A Semi-Automatic Software for Segmentation and Feature Extraction from Optical Coherence Tomography Images (Semi SFE)

Mansooreh Montazerin, Zahra Sajjadifar, Rahele Kafieh

Abstract:

The Ophthalmology Science has recently witnessed marked progress due to the advent of divergent imaging techniques, especially Optical Coherence Tomography which has caught many physicians’ attention for being exact, rapid, non-invasive and low-cost. In this paper, we wish to solve the difficulties associated with OCT image segmentation by offering a handy software written in the MATLAB [1] App Designer environment which helps researchers and clinicians to easily segment the images, save the numerical outcomes and send them for proper analysis. Serving an unambiguous user interface along with a unified platform in which all necessary functions have been incorporated graphically has made this software so unique that it could be recommended to anyone tending to work on ocular OCT layers and fluids segmentation.

Keywords:

Optical Coherence Tomography; Semi-Automatic Layer Segmentation; Ocular Images; Fluid Segmentation
1. **Motivation and Significance**

Optical Coherence Tomography (OCT) is a non-invasive, relatively inexpensive imaging technique based on low-coherence interferometry and captures high-resolution multi-dimensional images from biological tissue especially the retina [1]. Macular OCT images are widely used to assist doctors in diagnosing ocular deformities such as Diabetic Macular Edema (DME), Glaucoma, Retinal Detachment and Macular Degenerations [1]. In addition to that, their interesting application in diagnosis and thus, effective treatment of some neurodegenerative diseases like Multiple Sclerosis (MS) and Neuromyelitis Optica (NMO) has attracted a lot of doctors all over the world since these abnormalities have recently turned out to show their early signs in multiple parts of the eye’s tissue and therefore, OCTs [1].

Retina is generally formed by a number of different layers each with a particular shape and thickness. These layers typically lose their standard features with the occurrence of different diseases and measuring the quantitative amount of their structure conversion, provides instructive information about the type, severity and the must-be-employed treatment procedure of that disease [1]. Furthermore, the eye is filled with intraocular fluid which maintains sufficient pressure in the eyeball and is divides into two main portions: aqueous and vitreous humor. When the intraocular fluid changes its position and is leaked into the macula by abnormal blood vessels other kinds of diseases may be caused and fluids may be revealed in OCT images [1]. Regarding the retina anatomy, B-Scans are two-dimensional cross-sectional views of the central portion of the retina, called macula and a peripapillary OCT image is taken from the area surrounding the bundle of nerve fibers at the back of the eye, called optic nerve head.

In order for segmentation of B-Scans’ layers and fluids, assorted manual, semi-automatic and full-automatic approaches have been suggested. Whereas, the manual segmentation is both time-consuming and exposed to probable errors, automatic and semi-automatic algorithms have been introduced to solve these problems and are proved to satisfy most doctors (or computer users) who are in charge of OCT segmentation. Since hardly are full-automatic methods capable of being nicely applied to all sorts of OCT images, the semi-automatic segmentation concept has been offered to not only address this issue, but also supply the Gold Standard data for the full-automatic method’s testing and training stage. An abstract table containing a list of previous works on semi-automatic OCT segmentation is presented in Table 1. However, such algorithms suffer from noticeable constraints because they mostly support one data format, do not suggest various segmentation methods to the user, do not perform a detailed process on the input data like denoising or distinct filtering and finally, they are not integrated in an open-source, user-friendly software environment to make doctors contented by easing the segmentation proceedings for them.

Our proposed software consists of independent tabs each responsible for a special function and capable of being used by either professionals or amateurs. In the “File” tab, the user could open his desired data format and convert it to an exclusive format compatible with any other MATLAB code substituted with ours and put aside OCT data format converter software. The next three tabs are assigned for layer and fluid segmentation of macula’s B-Scans. The “Manual Layer
Segmentation” tab asks the user to input completely manual segmentation of retinal layers and could be used for construction of Gold Standards and error calculation. This block saves useful information about segmented layers in a MATLAB “.mat” file in a folder with a predetermined name. Meanwhile, we have designed the “Auto Layer Segmentation” tab to take the responsibility of the software’s main algorithm, semi-automatic segmentation, and to obtain each boundary in the order we have defined. It also has an option for a semi-automatic layer correction on the assumption that a boundary should be reformed. Fluids’ identification and localization is the “Fluid Segmentation” tab’s duty which adopts both manual and semi-automatic techniques and maintains all its classified information in a MATLAB structure. Lastly, there is a block put for vascular, arcuate peripapillary images to be pre-processed, denoised and segmented in the fifth tab named the “Peripapillary” tab.

