annotating images, e.g., tracing regions or marking locations in digital retinal images. This is essential to allow clinicians to generate ground truth against which to compare the results of VAMPIRE algorithms. The process of comparing automatic and manual answers is called *validation*.

2.0 Purpose:

The purpose of this document is to describe the procedure that the user must follow when using "VAMPIRE-Annotation Tool ".
This Tool allows the manual annotation of the following retinal features:
-Optic Disc: small blind spot on the surface of the retina where the fibers of the retina leave the eye and become part of the optic nerve. It is the only part of the retina that is insensitive to light.
-Fovea: region of the retina with maximum density of photoreceptors.
-Junction: bifurcation of the blood vessels.
-Width: caliber of the blood vessels.
Attached to this document there is a video with a demo of the "VAMPIRE Annotation Tool: OD, Fovea, Junctions, Widths".

3.0 Procedure:

3.1 User identification

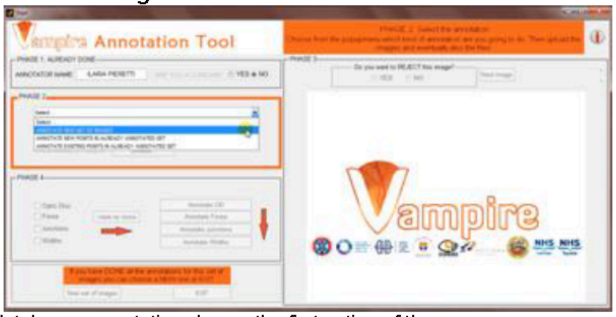


On the main screen top left side enter your name and specify if you are a clinician or not.

3.2 Selection of the annotation

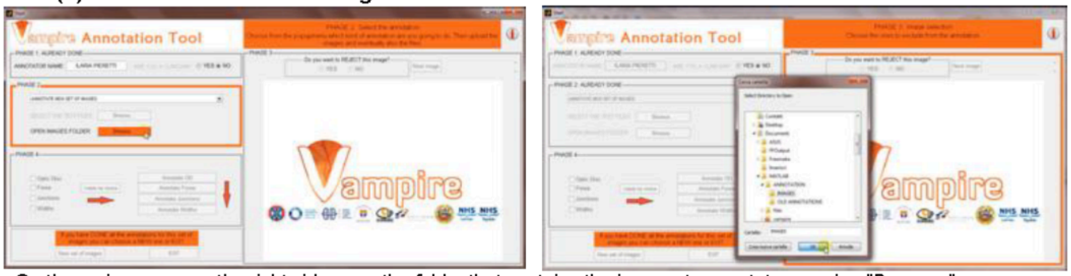
Immediately below by using a pop-up menu you have to indicate which is going to be your annotation path:
(a) annotate a new set of images
(b) annotate new points in already annotated set
(c) annotate existing points in already annotated set

3.3 (a) Annotate new set of images



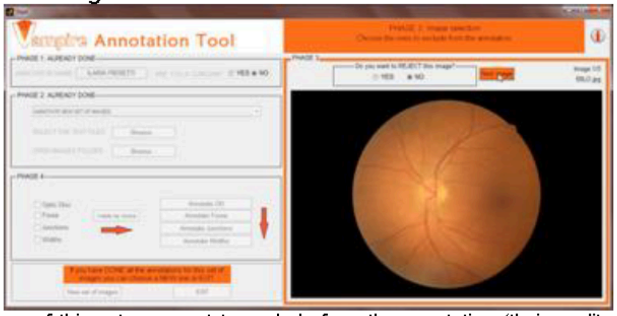
If you want to do a completely new annotation choose the first option of the menu.

3.4 (a) Choice of the set of images



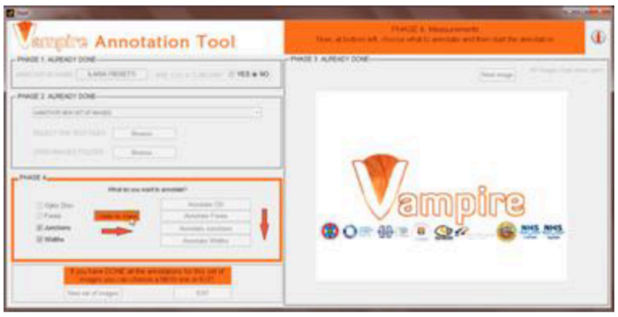
On the main screen on the right side open the folder that contains the images to annotate pressing "Browse..".

3.5 (a) Selection of the images to annotate



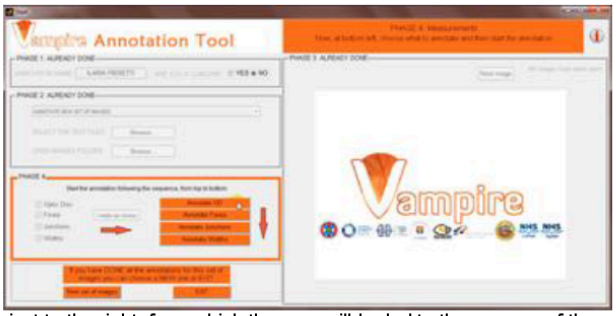
Now choose which images of this set you want to exclude from the annotation (their quality is too low, they are not centered, etc.) ticking YES or NO. To go to the next image press "Next image".

3.6 (a) What do you want to annotate?



When you have completed the selection on the images of this set, you have to choose what to annotate of these images ticking the checkboxes on bottom left side of the screen. The annotation of OD and fovea for choice (a) is compulsory so their checkboxes are already ticked. Once the choice has been made press "I made my choice".

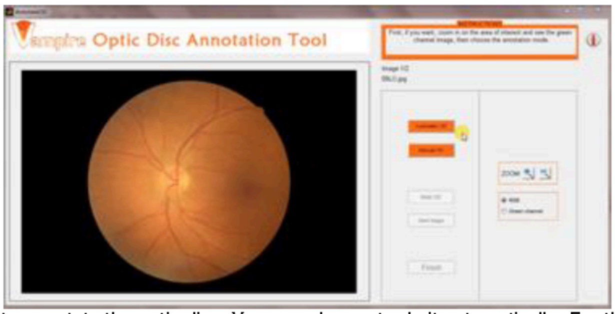
3.7 (a) Go to the sections of annotation



This will activate a menu just to the right, from which the user will be led to the screens of the annotations of OD, Fovea, Junctions and Widths.

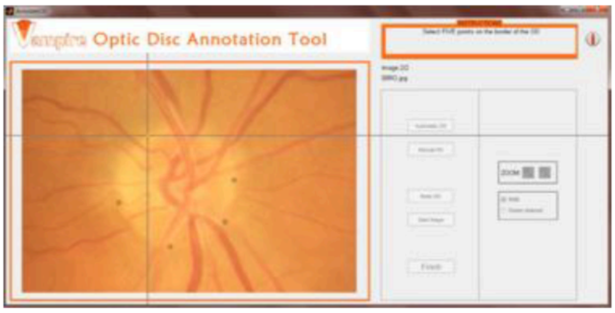
3.8 (a) Optic Disc Annotation Tool

3.8.1 (a)



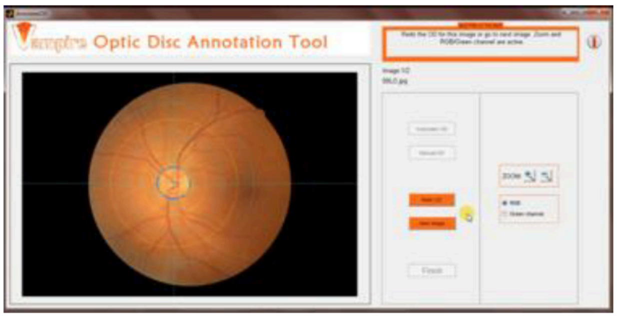
In this section you have to annotate the optic disc. You can choose to do it automatically. For the automatic mode press "Automatic OD" and wait until the calculated OD appears.

3.8.2 (a)



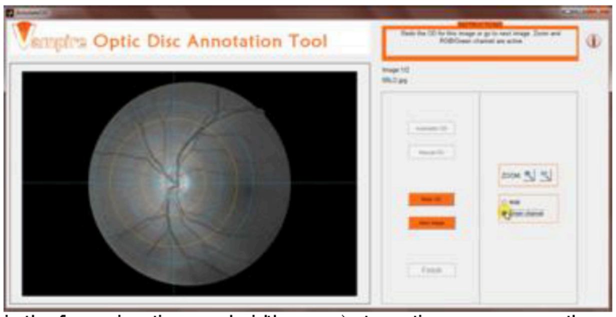
For the manual mode press "Manual OD" and then select five points on the edge of the OD.

3.8.3 (a)



When the ellipse appears, if you do not like it you can redo it (press "Redo OD"), otherwise you can go to the next image pressing "Next image".

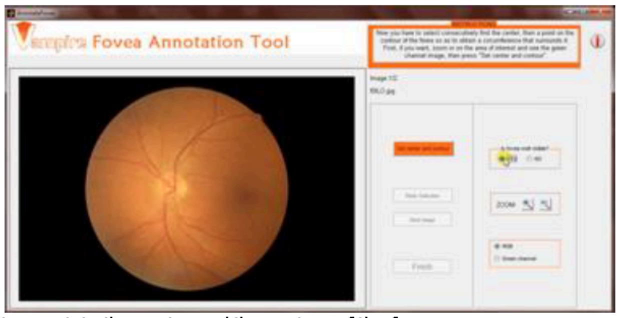
3.8.4 (a)



In this section (and also in the fovea, junctions and widths ones) at any time you can see the green channel image using the appropriate panel.

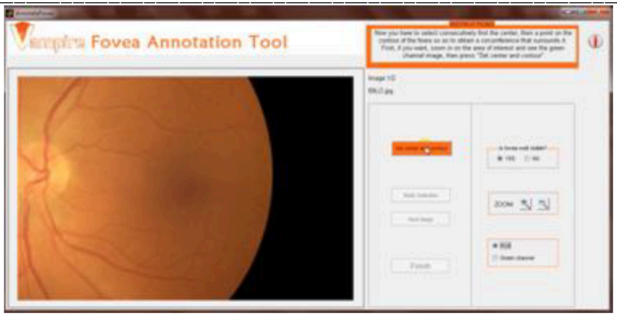
3.9 (a) Fovea Annotation Tool

3.9.1 (a)



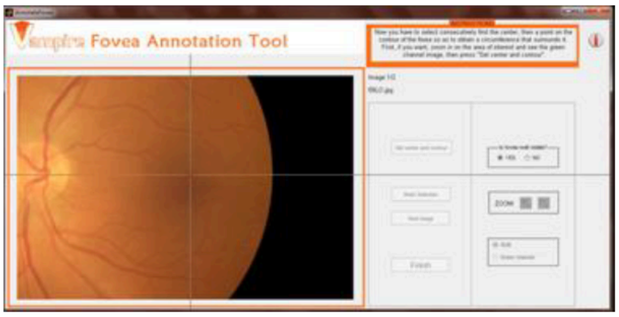
In this section you have to annotate the center and the contour of the fovea.
First of all, you are asked if the fovea in this image is well visible or not. In this way you save the accuracy of the annotation you are going to do.

