

**OPTIC DISC LOCALIZATION USING VESSEL
CLUSTERING AND ROTATIONAL 2D VESSEL
PROJECTION**

BY

PONGSATE TANGSENG

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF MASTER OF
ENGINEERING (INFORMATION AND COMMUNICATION
TECHNOLOGY FOR EMBEDDED SYSTEMS)
SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY
THAMMASAT UNIVERSITY
ACADEMIC YEAR 2014**

**OPTIC DISC LOCALIZATION USING VESSEL
CLUSTERING AND ROTATIONAL 2D VESSEL
PROJECTION**

BY

PONGSATE TANGSENG

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF MASTER OF
ENGINEERING (INFORMATION AND COMMUNICATION
TECHNOLOGY FOR EMBEDDED SYSTEMS)**

SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY

THAMMASAT UNIVERSITY

ACADEMIC YEAR 2014



OPTIC DISC LOCALIZATION USING VESSEL CLUSTERING AND
ROTATIONAL 2D VESSEL PROJECTION

A Thesis Presented

By
PONGSATE TANGSENG

Submitted to
Sirindhorn International Institute of Technology
Thammasat University

In partial fulfillment of the requirements for the degree of
MASTER OF ENGINEERING (INFORMATION AND COMMUNICATION
TECHNOLOGY FOR EMBEDDED SYSTEMS)

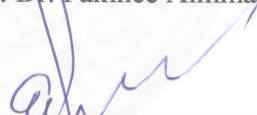
Approved as to style and content by

Advisor and Chairperson of Thesis Committee



(Asst. Prof. Dr. Pakinee Aimmanee, Ph.D.)

Co-Advisor




(Prof. Dr. Stanislav S. Makhanov, Ph.D.)

Committee Member and
Chairperson of Examination Committee



(Dr. Thepchai Supnithi, Ph.D.)

Committee Member



(Prof. Dr. Hirohiko Kaneko, Ph.D.)

JULY 2015

Acknowledgements

Foremost, I wish to express his profound to Asst. Prof. Pakinee Aimmanee, the thesis advisor, for his advice on the development of this thesis. She always provided me the insightful research plans and suggestions. Over the past two years, it is valuable time of working under her guidance.

Beside my advisor, I would like to thank all my thesis committee: Prof. Dr. Stanislav S. Makhanov, chairperson of examination committee from Sirindhorn International Institute of Technology (SIIT), Dr. Thepchai Supnithi, thesis committee from National Electronics and Computer Technology Center (NECTEC), and Prof. Dr. Hirohiko Kaneko, thesis committee from Tokyo Institute of Technology (Tokyo-Tech) for their excellence comments and suggestions.

I would like to thank Prof. Dr. Thanaruk Theeramunkong, the head of the school. He had introduced me about this master degree program when I was an undergraduate student. Finally, I would like to thank Thailand Advanced Institute of Science and Technology (TAIST), National Science and Technology Development Agency (NSTDA), Tokyo Institute of Technology, and Sirindhorn International Institute of Technology (SIIT), Thammasat University (TU) for financial support. .

Abstract

OPTIC DISC LOCALIZATION USING VESSEL CLUSTERING AND ROTATIONAL 2D VESSEL PROJECTION

by

PONGSATE TANGSENG

Bachelor of Science, Sirindhorn International Institute of Technology, 2011

Detecting optic disc (OD) in a retinal image is important for the diagnosis of numerous eye diseases. It helps ophthalmologists identify health condition of an eye. This thesis presents two effective methods for analyzing the retinal vessel structure that would be used for locating OD.

The first method is Retinal Vessel Clustering. After acquiring vessel data from retinal vessel segmentation, length and average contrast of each vessel is measured. Vessels which are too short or too obscure are removed. The remaining vessels are grouped based on distance between them. Each group is called cluster. With an assumption that the location of OD is the centroid of the main branches of vessels, clusters are used to find the location of the OD.

The second method is called Rotational 2D Vessel Projection. The main advantage of this method is that its performance is not affected by rotation of the retinal image and it is robust to abnormalities caused by diseases. Vessels in retina emerge from OD and can be observed in a retinal image as two main branches opposite to each other. If the image is rotated to proper angles, those two main branches will orient vertically. As a result, the area around the OD contains mostly vertical vessel and a few or none of horizontal vessels. As, OD is a high-contrast area because of dark vessels lying on a bright area, by rotating the image and repeatedly finding OD using these properties, the position of OD can be identified.

Keywords: Optic Disc Detection, Medical Image Processing, Data Clustering

Table of Contents

Chapter Title	Page
Signature Page	i
Acknowledgements	ii
Abstract	iii
Table of Contents	iv
List of Figures	vii
List of Tables	ix
1 Introduction	1
1.1 Background	1
1.1.1 Human eyes	1
1.1.2 Human retina, optic nerve and optic disc (OD)	2
1.1.3 Glaucoma	3
1.2 Problem statements	5
1.3 Purpose of study	6
1.4 Arrangement of thesis	6
2 Preliminary and Literature Review	7
2.1 Image processing algorithms	7
2.1.1 Image rotation	7
2.1.1.1 Nearest-neighbor interpolation	8
2.1.1.2 Bilinear interpolation	9
2.1.2 Image threshold technique	10
2.1.3 Morphological operations	11
2.1.3.1 Dilation	12
2.1.3.2 Erosion	13

2.1.3.3	Opening	14
2.1.3.4	Closing	14
2.1.3.5	Top-hat	15
2.2	Graph theories	16
2.2.1	Directed and undirected graph	16
2.2.2	Weighted graph	16
2.2.3	Complete graph	17
2.2.4	Tree	17
2.2.5	Spanning tree	18
2.2.6	Minimum spanning tree	18
2.3	Existing methods of optic disc detection	19
3	Methodology	21
3.1	Approaches	21
3.1.1	Vessel Clustering	21
3.1.1.1	Input	21
3.1.1.2	Remove low contrast segment	23
3.1.1.3	Remove short segments	24
3.1.1.4	Group adjacent segments	25
3.1.1.5	Select thickest clusters and group adjacent clusters	26
3.1.1.6	OD location from final clusters	26
3.1.2	Rotational 2D Vessel Projection	27
3.1.2.1	Vessel segmentation	28
3.1.2.2	Finding possible OD locations	28
3.1.2.3	OD location voting algorithm	30
3.2	Experimental setup	33
3.2.1	Vessel Clustering	33
3.2.2	Rotational 2D Vessel Projection	34
4	Result and Discussion	35

4.1 Result of Vessel Clustering Algorithm	35
4.2 Results of Rotational 2D Vessel Projection algorithm	37
5 Conclusion and Recommendation	40
References	41
Publication	45
Appendices	46
Appendix A: Vessel Clustering MATLAB source code	47
Appendix B: Rotational 2D Vessel Projection MATLAB source code	68

List of Figures

Figures	Page
1.1 Parts of a human eye	1
1.2 A retinal image	3
1.3 Normal vision and vision of people with glaucoma	4
1.4 Retinal image of a healthy eye and an eye with glaucoma	4
2.1 The value of destination pixel is copied from the source.	8
2.2 Original and rotated image using nearest-neighbor interpolation	8
2.3 Concept of bilinear interpolation	9
2.4 Original, nearest-neighbor, and bilinear interpolated image	10
2.5 Original and Otsu thresholded image	10
2.6 Process of dilation to a pixel	12
2.7 Original and dilated image	12
2.8 Process of erosion to a pixel	13
2.9 Original and eroded image	13
2.10 Original and opened image	14
2.11 Original and closed image	15
2.12 Original and top-hat filtered image	15
2.13 Undirected and directed graph	16
2.14 Weighted directed graph	17
2.15 Complete undirected graph	17
2.16 Tree	18
2.17 Minimum spanning tree	21
3.1 Vessel Clustering algorithm	21
3.2 Input image and segmented vessel	22
3.3 Vessel segmentation breaks vessels at junctions	22
3.4 Area around a vessel pixel	24
3.5 After remove low contrast segments	24
3.6 Using minimum spanning tree with points in a segment	25
3.7 After remove short segments	25

3.8 Clusters	25
3.9 Final clusters	26
3.10 OD location from final clusters	27
3.11 Overview of Rotational 2D Vessel Projection	27
3.12 Vessel segmentation	28
3.13 Cao et al.'s method	29
3.14 OD's location from Cao et al.'s method on various angles	30
3.15 Rotational 2D Vessel Projection algorithm	31
3.16 Images from STARE database	33
4.1 The OD location presented as the blue X mark from the Vessel Clustering	35
4.2 Incorrect Vessel Clustering result from segmentation error	36
4.3 Incorrect Vessel Clustering result from a conflict with the assumption	37
4.4 Rotational 2D Vessel Projection result	37
4.5 Examples of successful cases	39
4.6 Examples of unsuccessful cases	39

List of Tables

Tables	Page
3.1 A part of input to the algorithm	23
4.1 Accuracy comparison in percent	38



Chapter 1

Introduction

1.1 Background

1.1.1 Human eyes

Eyes are organs used to perceive visual information. They convert light to electrical information and send it to our brain. They are one of the most complex organs in human body and consist of many parts, which are crucial to our sense of vision. The following figure illustrates parts of a human eye.

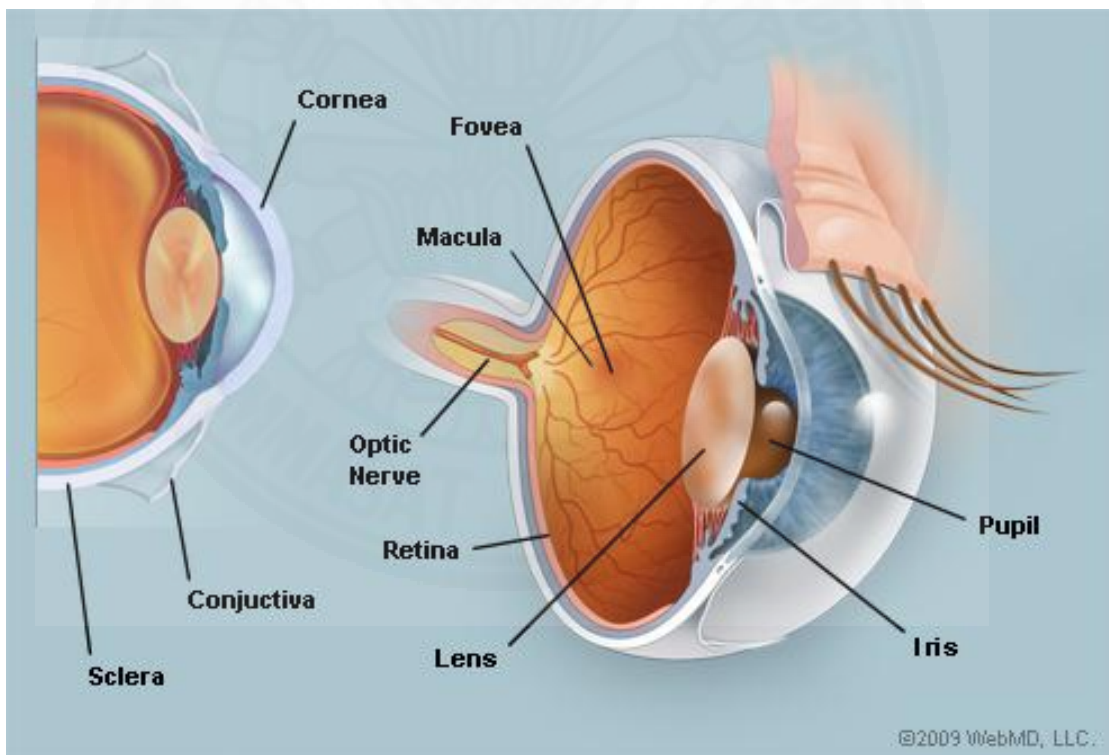


Figure 1.1: Parts of a human eye [1]

The eye is a slightly asymmetrical globe, about an inch in diameter. At the back of inner surface of an eye, there are photosensitive cells that can be simulated by light and sent electrical signals to the brain.

When light is reflected from an object of interest to an eye, the cornea refracts the light, lead it through the pupil, to the lens. There is fluid called “aqueous humor” filled in the space between lens and cornea. It provides nutrition to lens and cornea, and inflates the cornea. The iris controls the amount of light entering the eye by adjusting the size of pupil. Lens refracts and focuses the incoming light into an image of the object of interest on the retina. Inside an eye, there is fluid called *Vitreous Humor* that fills the central cavity and occupies approximate 80% of an eye. This fluid is a transparent, jelly-like substance produced at birth and remains unchanged throughout life. The retina is the most crucial part of vision and is described in the next section.

1.1.2 Human retina, optic nerve and optic disc (OD)

The neurosensory retina, usually called retina, is the second largest part of an eye follows the vitreous humor, and is the largest part of the fundus which is the interior surface of the eye. It is composed of light-sensitive layers. There are 2 types of photoreceptor cells on retina: rod cells and cone cells, named after its shape. Rod cells are sensitive to light and movement. The cells are scattered all over the retina. In contrast, cone cells are sensitive to colors. There are 3 types of cone cell correspond to three primary colors of visible light: red, green and blue. Cone cells cannot work very well in the dark, so human can only see in grayscale in low-light environment. They can be found only at the center of the retina (the area on the opposite side of the pupil) called “fovea”. These cells convert light into electrical impulses and send it to our brain via optic nerve to produce image.

The optic nerve is a paired nerve that transmits visual information from the retina to the brain. The optic nerve head or the optic disc (OD) is the point of exit for optic nerve’s axons leaving the eye and can be observed as a round-shaped bright spot inside a retinal image as seen in figure 1.2. Damages to optic nerve can causes loss of vision in the eye. The following section gives an example of the eye disease, Glaucoma, which affects the physical appearance of the OD.

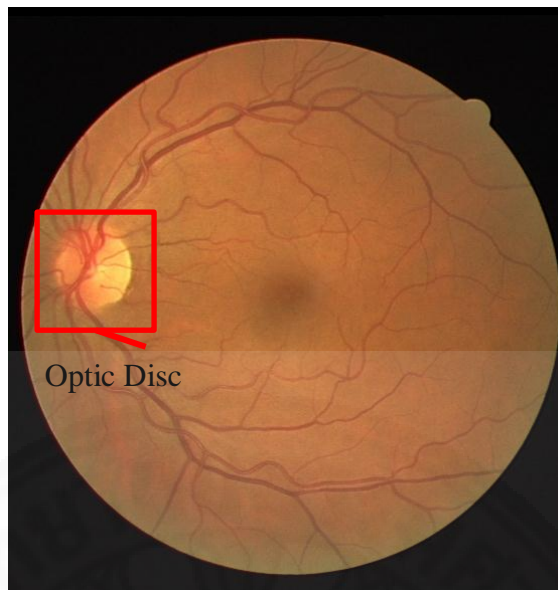


Figure 1.2: A retinal image. [2]

1.1.3 Glaucoma

Glaucoma is an eye-disease that can causes blindness by damaging eye's optic nerve. There are many types of glaucoma. One of them is called "Open-angle Glaucoma". The cause of open-angle glaucoma is the increasing of pressure inside the eye (intraocular pressure).

The aqueous humor, the fluid that filled the space between cornea and lens, is normally constantly produced by the ciliary epithelium, which is a structure supporting the lens. As fresh aqueous humor is produced by ciliary body, the part of the eye that lies just behind the iris, an equal amount must be drained through a drainage passageway (trabecular meshwork) around the edge of cornea. However, if the drainage is damaged, the fluid will be drained too slowly, the intraocular pressure will be increased. The effects of intermittent or persistent high pressure on delicate retinal nerve fibers and the optic nerve damage it; result in permanent vision loss, as shown in figure 1.3.



Figure 1.3: Normal vision (left) and vision of people with glaucoma (right) [3]

At first, open-angle glaucoma has no symptom. It does not cause any pain, and the vision loss is not noticeable. The patient will notice the vision loss only in the later stages of the disease, which make early detection utmost important. The vision loss comes from damage to the optic nerve. At the present, there is no way to repair the damaged optic nerve, so the vision loss is irreversible.