Table 1. Description of preceding algorithms for OCT semi-automatic layer and fluid segmentation

| Algorithm’s Name   | Input                                      | Number of Detected Layers                                      | Location of Segmentation |
|--------------------|--------------------------------------------|----------------------------------------------------------------|--------------------------|
| EdgeSelect [2]     | SD-OCT (Heidelberg Spectralis)            | 3 retinal layers/4 boundaries (ILM, IS/OS, RPE, BM)*           | Macula                   |
| Kago-Eye2 [3]      | SD-OCT                                    | 2 borders (C-S-S-H)                                            | Choroid                  |
| Zhao’s Method [4]  | SD-OCT                                    | 9 retinal layers (ILM, NFL/GCL, IPL/INL, INL/OPL, OPL/ONL, ELM, IS/OS, OS/RPE, RPE/CH)* | Macula                   |
| Liu’s Method [5]   | Duke Diabetic Macular Edema & POne datasets | 8 categories (ILM, NFL-IPL, INL, OPL, ONL-ISM, ISE, OSE-RPE, Fluids)* | Macula                   |
| SAMIRIX [6]        | Spectralis SD-OCT & Heidelberg Eye Explorer (HEYEX) (.vol format) | 9 boundaries (ILM, RNFL-GCL, IPL-INL, INL-OPL, OPL-ONL, ELM, IS/OS, OPT-RPE, BM)* | Macula                   |

* Each retinal layer’s abbreviation stands for as follows: ILM: Inner Limiting membrane, NFL: Nerve Fiber Layer, GCL: Ganglion Cell Layer, IPL: Inner Plexiform Layer, INL: Inner Nuclear Layer, OPL: Outer Plexiform Layer, ONL: Outer Nuclear Layer, ELM: External Limiting Membrane, IS/OS: Inner and Outer Segment, RPE: Retinal Pigment Epithelium, BM: Bruch’s Membrane, CH: Choroid.

2. **Software Description**

The software comprises 5 main tabs, namely File, Manual Layer Segmentation, Auto Layer Segmentation, Fluid Segmentation and Peripapillary, each of which has its own specific function. In addition to being able to work with different OCT data formats, with the aid of our designed software, the user would reap the benefits of various segmentation methods for OCTs and fluids, a decent layer correction procedure, a precise and suitable algorithm for peripapillary
segmentation and ultimately, a saving functionality which is considered to be the most important part of the software and stores all relevant coordinates, masks and images in a classified manner.

2.1. **File Tab**

OCT images can be of various formats. We've put three formats that are most commonly used so that the user can select his desired data format and load it into the MATLAB software environment. In this section, after loading data with one of the “.mat”, “.octbin” or ”.bin” formats, the user has an overview of different B-Scans using the top spinner provided that the loaded file has multiple B-Scans. The rotate button is set to rotate the image by 90, 180 or 270 degrees. Additionally, there is a text box in which the user types an appropriate path ending with a specific file name (e.g. the name of the patient) to save all layers and fluids’ coordinates and information (Fig. 1).

![File Tab](image)

Fig. 1. File tab

2.2. **Manual Layer Segmentation Tab**

If the user wishes to entirely acquire a boundary manually, he should make use of this tab which works with MATLAB “imfreehand” function. The user pushes mouse and drags it over the boundary. Every time his hand is picked up, a question dialogue appears on the figure and asks whether the user wants to continue or not (Fig. 2). As long as the answer is yes, multiple parts of boundary are obtained and once the answer turns to no, these parts are joined together and smoothed to make an entity (Fig. 3).
2.3. Auto Layer Segmentation Tab

2.3.1. Semi-Automatic Tab

In this tab, the user chooses his desired B-Scan as well as the boundary to-be-detected on the left hand side of the app window. Then, a MATLAB figure opens up waiting for a click on the first pixel of the opted boundary to cause Dijkstra’s algorithm \[7\] to start running. The algorithm finds the shortest path between the initial node (pixel) and other nodes (pixels) in a graph (image) by applying its interactive version, the livewire segmentation technique \[8\]. After clicking on the initial pixel of
the boundary, as the user moves the mouse on the boundary, the livewire displays the smallest cost path based on pixels’ brightness on the B-Scan. He should drag the mouse on the boundary so as to discover a route which best fits that boundary and pause the livewire by clicking on, whenever he observes that the route has become inappropriate. He, then, resumes by clicking on the next pixel beside the last one as another initial node and proceeds until the entire boundary is acquired (Fig. 4). The user should press the enter key at this time to stop running the livewire and to close the figure.