3.9.2 (a)



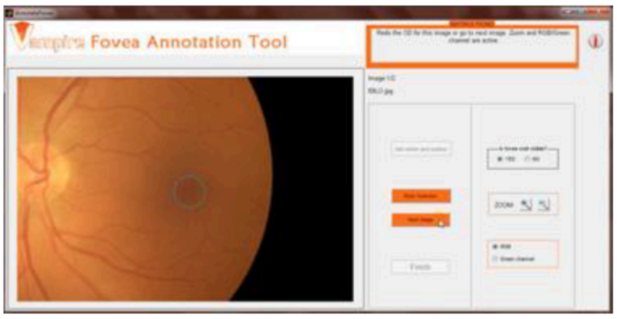
Before the selection of center and contour you have the possibility to zoom on the area of the image where the fovea is and/or see the green channel image.
To select the center and the contour of the fovea press "Set center and contour" …

3.9.3 (a)



… and then click on the image first on the center then on the contour.

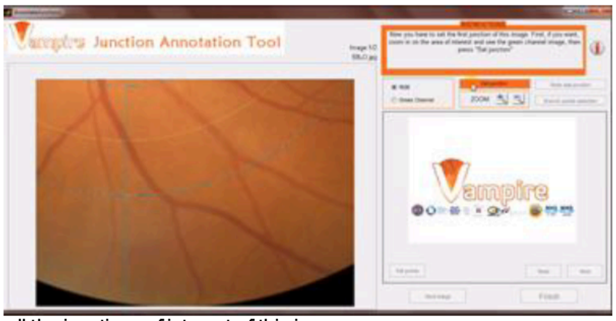
3.9.4 (a)



When the circle appears, if you do not like it can redo it (press "Redo Selection"), otherwise you can go to the next image pressing "Next image".

3.10 (a) Junctions Annotation Tool

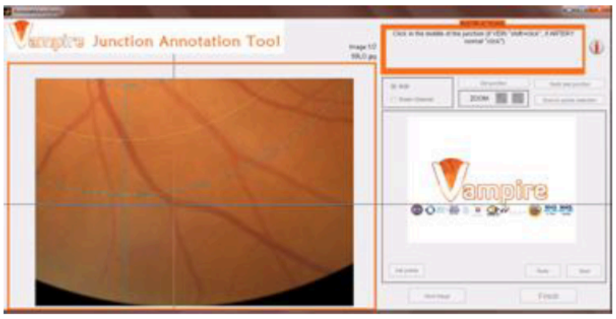
3.10.1 (a)



Now you have to set the first junction of this image. First, if you want, zoom in on the area of interest and see the green channel image, then press "Set junction"

The first step is to select all the junctions of interest of this image.
Before the selection of each point you have the possibility to zoom on the area of the image where the junction is and/or see the green channel image. To select each junction press "Set junction".

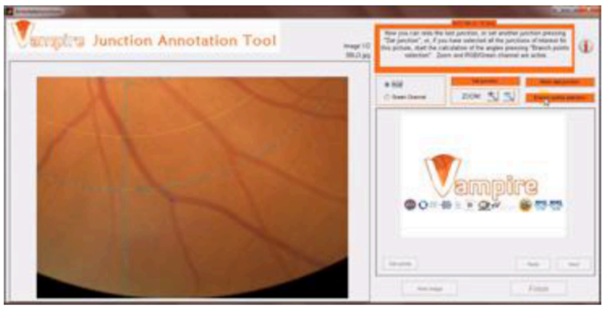
3.10.2 (a)



Click at the middle of the junction (if VEIN "shift+click", if ARTERY normal "click")

Then click on the image in the point you have chosen (to indicate a point belonging to a vein press simultaneously shift key while clicking and the point will appear blue, to indicate an artery do a simple click and the point will appear red).

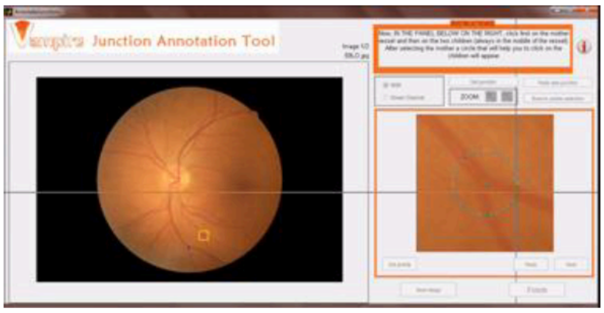
3.10.3 (a)



Now you can undo the last junction, or set another junction pressing "Set junction", or, if you have selected all the junctions of interest for this picture, start the calculation of the angles pressing "Branch points selection". Zoom and RGB/Green channel are active.

If you want to redo the last junction/s press "Redo last junction", otherwise set another junction on this image (press "Set junction") or, if you have selected all the junctions for this picture start the calculation of the angles (press "Branch points selection").

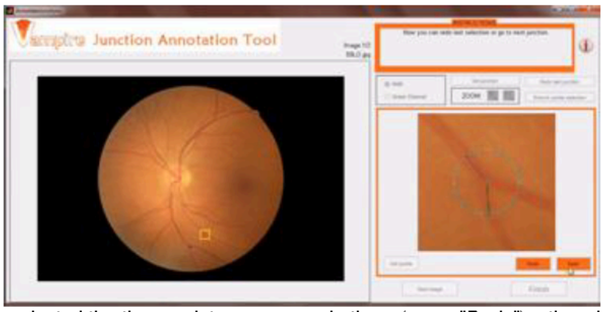
3.10.4 (a)



Now, IN THE PANEL BELOW ON THE RIGHT, click first on the mother vessel and then on the two children (always in the middle of the vessel). After selecting the mother a circle that will help you to click on the children will appear.

To calculate the bifurcation's angles you have to select (within the little screen just appeared on the right side) the three points belonging to the vessels involved in the bifurcation (first click on the mother vessel, then on the two children).
At the same time a yellow square will appear in the big screen on the left to show you the junction you are working on.

3.10.5 (a)

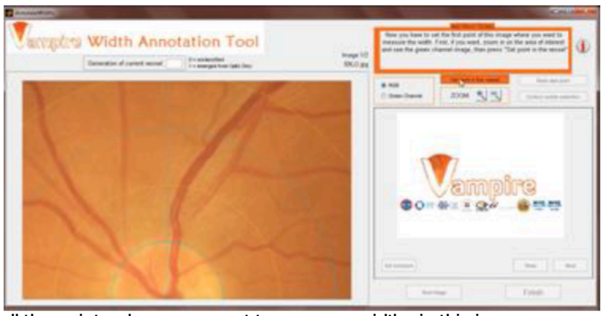


Now you can redo last selection or go to next junction.

If you do not like how you selected the three points you can redo them (press "Redo"), otherwise go to the next junction of this image (press "Next").

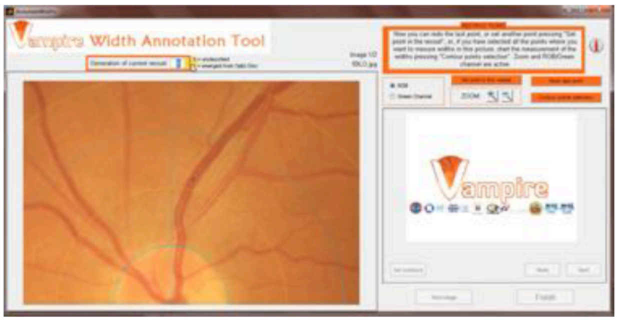
3.11 (a) Widths Annotation Tool

3.11.1 (a)



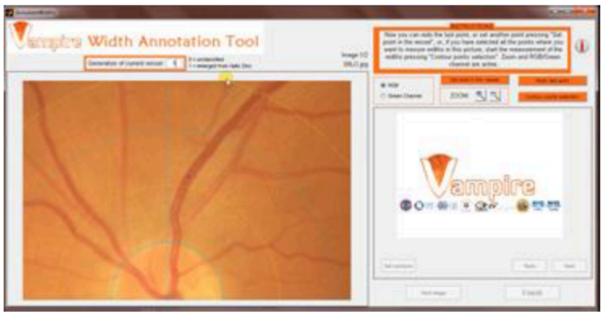
The first step is to select all the points where you want to measure widths in this image.
Before the selection of each point you have the possibility to zoom on the area of the image where the vessel is and/or see the green channel image. To select each point press "Set point in the vessel".

3.11.2 (a)



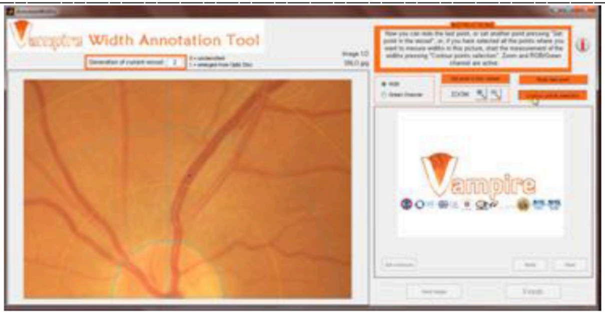
Then click on the image in the point you have chosen (to indicate a point belonging to a vein press simultaneously shift key while clicking and the point will appear blue, to indicate an artery do a simple click and the point will appear red).

3.11.3 (a)



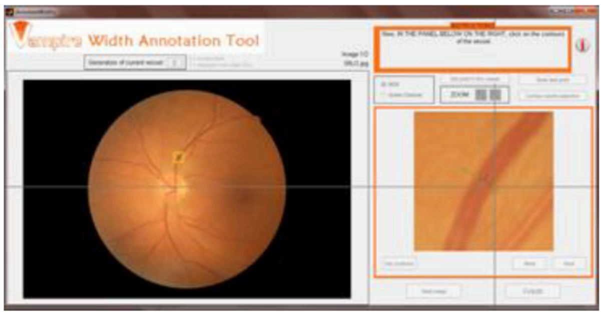
After selecting the point on the image, you can specify the generation of the vessel (where the point is) in the box above the picture.

3.11.4 (a)



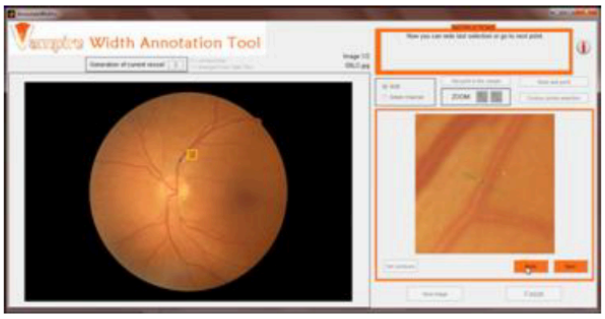
If you want to redo last point/s press "Redo last point", otherwise set another point on this image (press "Set point in the vessel ") or, if you have selected all the points for this picture start the measurement of the widths (press "Contour points selection").