One way to detect glaucoma at its early stage is to analyze the properties of OD from the retina image. The ophthalmologist can analyze and diagnose any abnormality from it. The examples of retina images without glaucoma and with glaucoma are shown in figure 1.4.

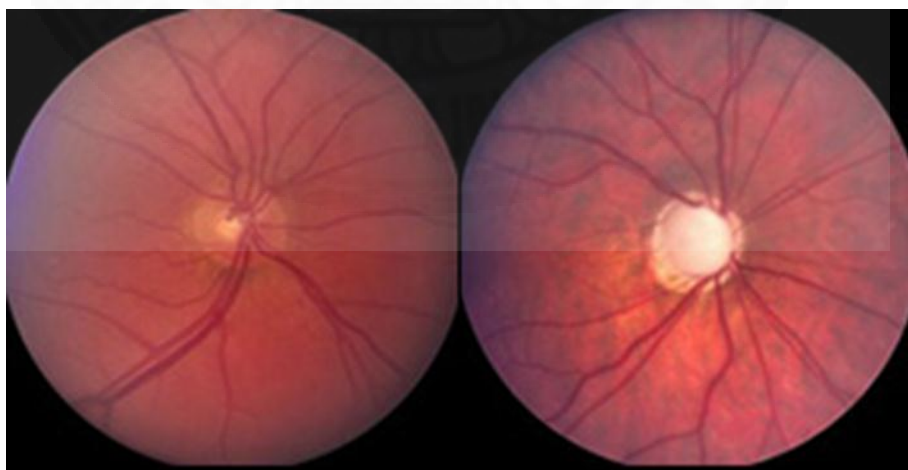


Figure 1.4: Retinal images of a healthy eye (left) and an eye with glaucoma (right) [4]

1.2 Problem statements

As stated in the previous section that one method of glaucoma detection is diagnosing the structure of OD, however, the procedure is costly and need high qualified staff. The computerized methods for optic disc detection and diagnosis could reduce the cost of the process and the number of required specialists.

This thesis concerns the development of an automatic, robust optic disc (OD)-detection method. The performance of this method is compared to the performance of several other automatic algorithms. OD-detection is an integral part of the screening system for glaucoma disease.

To locate the OD inside the retina photography, vessel structure and intensity of regions are used. Many OD detection techniques were proposed. Most of them are based on features of OD such as round shape and high brightness object in the image. However, these techniques have limitations as there are other artifacts resulted from diseases which may also be bright and round. Some techniques are based on geographic properties of the vessel network. Some take advantages of both approaches. Unfortunately, geographic properties of the vessel network are often dependent to orientation of the photo of retina, which affect the process significantly. As a result, new methods that robust to orientation of image are needed.

This thesis presents two approaches to determine position of the optic disc in a retinal image. These approaches mainly use vessels' properties which are robust to noises from diseases. The first one groups vessels into clusters, and then uses centroid of the clusters as the OD. The second one uses voting result from geometry-and-intensity based OD detection algorithm. Both approaches are robust to artifacts resulted from diseases and orientation of the image.

1.3 Purpose of study

To create OD detection algorithms that can locate the OD in retinal images effectively and robust to noise and other disturbance caused by eye diseases and/or image quality.

1.4 Arrangement of thesis

The proposal is organized as follows: In chapter 2, related theories and the existing methods for locating the OD are presented. In chapter 3, methodology of Vessel Clustering and Rotational 2D Vessel Projection is presented. Chapter 4 contains experimental setup, result and discussion of the proposed methods. Conclusion and recommendation are in chapter 5, followed by references, and list of my publication. MATLAB source code of Vessel Clustering and Rotational 2D Vessel Projection are in Appendix A and B respectively.

Chapter 2

Preliminary Principles and Literature Review

This chapter contains preliminary principles and related theories in order to provide readers the technical background about the thesis. It includes principles of image processing algorithms and few topics in graph theories. At the end of this chapter, existing methods of optic disc detection are reviewed.

2.1 Image processing algorithms

Image processing is a process of performing specific operations on an image in order to obtain desired information. This section describes image processing algorithms used in this thesis.

2.1.1 Image rotation

Image rotation performs a geometric transform from source image to destination image. Value of every pixel of destination image is copied from source image one pixel at a time. For a rotation at a clockwise angle (θ) around user-specific reference point (x_r, y_r) , value of destination pixel (x_2, y_2) is copied from source pixel (x_1, y_1) . The coordinate of source pixel (x_1, y_1) related to destination pixel (x_2, y_2) can be calculated as:

$$x_1 = \sqrt{(x_2 - x_r)^2 + (y_r - y_2)^2} * \cos(\tan^{-1}\left(\frac{y_r - y_2}{x_2 - x_r}\right) - \theta) + x_r \quad (2.1)$$

$$y_1 = y_r - \sqrt{(x_2 - x_r)^2 + (y_r - y_2)^2} * \sin(\tan^{-1}\left(\frac{y_r - y_2}{x_2 - x_r}\right) - \theta) \quad (2.2)$$

Note that the above equations apply to image rotation and y-axis of the image coordination system goes downward. This process is illustrated in figure 2.1.

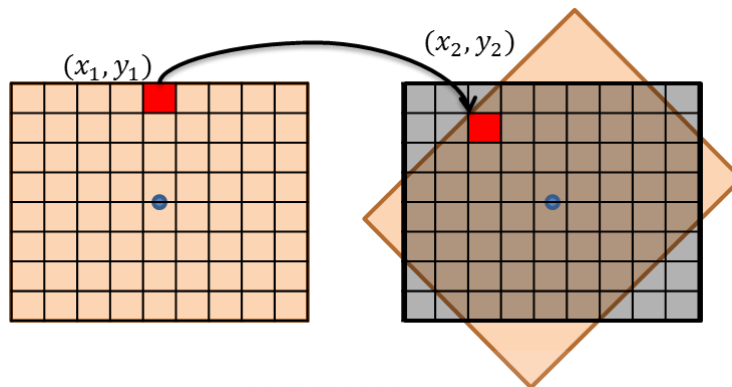


Figure 2.1: The value of destination pixel is copied from the source. The blue dot is the reference point in this rotation.

The coordinates at each pixel in a digital image must be integers but, results (x_1, y_1) from above equations are not. This section presents two approaches to cope with this problem: nearest-neighbor interpolation and bilinear interpolation.

2.1.1.1 Nearest-neighbor interpolation

When the desired position is not in integer values, the simplest way to do is just round them to integer. This method is called “Nearest-neighbor interpolation”, because we round the floating-point numbers into the nearest integer. The weak point of this method is that it gives blocky result since one or more pixels’ data of destination image are copied from one pixel of source image. Figure 2.2 shows an example of image rotation using nearest-neighbor interpolation.

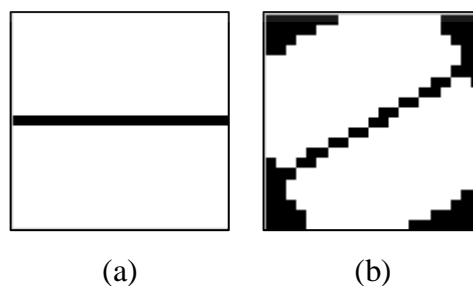


Figure 2.2: (a) original image, (b) 30° rotated image using nearest-neighbor interpolation

2.1.1.2 Bilinear interpolation

Instead of just rounding (x_1, y_1) coordinate into their nearest integers and using value of those pixels, the interpolation of surrounding pixels is used to get a better result. Bilinear interpolation uses value of 4 neighbor pixels: top-left, top-right, bottom-left, and bottom-right of the source image to calculate the value of a pixel of destination image. Figure 2.3 illustrates the concept of bilinear interpolation.

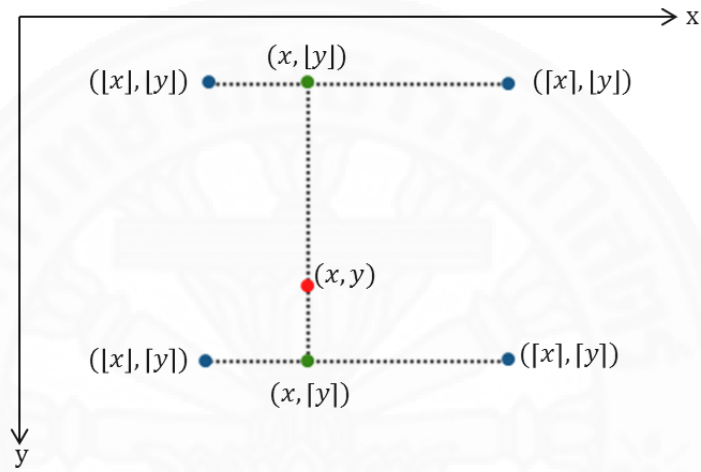


Figure 2.3: Concept of bilinear interpolation. Blue dots are pixels with known values. Green dots come from interpolations of two top and two bottom dots, respectively. The red dot is the final result, come from an interpolation of two green dots.

Assume that (x_1, y_1) resulted from equation 2.1 and 2.2 is the red dot in figure 2.3. Since it is in the middle of four known value pixels, we can estimate its value using the following equations:

$$\Delta x = x_1 - [x_1] \quad (2.3)$$

$$\Delta y = y_1 - [y_1] \quad (2.4)$$

$$t = (1 - \Delta x) * src([x_1], [y_1]) + \Delta x * src([x_1], [y_1]) \quad (2.5)$$

$$b = (1 - \Delta x) * src([x_1], [y_1]) + \Delta x * src([x_1], [y_1]) \quad (2.6)$$

$$dest(x_2, y_2) = (1 - \Delta y)t + (\Delta y)b \quad (2.7)$$

where $src(x, y)$ is value of pixel at coordinate (x, y) of source image

$dest(x, y)$ is value of pixel at coordinate (x, y) of destination image

From the equations, the value of surrounding source pixels are interpolated into the destination pixels. The rotated image will appear blurry instead of pixelated. Figure 2.4 compares results of two interpolation methods.

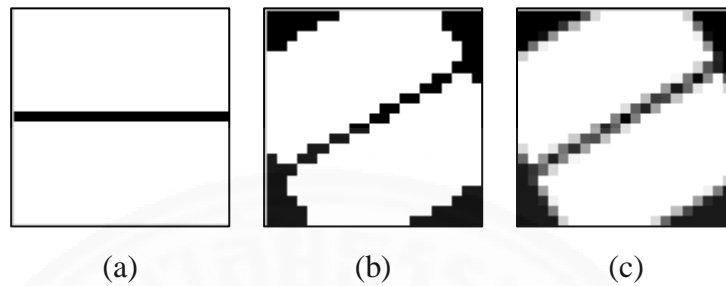


Figure 2.4: (a) original image, (b) and (c) 30° rotated image using nearest-neighbor interpolation and bilinear interpolation respectively.

2.1.2 Image thresholding technique

Thresholding is a process to separate pixels in an image into two groups, usually based on intensity of each pixel. This thesis uses image thresholding for extracting blood vessel, which is usually darker, in retinal images from background which is usually brighter. The threshold value used in this thesis is calculated using Otsu's method. Figure 2.5 shows the result of a grayscale image before and after being thresholded with Otsu's method.

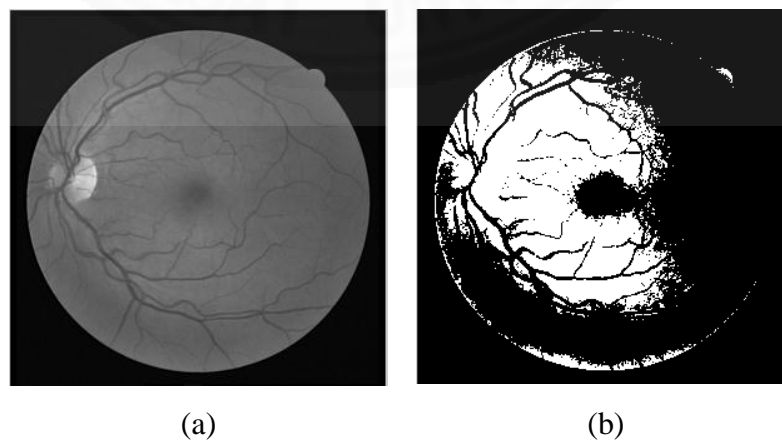


Figure 2.5: (a) original image, (b) thresholded image using Otsu's method to calculate the threshold value

Otsu's method selects the threshold value that maximizes the difference between background and foreground. In other word, the selected threshold value will maximize inter-class variance. It can be calculated by the following algorithm:

for each possible threshold value

$B = \text{set of pixels which its value} > \text{threshold}$

$F = \text{set of pixels which its value} \leq \text{threshold}$

$n_B = \text{number of elements in } B$

$n_F = \text{number of elements in } F$

$N = \text{number of pixels in the image}$

$w_B = n_B/N$

$w_F = n_F/N$

$\mu_B = (\sum \text{intensity of pixels in } B)/n_B$

$\mu_F = (\sum \text{intensity of pixels in } F)/n_F$

$\sigma^2 = w_B w_F (\mu_B - \mu_F)^2$

return threshold value with maximum σ^2

Algorithm 2.1: Otsu's method for threshold value of a grayscale image.

2.1.3 Morphological operations

Morphological operations alter shapes inside an image. They are suitable in many applications ranging from object segmentation to noise reduction. The basic morphological operations are dilation, erosion opening and closing. These basic operations can be combined into more complicated operations. All of them require two operators: an input image and a structuring element. An input image can be binary or grayscale. A structuring element is a binary matrix $m \times n$, while m and n must be odd numbers.

2.1.3.1 Dilation

The value of each pixel after applying the operation is equal to the maximum value of its neighborhood. The neighborhood is the surrounding pixels in the same shape and size of the structuring element centered at that pixel. Figure 2.6 shows the processing of a particular pixel in the input image. Figure 2.7 shows the input image, structuring element and the output after being dilated with structuring element.

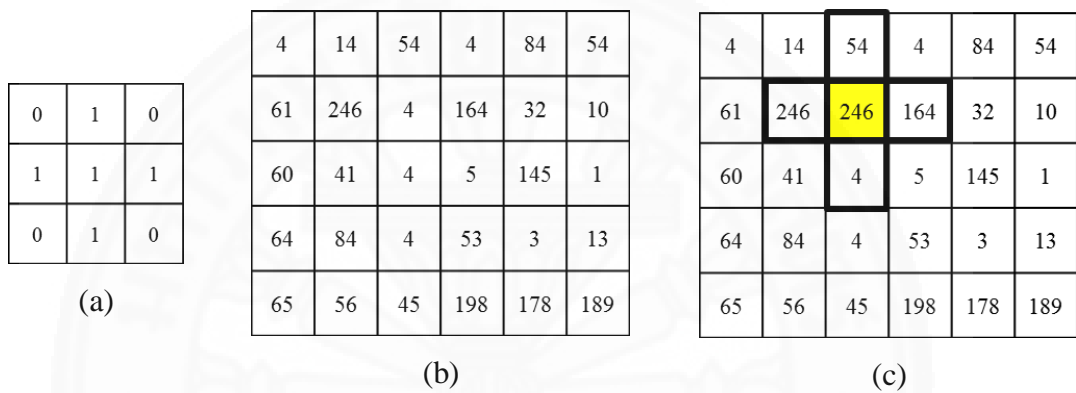


Figure 2.6: (a) structuring element, (b) input image, (c) a dilated pixel

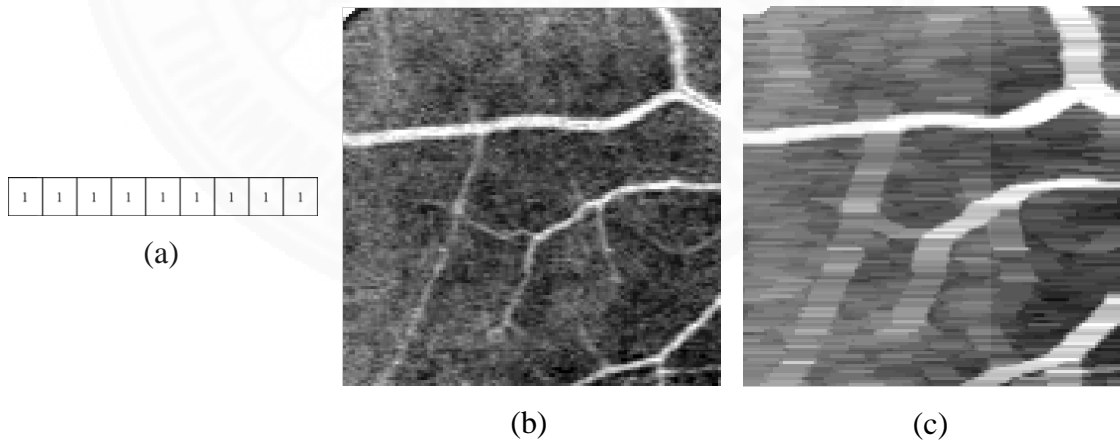


Figure 2.7: (a) structuring element, (b) input image, (c) dilated image

2.1.3.2 Erosion

The value of each pixel after applying the operation is equal to the minimum value of its neighborhood. The neighborhood is the surrounding pixels in the same shape and size of the structuring element centered at that pixel. Figure 2.8 shows the processing of a particular pixel in the input image. Figure 2.9 shows the input image, structuring element and the output after being eroded with structuring element.