![Image](image1.png)

Fig. 2. Semi-automatic Segmentation with “Livewire” function

Livewire does not give a proper result for finding the exact location of boundaries unless the desired boundary is completely isolated from the rest of the background image. Therefore, for each boundary, we have to select an image, in which, the outline of that boundary is sufficiently discernible so that it can be captured by the livewire function with the least number of clicks. The novelty of our introduced semi-automatic segmentation scheme is the implementation of the livewire on different images resulting in different boundaries acquisition. We find the best edge-detected image utilizing diverse methods of edge detection [9],[10],[11],[12],[13]. A brief illustration of conducted operations on the main OCT image, leading to an apt background image for segmentation is shown in Fig. 5 and Fig. 6.

![Image](image2.png)

Fig. 5. An overview of operations done for production of an apt background image for segmentation
Here, it is of vital importance that user follows a specific sequence in acquiring the boundaries. Hence, there are colored lamps demonstrating the state of each boundary to guide the user. Whereas the green color is put to indicate that the boundary is acquired completely, yellow and red colors are set to be a sign of partially and not acquired boundaries respectively. As the GCL/IPL in the
macula’s B-Scans is not clearly found, it cannot be obtained by our implemented methods. Thus, the user should acquire the fourth boundary immediately after the second, while the third boundary is full-automatically found by locations of the minimum brightness between boundaries 2 and 4 in the Y-Gradient of the main image being attributed to it. After smoothing, the attained boundary is plotted on each B-Scan and its Y coordinates are saved in a “.mat” file (Fig. 7). Notice that the number of clicks depend both on the image quality and how that boundary is located on the image.

The mean number of clicks and the required time (in seconds) for each boundary to be entirely acquired, is calculated by averaging over 10 images and is indicated in Table 2.

Table 2. Number of clicks and required time for the semi-automatic method

|                  | ILM | NFL-GCL | IPL-INL | INL-OPL | OPL-ONL | ONL-IS/OS | IS/OS-RPE | BM-Choroid | total |
|------------------|-----|---------|---------|---------|---------|-----------|-----------|------------|-------|
| Number of clicks | 2.1 | 3.1     | 2.5     | 4.3     | 4       | 3.3       | 2.4       | 2.7        | -     |
| Required time    | 3.58| 5.64    | 6.14    | 8.92    | 5.98    | 6.34      | 7.62      | 4.47       | 48.69 |

Fig. 7. Semi-Automatic tab of the Auto Layer Segmentation tab
2.3.2. Manual-Grid Tab

The user chooses the to-be-segmented boundary and enters the number of adequate vertical lines by which that boundary should be gridded. After that, a gridded B-Scan opens up in a figure in order for the user to click the boundary exactly on the plotted vertical lines (Fig. 8). When finished, the interpolated boundary is depicted on the B-Scan and its Y coordinates are saved in a “.mat” file (Fig .9). Unlike the manual method implemented by MATLAB “imfreehand” function, in the manual-grid method, the user clicks on a limited number of points and the whole boundary is found through interpolation.

Table 3 represents the calculated time (in seconds) for obtaining a boundary equally divided into 10 portions and has no integral distortion (if distortion increases, number of lines would reach maximum of 15):

|                | ILM | NFL-GCL | GCL-IPL | IPL-INL | INL-OPL | OPL-ONL | ONL-IS/OS | IS/OS-RPE | BM-Choroid | total   |
|----------------|-----|---------|---------|---------|---------|---------|-----------|-----------|------------|---------|
| Required time  | 23  | 22      | 22      | 21      | 20      | 21.5    | 20.25     | 20.2      | 16         | 185.95  |

Fig. 8. Clicking on vertical lines in the manual-grid method
2.3.3. Layer Correction Tab