3.11.5 (a)



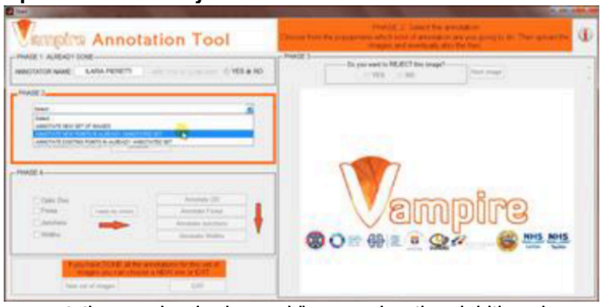
To annotate the width you have to select (within the little screen just appeared on the right side) two points on the edges of the vessel near the point previously selected. A green line will appear through the selected points. It should match with the cross section of the vessel passing through the point within it; if not, it is advisable to reselect the points on the contour to have a better measure of width. At the same time a yellow square will appear in the big screen on the left to show you the point you are working on.

3.11.6 (a)



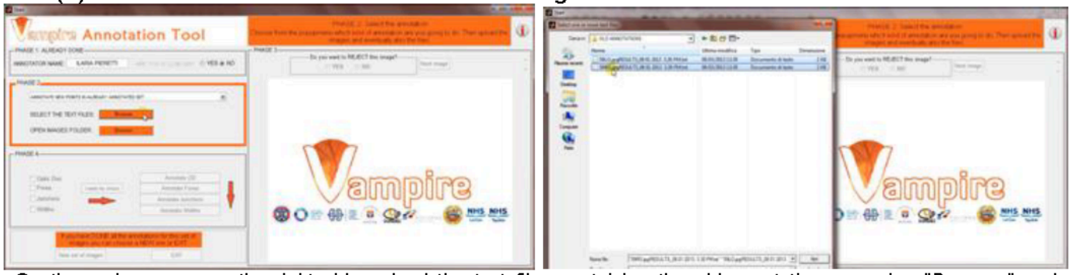
If you do not like how you selected the two points you can redo them (press "Redo"), otherwise go to the next point of this image (press "Next").

3.3 (b) Annotate new points in already annotated set



If you want to continue an annotation previously done adding new junctions/widths, choose the second option of the menu.

3.4 (b) Choice of the text files and the set of images



On the main screen on the right side upload the text files containing the old annotations pressing "Browse.." and selecting the files in their folder.
Look at the 3.4 (a) to see the part concerning the set of images.

3.5 (b) Selection of the images to annotate

The same as for the choice (a) so look at the 3.5 (a).

3.6 (b) What do you want to annotate?

The same as for the choice (a) so look at the 3.6 (a). The only difference is that in this case the annotation of OD and fovea should NOT be taken so their checkboxes are NOT ticked.

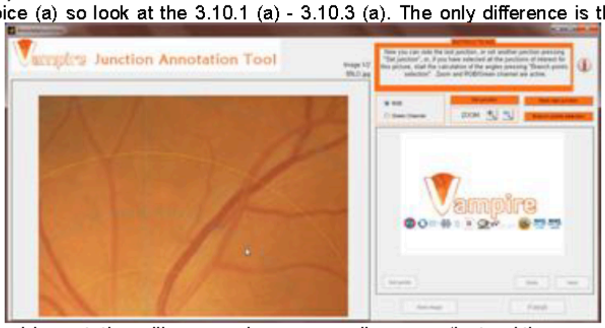
3.7 (b) Go to the sections of annotation

The same as for the choice (a) so look at the 3.7 (a).

3.8 (b) Junctions Annotation Tool

3.8.1 (b) - 3.8.3 (b)

The same as for the choice (a) so look at the 3.10.1 (a) - 3.10.3 (a). The only difference is that on the big screen the



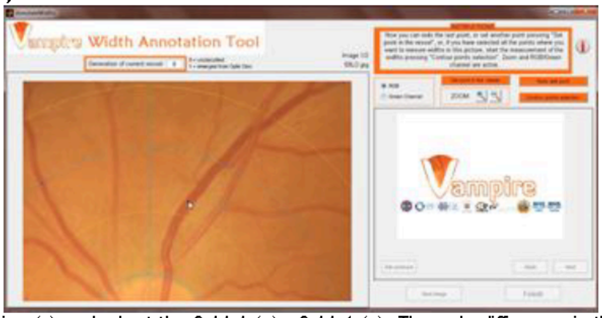
junctions annotated in the old annotation will appear shown as small crosses (instead the new ones always as dots).

3.8.4 (b) - 3.8.5 (b)

The same as for the choice (a) so look at the 3.10.4 (a) - 3.10.5 (a).
Note: only the new junctions are uploaded into the small screen on the right to measure the branching angles.

3.9 (b) Widths Annotation Tool

3.9.1 (b) - 3.9.4 (b)

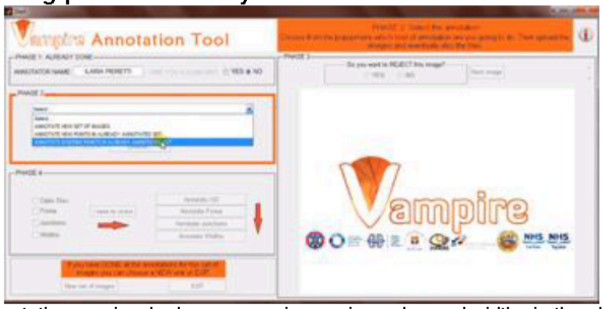


The same as for the choice (a) so look at the 3.11.1 (a) - 3.11.4 (a). The only difference is that on the big screen the points annotated in the old annotation will appear shown as small crosses (instead the new ones always as dots).

3.8.5 (b) - 3.8.6 (b)

The same as for the choice (a) so look at the 3.11.5 (a) - 3.11.6 (a).
Note: only the new points are uploaded into the small screen on the right to measure the width.

3.3 (c) Annotate existing points in already annotated set



If you want to redo an annotation previously done measuring again angles and widths in the old points, choose the third option of the menu.

3.4 (c) Choice of the text files and the set of images

The same as for the choice (b) so look at the 3.4 (b).

3.5 (c) Selection of the images to annotate

The same as for the choice (a) so look at the 3.5 (a).

3.6 (c) What do you want to annotate?

The same as for the choice (a) so look at the 3.6 (a). The only difference is that in this case the annotation of OD and fovea should NOT be taken so their checkboxes are NOT ticked.

3.7 (c) Go to the sections of annotation

The same as for the choice (a) so look at the 3.7 (a).

3.8 (c) Junctions Annotation Tool

Here the junctions on the big images are automatically uploaded (and plotted) from an old annotation so the tool skips directly to the part of the selection of the branching points on the little screen on the right.

3.8.1 (c) - 3.8.2 (c)

The same as for the choice (a) so look at the 3.10.4 (a) - 3.10.5 (a).

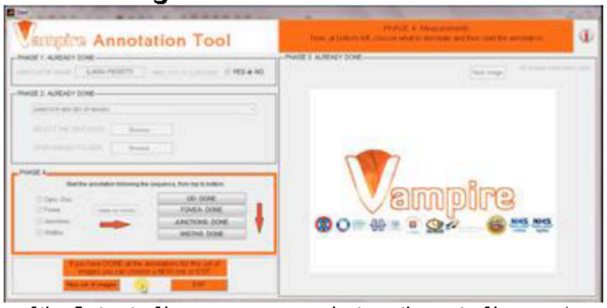
3.9 (c) Widths Annotation Tool

Here the points on the big images are automatically uploaded (and plotted) from an old annotation so the tool skips directly to the part of the selection of the points on the vessels' contour on the little screen on the right.

3.9.1 (c) - 3.9.2 (c)

The same as for the choice (a) so look at the 3.11.5 (a) - 3.11.6 (a).

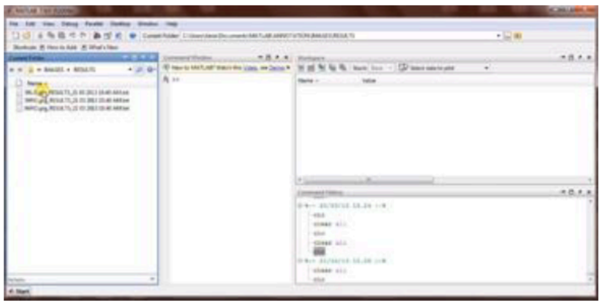
3.12 Chose another set of images or EXIT the Tool



After the whole annotation of the first set of images, you can select another set of images (press "New set of images") or you can exit the tool (press "EXIT").

<u>4.0 The output text file:</u>

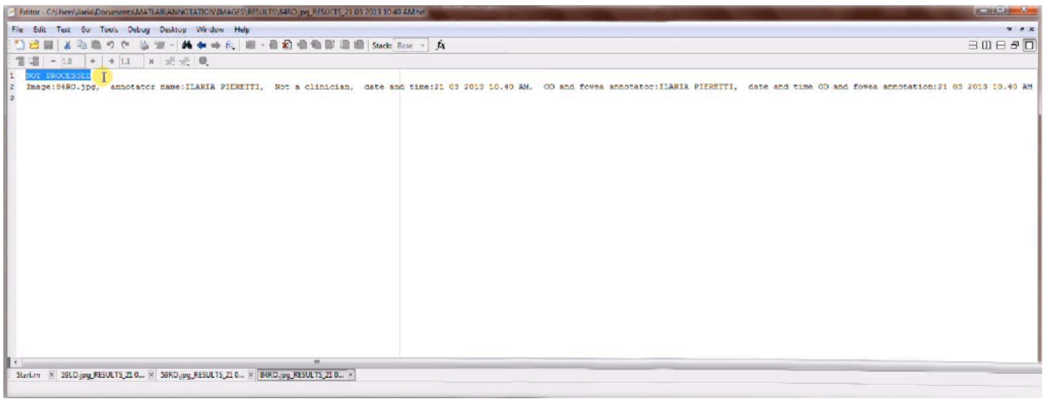
The program returns in output one text file for each image of the set. This files are automatically saved in a folder called "RESULTS" created automatically in the folder of the images.