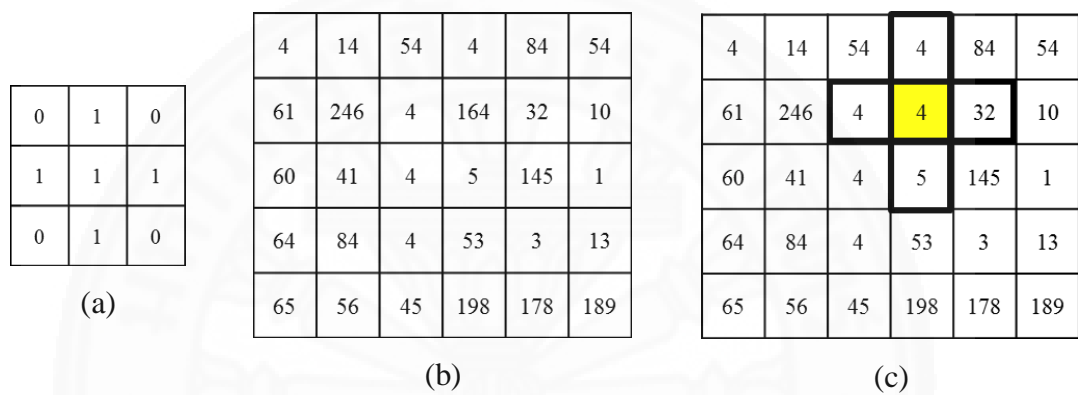


Figure 2.8: (a) structuring element, (b) input image, (c) an eroded pixel

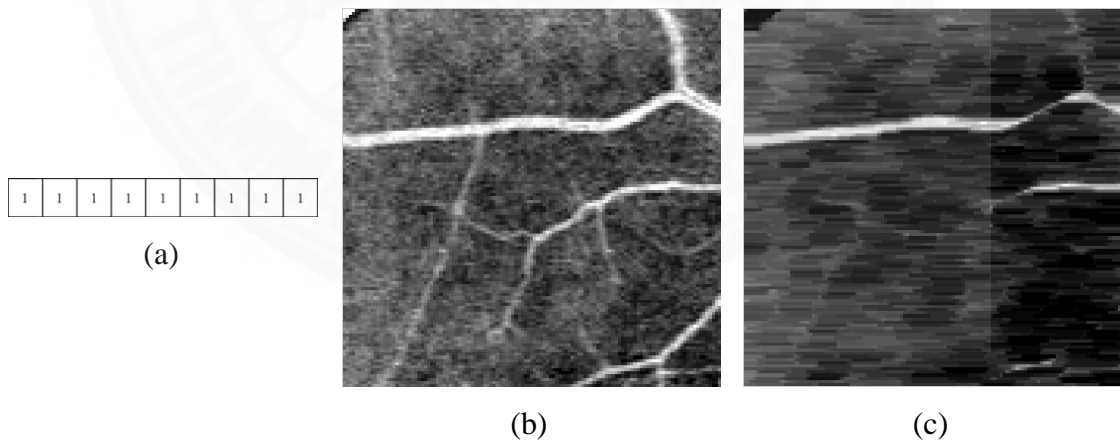


Figure 2.9: (a) structuring element, (b) input image, (c) eroded image

2.1.3.3 Opening

An opening operation is an erosion followed by a dilation using the same structuring element. The operation removes parts of foreground that the element

cannot fit in, and then expands every foreground pixel to the element. As a result, small foreground objects are removed while big foreground objects are expanded. Figure 2.10 shows the input image, structuring element and the output after being opened with structuring element.

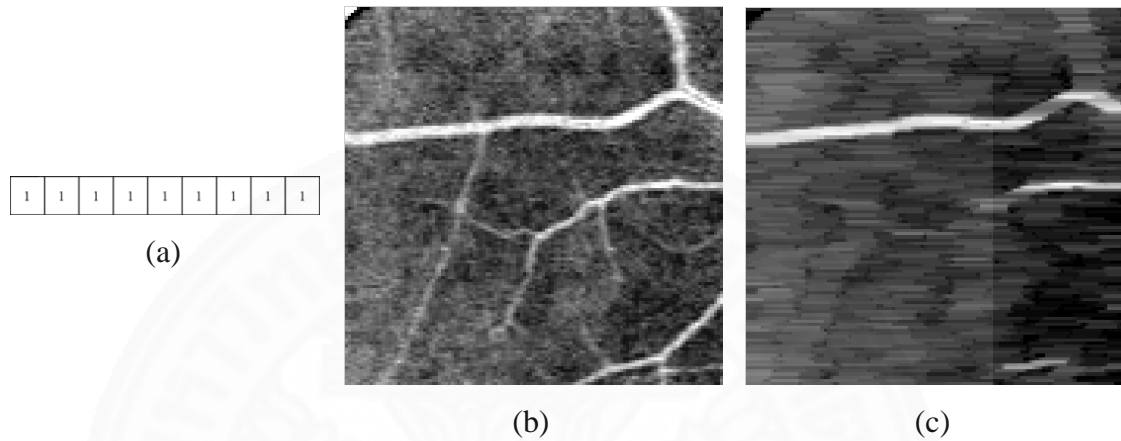


Figure 2.10: (a) structuring element, (b) input image, (c) opened image

2.1.3.4 Closing

A closing operation is a dilation followed by an erosion using the same structuring element. The operation expands every foreground pixel to the element, and then removes parts of foreground that the element cannot fit in. As a result, holes and spaces that are dwindled according to the element. Figure 2.11 shows the input image, structuring element and the output after being closed with structuring element.

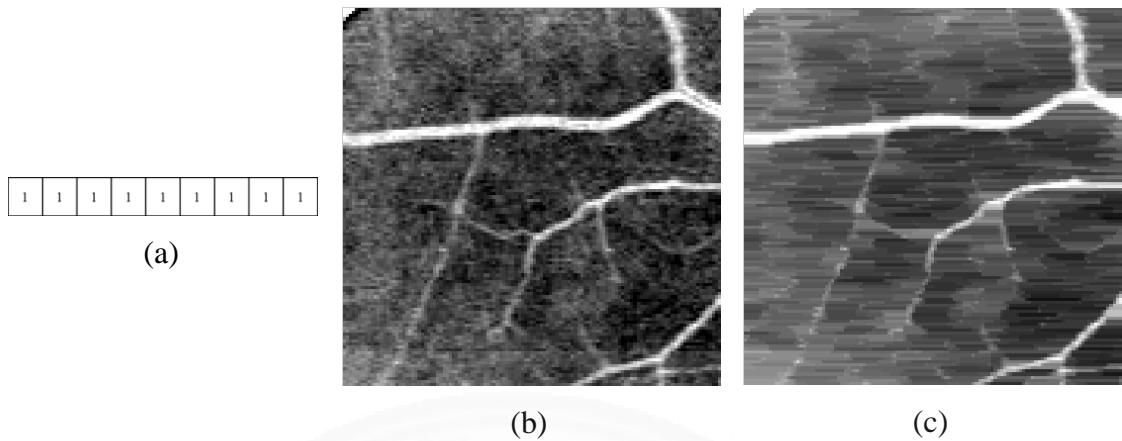


Figure 2.11: (a) structuring element, (b) input image, (c) closed image

2.1.3.5 Top-hat

A top-hat operation enhances bright points in an image. The operation can be described as $I' = I - (I \circ n)$ where I is an input image, n is the structuring element, \circ denotes opening operation and I' is an output image. Figure 2.12 shows the input image, structuring element and the output after being top-hat filtered with structuring element.

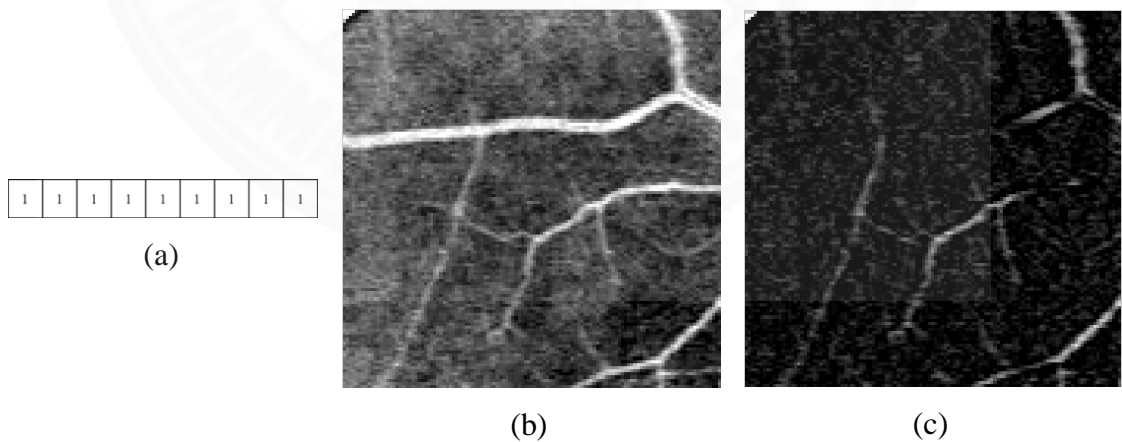


Figure 2.12: (a) structuring element, (b) input image, (c) top-hat filtered image

2.2 Graph theories

2.2.1 Directed and undirected graph

A graph G is a set of vertices V and edges E , $G = (V, E)$, where E is a set of two-element subset of V . A graph can be undirected or directed graph. In an undirected graph, the order of elements of each edge does not matter. In contrast, it does matter in a directed graph. There is an arrow at one end of edge indicates the direction. Figure 2.13 compares directed and undirected graph.

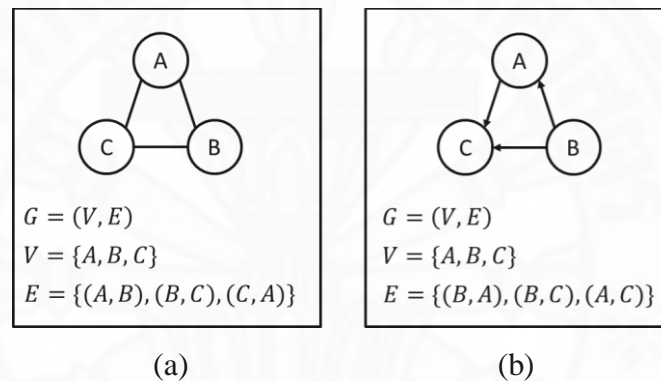
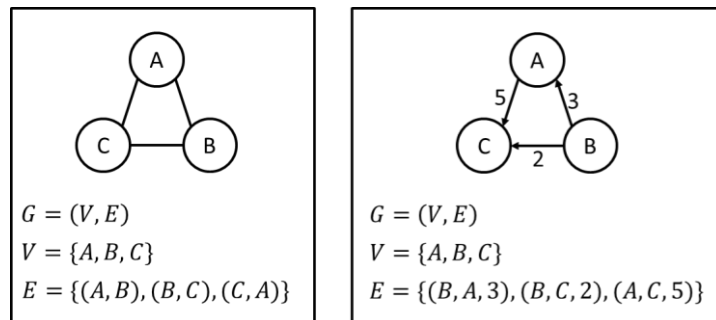


Figure 2.13: (a) undirected graph, (b) directed graph

2.2.2 Weighted graph

In addition, each edge in a graph may associates with a value “weight”. The weight in an edge can be used in various applications. For example, if a graph represents a network of cities connected by roads, the weight of each edge might represent the length of that road. Weight and direction can be used together. Figure 2.14 compares an undirected graph and a weighted directed graph.



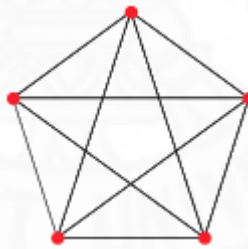
(a)

(b)

Figure 2.14: (a) undirected graph, (b) weighted directed graph

2.2.3 Complete graph

A complete undirected graph is a graph which every pair of distinct vertices is connected by an edge. A complete directed graph is a graph which every pair of distinct vertices is connected by a pair of unique edges (one for each direction). A complete undirected graph K_n has exactly $n(n - 1)/2$ edges where n is the number of vertices in the graph. Figure 2.15 shows a 5-vertex complete undirected graph, K_5 .

Figure 2.15: 5-vertex complete undirected graph K_5

2.2.4 Tree

A tree is a special kind of an undirected graph that contain no cycle, e.g. there is exactly one path connecting two vertices in the tree. A tree T_n has exactly $n - 1$ edges where n is the number of vertices in the graph. Figure 2.16 shows a tree with 5 vertices, T_5 .

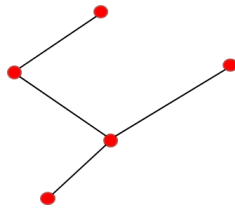


Figure 2.16: tree with 5 vertices, T_5 .

2.2.5 Spanning tree

A spanning tree of an undirected graph is a subgraph, which is also a tree, of a graph that contains all vertices of that graph. A graph may have more than one spanning tree.

2.2.6 Minimum spanning tree

A minimum spanning tree of an undirected weighted graph is one of spanning trees of that graph that has the smallest possible weight. A minimum spanning tree of an undirected unweight graph can be any spanning tree of that graph. Figure 2.17 shows the minimum spanning tree of a graph.

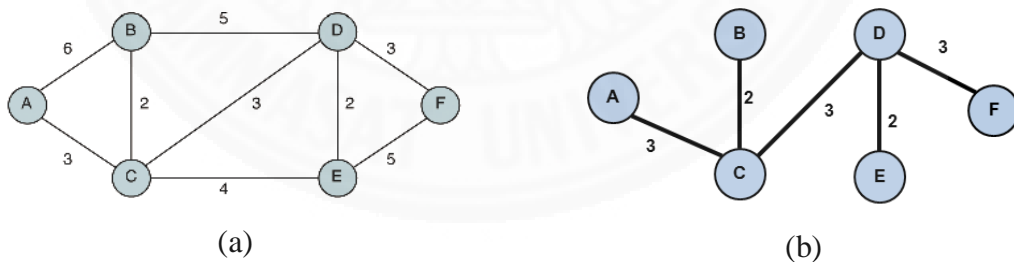


Figure 2.17: (a) undirected weighted graph (b) its minimum spanning tree

A commonly used algorithm to find a minimum spanning tree of a graph is created by Joseph Kruskal in 1956, and can be described as follows:

To find a minimum spanning tree of graph $G = (V_G, E_G)$

Let $minspt = graph(V_{minspt}, E_{minspt})$ where $V_{minspt} = V_G$ and $E_{minspt} = \emptyset$

Let $E' = E_G$ sorted by weight in increasing order

while $|E_{minspt}| \neq |V_{minspt}| - 1$

let $e = \text{first element in } E'$

$E' = E' - e$

$E_{minspt} = E_{minspt} \cup e$

if $minspt$ contains a cycle then

$E_{minspt} = E_{minspt} - e$

endif

endwhile

return $minspt$

Algorithm 2.2: Kruskal's algorithm for finding minimum spanning tree of graph G

2.3 Existing methods of optic disc detection

In this section, different techniques employed to automatically detect the OD inside a retina image are reviewed.