There is a chance put for the user to manually correct the boundary that is defectively acquired. The user should first, click on the beginning and end of the intended path for correction. This path is omitted and gridded by some vertical lines whose number has been set by the “Grid” textbox and is determined exactly like the Manual-Grid layer segmentation explained above (Fig. 10). When finished, the corrected boundary is replaced with the inaccurate one in the corresponding “.mat” file (Fig. 11).
2.3.4. **Save Layer Info Tab**

“Show All Layers and Save” button in this tab is utilized only if all boundaries have been acquired (Fig. 12). It saves all information related to the app’s semi-automatic section.
2.4. **Fluid Segmentation Tab**

Fluids appear as cysts due to various diseases, including diabetes and hypertension [ ]. There are different types of fluids with regard to their position. Intra-Retinal Fluid (IRF) normally appears above Outer Plexiform Layer (OPL), Sub-Retinal Fluid (SRF) is dark accumulations of fluid beneath the Outer Segment Layer (OSL) and PED (Pigment Epithelia Detachment) means that there is fluid beneath the Retinal Pigment Epithelium (RPE) which is the layer of cells under the retina. In this section, we have employed two distinct techniques and are going to locate all kinds of fluids through them (Fig. 13).

![Image](image.png)

*Fig. 13. File tab, appropriate B-Scan selection*

### 2.4.1. Semi-Automatic Tab

The first tab enables the user to segment fluids by a semi-automatic approach using the livewire function process of which is elaborated in the “Semi-Automatic Layer Segmentation” tab. Briefly, the user enters the desired B-Scan number and then, selects the fluid type he would like to find from the three types shown in the left hand side (IRF, SRF, PED). So, the program enters an infinite loop and receives as many objects as the user wants and the text box value indicating the Number of Objects is added automatically when each object is taken. If the user finds an object incorrectly and prefers to delete it, he can click on the “Delete” button and obtain the deleted object again. After all planned fluids are detected, by clicking on the “Finish” button, all fluid coordinates and their mask images will be stored in a MATLAB structure (Fig. 14).
2.4.2. Manual Tab

The second tab acts exactly like the first tab except that the fluids are found manually using the MATLAB “imfreehand” function. Here, unlike the “Manual Layer Segmentation” tab, there is no question dialogue when user’s hand is picked up. Instead, every time the user pushes the mouse and drags it over the fluid’s boundary and picks his hand up, one separate fluid is recognized (Fig. 15).
2.5. Peripapillary Tab

This tab is allocated to segment layers of peripapillary images semi-automatically. However, in contrast to B-Scans, existence of veins and the arcuate structure of these images can disturb the process of livewire function in a way that some boundaries cannot be obtained simply. Thus, useful approaches are adopted here for image enhancement to overcome the problems. The general procedure of this tab is illustrated in Fig. 16.

![Fig. 16. The block diagram of the Peripapillary tab](image)

2.5.1 Alignment

To align all boundaries, we need to circularly shift each column of the image by its corresponding value in a shift vector which is produced by subtracting all Y-coordinates of a reference boundary (e.g. the 6th boundary) from the maximum amount. This vector is the same length as the image’s width and its elements show the number of pixels causing the arcuate structure of the reference boundary. By choosing the sixth boundary as a reference and easily obtaining it with the livewire function, the shift vector for the MATLAB “circshift” function is determined and subsequently, the arc becomes flattened.

2.5.2 Vein Detection

In order for veins to be omitted, the user must acquire the third boundary on the flattened image to calculate the mean value of every column from the third boundary to bottom of the image. Since
veins appear in black in ocular images, local minima and maxima of these mean values are their exact locations. To find the beginning, the end and consequently the width of the veins, derivative of the mean vector suggested above is computed and by employing “islocalmin” and “islocalmax” MATLAB functions, relative extrema are figured out while one of which (minima), introduces the beginning and the other (maxima), offers the end of the veins' locations (Fig. 17). It is apparent that the width of the veins is achieved by subtracting the minima from maxima. Finally, similar to the left side half, the right side half of the vein is replaced by adjacent right (or left) columns; Therefore, the veins are quite eliminated and the peripapillary image is prepared to be segmented.