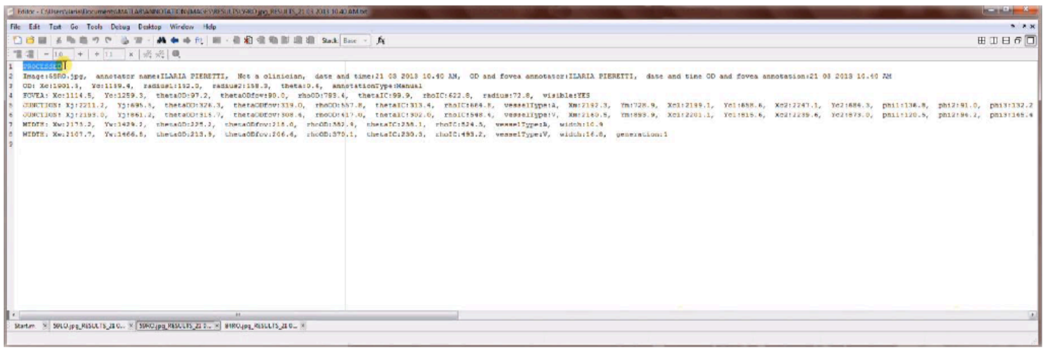
The name of the file is always like: *ImageName*_RESULTS_*DateAndTime*.txt
The file contains information about the user, the image and the annotations taken.
In the first line it is always shown if the image has been processed ('PROCESSED') or if it has been rejected ('NOT PROCESSED') by the user.
In the second line you can always find the name of the image, information about the user (name and belonging to the clinicians), date and time of the annotation, information about the annotation of OD and fovea (annotator and date and time that can coincide with the first -choice (a)- or not -choice (b) and (c)).
If the image has been processed by the user, from third line onwards you can find:
-third line: always information about the OD. Cartesian coordinates of its center, the two radii and the coefficient theta of the ellipse that fits the OD.
-fourth line: always information about the fovea. Cartesian and polar coordinates of its center, the radius.
From fifth line onwards you can find information about junctions and widths (depending on the case the number of

____

junctions and widths could range from zero to many):
- junctions: Cartesian and polar coordinates of the point in the center of the junction, the type of the vessel (vein or artery) where the junction is, Cartesian coordinates of the three points belonging to the three vessels involved in the junction, the three bifurcation angles.
-widths: Cartesian and polar coordinates of the point within the vessel where you measured the width, the type of the vessel (vein or artery) where the point is, the width and the generation of this vessel.



```
NOT PROCESSED
Image:84RO.jpg,  annotator name:ILARIA PIERETTI,  Not a clinician,  date and time:21 03 2013 10.40 AM,  OD and fovea annotator:ILARIA PIERETTI,  date and time OD and fovea annotation:21 03 2013 10.40 AM
```



```
PROCESSED
Image:59RO.jpg,  annotator name:ILARIA PIERETTI,  Not a clinician,  date and time:21 03 2013 10.40 AM,  OD and fovea annotator:ILARIA PIERETTI,  date and time OD and fovea annotation:21 03 2013 10.40 AM
OD: Xc:1901.1,  Yc:1159.4,  radius1:152.0,  radius2:158.3,  theta:0.6,  annotationType:Manual
FOVEA: Xc:1114.5,  Yc:1259.3,  thetaOD:97.2,  thetaODfov:90.0,  rhoOD:793.4,  thetaIC:99.9,  rhoIC:622.8,  radius:72.8,  visible:YES
JUNCTION: Xj:2211.2,  Yj:695.5,  thetaOD:326.3,  thetaODfov:319.0,  rhoOD:557.8,  thetaIC:313.4,  rhoIC:684.8,  vesselType:A,  Xm:2192.3,  Ym:728.9,  Xc1:2199.1,  Yc1:858.6,  Xc2:2247.1,  Yc2:684.3,  phi1:138.8,  phi2:91.0,  phi3:132.2
JUNCTION: Xj:2193.0,  Yj:861.2,  thetaOD:315.7,  thetaODfov:308.4,  rhoOD:417.0,  thetaIC:302.0,  rhoIC:548.4,  vesselType:V,  Xm:2160.5,  Ym:899.9,  Xc1:2201.1,  Yc1:815.6,  Xc2:2239.6,  Yc2:873.0,  phi1:120.5,  phi2:96.2,  phi3:165.4
WIDTH: Xw:2173.2,  Yw:1429.2,  thetaOD:225.2,  thetaODfov:218.0,  rhoOD:382.4,  thetaIC:238.1,  rhoIC:524.5,  vesselType:A,  width:10.9
WIDTH: Xw:2107.7,  Yw:1466.8,  thetaOD:213.9,  thetaODfov:206.6,  rhoOD:375.1,  thetaIC:230.3,  rhoIC:493.2,  vesselType:V,  width:16.8,  generation:1
```

Appendix B

Code of paragraph 3.3.1

```
% --- Executes during object creation, after setting all properties.
function Name_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
  defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function Name_Callback(hObject, eventdata, handles)

handles.Annotator.Name = get(hObject,'String');
% update handles
guidata(hObject, handles);
```

```
% --- Executes when selected object is changed in uipanelClinician.
function uipanelClinician_SelectionChangeFcn(hObject, eventdata, handles)

if hObject==handles.YesClinician
    % if user check this button save 'Clinician'
    handles.Annotator.Clinician='Clinician';
elseif hObject==handles.NoClinician
    % if user check this button save 'Not a clinician'
    handles.Annotator.Clinician='Not a clinician';
end
% if the user information is not fully inputted, exit
if isempty(handles.Annotator.Name)||isempty(handles.Annotator.Clinician)
    return
end
```

```
% update handles
guidata(hObject, handles);
```

```
% --- Executes during object creation, after setting all properties.
function popupmenu_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
  defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on selection change in popupmenu.
function popupmenu_Callback(hObject, eventdata, handles)

contents=cellstr(get(hObject,'String'));
Aselection=contents{get(hObject,'Value')};
if strcmp(Aselection,'ANNOTATE NEW SET OF IMAGES')
    set(handles.openFileButton,'Enable','off','BackgroundColor
      ',[0.941,0.941,0.941]);
    set(handles.openFolderButton,'Enable','on','BackgroundColor
      ',[1,0.345,0.137]);
elseif strcmp(Aselection,'ANNOTATE NEW POINTS IN ALREADY ANNOTATED SET')
    set(handles.openFolderButton,'Enable','on','BackgroundColor
      ',[1,0.345,0.137]);
    set(handles.openFileButton,'Enable','on','BackgroundColor
      ',[1,0.345,0.137]);
elseif strcmp(Aselection,'ANNOTATE EXISTING POINTS IN ALREADY ANNOTATED
  SET')
    set(handles.openFolderButton,'Enable','on','BackgroundColor
      ',[1,0.345,0.137]);
    set(handles.openFileButton,'Enable','on','BackgroundColor
      ',[1,0.345,0.137]);
elseif strcmp(Aselection,'Select ...')
    set(handles.openFolderButton,'Enable','off','BackgroundColor
      ',[0.941,0.941,0.941]);
    set(handles.openFileButton,'Enable','off','BackgroundColor
      ',[0.941,0.941,0.941]);
end
% update handles
guidata(hObject, handles);
```

```
% --- Executes on button press in openFolderButton.
function openFolderButton_Callback(hObject, eventdata, handles)

% Get folder:
directory=uigetdir(handles.files.directory);
% If the user pressed cancelled, then exit:
if directory==0
    return
end
% clear previous results:
handles.files=[];
% upload images and save handles:
handles=openFilesFromDirectory(directory, handles);
handles.imageID=1;

% create the folder RESULTS and the text file for the current image
resFold='RESULTS';
mkdir(directory, resFold);
fileID=fopen(strcat(directory,'\',resFold,'\',cell2mat(handles.files.name(
  handles.imageID)),'_','RESULTS','_', dateNtime,'.','txt'),'a+');
fclose(fileID);

% open image
handles.image=imread([handles.files.directory filesep cell2mat(handles.
  files.name(handles.imageID))]);
% plot image
axes(handles.mainFigure)
imshow(handles.image); hold on;
% display the number of images and file's name
```

```
c=strcat('Image:',num2str(handles.imageID),'/',num2str(handles.files.
  nFiles));
set(handles.display2,'String',c);
c1=handles.files.name(handles.imageID);
set(handles.text25,'String',c1);

setButtonStateFor_NextOFFProcessON(handles);
% updating handles
guidata(hObject, handles);
```

```
% --- Executes on button press in openFileButton.
function openFileButton_Callback(hObject, eventdata, handles)

[FileNameTemp,PathName,FilterIndex]=uigetfile('*.txt','Select one or more
  text files','Multiselect','on');
% add this directory to path and extract file's name
idx=strfind(PathName,'\');
idx2=idx(end-1);
my_path=PathName(1:idx2);
my_dir=PathName(idx2+1:end);
my_dir=my_dir(1:end-1);
addpath my_dir;
if ischar(FileNameTemp)
    FileNameTemp=strread(FileNameTemp,'%s','whitespace','''{}')';
end
FileLIST={FileName{1,:}, FileNameTemp{1,:}};
FileName=FileLIST;
% update handles
guidata(hObject, handles);
```

```
% --- Executes on button press in YesProcess.
function YesProcess_Callback(hObject, eventdata, handles)

% label the image as processed
handles.Processed(handles.imageID).Processed='PROCESSED';
```

```
%processed is a vector that contains 0 and 1, one for each image. 1 means
  the image has to be processed, 0 not to be.
processed(handles.imageID)=1;

setButtonStateFor_NextONProcessOFF(handles);
% update handles
guidata(hObject, handles);
```

```
% --- Executes on button press in NoProcess.
function NoProcess_Callback(hObject, eventdata, handles)
% label the image as not processed
handles.Processed(handles.imageID).Processed='NOT PROCESSED';

%processed is a vector that contains 0 and 1, one for each image. 1 means
  the image has to be processed, 0 not to be.
processed(handles.imageID)=0;

setButtonStateFor_NextONProcessOFF(handles);
% update handles
guidata(hObject, handles);
```

```
% --- Executes on button press in NextImage.
function NextImage_Callback(hObject, eventdata, handles)

if strcmp(Aselection,'ANNOTATE NEW SET OF IMAGES')
    firstAnnotator=handles.Annotator.Name;
    firstDate=dateNtime;
    %save first data in the text file (user informations, name of image,
      etc)
    resFold='RESULTS';
    fileID=fopen(strcat(directory,'\',resFold,'\',cell2mat(handles.files
      .name(handles.imageID)), '_', 'RESULTS', '_', dateNtime, '.', 'txt')
      , 'a+');
    write_GeneralInformations(fileID, handles.Processed(handles.imageID).
      Processed, cell2mat(handles.files.name(handles.imageID)), num2str(
      handles.Annotator.Name), handles.Annotator.Clinician, num2str(
      handles.dateNtime), firstAnnotator, firstDate);
                 fclose(fileID);
elseif strcmp(Aselection,'ANNOTATE EXISTING POINTS IN ALREADY ANNOTATED
  SET') || strcmp(Aselection,'ANNOTATE NEW POINTS IN ALREADY ANNOTATED SET
  ')
    %read file file
    for rf=1:size(FileLIST,2)
        nametemp=FileLIST{1,rf};
        if strcmp(handles.files.name{handles.imageID,1},nametemp(1,1:size(
          handles.files.name{handles.imageID,1},2)))
            fileName=nametemp;
        end
    end
    cd(PathName)
    %save in structure:
    res=readTextFile(fileName);
    firstAnnotator=char(res{2,4});
    firstDate=char(res{2,7});
    data=[str2double(res{3,3}),str2double(res{3,5}),str2double(res{3,7}),
      str2double(res{3,9}),str2double(res{3,11})];
    center=[str2double(res{4,3}),str2double(res{4,5}),str2double(res{4,7})
      ,str2double(res{4,9}),str2double(res{4,11}),str2double(res{4,13}),
      str2double(res{4,15})];
    radius=str2double(res{4,17});
    %save first data in the text file (user informations, name of image,
      etc)and also OD and fovea information in the text file
    resFold='RESULTS';
    fileID=fopen(strcat(directory, '\', resFold, '\', cell2mat(handles.
      files.name(handles.imageID)), '_', 'RESULTS', '_', dateNtime, '.', '
      txt'),'a+');
                 write_GeneralInformations(fileID, handles.Processed(
                   handles.imageID).Processed, cell2mat(handles.files.name(
                   handles.imageID)), num2str(handles.Annotator.Name),
                   handles.Annotator.Clinician, num2str(handles.dateNtime),
                     firstAnnotator ,  firstDate ) ;
                write_OD2 ( fileID , data ) ;
                write_Fovea2 ( fileID , center , radius ) ;
                fclose ( fileID ) ;
end

%reset the YES/NOprocess buttons
set ( handles . YesProcess , 'Value' ,0) ;
set ( handles . NoProcess , 'Value' ,0) ;

% read the next image
handles . imageID=handles . imageID+1;

if handles . imageID > handles . files . nFiles %no more images
    c='All images have been seen ';
    set ( handles . display2 , 'String ' , c , 'Enable ' , 'off ') ;
    c1=' ';
    set ( handles . text25 , 'String ' , c1 ) ;
    image2process=size ( find ( processed==1) ,2) ;
    if image2process==0 % no images to process
        % display worning if the user didn't select any image
        w='WARNING: you have rejected all the images in this folder !!! ';
        set ( handles . warning , 'Visible ' , 'on ' , 'String ' ,w) ;
        s2='If you have DONE all the annotations for this set of images
          you can choose a NEW one or EXIT ';
        set ( handles . textNewSetOrExit , 'Visible ' , 'on ' , 'String ' , s2 ) ;
         setButtonStateFor_FinishGoToMainMenu ( handles ) ;
        % highlight exit buttons
        set ( handles . ExitButton , 'Enable ' , 'on ' , 'BackgroundColor
          ' ,[1 ,0.345 ,0.137]) ;
        set ( handles . NewSetImagesButton , 'Enable ' , 'on ' , 'BackgroundColor
          ' ,[1 ,0.345 ,0.137]) ;
        set ( handles . NextImage , 'Enable ' , 'off ' , 'BackgroundColor
          ' ,[0.941 ,0.941 ,0.941]) ;
        % delate retinal images and replot vampire logo
```

```
            imageLogo=imread('eyeLogo.png');
            axes(handles.mainFigure)
            cla
            axis auto
            imshow(imageLogo);
            return;
        end
        setButtonStateFor_ImageFinishedWhatAnnotate(handles);
        return;
end

%images not finished:
c=strcat('Image:',num2str(handles.imageID),'/',num2str(handles.files.
   nFiles));
set(handles.display2,'String',c);
c1=handles.files.name(handles.imageID);
set(handles.text25,'String',c1);
% open image
handles.image = imread([ handles.files.directory filesep cell2mat(handles.
   files.name(handles.imageID)) ]);

% updates handles
guidata(hObject, handles);

% plot image
axes(handles.mainFigure)
imshow(handles.image);hold on;

setButtonStateFor_NextOFFProcessON(handles);
```

```
% --- Executes on button press in ChooseAnnotationsButton.
function ChooseAnnotationsButton_Callback(hObject, eventdata, handles)

setButtonStateFor_FinishGoToMainMenu(handles);
```

```
% save all the exit of the checkbox in one matrix so that in MainMenu it's
   simpler set buttons on/off:
global annotateButtonsMainMenu;
annotateButtonsMainMenu=[];
annotateButtonsMainMenu(1)=get(handles.ODButton,‘Value’);
annotateButtonsMainMenu(2)=get(handles.FoveaButton,‘Value’);
annotateButtonsMainMenu(3)=get(handles.WidthsButton,‘Value‘);
annotateButtonsMainMenu(4)=get(handles.JunctionsButton,‘Value’);

if get(handles.ODButton, ‘Value’)==get(handles.FoveaButton, ‘Value’)==get(
  handles.WidthsButton, ‘Value’)==get(handles.JunctionsButton, ‘Value’)==0
     set(handles.ExitButton,‘Enable’, ‘on’, ‘BackgroundColor’,
        [1,0.345,0.137]);
     set(handles.NewSetImagesButton, ‘Enable’, ‘on’, ‘BackgroundColor’,
        [1,0.345,0.137]);
end
% PUSH−BUTTONS MENU:
MenuAnnotationSelection(hObject, eventdata, handles);
```

```
function MenuAnnotationSelection(hObject, eventdata, handles)

if annotateButtonsMainMenu(1)==1 && annotateButtonsMainMenu(2)==1 &&
  annotateButtonsMainMenu(3)==0 && annotateButtonsMainMenu(4)==0
      setButtonStateFor_OD_F(handles)
elseif annotateButtonsMainMenu(1)==1 && annotateButtonsMainMenu(2)==1 &&
  annotateButtonsMainMenu(3)==1 && annotateButtonsMainMenu(4)==0
     setButtonStateFor_OD_F_W(handles);
elseif  annotateButtonsMainMenu(1)==1 && annotateButtonsMainMenu(2)==1 &&
  annotateButtonsMainMenu(3)==0 && annotateButtonsMainMenu(4)==1
      setButtonStateFor_OD_F_J(handles);
elseif annotateButtonsMainMenu(1)==1 && annotateButtonsMainMenu(2)==1 &&
  annotateButtonsMainMenu(3)==1 && annotateButtonsMainMenu(4)==1
     setButtonStateFor_ALL(handles);
elseif annotateButtonsMainMenu(1)==0 && annotateButtonsMainMenu(2)==0 &&
  annotateButtonsMainMenu(3)==1 && annotateButtonsMainMenu(4)==0
```

```
    setButtonStateFor_W(handles);
elseif annotateButtonsMainMenu(1)==0 && annotateButtonsMainMenu(2)==0 &&
  annotateButtonsMainMenu(3)==0 && annotateButtonsMainMenu(4)==1
    setButtonStateFor_J(handles);
elseif annotateButtonsMainMenu(1)==0 && annotateButtonsMainMenu(2)==0 &&
  annotateButtonsMainMenu(3)==1 && annotateButtonsMainMenu(4)==1
    setButtonStateFor_W_J(handles);
end
```