Several OD detection methods are available in literature. Features of OD such as its brightness and round shape are often used to locate it. For example, Tolia and Panas [5], Lalonde [6], Lu et al. [7], and Akram [8] used brightness as the main feature to roughly determine OD position. Sinthanayothin et al. [9], Jelinek [10], and Lupascu et al. [11] recognized the area with highest variance of intensity as OD. Sagar et al. [12] implemented OD localization using Principle Component Analysis (PCA) with pixels in the test image with the highest 5% intensity level and hue value in the yellow color as candidate regions. A template of bright circular region is also used by Osareh et al. [13] to locate OD.

These methods perform quite well with images of healthy retina. However, the methods may not work on images with poor quality resulting from poor lighting condition, unhealthy retina, and/or other artifacts as they share a lot of common features with the OD.

Vessel structure is another feature in the retinal image that can be used to detect OD. It is usually resistant to lighting condition and relatively prominent despite in an unhealthy retina. Hoover and Goldbaum [14] used a technique called fuzzy convergence to determine the origin of the blood vessel network which is OD. A parabola-like model of vessel network is used by Ruggeri et al. [15], and Zhang and Zhao [16] to locate the OD.

Vessel structure and OD's brightness can also be used together for OD location detection. Kavitha and Devi [17] used brightness and convergent point of blood vessels. Abramoff and Niemeijer [18] detected the approximate position of the optic disc using kNN regression with feature vectors including number of vessels, vessels' width, and intensity. Mahfouz and Fahmy [19] and Cao et al. [20] transformed the localization problem into two one dimension problems by projecting the image features, namely, vessel structure, brightness and the size of the OD, onto two perpendicular directions. Youssif et al. [26] used a vessels' direction matched filter which is based on vessels direction and image intensity to locate OD. Mahfouz and Fahmy [19], Youssif et al. [26], and Cao et al [20]'s work have a major disadvantage. Methods may not work well when the orientation of vessel is changed. However, it is robust to noise and artifacts in images.

This work proposes two methods that work even when the images are rotated. The first method combines vessels into a few clusters, then find the centroid of them and used it as the location of OD. The second one uses an improved version of Cao's approach to detect OD even when the image is tilted while still preserve its robustness.

Chapter 3

Methodology

3.1 Approaches

3.1.1 Vessel Clustering

Vessel Clustering is a new algorithm for clustering the main blood vessels in retinal images. Grouping vessels that are relatively close and have similar properties such as intensity and thickness are key concepts in our clustering algorithm. The purposed method aims to increase the correctness of Vessel Clustering, which is used to find the location of optic disc (OD). The algorithm is illustrated in figure 3.1.

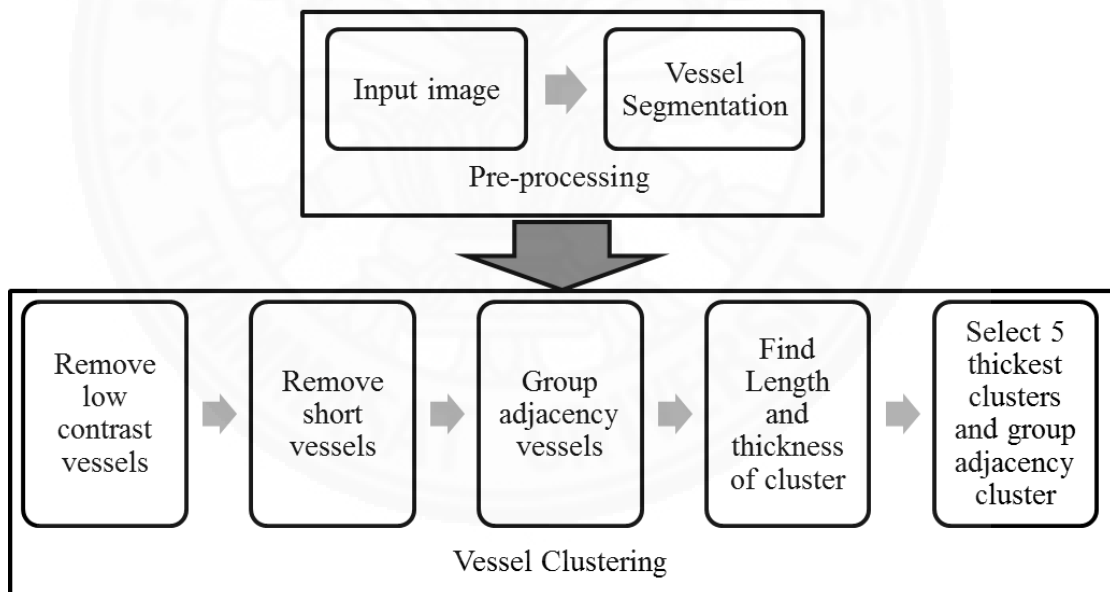


Figure 3.1: Vessel Clustering algorithm

3.1.1.1 Input

Input to the algorithm is a color retinal image and vessels data extracted from it. Figure 3.2 shows segmented vessel of an input image. The segmentation algorithm breaks the vessel network in a retinal image into many segments at bifurcation (one

vessel splits into two) and crossover (two vessel crossing) as shown in figure 3.3. The data is points in Cartesian coordinate. Each point is a point in segment in the image. In addition to x, y position, a point also includes segment ID, thickness of segment at that point and intensity of green channel in the image as shown in table 3.1.

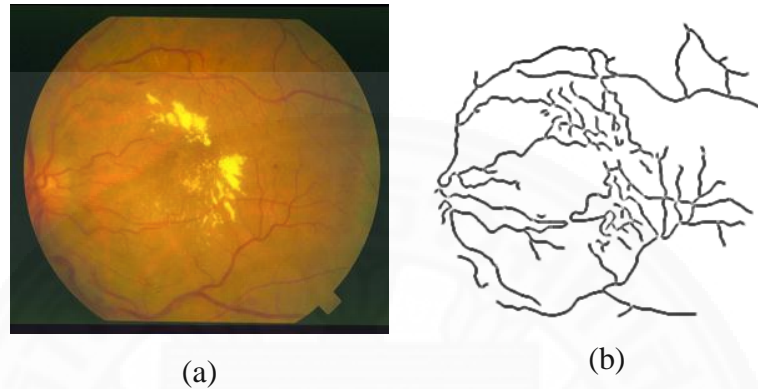


Figure 3.2: (a) input image, (b) segmented vessel

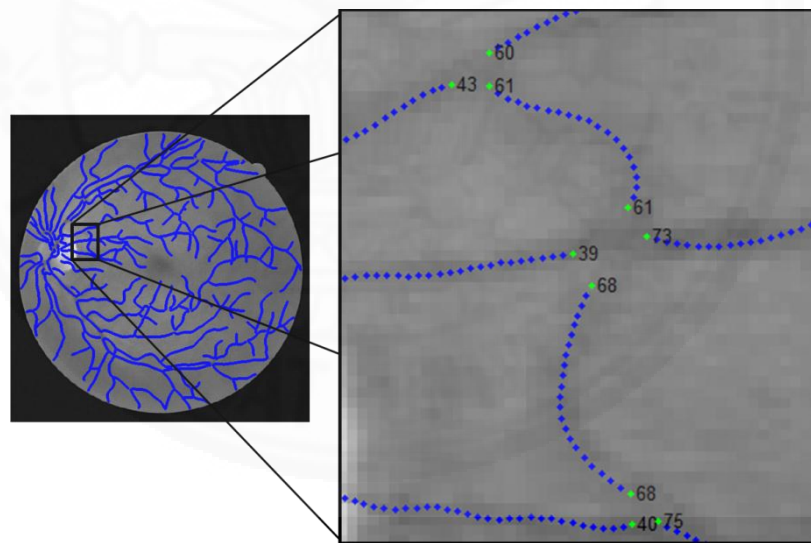


Figure 3.3: Vessel segmentation breaks vessel at junctions. Blue dots are vessel points. The green dots mark segment's end and the numbers are segments' ID.

Table 3.1: A part of input to the algorithm. Each row is a point in a retinal image.

Segment ID	X	Y	Thickness	Green value
1	288.9723	19.1595	3.8472	2
1	288.0513	19.5249	3.8451	39
1	287.2596	20.0747	4.4436	39
1	286.5719	20.7743	3.9126	103
2	285.9628	21.5892	3.163	124
2	285.4069	22.4847	2.5235	124
2	284.8787	23.4264	3.206	115

3.1.1.2 Remove low contrast segment

Contrast of a segment is a key feature. Segments with low contrast usually insignificant and will be removed. Contrast of a segment shows how the segment stands out from background. The process to calculate contrast of each segment is as followed: At each point, an area in input image around the point with dimension $2 \times \text{thickness} + 1 \times 2 \times \text{thickness} + 1$ pixels is selected. The mean of green value in the area is calculated and used as the threshold to identify background pixels in which brighter than vessel pixels. Then, the mean of green value of background pixels is calculated. The difference between green value of the point and the mean of background pixels is kept as difference in green (DG) of the point. Means of DG of all points in a segment is used as contrast of that segment. The threshold of DG value is the difference between mean and standard deviation value of DG values of all points. Segments with lower DG value than the threshold are removed. Let consider an example shown in figure 3.4. Vessel pixels are gray-shaded pixels and the others are background. Numbers are green value of each pixel. Let 45 be the green value of the point. The thickness of the point is 3 so the size of window is 7×7 pixels. The threshold is mean of green value of every point which is 71.87. The mean of background pixels is 82.6. As a result, the DG of this point is 37.6. Figure 3.5 shows a comparison between plotted vessel data before and after this step.

90	97	99	87	76	45	78
94	94	93	86	47	46	76
84	82	85	45	43	80	82
84	82	46	45	45	79	83
79	79	44	44	84	82	85
77	77	44	48	79	79	78
72	46	43	78	77	77	77

Figure 3.4: A square patch centered at a focused point (red) of a vessel which has thickness of 3.

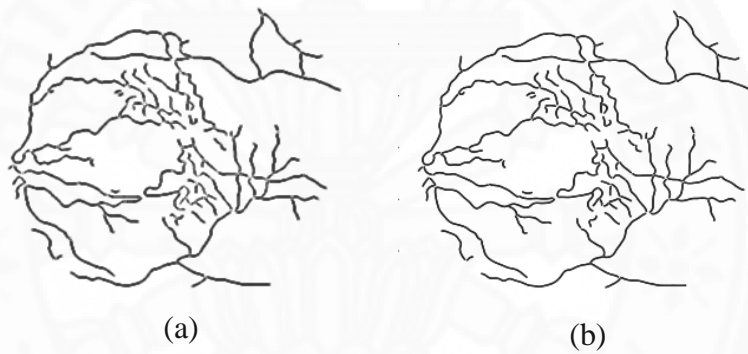


Figure 3.5: (a) segmented vessel, (b) after remove low contrast segments

3.1.1.3 Remove short segments

The length of a segment is also a key feature. Short segments usually are noise created in vessel segmentation step. The length of each segment is measured by consider a segment as a weighted complete graph with points as nodes and distance between nodes as weight of the edge. From the graph, minimum spanning tree is created. And total weight of the tree is length of the segment as illustrate in figure 3.6. Short segments are removed. The length of the vessels to be removed is equal to the difference between mean and 0.25 times standard deviation value of all segment's length. Figure 3.7 shows a comparison between plotted vessel data before and after this step.

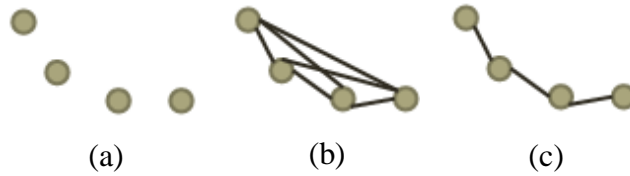


Figure 3.6: (a) points in a segment, (b) complete weighted graph with distance between each pair of points as weight, (c) minimum spanning tree

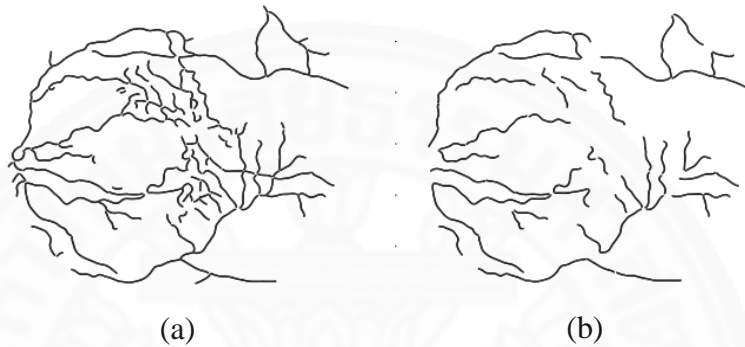


Figure 3.7: (a) after remove low contrast segment, (b) after remove short segments

3.1.1.4 Group adjacent segments

Remaining segments are parts of significant segments. Distance between two segments is the distance between the closest pair of points from those two segments. Distance between every possible pairs of segments is calculated. Then, segments that stay closer than 10 pixels are merged into a cluster. Figure 3.8 shows a comparison between plotted vessel data before and after this step.

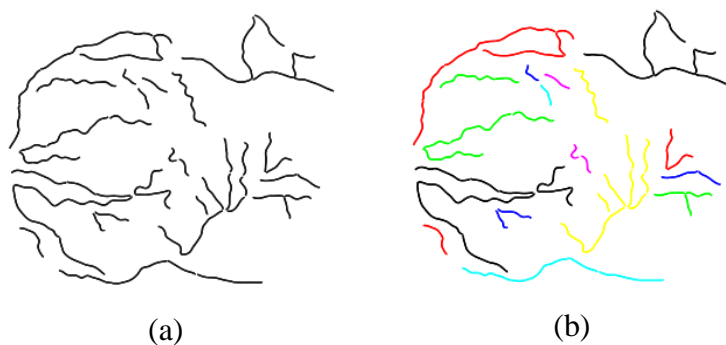


Figure 3.8: (a) after remove short segments, (b) clusters. Segments in same cluster are marked with the same color.

3.1.1.5 Select thickest clusters and group adjacent clusters

Summation of thickness (SoT) of each point in a cluster is used to determine significant clusters. We selected 5 clusters with the most SoT as main clusters. At last, main clusters that stay together closer than 25 pixels are grouped together and become the output of the algorithm. Figure 3.9 shows a comparison between plotted vessel data before and after this step.

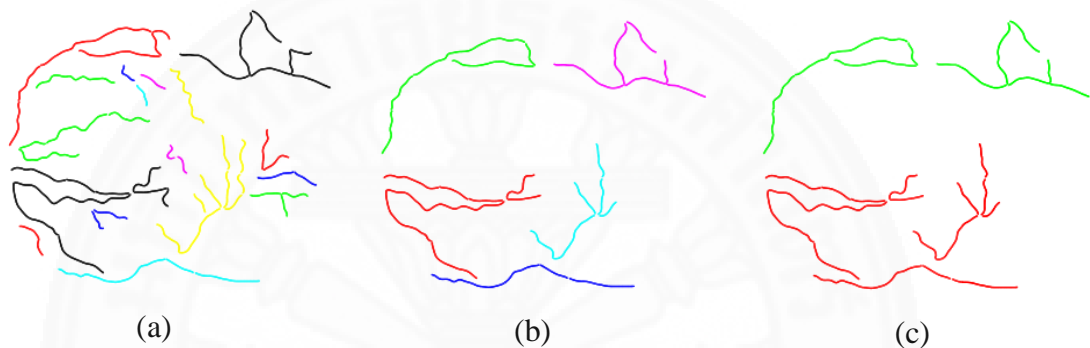


Figure 3.9: (a) clusters, (b) 5 clusters with the most SoT, (c) final clusters. Segments in same cluster are marked with the same color.

3.1.1.6 OD location from final clusters

After final clusters are created, we use the assumption that the point that is closest to all clusters is the center of the OD [22] to conclude the location of OD. Figure 3.10 shows the result of applying the method in [22] to final clusters.