2.5.1 Pre-Processing Block

Similar to macula’s semi-automatic method, for each boundary in peripapillary OCTs, we need to find an image which that boundary is distinguishable and can be captured by livewire function with the least number of clicks (Fig. 18). We mostly use gradient and canny of images in this section (Fig. 19).
Fig. 18. Illustration of the pre-processing block in Fig. 16

Fig. 19. Peripapillary tab
2.6. *Functions Description*

Table 4. Semi SFE software’s main functions

| Tab                  | Section       | Functions                      |
|----------------------|---------------|--------------------------------|
| File                 | .vol          | Read_VOL_func                  |
|                      | .octbin       | readbin                        |
| Manual Layer         | Boundaries    | Free_hand                      |
| Segmentation         | Semi-Automatic| lwcontour, resize_contour      |
|                      | Manual-Grid   | ginputc, resize_contour        |
|                      | Layer Correction| manual_correction             |
| Auto Layer           | Semi-Automatic| lwcontour                      |
| Segmentation         | Manual        | MATLAB imfreehand              |
| Fluid Segmentation   | Semi-Automatic| lwcontour, resize_contour      |
|                      | Manual        |                              |
| Peripapillary        | Boundaries    | lwcontour, resize_contour      |
|                      |               | find_vein                      |
|                      |               | alignment                      |

3. *Illustrative Example*

Fig. 20 is brought to visualize an overview of the software for the reader. While each tab could be utilized independently from others, results can be merged together and compared for research purposes.
Fig. 20. A review of Semi SFE’s primary tabs
4. **Impact**

Our software is principally devised to benefit doctors by reducing the dedicated time to OCT layer segmentation as well as providing a straightforward, easy to manipulate environment for them. Its structure design makes it to be quickly learned and conveniently used without facing any major problem. Lack of an appropriate user interface for OCT layer and fluid segmentation raised our enthusiasm to work on this paper and we tried to fully clarify every helpful point which might be critical to running this software for the readers. Before writing this paper, we assessed practical aspects of our software by distributing it among some biomedical engineering graduate students and two ophthalmologists whose satisfaction with its time saving, multifunctional and costless features motivated us to improve our job through handing out the software to larger communities.

5. **Conclusions**

According to a thorough evaluation of our semi-automatic devised algorithm, we deduced that the method is feasible to be employed on real and artificial data, significantly reduces the amount of time layers and fluids need to be segmented and also allows researchers to combine it with their own codes in order to achieve the best result. On account of being an open-source product, it can be upgraded to newer versions by either adding more essential tabs to it or by improving current tabs and algorithms so that they can be applied more efficiently.
References

1. MathWorks, "MATLAB version R2018a", N. ed: The Mathworks Mathworks, Editor. 2018.
2. Huang, Y., et al., Development of a semi-automatic segmentation method for retinal OCT images tested in patients with diabetic macular edema. PloS one, 2013. 8(12).
3. Sonoda, S., et al., Kago-Eye2 software for semi-automated segmentation of subfoveal choroid of optical coherence tomographic images. Japanese journal of ophthalmology, 2019. 63(1): p. 82-89.
4. Zhao, L., et al., Semi-automatic OCT segmentation of nine retinal layers. Investigative Ophthalmology & Visual Science, 2012. 53(14): p. 4092-4092.
5. Liu, X., et al., Semi-supervised automatic segmentation of layer and fluid region in retinal optical coherence tomography images using adversarial learning. IEEE Access, 2018. 7: p. 3046-3061.
6. Motamedi, S., et al., Normative data and minimally detectable change for inner retinal layer thicknesses using a semi-automated OCT image segmentation pipeline. Frontiers in Neurology, 2019. 10: p. 1117.
7. Dijkstra, E.W., A note on two problems in connexion with graphs. Numerische mathematik, 1959. 1(1): p. 269-271.
8. Chodorowski, A., et al. Color lesion boundary detection using live wire. in Medical Imaging 2005: Image Processing. 2005. International Society for Optics and Photonics.
9. Canny, J., A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence, 1986(6): p. 679-698.
10. Sobel, I., History and definition of the sobel operator. Retrieved from the World Wide Web, 2014. 1505.
11. Sobel, I. and G. Feldman, A 3x3 isotropic gradient operator for image processing. a talk at the Stanford Artificial Project in, 1968: p. 271-272.
12. Serra, J., Image analysis and mathematical morphology. 1983: Academic Press, Inc.
13. Matheron, G. and J. Serra. The birth of mathematical morphology. in Proc. 6th Intl. Symp. Mathematical Morphology. 2002. Sydney, Australia.