Code of paragraph 3.3.2

```
% --- Outputs from this function are returned to the command line.
function varargout = AnnotateOD_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;
%save directory
handles.files.directory=directory;
% number of images
handles.files.nFiles = 0;
handles.imageID=1;
% load VAMPIRE logo
littleLogo=imread('littleLogo.png');
axes(handles.logo);
imshow(littleLogo);
%load zoom button's logo
zoomin=imread('zoomin.png');
set(handles.zoomIN,'CData',zoomin);
zoomout=imread('zoomout.png');
set(handles.zoomOUT,'CData',zoomout);
%load info button's logo
info=imread('infoLogo.png');
set(handles.info_button,'CData',info);
% clear previous results if any
handles.files=[];
```

```
% upload images and save handles
handles=openFilesFromDirectory(directory, handles);
% number of these images to be processed
handles.image2process=0;
handles.files.nFiles2process=size(find(processed==1),2); %processed loaded
   from START code
% read first image
while handles.imageID<=handles.files.nFiles
    if processed(handles.imageID)==0 %imageID not to be annotated so don't
       open it
        handles.imageID=handles.imageID+1;
    else % imageID has to be annotated so open it
        handles.image2process=handles.image2process+1;
        % display the number of images
        c=strcat('Image:',num2str(handles.image2process),'/',num2str(
          handles.files.nFiles2process));
        set(handles.display,'String',c);
        c1=handles.files.name(handles.imageID);
        set(handles.display1,'String',c1);
        handles.image=imread([handles.files.directory filesep cell2mat(
          handles.files.name(handles.imageID))]);
        % plot image
        axes(handles.mainFigure)
        imshow(handles.image);hold on;

        setButtonStateFor_ChoseAnnotation(handles);
        % updates handles
        guidata(hObject, handles);
        break
    end
end
% if the images are finished
    if handles.imageID > handles.files.nFiles
        c='All images have been annotated';
        set(handles.display, 'String', c);
        c1=' ';
        set(handles.display1, 'String', c1);
        s='Press "Finish" to go to Start Menu';
        set(handles.suggestions, 'String', s);
        % remove retinal image from screen and replot vampire logo
        imageLogo=imread('eyeLogo.png');
        axes(handles.mainFigure)
        cla
        axis auto
        imshow(imageLogo);
        setButtonStateFor_Finish(handles);
        return;
    end
% updates handles
guidata( hObject, handles )
```

```
% --- Executes on button press in autolocateButton.
function autolocateButton_Callback(hObject, eventdata, handles)

c='Wait, locating OD ...';
set(handles.suggestions,'String',c);
setButtonStateFor_InProcess(handles);

% Mark selected image as non-located OD
handles.results(handles.image2process).ODborders=0;
handles.results(handles.image2process).ODellipse=0;
handles.results(handles.image2process).ODcenter=0;
handles.results(handles.image2process).Coefficient=0;
handles.results(handles.image2process).Residual=0;
handles.results(handles.image2process).ODradius=0;
% run the VAMPIRE's function automaticODlocation
handles=automaticODlocation(handles);
% show located OD (draw the circles)
if handles.image2process>0
    if handles.results(handles.image2process).ODradius>0
```

```
        [cx1,cy1,cx2,cy2,cx3,cy3]=drawCircles(handles, handles.
          image2process);
    end
end
% label the OD of this image as "Automatic"
handles.results(handles.image2process).AnnotType='Automatic';

setButtonStateFor_ClearOrNext(handles);

c='Redo the OD for this image or go to next image. Zoom and RGB/Green
  channel are active.';
set(handles.suggestions,'String',c);
% update handles:
guidata(hObject, handles);
```

```
% --- Executes on button press in locateButton.
function locateButton_Callback(hObject, eventdata, handles)

c='Select FIVE points on the border of the OD';
set(handles.suggestions,'String',c);
set(handles.uipanelFig,'BorderWidth',2,'ShadowColor',[1,0.345,0.137],'
  HighlightColor',[1,0.345,0.137]);

setButtonStateFor_InProcess(handles);

% Mark selected image as non-located OD
handles.results(handles.image2process).ODborders=0;
handles.results(handles.image2process).ODellipse=0;
handles.results(handles.image2process).ODcenter=0;
handles.results(handles.image2process).Coefficient=0;
handles.results(handles.image2process).Residual=0;
handles.results(handles.image2process).ODradius=0;
% Obtain borders coordinate from the user
for i=1:5 % five points
    OD_borders(i,:) = ginput(1);
    plot(OD_borders(i,1),OD_borders(i,2),'ko');
end
% Fit the ellipse
a=fitellip(OD_borders(:,1),OD_borders(:,2));
% Calaculate the residual of ellipse
ResSum=0;
for i=1:length(OD_borders)
    x=OD_borders(i,1);
    y=OD_borders(i,2);
    residual=a(1)*x*x + a(2)*x*y + a(3)*y*y + a(4)*x + a(5)*y + a(6);
    ResSum=ResSum+residual;
end
% e contains ellipseXcenter,ellipseYcenter, Radius1, Radius2, theta
e=fitellipse(OD_borders(:,1),OD_borders(:,2));
%Save results
handles.results(handles.image2process).ODborders=OD_borders;
handles.results(handles.image2process).ODellipse=e;
handles.results(handles.image2process).ODcenter=handles.results(handles.
  image2process).ODellipse(1:2);
handles.results(handles.image2process).Coefficient=a;
handles.results(handles.image2process).Residual=ResSum;
% Save R=radius1 + radius2
handles.results(handles.image2process).ODradius=(handles.results(handles.
  image2process).ODellipse(3)+handles.results(handles.image2process).
  ODellipse(4))/2;
% Draw Ellipse
drawellip(a,OD_borders,handles);
% show located OD (draw the circles)
if handles.image2process>0
    if handles.results(handles.image2process).ODradius > 0 %&& handles.
      showOD
        [cx1,cy1,cx2,cy2,cx3,cy3]=drawCircles(handles, handles.
          image2process);
    end
end
```

```
% label the OD of this image as "Manual"
handles.results(handles.image2process).AnnotType='Manual';

setButtonStateFor_ClearOrNext(handles);

c='Redo the OD for this image or go to next image. Zoom and RGB/Green
  channel are active.';
set(handles.suggestions,'String',c);
set(handles.uipanelFig,'BorderWidth',1,'ShadowColor','k','HighlightColor
  ','w');
% updates handles
guidata(hObject, handles);
```

```
% --- Executes on button press in zoomOUT.
function zoomOUT_Callback(hObject, eventdata, handles)

button_state = get(hObject,'Value'); % save the state of the button (
  pressed or not)
if button_state == get(hObject,'Max')   % toggle button is pressed
   zoom on; % activate zoom
   get(zoom, 'Direction');
   set(zoom, 'Direction', 'out'); % set direction of the zoom = zoom out
elseif button_state == get(hObject,'Min')    % toggle button is not
  pressed
  disp('off');
end
```

```
% --- Executes on button press in zoomIN.
function zoomIN_Callback(hObject, eventdata, handles)

button_state = get(hObject,'Value'); % save the state of the button (
  pressed or not)
if button_state == get(hObject,'Max')   % toggle button is pressed
   zoom on; % activate zoom
   get(zoom, 'Direction');
```

```
    set(zoom, 'Direction', 'in'); % set direction of the zoom = zoom in
elseif button_state == get(hObject,'Min')    % toggle button is not
  pressed
  disp('off');
end
```

```
% --- Executes when selected object is changed in uipanelC_G.
function uipanelC_G_SelectionChangeFcn(hObject, eventdata, handles)

if hObject==handles.colourButton %user chose RGB image
    axes(handles.mainFigure)
    imshow(handles.image); hold on;
elseif hObject==handles.Gbutton %user chose green channel image
    handles.imageG=handles.image(:,:,2);
    axes(handles.mainFigure)
    imshow(handles.imageG); hold on;
end
% plot again what necessary
[cx1,cy1,cx2,cy2,cx3,cy3]=drawCircles(handles, handles.image2process);
% updates handles
guidata( hObject, handles )
```

```
% --- Executes on button press in clearButton.
function clearButton_Callback(hObject, eventdata, handles)
%reset the OD data for this image:
handles.results(handles.image2process).ODborders=[];
handles.results(handles.image2process).ODellipse=[];
handles.results(handles.image2process).ODcenter  =[];
handles.results(handles.image2process).Coefficient=[];
handles.results(handles.image2process).Residual=[];
handles.results(handles.image2process).ODradius=[];

set(handles.colourButton,'Value',1);
set(handles.Gbutton,'Value',0);
```

```
% re-plot the image without the wrong OD just calculated
axes(handles.mainFigure);
imshow(handles.image);hold on;

setButtonStateFor_ChoseAnnotation(handles);

s='First, if you want, zoom in on the area of interest and see the green
  channel image, then choose the annotation mode.';
set(handles.suggestions,'String',s);
% updates handles
guidata(hObject, handles);
```

```
% --- Executes on button press in nextImageButton.
function nextImageButton_Callback(hObject, eventdata, handles)
% once next button is pressed, the information of the previous image
  should be saved
resultInfo(handles.image2process,:)=[handles.results(handles.image2process
  ).ODellipse];
resultInfoAdditional(handles.image2process,:)=[handles.results(handles.
  image2process).Coefficient',handles.results(handles.image2process).
  Residual];
% save results in .txt file
fileID=fopen(strcat(directory,'\','RESULTS','\',cell2mat(handles.files.
  name(handles.imageID)),'_','RESULTS','_', dateNtime,'.','txt'),'a+');
write_OD(fileID,resultInfo(handles.image2process,:),handles.results(
  handles.image2process).AnnotType);
fclose(fileID);

zoom out;
% read next image
handles.imageID=handles.imageID+1;
while handles.imageID<=handles.files.nFiles
    if processed(handles.imageID)==0 %imageID doesn't have to be annotated
        so don't open it
         handles.imageID=handles.imageID+1;
    else %imageID has to be annotated so open it
        handles.image2process=handles.image2process+1;
        % display the number of images
        c=strcat('Image:',num2str(handles.image2process),'/',num2str(
          handles.files.nFiles2process));
        set(handles.display,'String',c);
        c1=handles.files.name(handles.imageID);
        set(handles.display1,'String',c1);
        handles.image = imread([ handles.files.directory filesep cell2mat(
          handles.files.name(handles.imageID)) ]);
        % plot image
        axes(handles.mainFigure)
        imshow(handles.image);hold on;
        setButtonStateFor_ChoseAnnotation(handles);
        s='First , if you want, zoom in on the area of interest and see the
           green channel image, then choose the annotation mode.';
        set(handles.suggestions,'String',s);
        % updates handles
        guidata(hObject, handles);
        break;
    end
end
% if the images are finished
if handles.imageID > handles.files.nFiles
    c='All images have been annotated';
    set(handles.display,'String',c);
    c1=' ';
    set(handles.display1,'String',c1);
    s='Press "Finish" to go to Start Menu';
    set(handles.suggestions,'String',s);
    % delate retinal images and replot vampire logo
    imageLogo=imread('eyeLogo.png');
    axes(handles.mainFigure)
    cla
    axis auto
    imshow(imageLogo);
    setButtonStateFor_Finish(handles);
    return;
end
% updates handles
guidata(hObject, handles);
```

```
% --- Executes on button press in finish.
function finish_Callback(hObject, eventdata, handles)
close;
```

```
function setButtonStateFor_ChoseAnnotation(handles)
  set(handles.locateButton,'Enable','on','BackgroundColor
    ',[1,0.345,0.137]);
  set(handles.clearButton,'Enable','off','BackgroundColor
    ',[0.941,0.941,0.941]);
  set(handles.autolocateButton, 'Enable','on','BackgroundColor
    ',[1,0.345,0.137]);
  set(handles.zoomIN,'Enable','on','BackgroundColor',[1,0.345,0.137]);
  set(handles.zoomOUT,'Enable','on','BackgroundColor',[1,0.345,0.137]);
  set(handles.uipanelZoom,'ShadowColor',[1,0.345,0.137]);
  set(handles.uipanelC_G,'ShadowColor',[1,0.345,0.137]);
  set(handles.colourButton,'Enable','on');
  set(handles.Gbutton,'Enable','on');
  set(handles.nextImageButton,'Enable','off','BackgroundColor
    ',[0.941,0.941,0.941]);
  set(handles.finish,'Enable','off','BackgroundColor',[0.941,0.941,0.941])
    ;
```