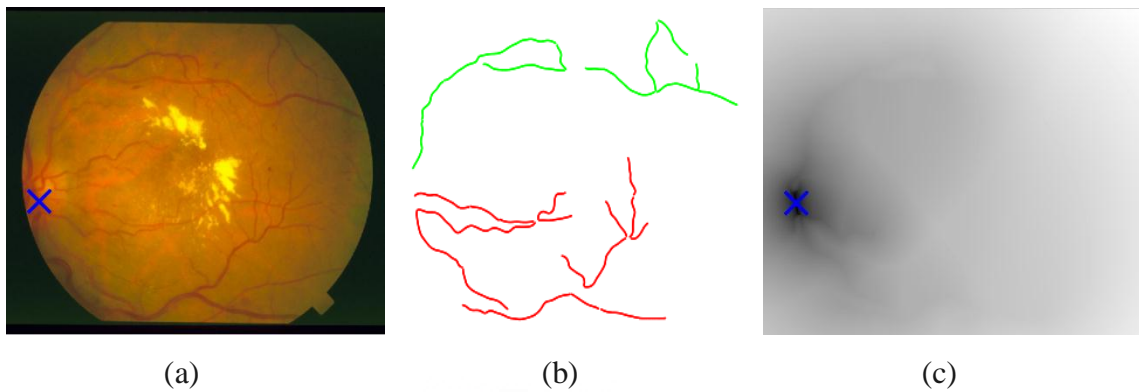


Figure 3.10: OD location from applying [22] to final clusters (a) original image, (b) final clusters, (c) OD location

3.1.2 Rotational 2D Vessel Projection

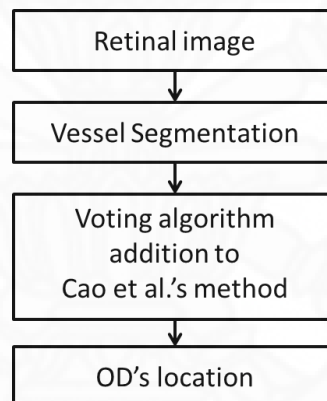


Figure 3.11: The overview of Rotational 2D Vessel Projection

The overview of the method is illustrated in figure 3.11. A grayscale image is obtained by extracting the green channel from the input image. The vessel segmentation algorithm is applied to obtain a binary image of vessels. OD location candidates come from result of Cao et al. [20] algorithm. Using our voting algorithm, the final location of OD in the retinal image can be concluded. The details of each step are presented in the following subsections.

3.1.2.1 Vessel segmentation

Vascular structure is an important feature used by our method. Thus, first we apply the vessel segmentation algorithm to extract vascular network from the grayscale retinal image. To extract the vessel, we perform top-hat filtering on complement of grayscale image with +-shaped structure to highlight vessel region. Then, Otsu's method is used to calculate threshold value of intensity of filtered image and convert it into binary image. Figure 3.12 shows an example of this process.

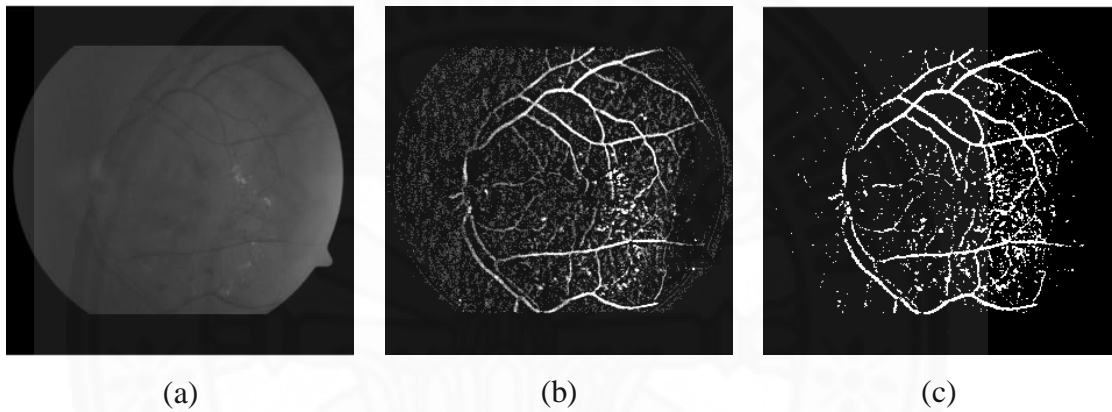


Figure 3.12: Vessel segmentation (a) grayscale image (b) top-hat filtered image (c) vessel image

3.1.2.2 Finding possible OD locations

We used an algorithm proposed by Cao et al. [20] to approximate OD location. Their assumption are, firstly the area around OD contains high density of vertical vessels and contains low density of horizontal vessels, and secondly OD is where the dark vessels lined on top of the bright area of the OD, thus produces high intensity variance. After getting vessel image from vessel segmentation, two morphological opening operations are performed to extract vessels in vertical and horizontal direction from the vessel image.

By subtracting number of vessel pixels in horizontal direction from vertical direction at different value of x 's, horizontal position of OD is found. Then, using

variance of image intensity along vertical line on that horizontal position, location of OD is obtained. Figure 3.13 illustrates Cao et al.'s algorithm.

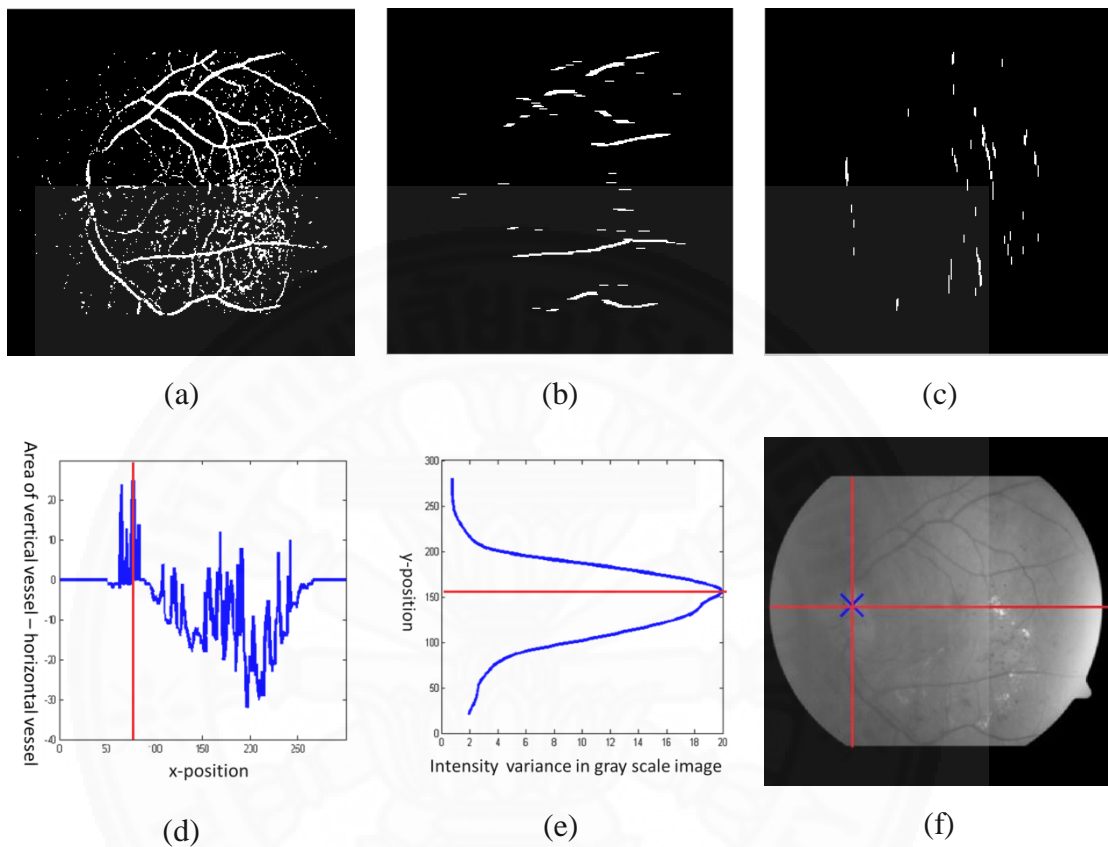


Figure 3.13: Cao et al.'s method (a) segmented vessel, (b) horizontal vessel, (c) vertical vessel, (d) graph of area of vertical vessel minus horizontal vessel at each x-position, (e) variance of intensity along $x=a$ where a is x-position that is the highest peak (red line in (d)), (f) Result

3.1.2.3 Rotational 2D Vessel Projection

Cao et al.'s algorithm described in earlier section alone cannot be used with tilted image because changing in orientation invalidates relationship of vertical and horizontal vessel in their assumption. Figure 3.14 shows the result when the algorithm is used on tilted images.

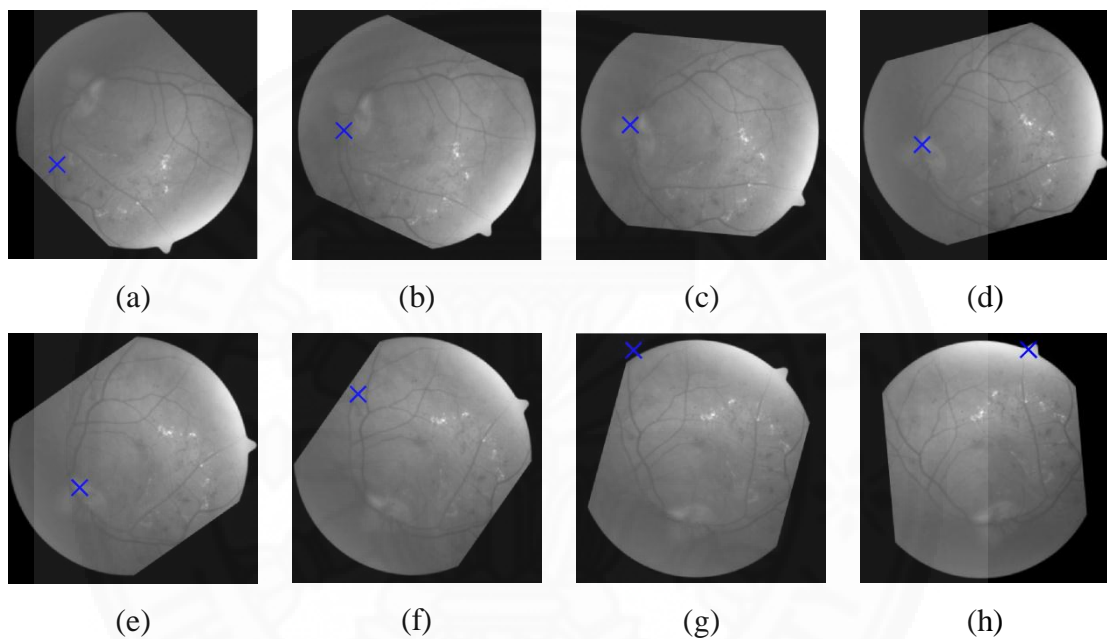


Figure 3.14: OD location from Cao et al.'s method on various angles in degree (a) -45 (b) -25 (c) -5 (d) 15 (e) 35 (f) 55 (g) 75 (h) 95

As shown in figure 3.14, OD location from Cao et al.'s method is depended on the orientation of the image. However, in some appropriate angles as in figure 3.14 (b)-(e), the method can locate OD correctly. Using this knowledge, the proposed method can localize OD on the tilted image by iteratively rotating the input image in small degree per step, for each iteration we repeat the following steps: apply Cao et al.'s algorithm, record the OD location result from Cao et al.'s algorithm, convert the OD coordinate obtained from Cao's back by the same amount of degree to get the tentative OD coordinate, and record a region of a circle centered at this tentative OD coordinate. After the algorithm ends, the regions where the circles are on the most is the location of OD. From the voting result from tentative OD coordinate, the

algorithm can roughly approximate the area in the image containing OD. The whole process may take time due to the number of iterations it needs to perform. To speed it up, the image's width is downsized to 300 pixels while reserving the image's aspect ratio. Then the image is patched to a square shape. In our experiment, the image is rotated from 0 until 180 degrees with increment of 5 degrees every iteration for every iteration. According to our empirical study with 340 images from 4 public databases: DRIVE [2], STARE [21], DIARETDB0 [24] and DIARETDB0 [25], OD in 339 images lie in the middle-height of the image. As a result, we eliminate the locations that do not meet this requirement. Steps of our voting algorithm are described in algorithm 3.1. Figure 3.15 illustrates steps in Rotational 2D Vessel Projection algorithm.

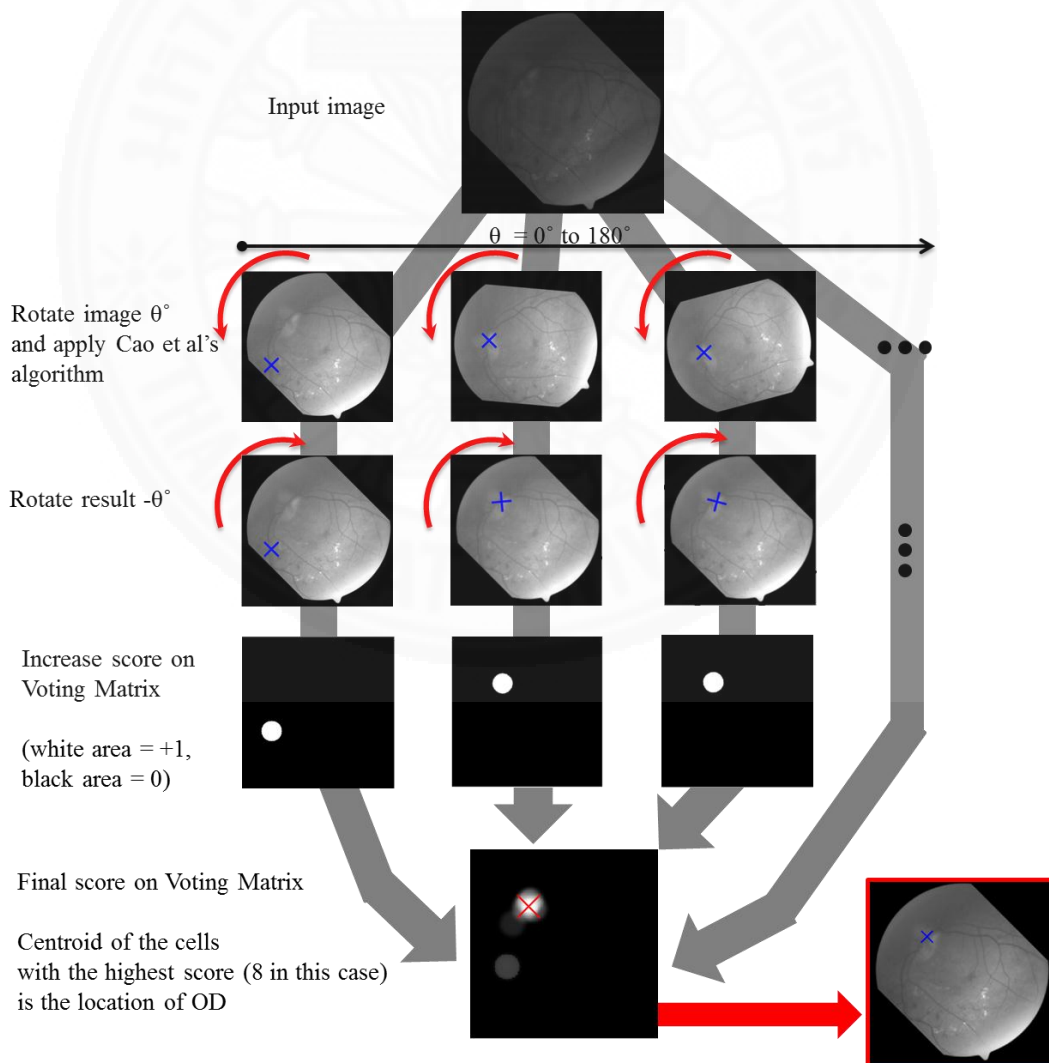


Figure 3.15: Rotational 2D Vessel Projection algorithm

Input: Grayscale retina image I
 $inc = 5$
 r : approximate optic disc radius = 20
 w : image width = 300
 $\theta = 0$
 Voting score matrix $V_{w \times w}$ with initial values 0's
 While $\theta < 180$

1. $I' =$ Rotate I counterclockwise by θ degree with bilinear interpolation
2. $(a, b) =$ OD location returned by applying Cao's algorithm to I' .
3. if $b < w/4$ or $b > 3w/4$ then
 - $\theta = \theta + inc$
 - continue
- endif
4. Convert (a, b) to polar coordinate system originated at $(w/2, w/2)$.
 $(a, b) = (\rho \cos(\theta_2), \rho \sin(\theta_2))$

 By

$$a' = a - w/2$$

$$b' = w/2 - b$$

$$\theta_2 = \tan^{-1}(b'/a')$$

$$\rho = \sqrt{a'^2 + b'^2}$$

5. Rotate (ρ, θ_2) back to original angle
 $\theta_2 = \theta_2 - \theta$
6. Convert (ρ, θ_2) back to Cartesian coordinate system
 $a'' = \rho \cos \theta_2 + w/2$
 $b'' = w/2 - \rho \sin \theta_2$
7. Compute the score and update the voting score
 $V(x, y) = V(x, y) + f(x, y, a'', b'')$

 Where

$$f(x, y, a'', b'') = \begin{cases} 1: (x - a'')^2 + (y - b'')^2 \leq r^2 \\ 0: otherwise \end{cases}$$

8. $\theta = \theta + inc$

 endwhile
 9. $m = \max(V)$
 10. $S = \{(x, y) \in \mathbb{Z}^2 \mid V(x, y) = m\}$
 11. $C = \sum S / |S|$
 return C

Algorithm 3.1: Rotational 2D Vessel Projection.