Code of paragraph 3.3.3

```
% --- Executes when selected object is changed in uipanelFovea.
function uipanelFovea_SelectionChangeFcn(hObject, eventdata, handles)
global countV;
```

```
if countV==0
    if hObject==handles.YESvisible
        handles.results(handles.image2process).visible='YES';
    elseif hObject==handles.NOvisible
        handles.results(handles.image2process).visible='NO';
    end
    countV=countV+1;
    s='Now you have to select consecutively first the center, then a point
       on the contour of the fovea so as to obtain a circumference that
      surrounds it. First, if you want, zoom in on the area of interest
      and see the green channel image, then press "Set center and contour
      ".';
    set(handles.suggestions,'String',s);
    setButtonStateFor_setCenterContour(handles);
else
    if hObject==handles.YESvisible
        handles.results(handles.image2process).visible='YES';
    elseif hObject==handles.NOvisible
        handles.results(handles.image2process).visible='NO';
    end
    countV=countV+1;
end
% update handles:
guidata(hObject, handles);
```

```
function setCenterContour_Callback(hObject,eventdata,handles)

getUserInput();
% calculation of the polar_coordinates, for center:
[thetaODc,thetaODfovc,rhoODc,thetaICc,rhoICc]=polar_coordinates(xFOV, yFOV
   , x_image_size, y_image_size, xOD, yOD, xFOV, yFOV);
handles.results(handles.image2process).FoveaCenter=[center(handles.
  image2process,:), polarFoveaCenter(handles.image2process,:)];
% for contour:
[thetaODb,thetaODfovb,rhoODb,thetaICb,rhoICb]=polar_coordinates(xContour,
  yContour, x_image_size, y_image_size, xOD, yOD, xFOV, yFOV);
handles.results(handles.image2process).FoveaContour=[contour(handles.
  image2process,:), polarFoveaContour(handles.image2process,:)];
% calculate radius:
r(handles.image2process)=sqrt((center(handles.image2process,1)-contour(
  handles.image2process,1))^2+(center(handles.image2process,2)-contour(
  handles.image2process,2))^2);
handles.results(handles.image2process).FoveaRadius=r(handles.image2process
  );
% draw circle:
drawcirclefovea(center(handles.image2process,1),center(handles.
  image2process,2),r(handles.image2process));
```

```
function getUserInput()
hold on
% center:
c=ginput(1);
plot(c(1),c(2),'.g');
center=[center; c];
% contour:
b=ginput(1);
plot(b(1),b(2),'.b');
contour=[contour; b];
```

```
function [thetaOD,thetaODfov,rhoOD,thetaIC,rhoIC]=polar_coordinates(xp,yp,
  xpix,ypix,xod,yod,xfov,yfov)
    %shift the origin in OD:
    xpod=xp-xod;
    ypod=yod-yp;
    %shift the origin in the center of the image:
    xpic=xp-(xpix/2);
    ypic=(ypix/2)-yp;
    %%polar coord related to the positive horizontal axis (angle
      counterclockerwise)
    %pole in OD:
```

```
[tODrad,rhoOD]=cart2pol(xpod, ypod);
tOD=tODrad*180/pi;
%pole in IC:
[tICrad,rhoIC]=cart2pol(xpic, ypic);
tIC=tICrad*180/pi;
%%rotation of 90  so that the polar axis is the positive vertical one
%pole in OD:
if tOD<90
    thetaOD=270+tOD;
elseif tOD>=90
    thetaOD=tOD-90;
end
%pole in IC:
if tIC<90
    thetaIC=270+tIC;
elseif tIC>=90
    thetaIC=tIC-90;
end
%right eye is OK, change rotation for left eye:
if xfov>xod
    thetaOD=360-thetaOD;
    thetaIC=360-thetaIC;
end

%% thetaODfov:
%calculate the angle difference between the horizontal axis of the
  image and the OD-fovea axis:
% hor=vector // horizontal axis of the image
hor=[xfov-xod 0];
% odfov=vector from OD to fovea
odfov=[xfov-xod yfov-yod];
% alpha=angular difference between these 2 vectors
alpha=-(180/pi)*atan2(hor(1)*odfov(2)-hor(2)*odfov(1), hor(1)*odfov(1)
  +hor(2)*odfov(2));
alpha=abs(alpha);
```

```
    % calculate thetaODfov
    if yfov>yod
        tODfov=thetaOD-alpha;
        if tODfov<0
            thetaODfov=thetaOD-alpha+360;
        else
            thetaODfov=tODfov;
        end
    elseif yfov<yod
        tODfov=thetaOD+alpha;
        if tODfov>360
            thetaODfov=thetaOD+alpha-360;
        else
            thetaODfov=tODfov;
        end
    elseif yfov==yod
        thetaODfov=thetaOD;
    end
end
```

```
function drawcirclefovea(xc,yc,r)
theta=linspace(0,2*pi,40);
x_circle=r*cos(theta);
y_circle=r*sin(theta);
cx=x_circle+xc;
cy=y_circle+yc;
plot(cx, cy, '-c');
```

```
function RemoveButton_Callback(hObject, eventdata, handles)
center(end,:) = []; %remove last item.
contour(end,:) = [];
%update the axis by re-plotting the image
cla;
imshow(handles.image);
```