3.2 Experimental setup

3.2.1 Vessel Clustering

Images from the Structured Analysis of the Retina (STARE) research project [21] are used to evaluate performance of OD localization by the proposed Vessel Clustering algorithm and Rotational 2D Vessel Projection. STARE images are provided by the Shiley Eye Center at the University of California, San Diego, and by the Veterans Administration Medical Center in San Diego. The resolution of all images is 700×605 pixels. There are 81 images in total, taken from both healthy and unhealthy eyes. Images from the healthy group are clear. Vessels and OD are prominent and the background is evenly illuminated. On the other hand, the unhealthy ones' vessels are not clear, often difficult to be identified. They also contain abnormalities; some of them are bright and in circular shape like OD. In some cases, the OD is also severely damaged and loss its round shape. Some images are badly illuminated; cause the vessel and OD to fade which is hard to find even with the human eye. Figure 3.16 shows example of images from STARE database.

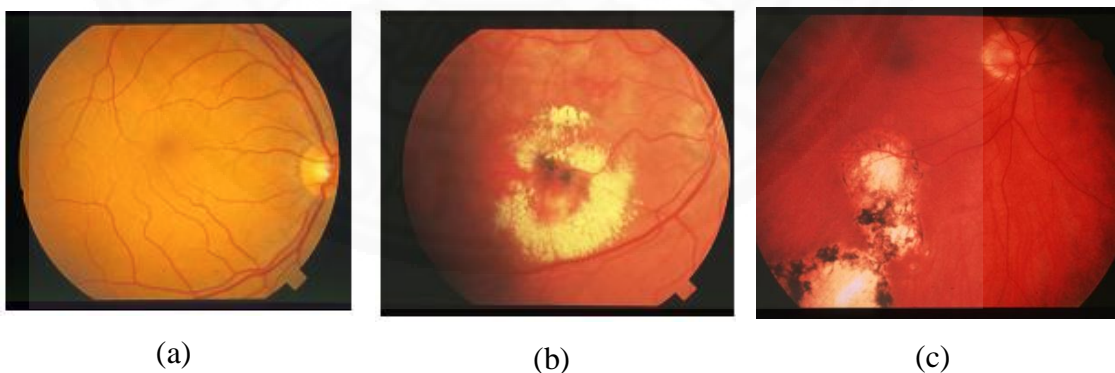


Figure 3.16: Images from STARE database. (a) healthy image, (b), (c) unhealthy images

Using the assumption of that the point that is closest to all cluster is the center of the OD [22], the Vessel Clustering algorithm is evaluated [23] by finding the accuracy of getting the location of OD. This algorithm is implemented and executed

using MATLAB running on a Microsoft Windows 7 desktop with Intel Core i7-4500U @3GHz with 8GB RAM.

3.2.2 Rotational 2D Vessel Projection

In addition to the dataset used with Vessel Clustering described in section 4.1.1, three additional datasets: DRIVE [2], DIARETDB0 [24], and DIARETDB1 [25], which consists of 40, 130 and 89 images, respectively, are used to evaluate performance of Rotational 2D Vessel Projection algorithm. With 81 images from STARE, a total of 340 retinal images were used. DRIVE images were obtained from a diabetic retinopathy screening program in Netherlands. DIARETDB0 and DIARETDB1 are provided by Laboratory of Information Processing, Lappeenranta University of Technology. The emulation of image rotation is also performed to validate the performance of the algorithm on rotated images. To emulate rotation of retinal images, the input images are patched to square-shape and rotated to 8 angles: 2, 29, 58, 75, 103, 123, 142, and 172 degree counter-clockwise using bilinear interpolation. These angles are chosen at random. The algorithm's performance is compared to that of Cao et al. Rotational 2D Vessel Projection algorithm is implemented and executed using MATLAB running on a Microsoft Windows 7 laptop with Intel Core i7-4500U @3GHz with 8GB RAM.

Chapter 4

Result and Discussion

In this chapter we report the result and discussion from two approaches presented in the previous chapter. For evaluation scheme, the detected location of OD is considered correct if it is within the OD region of the groundtruth. Results from two approaches are described in the following sections.

4.1 Results of Vessel Clustering algorithm

Using the proposed Vessel Clustering algorithm to locate the OD in retinal images from STARE dataset, the accuracy and computational time per image obtained are 92.59 % and 76 seconds per image. The examples of location obtained from our first approach are provided in Figure 4.1

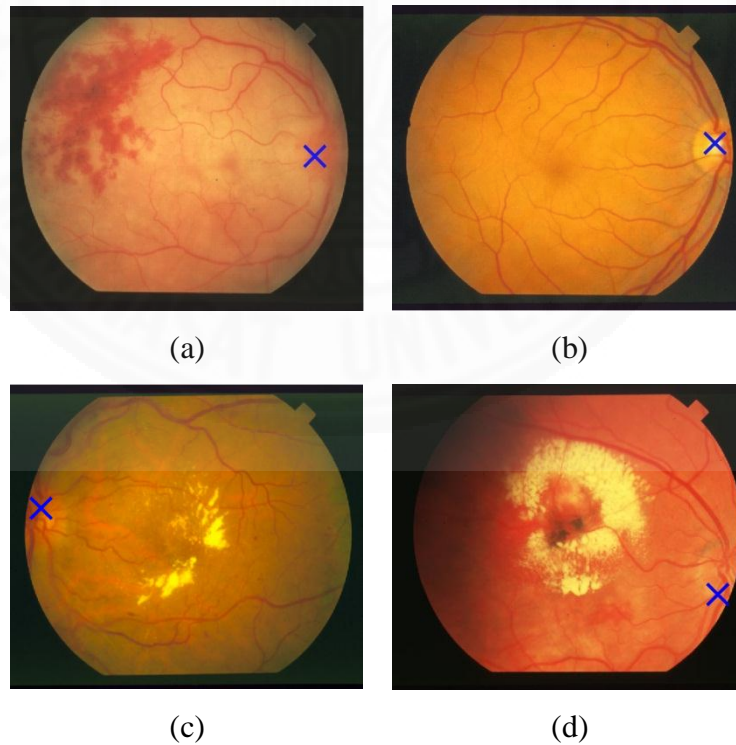


Figure 4.1: The OD location presented as the blue X mark from the Vessel Clustering.

Despite very high performance of the proposed Vessel Clustering algorithm (92.59%), it is still not perfect. There are three causes of imperfection of the algorithm.

First, since this algorithm's input is the segmented vessel data, it relies on vessel segmentation algorithm. The proposed algorithm can resist some amount of vessel segmentation errors such as detecting abnormalities as vessels by removing short and low contrast segments as stated in methodology. However, in some images, the disease damages the eye so much so almost all vessels and OD disappear or are almost completely unrecognizable resulting incorrect results.

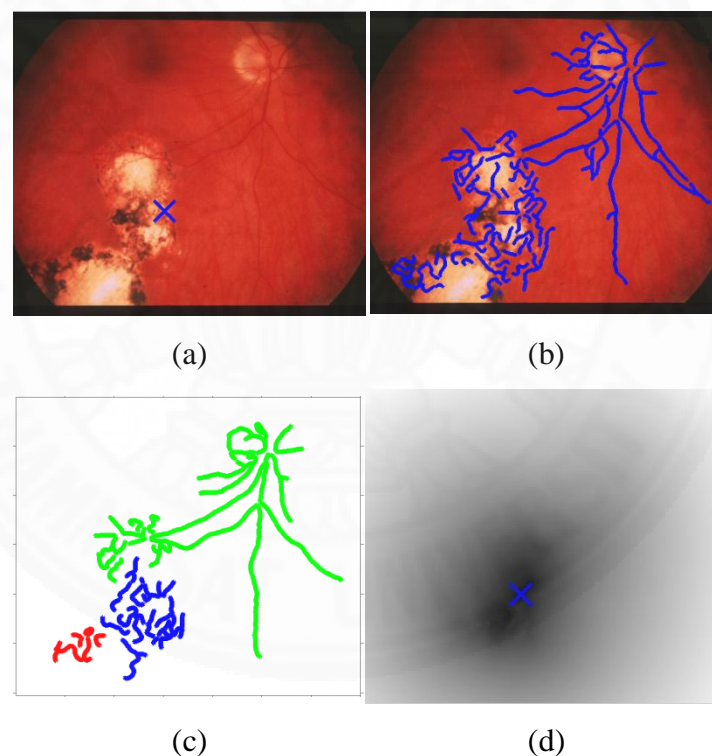


Figure 4.2: Incorrect Vessel Clustering result from segmentation error (a) original images, (b) vessel data, (c) final clusters, (d) OD location

Second, normally, there are few, usually two, main branches of vessel emerged from OD, thus the assumption is the centroid of vessel clusters (branches) is the position of OD. However, in some images, the vessels inside OD are connected into one branch of vessel which makes it impossible for the algorithm to work.

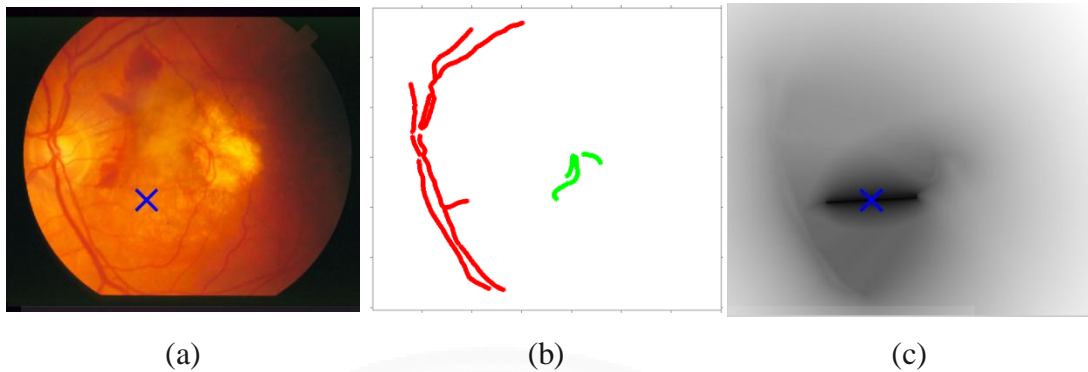


Figure 4.3: Incorrect Vessel Clustering result from a conflict with the assumption (a) original images, (b) final clusters, (c) OD location

Lastly, since this algorithm uses parameters such as the threshold of DG, vessel length and distance between vessel segments to remove noise and grouping vessel segments, then value of these parameters can be imperfect, result in error in clustering steps.

4.2 Results of Rotational 2D Vessel Projection algorithm

For Rotational 2D Vessel Projection, the accuracy of Rotational 2D Vessel Projection algorithm on STARE database is 87.65 % with average computation time of 2.11 seconds per image. Figure 4.4 shows OD locations detected using Rotational 2D Vessel Projection on images from STARE database.

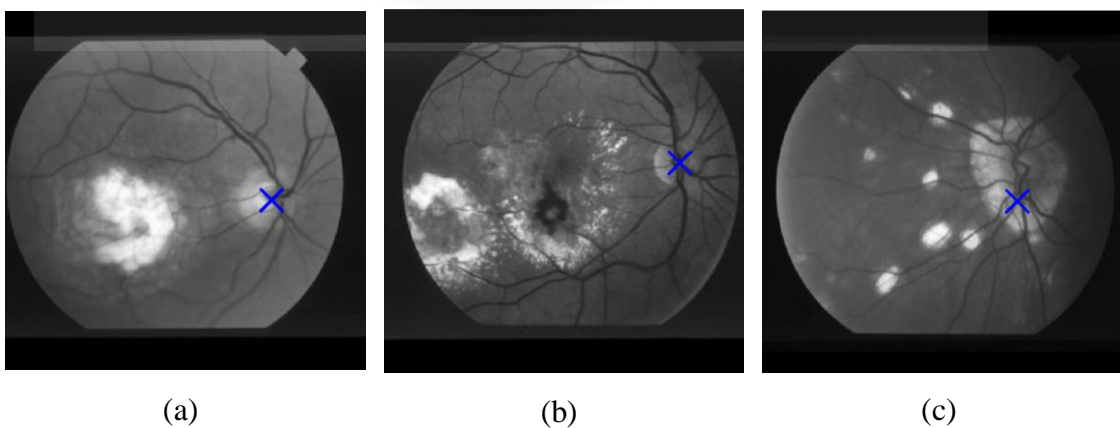


Figure 4.4: Rotational 2D Vessel Projection result. The blue crosses mark the OD.

Additional experiments are performed to evaluate performance on tilted images with DRIVE [2], STARE [21], DIARETDB0 [24] and DIARETDB1 [25] database. In all angles, the method is able to correctly locate the OD with average accuracy 94.93% and average 2.12 seconds computing time. Table 4.1 shows the results of the proposed method compare to the base method --Cao et al.'s. As it was proven experimentally that it is superior to other methods ([9], [13], [14], [15], [19], [26]) in terms of accuracy and computation time. Because our proposed method outperforms Cao's, it is thus better than those methods as well.

Table 4.1: Accuracy comparison in percent

Tilt angle (degree)	DRIVE		STARE		DIARETDB0		DIARETDB1	
	Cao's	RVP	Cao's	RVP	Cao's	RVP	Cao's	RVP
2	100	100	71.60	82.72	91.36	97.69	92.59	98.88
29	87.5	100	64.20	83.95	88.89	97.69	90.12	98.88
58	15	100	32.10	86.42	35.80	96.92	23.46	98.88
75	12.5	100	25.93	83.95	17.28	96.15	4.94	98.88
103	10	100	24.69	85.19	13.58	96.15	2.47	98.88
123	20	100	28.40	85.19	37.04	96.15	19.75	98.88
142	80	100	60.49	85.19	79.01	96.92	82.72	98.88
172	100	100	60.49	86.42	85.19	97.69	87.65	98.88
Average	53.125	100	45.99	84.88	56.02	96.92	50.46	98.88

Figure 4.5 shows examples of correctly located OD using Rotational 2D Vessel Projection. As shown in Table 4.1, effect of rotation on the performance of the algorithm is very small, thus the algorithm is robust even when retinal image is tilted.

Despite high performance, there are some errors. The errors are due to three factors: uneven illumination, distractions from pathological changes, and uneven background level. Figure 4.6 shows examples of images which OD is located incorrectly. Figure 4.6a, vessel segmentation failed to extract vessel structure correctly due to uneven illumination which affect segmentation of vessels. In figure 4.6b, distraction is falsely segmented as vessel, while in figure 4.6c, the distraction

causes high intensity variance near vessels. Figure 4.6d shows an image with uneven background level, cause high intensity variance at the edge of retina.