Code of paragraph 3.3.4

```
%--- Executes on button press in setJunctionButton.
function setJunctionButton_Callback(hObject, eventdata, handles)

setButtonStateFor_InProcess(handles);

getUserInputJunction();
j=j+1; %counter of the junctions

% check if shift key is pressed
shiftValue=get(handles.figure1, 'SelectionType');
hold on;
if strcmp(shiftValue, 'extend') %if shift pressed=vein
    vesselType = 'V';
    plot(junctions(j,1), junctions(j,2), '.b');
else
    vesselType = 'A'; %if shift not pressed=artery
    plot(junctions(j,1), junctions(j,2), '.r');
end

%% polar_coordinates input: xp,yp,xpix,ypix,xod,yod,xfovea,yfovea
[thetaOD,thetaODfov,rhoOD,thetaIC,rhoIC]=polar_coordinates(junctions(j,1),
   junctions(j,2), size(handles.image,2), size(handles.image,1),
  resultInfo(handles.image2process,1), resultInfo(handles.image2process,2)
  , xFovea(handles.image2process), yFovea(handles.image2process));
tempPolar(j,:)=[thetaOD,thetaODfov,rhoOD,thetaIC,rhoIC];
handles.Junctions(handles.image2process).Junctions=[junctions tempPolar
  VeinOrArt'];

setButtonStateFor_ReadyJunctions(handles);
s='Now you can redo the last junction, or set another junction pressing "
  Set junction", or, if you have selected all the junctions of interest
  for this picture, start the calculation of the angles pressing "Branch
  points selection". Zoom and RGB/Green channel are active.';
```

```
set(handles.suggestions,'String',s);
% updates handles
guidata( hObject, handles )
```

```
function getUserInputJunction()

ju=ginput(1);
junctions=[junctions;ju];
```

```
% --- Executes on button press in setVesselsPointsButton.
function setVesselsPointsButton_Callback(hObject, eventdata, handles)

i=0;
zoom out;

if i==size(junctions,1) % junctions for this image are finished
    setButtonStateFor_NextImage(handles);
    set(handles.uipanelLittleImage,'BorderWidth',1,'ShadowColor','k','
      HighlightColor','w');
    s='The annotation of this image is finished, go to the next one';
    set(handles.suggestions,'String',s);
    return;
end

s='Now, IN THE PANEL BELOW ON THE RIGHT, you have to set the points in the
   branches. Press ''Set points'' to start.';
set(handles.suggestions,'String',s);

i=i+1;
%plot rectangle on Main Figure
axes(handles.mainFigure)
imshow(handles.image);hold on;
for temp=1:size(junctions,1)
    if strcmp(char(VeinOrArt(temp)),'V')
        plot(junctions(temp,1), junctions(temp,2), '.b');
    else
        plot(junctions(temp,1), junctions(temp,2), '.r');
    end
end
w=100; %width of the rectangle
h=100; %high of the rectangle
xr=junctions(i,1)-(w/2);
yr=junctions(i,2)-(h/2);
rectangle('Position',[xr,yr,w,h],'Curvature',[0,0],'LineWidth',2,'
  LineStyle','-','EdgeColor','y');

%plot OLD junctions:
if not(isempty(oldJunctions))
    for k=1:size(oldJunctions,1)
        if oldJunctions(k,3)==1
            plot(oldJunctions(k,1),oldJunctions(k,2),'bx');
        elseif oldJunctions(k,3)==0
            plot(oldJunctions(k,1),oldJunctions(k,2),'rx');
        end
    end
end

% plot image on littleFigure
axes(handles.littleFigure)
imshow(handles.image);hold on;
plot(junctions(i,1), junctions(i,2), '.c');
%zoom in on point location
xmin = junctions(i,1)-100;
xmax = junctions(i,1)+100;
ymin = junctions(i,2)-100;
ymax = junctions(i,2)+100;
axis([xmin xmax ymin ymax]);

setButtonStateFor_ReadyPoints(handles);
% updates handles
```

```
guidata( hObject, handles )
```

```
% --- Executes on button press in PointsForNextJunctionButton.
function PointsForNextJunctionButton_Callback(hObject, eventdata, handles)

s='Now, IN THE PANEL BELOW ON THE RIGHT, click first on the mother vessel
  and then on the two children (always in the middle of the vessel). After
   selecting the mother a circle that will help you to click on the
  children will appear.';
set(handles.suggestions,'String',s);
setButtonStateFor_InProcess(handles);

getUserInputPoints();
s='Now you can redo last selection or go to next junction.';
set(handles.suggestions,'String',s);
setButtonStateFor_RedoOrNext(handles);
% call the function calculateAngles
angles(i,:)=calculateAngles(junctions(i,:),m(i,:),p1(i,:),p2(i,:));
handles.Junctions(handles.image2process).VesselPoints=[m p1 p2];
handles.Junctions(handles.image2process).Angles=angles;
% updates handles
guidata(hObject, handles)
```

```
function getUserInputPoints()

hold on
pointm=ginput(1); %mother
plot(pointm(1),pointm(2),'.b');
rj=sqrt((junctions(i,1)-pointm(1))^2+(junctions(i,2)-pointm(2))^2); %
  radius
drawcirclefovea(junctions(i,1),junctions(i,2),rj); %circle with center in
  junctions(i,:) and radius rj
m=[m; pointm];

point1=ginput(1); %children 1
```

```
plot(point1(1),point1(2),'.g');
p1=[p1; point1];

point2=ginput(1); %children 2
plot(point2(1),point2(2),'.r');
p2=[p2; point2];
```

```
%function to calculate angles
function phi=calculateAngles(ju,m,p1,p2)

mother_vector=[m(1)-ju(1) m(2)-ju(2)];
child1_vector=[p1(1)-ju(1) p1(2)-ju(2)];
child2_vector=[p2(1)-ju(1) p2(2)-ju(2)];
%Plot vectors
plot([m(1) ju(1)], [m(2) ju(2)], [p1(1) ju(1)], [p1(2) ju(2)], [p2(1) ju
  (1)], [p2(2) ju(2)]);
%calculate angles:
a1=-(180/pi)*atan2(mother_vector(1)*child1_vector(2)-mother_vector(2)*
  child1_vector(1), mother_vector(1)*child1_vector(1)+mother_vector(2)*
  child1_vector(2));
a2=-(180/pi)*atan2(child1_vector(1)*child2_vector(2)-child1_vector(2)*
  child2_vector(1), child1_vector(1)*child2_vector(1)+child1_vector(2)*
  child2_vector(2));
a3=-(180/pi)*atan2(child2_vector(1)*mother_vector(2)-child2_vector(2)*
  mother_vector(1), child2_vector(1)*mother_vector(1)+child2_vector(2)*
  mother_vector(2));
if a2>0 %the user has selected the vectors counterclockwise
    if abs(a3)>abs(a1)
        phi1=abs(a1);
        phi2=abs(a2);
        phi3=360-phi1-phi2;
    else %abs(a3)<abs(a1)
        phi2=abs(a2);
        phi3=abs(a3);
        phi1=360-phi2-phi3;
    end
elseif a2<0 %the user has selected the vectors clockwise
    if abs(a3)>abs(a1)
        phi2=abs(a2);
        phi3=abs(a1);
        phi1=360-phi2-phi3;
    else %abs(a3)<abs(a1)
        phi1=abs(a3);
        phi2=abs(a2);
        phi3=360-phi1-phi2;
    end
end
phi=[phi1,phi2,phi3];
```

Code of paragraph 3.3.5

```
% --- Executes during object creation, after setting all properties.
function generation_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
  defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function generation_Callback(hObject, eventdata, handles)

handles.Widths(handles.image2process).Generation(j)=get(hObject,'String');
% updating handles
guidata(hObject, handles);
```

```
% --- Executes on button press in ContoursForNextPointButton.
function ContoursForNextPointButton_Callback(hObject, eventdata, handles)