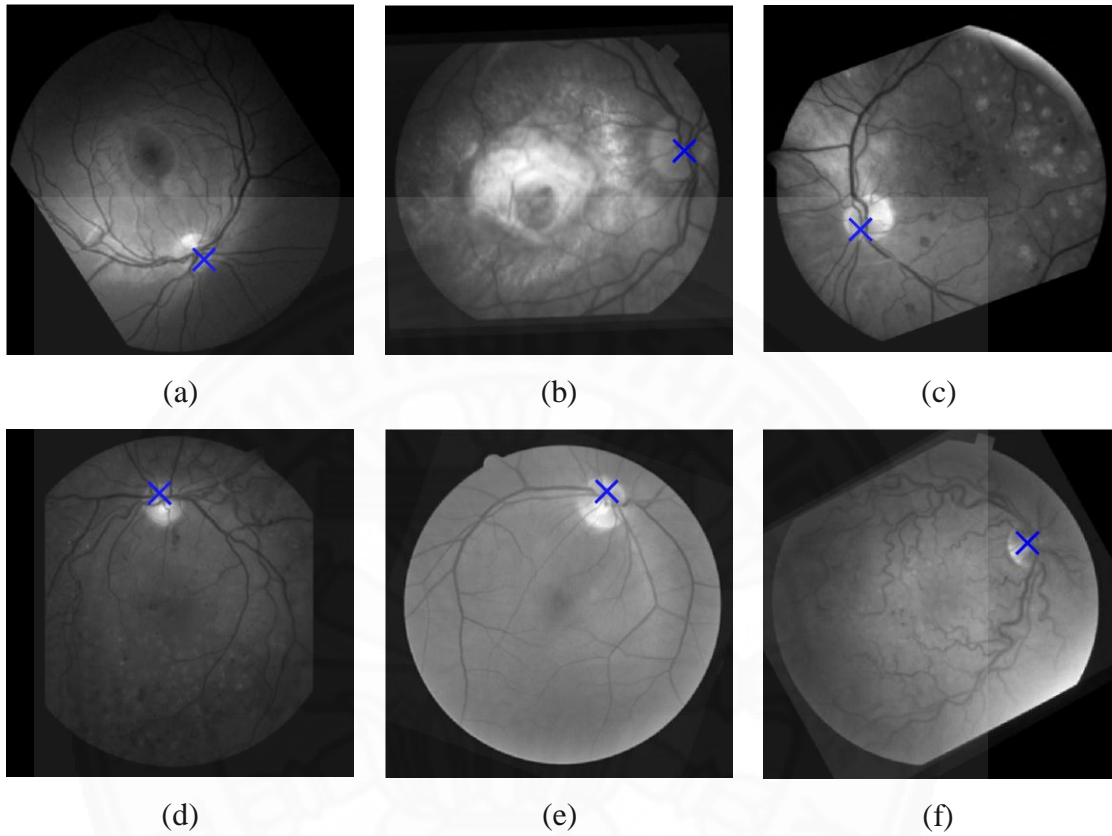


Figure 4.5: Examples of successful cases

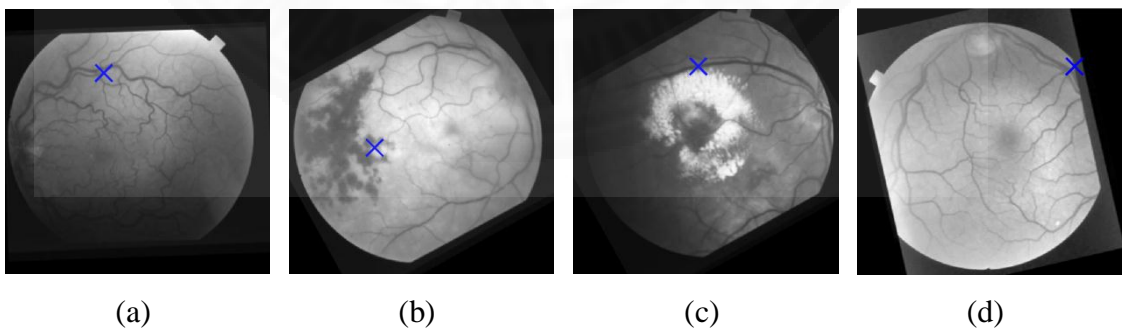


Figure 4.6: Examples of unsuccessful cases (a) uneven illumination (b) distraction detected as vessels (c) distraction with high contrast near thick vessel (d) uneven background level

Chapter 5

Conclusion and Recommendation

In this thesis, we proposed a new clustering algorithm which is designed specifically to group the main blood vessels into clusters. These clusters represent main branches of blood vessels. The proposed method uses length, intensity, and thickness of the vessels to detect main vessels and uses distances between vessels to cluster two vessels or clusters that are close together. With assumption that the centroid of the cluster is the location of OD, we find that the accuracy of the algorithm in getting the location of the OD of images from STARE dataset is 92.59% with 76 seconds computation time.

Furthermore, inspired by projection idea and its limitations, a new method for optic disc localization is also presented. In this work, a majority voting of OD location reported based on vessel structure and intensity variance in various angles is used. The method is able to localize the OD accurately despite rotational angle of the image. The 94.93% accuracy on 8 random angles of 340 images from 4 public datasets (2,720 images in total) with only slightly more than 2 seconds computing time per image shows that the algorithm is very robust and fast.

Although the performance of both algorithms is high, there are still rooms for improvements. Both algorithms could use better algorithms for determining value of parameters. Computation time of Vessel Clustering can be improved by optimizing the step of measuring distance between segments. Rotational 2D Vessel Projection could use better vessel segmentation algorithm that capable to detect and remove distraction and can solve the problem of uneven background. It is also possible to use the algorithm to specifying region of interest to detect boundary of retina in an image.

References

- [1] *Picture of the Eyes*. Digital image. *The Eyes (Human Anatomy)_ Diagram, Optic Nerve, Iris, Cornea, Pupil, & More*. WebMD, LLC., 2013. Web. 19 May 2015.
- [2] J.J. Staal, M.D. Abramoff, M. Niemeijer, M.A. Viergever, B. van Ginneken, "Ridge based vessel segmentation in color images of the retina", *IEEE Transactions on Medical Imaging*, 2004, vol. 23, pp. 501-509.
- [3] *Normal Vision and the Same Scene as Viewed by a Person with Glaucoma*. Digital image. *Facts About Glaucoma | National Eye Institute*. The National Eye Institute (NEI), n.d. Web. 19 May 2015.
- [4] Burk, Scott, John S. Cohen, and Harry Quigley. *Healthy Optic Nerve and Optic Nerve in Eye with Glaucoma*. Digital image. *Optic Nerve Cupping | Glaucoma Research Foundation*. Glaucoma Research Foundation, 21 Aug. 2012. Web. 19 May 2015.
- [5] Y.A. Tolias, S.M. Panas. (1998). A fuzzy vessel tracking algorithm for retinal images based on fuzzy clustering, *IEEE Transactions on Medical Imaging* 17-2(1998) 263-273.
- [6] M. Lalonde, M. Beaulieu, L. Gagnon. (2001). Fast and robust optic disc detection using pyramidal decomposition and Hausdorff-based template matching, *IEEE Transactions on Medical Imaging* 20-11(2001) 1193-1200.
- [7] S. Lu, J. Liu, J.H. Lim, Z. Zhang, N.M. Tan, W.K. Wong, T.Y. Wong. (2010). Automatic optic disc segmentation based on image brightness and contrast, *In SPIE Medical Imaging, International Society for Optics and Photonics* 76234J(2010) 1-8.

- [8] U.M. Akram, S.A. Khan. (2012). Automated detection of dark and bright lesions in retinal images for early detection of diabetic retinopathy, *Journal of Medical Systems* 36(2012) 3151-3162.
- [9] C. Sinthanayothin, J.F. Boyce, H.L. Cook, T.H. Williamson. (1999) Automated localisation of the optic disc, fovea, and retinal blood vessels from digital colour fundus images, *British Journal of Ophthalmology* 83-8(1999) 902-910.
- [10] H.F. Jelinek, C. Depardieu, C. Lucas, D.J. Cornforth, W. Huang, M.J. Cree. (2005). Towards vessel characterization in the vicinity of the optic disc in digital retinal images, *The Image and Vision Computing Conference*.
- [11] Lupascu, C.A.; Tegolo, D.; Di Rosa, L. (2008). Automated Detection of Optic Disc Location in Retinal Images, *Computer-Based Medical Systems, 2008. CBMS '08. 21st IEEE International Symposium on* , vol., no., pp.17,22
- [12] A.V. Sagar, S. Balasubramanian, V. Chandrasekaran. (2007). Automatic Detection of Anatomical Structures in Digital Fundus Retinal Images, *MVA IAPR Conference on Machine Vision Applications (2007)* 483-486.
- [13] A. Osareh, M. Mirmehdi, B. Thomas, R. Markham. (2002). Colour morphology and snakes for optic disc localisation, *The 6th Medical Image Understanding and Analysis Conference, BMVA Press* (2002) 21-24.
- [14] Hoover, A.; Goldbaum, M. (2003). Locating the optic nerve in a retinal image using the fuzzy convergence of the blood vessels, *Medical Imaging, IEEE Transactions on* , vol.22, no.8, pp.951,958
- [15] Ruggeri, A.; Forrachia, M.; Grisan, E. (2003). Detecting the optic disc in retinal images by means of a geometrical model of vessel network, *Engineering in Medicine and Biology Society, 2003. Proceedings of the 25th Annual International Conference of the IEEE* , vol.1, no., pp.902,905 Vol.1

- [16] Zhang, D.; Zhao, Y. (2014). Novel Accurate and Fast Optic Disc Detection in Retinal Images with Vessel Distribution and Directional Characteristics, *Biomedical and Health Informatics, IEEE Journal of* , vol.PP, no.99, pp.1,1
- [17] Kavitha, D.; Shenbaga Devi, S. (2005). Automatic detection of optic disc and exudates in retinal images, *Intelligent Sensing and Information Processing, 2005. Proceedings of 2005 International Conference on* , vol., no., pp.501,506
- [18] Abramoff, Michael D.; Niemeijer, M. (2006). The automatic detection of the optic disc location in retinal images using optic disc location regression, *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE* , vol., no., pp.4432,4435,
- [19] Mahfouz, A.E.; Fahmy, A.S. (2009). Ultrafast optic disc localization using projection of image features, *Image Processing (ICIP), 2009 16th IEEE International Conference on* , vol., no., pp.665,668
- [20] Qinghui Cao, Jianli Liu, Qihong Zhao. (2013). Fast Automatic Optic Disc Localization in Retinal Images, *Image and Graphics (ICIG), 2013 Seventh International Conference on* , vol., no., pp.827,831
- [21] Michael Goldbaum. (2003). *The STARE Project* Retrieved May, 2013, from <http://www.ces.clemson.edu/~ahoover/stare/>
- [22] Bornschlegel T., Muangnak N., Aimmanee P., Makhanov S. S., and Uyyanonvara B. (2012). Finding curvess convergence using delaunay search. *In Proceedings of The First ASIAN Conference on Information Systems (ACIS 2012) [CDROM]*

- [23] Maria Halkidi, Yannis Batistakis and Michalis Vazirgiannis. (2001). Clustering Validation Techniques. *Journal of Intelligent Information Systems*, 17:107–145
- [24] Kauppi, T., Kalesnykiene, V., Kamarainen, J.-K., Lensu, L., Sorri, I., Uusitalo, H., Klviinen, H., Pietil, J., (2006). DIARETDB0: Evaluation Database and Methodology for Diabetic Retinopathy Algorithms. Retrieved January 11, 2015
- [25] Kauppi, T., Kalesnykiene, V., Kamarainen, J.-K., Lensu, L., Sorri, I., Raninen A., Voutilainen R., Uusitalo, H., Kälviäinen, H., Pietil, J. (2007). DIARETDB1 diabetic retinopathy database and evaluation protocol. *Medical Image Understanding and Analysis, 2007 Eleventh International Conference on*, vol., no., pp
- [26] A.A.H. Abdel-Razik Youssif, A.Z. Ghalwash, A.A.S. Abdel-Rahman Ghoneim. (2008). Optic Disc Detection From Normalized Digital Fundus Images by Means of a Vessels' Direction Matched Filter. *Medical Imaging, IEEE Transactions on* , vol.27, no.1, pp.11,18, Jan. 2008

Publication

Tangseng, P., Muangnak, N., Aimmanee, P., Mekanov, S. S., & Uyyanonvara, B. (2013). A Study of Retinal Blood Vessel Clustering for Finding the Main Vessel Convergence. *Information Systems (ACIS), 2013 The Second Asian Conference on*, 31 Oct – 2 Nov. 2013



Appendices



Appendix A

Vessel Clustering MATLAB source code

findDG.m

```
function findDG

close all, clear all;
vessel_path = '../input/vessels/';

strDir      = strcat(vessel_path, 'im*.mat');
listFile    = dir(strDir);
listFileCell = [{listFile.name}'];
fileIDList  = strtok(listFileCell, 'im*.');
listFileCell = strcat(vessel_path, listFileCell);

for i = 1 : size(fileIDList, 1)
    imageNumber = i;
    imageTxt = fileIDList{imageNumber, 1};
    fileName = listFileCell{imageNumber, 1};

    imname = strcat('im', imageTxt);
    RawData = load(strcat('../output/', imname, '.csv'));
    dataSize = size(RawData);
    rawNumberOfPoints = dataSize(1);
    oimg = imread(strcat('../input/original/', imname, '.jpg'));
    oimg_green = oimg(:, :, 2);
    [height width depth] = size(oimg_green);

    DG = NaN(1, rawNumberOfPoints);

    window = NaN(20);
    for i=1:rawNumberOfPoints
        points = RawData(i, [2,3,4,5]);
        points = round(points);
        x = points(1);
        y = points(2);
    end
end
```

```

        thickness = points(3);
        green = points(4);

        if(isnan(thickness)) thickness=5; end

        xo = x-thickness;
        yo = y-thickness;
        xf = x+thickness;
        yf = y+thickness;

        if(xo<1) xo = 1;end
        if(yo<1) yo = 1;end
        if(xf>width) xf = width;end
        if(yf>height) yf = height;end

        window = oimg_green((yo:yf),(xo:xf));

        thres = mean(mean(window));
        meanBG = mean(window(window > thres));
        DG(i) = abs(green - meanBG);
    end
    RawData(:,6) = DG(:);

    imname = strcat('../output/',imname);
    csvwrite (strcat(imname,'DG.csv'),RawData);

end
end

```

vesselClustering.m

```

function vesselClustering(imnum)
%clear all;
%imnum =

if(imnum < 10)
    imname = strcat('im000',int2str(imnum));
elseif(imnum < 100)

```

```

        imname = strcat('im00',int2str(imnum));
elseif(imnum<1000)
        imname = strcat('im0',int2str(imnum));
end

RawData = load(strcat(imname,'DG.csv'));

dataSize = size(RawData);
rawNumberOfPoints = dataSize(1);
rawGreen = RawData(:,6);

% find threshold
[mu sigma] = normfit(rawGreen);
%remove junk
nOfVessel = RawData(end,1);
gOfVessel = zeros(1,nOfVessel);
j=1;
%Summation of green value of each vessel
while (j<= rawNumberOfPoints)
        gOfVessel(RawData(j,1)) = gOfVessel(RawData(j,1)) + rawGreen(j);
        j = j + 1;
end
%Find number of points of each vessel
nOfPoints = zeros(1,nOfVessel);
for i=1:nOfVessel
        nOfPoints(i) = histc(RawData(:,1),i);
end
gOfVessel = gOfVessel./nOfPoints;
%mark the vessels that green outside the range
for i=1:nOfVessel
        if(gOfVessel(i)<mu-sigma)
                gOfVessel(i) = -1;
        end
end
%delete the vessels that green outside the range
for j = 1:nOfVessel
        if(gOfVessel(j)==-1)
                i=1;