```

```
s='Now, IN THE PANEL BELOW ON THE RIGHT, click on the contours of the
  vessel.';
set(handles.suggestions,'String',s);
setButtonStateFor_InProcess(handles);
% call the function getUserInputContours
getUserInputContours();
s='Now you can redo last selection or go to next point.';
set(handles.suggestions,'String',s);
setButtonStateFor_RedoOrNext(handles);
% points on the contour
global contour1;
global contour2;
% call the function calculateWidths
widths(i,:)=calculateWidths(contour1(i,:),contour2(i,:));
handles.Widths(handles.image2process).Contours=[contour1 contour2];
handles.Widths(handles.image2process).Widths=widths;
% updates handles
guidata( hObject, handles )
```

```
function getUserInputContours()

hold on
b1=ginput(1);
plot(b1(1), b1(2), '.y');
contour1=[contour1; b1];
% vector from current point to central point
v=b1-points(i,:);
v=v./norm(v); % normalized
vp=[-v(2) v(1)]; % perpendicular
% plot line along vessel
if VeinOrArt(i)==86
    colourStr = '-b';
else
    colourStr = '-r';
end
```

```
plot([points(i,1) points(i,1)+v(1)*30],[points(i,2) points(i,2)+v(2)*30] ,
   '-g');
plot([points(i,1) points(i,1)+vp(1)*10],[points(i,2) points(i,2)+vp(2)*10]
   ,colourStr);
% plot line acros vessel (other side)
plot([points(i,1) points(i,1)-v(1)*30],[points(i,2) points(i,2)-v(2)*30] ,
   ':g');
% point on the other contour
b2=ginput(1);
plot(b2(1), b2(2), '.y');
contour2=[contour2; b2];
```

```
% function to calculate widths
function w=calculateWidths(p1,p2)
w=sqrt((p2(1)-p1(1))^2+(p2(2)-p1(2))^2);
```

Code of paragraph 3.3.6

```
h=actxcontrol('WMPlayer.OCX.7',[428 8 846 587]);
filename='videodemo.mp4';
pathname='C:\...\';
h.URL=[pathname filename];
h.controls.play;
```

```
% --- Executes on button press in SOPbutton.
function SOPbutton_Callback(hObject, eventdata, handles)
open('C:\...\SOPdocument.pdf');
```

Code of paragraph 3.3.7

```
function write_GeneralInformations(fileID,processed,img,annotator,
  clinician,dateNtime,firstAnnotator,firstDate)
```

```
fprintf(fileID,'%s\nImage:%s, annotator name:%s, %s, date and time:%s,
   OD and fovea annotator:%s, date and time OD and fovea annotation:%s\n',
  processed, img, annotator, clinician, dateNtime, firstAnnotator,
  firstDate);
```

```
function write_OD(fileID,data, annotType)
fprintf(fileID, 'OD: Xc:%.1f, Yc:%.1f, radius1:%.1f, radius2:%.1f,
  theta:%.1f, annotationType:%s\n', data, annotType);
```

```
function write_Fovea(fileID,center,radius, visible)
fprintf(fileID,'FOVEA: Xc:%.1f, Yc:%.1f, thetaOD:%.1f, thetaODfov:%.1f,
    rhoOD:%.1f, thetaIC:%.1f, rhoIC:%.1f, radius:%.1f, visible:%s\n',
  center, radius, visible);
```

```
function write_Junctions(fileID,junctionsNva,vesselpoints,angles)
fprintf(fileID,'JUNCTION: Xj:%.1f, Yj:%.1f, thetaOD:%.1f, thetaODfov
  :%.1f, rhoOD:%.1f, thetaIC:%.1f, rhoIC:%.1f, vesselType:%c, Xm:%.1f
  , Ym:%.1f, Xc1:%.1f, Yc1:%.1f, Xc2:%.1f, Yc2:%.1f, phi1:%.1f, phi2
  :%.1f, phi3:%.1f\n', junctionsNva, vesselpoints, angles);
```

```
function write_Widths_Generation(fileID,pointNva,w,generation)
fprintf(fileID,'WIDTH: Xw:%.1f, Yw:%.1f, thetaOD:%.1f, thetaODfov:%.1f,
    rhoOD:%.1f, thetaIC:%.1f, rhoIC:%.1f, vesselType:%s, width:%.1f,
  generation:%u\n', pointNva, w, generation);
```

```
function write_Widths(fileID,pointNva,w)
fprintf(fileID,'WIDTH: Xw:%.1f, Yw:%.1f, thetaOD:%.1f, thetaODfov:%.1f,
    rhoOD:%.1f, thetaIC:%.1f, rhoIC:%.1f, vesselType:%s, width:%.1f\n
  ', pointNva, w);
```

Code of paragraph 3.3.8

```
%data with label L:
lp=regexp(line,L); % label position
```

```
if not(isempty(lp))
 res{i,cc}=L; % res is the structure
 cc=cc+1;
 ls=size(L,2); % label size
 ssi=ls+lp; % start string interval
 vir=regexp(line,',');
 c=0;
 for v=1:size(vir,2)
  if vir(v)>lp
   esi=vir(v)-1; % end string interval
   c=1;
   break
  end
 end
 if c==0
  res{i,cc}=line(ssi:end); % the data is the last one of itse line
 elseif c==1
  res{i,cc}=line(ssi:esi); % the data isn't the last one of its line
 end
end
```

Bibliography

[1] N. Patton, T. M. Aslam, T. J. MacGillivray, I. J. Deary, B. Dhillon, R. H. Eikelboom, K. Yogesan, and I. J. Constable, “Retinal image analysis: concepts, applications and potential,” in Prog. Retin. Eye Res., vol. 25, no. 1, pp. 99127, 2006.

[2] B. Al-Diri, A. Hunter, D. Steel and M. Habib, “Manual Measurement of Retinal Bifurcation Features,” in Engineering in Medicine and Biology Society (EMBC) Annual International Conference of the IEEE, 2010.

[3] N. Patton, T. Aslam, T. J. MacGillivray, A. Pattie, I. J. Deary, and B. Dhillon, “Retinal vascular image analysis as a potential screening tool for cerebrovascular disease,” in Journal of Anatomy, vol. 206, pp. 318348, 2005.

[4] M. Niemeijer, X. Xu, A. V. Dumitrescu, P. Gupta, B. van Ginneken, J. C. Folk and M. D. Abrmoff, “Automated Measurement of the Arteriolar-to-Venular Width Ratio in Digital Color Fundus Photographs,” in IEEE Trans on Medical Imaging, vol. 30, no. 11, pp. 19411950, 2011.

[5] A. Perez-Rovira, T. MacGillivray, E. Trucco, K. S. Chin, K. Zutis, C. Lupascu, D. Tegolo, A. Giachetti, P. J. Wilson and A. Doney, “Vampire: Vessel assessment and measurement platform for images of the retina,” in Proc. IEEE Engineering in Medicine and Biology Society, pp. 33913394, 2011.

[6] E. Trucco, L. Ballerini, D. Relan, A. Giachetti, T. MacGillivray, K. Zutis, C. Lupascu, D. Tegolo, E. Pellegrini, G. Robertson, P. J. Wilson, A. Doney and B. Dhillon, “Novel VAMPIRE algorithms for quantitative analysis of the retinal vasculature,” in Biosignals and Biorobotics Conference (BRC), 2013.

[7] A. Giachetti, K. S. Chin, E. Trucco, C. Cobb, and P. J. Wilson, “Multiresolution localization and segmentation of the optical disc in fundus images using inpainted background and vessel information,” in ICIP, pp. 21452148, 2011.

[8] J. V. B. Soares, J. J. G. Leandro, R. M. Cesar, H. F. Jelinek and M. J. Cree, “Retinal vessel segmentation using the 2-D Gabor wavelet and supervised classification,” in IEEE Trans. on Med. Im., vol. 25, pp. 12141222, Sept. 2006.

[9] A. Cavinato, L. Ballerini, E. Trucco and E. Grisan, “Spline-based refinement of vessel contours in fundus retinal images for width estimation,” in ISBI, Apr. 2013.

[10] C. A. Lupascu, D. Tegolo and E. Trucco, “Ensembles of bagged decision trees for measuring retinal vessels using an extended multiresolution Hermite model,” submitted for publication.

[11] D. Sumukadas, R. Price, G. P. Leese, E. Trucco, M. E. T. McMurdo, “Does the European Working Group on Sarcopenia in Older People algorithm detect all those vulnerable?,” to appear in Age and Ageing.

[12] A. J. Cruz-Jentoft, J. P. Baeyens, J. M. Bauer, Y. Boirie, T. Cederholm, F. Landi, F. C. Martin, J. P. Michel, Y. Rolland, S. M. Schneider, E. Topinkov, M. Vandewoude and M. Zamboni, “Sarcopenia: European consensus on definition and diagnosis: Report of the European Working Group on Sarcopenia in Older People,” in Age and Ageing, vol. 39, issue 4, pp. 412-23, Jul 2010.

[13] M. D. Knudtson, K. E. Lee, L. D. Hubbard, T. Y. Wong, R. Klein and B. E. K. Klein, “Revised formulas for summarizing retinal vessel diameters,” in Curr. Eye Res, vol. 27, no. 3, pp.143149, 2003.

[14] http://www.mathworks.co.uk/help/matlab/index.html

[15] T. Y. Wong, M. D. Knudtson, R. Klein, B. E. . Klein, S. M. Meuer and L. D. Hubbard, “Computer-assisted measurement of retinal vessel diameters in the Beaver Dam Eye Study: methodology, correlation between eyes, and effect of refractive errors,” Ophthalmology, vol. 111, no. 6, pp. 11831190, Jun. 2004.

[16] D. Fiorin and A. Ruggeri, “Computerized analysis of narrow-field ROP images for the assessment of vessel caliber and tortuosity,” in Engineering in Medicine and Biology Society EMBC Annual International Conference of the IEEE, pp.2622-2625, 2011.

[17] D. N. Shah, C. M.Wilson, G. S. Ying, K. A. Karp, K. D. Cocker, J. Ng, E. Schulenburg, A. R. Fielder, M. D. Mills and G. E. Quinn, "Comparison of expert graders to computer-assisted image analysis of the retina in retinopathy of prematurity," in British Journal of Ophthalmology, vol. 95, issue 10, p.144, Oct 2011.

[18] C. G. Owen, A. R. Rudnicka, R. Mullen, S. A. Barman, D. Monekosso, P. H. Whincup, J. Ng and C. Paterson, "Measuring Retinal Vessel Tortuosity in 10-Year-Old Children: Validation of the Computer-Assisted Image Analysis of the Retina (CAIAR) Program," in IOVS, vol. 50, No. 5, May 2009.

[19] C. G. Owen, T. J. Ellis and E. G. Woodward, "A comparison of manual and automated methods of measuring conjunctival vessel widths from photographic and digital images," in Ophthalmic Physiol Opt., vol. 24, issue 2, pp.74-81, Mar 2004.

[20] B. Al-Diri, A. Hunter, D. Steel and M. Habib, "Manual Measurement of Retinal Bifurcation Features," in Engineering in Medicine and Biology Society (EMBC) Annual International Conference of the IEEE, 2010.

[21] E. Trucco, A. Ruggeri, T. Karnowski, L. Giancardo, E. Chaum, J. P. Hubschman, B. al-Diri, C. Y. Cheung, D. Wong, M. Abramoff, G. Lim, D. Kumar, P. Burlina, N. M. Bressler, H. F. Jelinek, F. Meriaudeau, G. Quellec, T. MacGillivray and B. Dhillon, "Validating Retinal Fundus Image Analysis Algorithms: Issues and a Proposal," in IOVS, vol. 54, No. 0, May 2013.