```

```

        while (i<=rawNumberOfPoints)
            if (RawData (i,1)==j)
                RawData (i,:)=[];
                i=i-1;
                rawNumberOfPoints = rawNumberOfPoints-1;
            end
            i=i+1;
        end
    end
end

dataSize = size (RawData);
rawNumberOfPoints = dataSize (1);

%find number of vessel
vnum=1;
oldVnumList = unique (RawData (:,1));
pointNum=1;
i=1;
while i<=numel (oldVnumList)
    if (pointNum > rawNumberOfPoints)
        break;
    elseif (RawData (pointNum,1)==oldVnumList (i))
        RawData (pointNum,1)=vnum;
        pointNum = pointNum+1;
    else
        i=i+1;
        vnum=vnum+1;
    end
end
end
numberOfVessel = vnum;

%find number of points in a vessel
nOfPointsInVessel = [];
for i=1:numberOfVessel
    nOfPointsInVessel (i) = histc (RawData (:,1),i);
end

%Find length of each vessel

```

```

vesselLength = NaN(numberOfVessel,1);
i=1;
for vnum=1:numberOfVessel
    vesselPoints = zeros(1,2);
    while(i<=rawNumberOfPoints)
        if(RawData(i,1)==vnum)
            point = [RawData(i,2) RawData(i,3)] ;
            vesselPoints = cat(1,vesselPoints,point);
            i = i+1;
        elseif(RawData(i,1)>vnum)
            vesselPoints(1,:)=[];
            vesselLength(vnum) = getVesselLength(vesselPoints);
            break;
        end
    end
end
vesselPoints(1,:)=[];
vesselLength(vnum) = getVesselLength(vesselPoints);
vesselNumberList = (1:numberOfVessel);

[mu sigma] = normfit(vesselLength);
threshold = mu-sigma/2

%mark the vessel to be deleted
for(i=1:numberOfVessel)
    if(vesselLength(i)<threshold)
        vesselNumberList(i)=-1;
    end
end

thresholdedNumberOfPoints = rawNumberOfPoints;
thresholdedVesselLength = vesselLength;
thresholdedNumberOfVessel = numberOfVessel;
thresholdedData = RawData;

%delete shorter-than-threshold vessel
for(j = 1:numberOfVessel)
    if(vesselNumberList(j)==-1)
        i=1;

```

```

        while (i<=thresholdedNumberOfPoints)
            if (thresholdedData (i,1)==j)
                thresholdedData (i,:)=[];
                i=i-1;
                thresholdedNumberOfPoints =
thresholdedNumberOfPoints-1;
            end
            i=i+1;
        end
    end
end
j=1;

%delete length of shorter-than-threshold vessel
while (j<=thresholdedNumberOfVessel)
    if (thresholdedVesselLength (j)<threshold)
        thresholdedVesselLength (j)=[];
        nOfPointsInVessel (j)=[];
        j=j-1;
        thresholdedNumberOfVessel = thresholdedNumberOfVessel -1;
    end
    j=j+1;
end

%vessel is 3D matrix where
%vessel (i,j,1) means x value of jth point of ith vessel and
%vessel (i,j,2) means y value of jth point of ith vessel

vessel = zeros (thresholdedNumberOfVessel,max (nOfPointsInVessel),5);
vessel (:,:,)=NaN;
n=1;
i=1;
for (j=1:thresholdedNumberOfPoints)
    vessel (i,n,1) = thresholdedData (j,2);
    vessel (i,n,2) = thresholdedData (j,3);
    vessel (i,n,3) = thresholdedData (j,4);
    vessel (i,n,4) = thresholdedData (j,5);
    vessel (i,n,5) = thresholdedData (j,6);
    if (n==nOfPointsInVessel (i))

```

```

        i=i+1;
        n=0;
    end
    n=n+1;
end

%find distance between every point to every vessel

%distanceMatrix is an NxMxK matrix whern N and K is the number of
vessel
%and M is maximum length of vessel
%The distanceMatrix(i,j,k) represents the distance between jth point
of ith vessel to kth vessel
%The distance is calculate by euclidean distance
%Since I can't use pdist effectively (for now), I created a function
distance2D(x1,y1,x2,y2) to calculate distance.

distanceMatrix =
NaN(thresholdedNumberOfVessel,max(nOfPointsInVessel),thresholdedNumbe
rOfVessel);
%i,l = vessel number
%j,m = point number in a vessel
for(i=1:thresholdedNumberOfVessel)
    for(j=1:nOfPointsInVessel(i))
        for(l=1:thresholdedNumberOfVessel)
            d= zeros(nOfPointsInVessel(l),1);
            for(m=1:nOfPointsInVessel(l))
                d(m) =
distance2D(vessel(i,j,1),vessel(i,j,2),vessel(l,m,1),vessel(l,m,2));
            end
            distanceMatrix(i,j,l) = min(d);
        end
    end
end
disBTWvessel = zeros(thresholdedNumberOfVessel);
%i,k = vessel numbet
%j = point number
for(i=1:thresholdedNumberOfVessel)
    for(k=1:thresholdedNumberOfVessel)

```



```

        if(i==k)
            disBTWvessel(i,k) = NaN;
        else
            disBTWvessel(i,k) = min(distanceMatrix(i,:,k));
        end
    end
end

thresholdDis = 15;
pairList = [0,0];

%make a list of pairs of vessel that the distance between them is
less than 10
%the list's name is pairList
for(i=1:thresholdedNumberOfVessel)
    for(j=1:i)
        if(disBTWvessel(i,j)<thresholdDis)
            pair = [i,j];
            pairList = cat(1,pairList,pair);
        end
    end
end
pairList(1,:)=[]

tmp = pairList;
clusterNumber = 1;
N=numel(tmp);

%make clusters
for i=1:N
    if(tmp(i)~=NaN)
        cluster=tmp(i);
        c=1;
        while(c<=numel(cluster))
            reference = cluster(c);
            j=1;
            while(j<=N)
                if(tmp(j)==reference) %found
                    if(j<=N/2)

```

```

        [reference tmp(j+N/2)];
        cluster = [cluster tmp(j+N/2)];
        tmp(j,:) = NaN;
    else
        [reference tmp(j-N/2)];
        cluster = [cluster tmp(j-N/2)];
        tmp(j-N/2,:) = NaN;
    end
end
    end
    j=j+1;
end
    c=c+1;
end
    if(numel(cluster)>1)
        clusterList{clusterNumber}=unique(cluster);
        clusterNumber=clusterNumber+1;
    end
end
end

maxEl = 0;
for i=1:numel(clusterList)
    [row num] = size(clusterList{i});
    if(maxEl < num)
        maxEl = num;
    end
end

clusterTable = NaN(numel(clusterList),maxEl);

for i=1:numel(clusterList)
    [row num] = size(clusterList{i});
    for j=1:num
        clusterTable(i,j) = clusterList{i}(1,j);
    end
end
clusterTable

pairedVesselNumberList = unique(pairList);

```

```

%create list of number of single vessel. => singleVesselNumberList
singleVesselNumberList = [1:thresholdedNumberOfVessel];
for(i=1:numel(pairedVesselNumberList))
    j=1;
    while(j<=numel(singleVesselNumberList))
        if(pairedVesselNumberList(i)==singleVesselNumberList(j))
            singleVesselNumberList(j)=[];
        end
        j=j+1;
    end
end
singleVesselNumberList

finalAnswer = zeros(1,6);
size(finalAnswer);

nOfVessel =
numel(pairedVesselNumberList)+numel(singleVesselNumberList);

%put single vessels into final answer
%i = single vessel's number
%point = [vesselNumber,x,y,thickness,length]
finalVesselNumber = 1;
[row col depth] = size(vessel);
for(i=1:numel(singleVesselNumberList))
    for(j=1:thresholdedNumberOfVessel)
        if(j==singleVesselNumberList(i))
            for(k=1:nOfPointsInVessel(j))
                point = [finalVesselNumber
reshape(vessel(j,k,:), [1,depth])];
                finalAnswer = cat(1,finalAnswer,point);
            end
        end
    end
    finalVesselNumber=finalVesselNumber+1;
end

[row col]=size(clusterTable);

```

```

nOfClusters = row;
%put clusters into final answer
%i = cluster's number
%j = number of vessel inside the ith cluster
for(i=1:row)
    for(j=1:col)
        vnum = clusterTable(i,j);
        if(isnan(vnum))
            break
        else
            for(k=1:nOfPointsInVessel(vnum))
                point = [finalVesselNumber
                reshape(vessel(vnum,k,:), [1,depth])];
                finalAnswer = cat(1,finalAnswer,point);
            end
        end
        finalVesselNumber=finalVesselNumber+1;
    end
end
finalAnswer(1,:)=[];

finalVesselNumber=finalVesselNumber-1;

%clusterNoOfVessel

%Since the final answer consists of many single vessels first then
many
%clusters later, we gonna make a list of length of clusters by put
the
%length of single vessels to the list first then find and put the
length
%of clusters.

%put the length of single vessels to the list
clusterLengthList = NaN(1,finalVesselNumber);
for i=1:numel(singleVesselNumberList)

clusterLengthList(i)=thresholdedVesselLength(singleVesselNumberList(i
));

```

```

end
start = i;
%find and put the length of clusters
for n=1:nOfClusters
    aCluster = clusterTable(n,:);
    aCluster = aCluster(~(isnan(aCluster)));
    nOfVessel = numel(aCluster);
    clusterGraph = zeros(nOfVessel);
    for i=1:nOfVessel
        for j=1:i
            if(i~=j)
                clusterGraph(i,j) =
disBTWvessel(aCluster(i),aCluster(j));
            end
        end
    end
    cSparse = sparse(clusterGraph);
    [ST,pred] = graphminspantree(cSparse);

    clusterLengthList(start+n) = sum(sum(full(ST)))+
sum(thresholdedVesselLength(aCluster));
end
clusterLengthList

%find summation of thickness in each cluster.

data = finalAnswer;
dataSize = size(data);
numOfCluster = data(end,1);

%prepare some data
[nOfCluster nOfVessel] = size (clusterTable);
tmp = NaN(numel(singleVesselNumberList),nOfVessel);
tmp(:,1) = singleVesselNumberList;
clusterTable2 = [tmp ;clusterTable];

clusterNumberList = data(1:dataSize,1)';
clusterXList = data(1:dataSize,2)';
clusterYList = data(1:dataSize,3)';

```

```

clusterNumberOfPoints = dataSize(1);
clusterThickness = data(1:dataSize,4)';
cnum = 1;
thickSum = 0;
count=0;

%Make a list of summation of thickness of each clusters =>
clusterThicknessList
for i=1:clusterNumberOfPoints
    if(clusterNumberList(i)==cnum)
        if(~isnan(clusterThickness(i)))
            thickSum = thickSum + clusterThickness(i);
        end
    else
        clusterThicknessList(cnum)=thickSum;
        count=0;
        thickSum = clusterThickness(i);
        cnum = clusterNumberList(i);
    end
end
clusterThicknessList(cnum)=thickSum;

mainClusterThicknessList = clusterThicknessList;
mainClusterLengthList = clusterLengthList;

%find thickness threshold ()
tmp = [];
tmp = sort(clusterThicknessList,'descend');
nOfElement = numel(tmp);

if(nOfElement>=6)
    thicknessThreshold = tmp(6)
    i=1;
    while i<=numel(mainClusterThicknessList)
        if(mainClusterThicknessList(i)<=thicknessThreshold)
            mainClusterThicknessList(i)=[];
            mainClusterLengthList(i)=[];
            clusterTable2(i,:)=[];
        else

```

```

        i=i+1;
    end
end
i=1;

%put thicker-than-threshold clusters to mainCluster

mainClusterNumber = 0;
mainCluster = zeros(1,6);
for(i=1:numel(clusterThicknessList))
    if(clusterThicknessList(i)>thicknessThreshold)
        mainClusterNumber = mainClusterNumber + 1;
        for(k=1:clusterNumberOfPoints)
            if(data(k,1)==i)
                point = [mainClusterNumber,data(k,2:end)];
                mainCluster = cat(1,mainCluster,point);
            end
        end
    end
end
mainCluster(1,:)=[];

else
    mainCluster = data;
end

%Find distance between each cluster disBTWcluster
clusterTable2(all(clusterTable2==0,2),:)=[];
[nOfCluster nOfVessel]=size(clusterTable2)
disBTWcluster=NaN(nOfCluster);
for i=1:nOfCluster
    for j=1:i
        if(i~=j)
            cDisList=[];
            for k=1:nOfVessel
                if(~isnan(clusterTable2(i,k)))
                    vlist = clusterTable2(j,:);
                    vlist(isnan(vlist))=[];
                    vDisList = [];

```

```

        for ii=1:numel(vlist)
            vDisList(ii) =
disBTWvessel(clusterTable2(i,k),vlist(ii));
            end
            cDisList(k) = min(vDisList);
        else
            break;
        end
    end
    disBTWcluster(i,j)=min(cDisList);
end
end
end

thresholdDis = 25;
pairList = [0,0];
%make a list of pairs of cluster that the distance between them is
less
%than 25. the list's name is pairList
for(i=1:nOfCluster)
    for(j=1:i)
        if(disBTWcluster(i,j)<thresholdDis)
            pair = [i,j];
            pairList = cat(1,pairList,pair);
        end
    end
end
end
pairList(1,:)=[]

pairedClusterNumberList = unique(pairList);

singleClusterNumberList = [1:nOfCluster];
for(i=1:numel(pairedClusterNumberList))
    j=1;
    while(j<=numel(singleClusterNumberList))

        if(pairedClusterNumberList(i)==singleClusterNumberList(j))
            singleClusterNumberList(j)=[];
        end
    end
end

```



```

        j=j+1;
    end
end
singleClusterNumberList

tmp = pairList;
superClusterNumber = 1;
N=numel(tmp);

%make superClusters
for i=1:N
    if(tmp(i)~=NaN)
        superCluster=tmp(i);
        c=1;
        while(c<=numel(superCluster))
            reference = superCluster(c);
            j=1;
            while(j<=N)
                if(tmp(j)==reference) %found
                    if(j<=N/2)
                        [reference tmp(j+N/2)];
                        superCluster = [superCluster tmp(j+N/2)];
                        tmp(j,:)=NaN;
                    else
                        [reference tmp(j-N/2)];
                        superCluster = [superCluster tmp(j-N/2)];
                        tmp(j-N/2,:)=NaN;
                    end
                end
                j=j+1;
            end
            c=c+1;
        end
        if(numel(superCluster)>1)
            superClusterList{superClusterNumber}=unique(superCluster);
            superClusterNumber=superClusterNumber+1;
        end
    end
end

```

```

        end
    end

    maxEl = 0;
    for i=1:numel(superClusterList)
        [row num] = size(superClusterList{i});
        if(maxEl < num)
            maxEl = num;
        end
    end

    superClusterTable = NaN(numel(superClusterList),maxEl);

    for i=1:numel(superClusterList)
        [row num] = size(superClusterList{i});
        for j=1:num
            superClusterTable(i,j) = superClusterList{i}(1,j);
        end
    end

    end
    superClusterTable

    data = mainCluster;

    [nOfSupercluster nOfSubcluster] = size(superClusterTable);
    tmp = NaN(numel(singleClusterNumberList),nOfSubcluster);
    tmp(:,1) = singleClusterNumberList';
    superClusterTable=[tmp;superClusterTable];

    %Put points from mainCluster to superCluster
    [nOfSupercluster nOfSubcluster] = size(superClusterTable);
    [nOfPoints col] = size(data);
    superCluster = zeros(1,col);
    for suCnum=1:nOfSupercluster
        for cNum=1: nOfSubcluster
            if(isnan(superClusterTable(suCnum,cNum)))
                break;
            else
                for i=1:nOfPoints
                    if(data(i,1)==superClusterTable(suCnum,cNum))

```

```

        point = [suCnum data(i,2:end)];
        superCluster = cat(1,superCluster,point);
    end
end
end
end
end
superCluster(1,:)=[];

mainCluster = superCluster;
size(mainCluster);
resultPath =
strcat('C:\Users\pongsatel\Documents\MATLAB\output\', imname);
save (resultPath, 'mainCluster');
draw_clusteredVessels(imname);

```

getVesselLength.m

```

function length = getVesselLength(vesselPoints)
%getVesselLength find length of a vessel
%by considers the point(x,y) in a vessel as a node of a weighted
complete
%graph which distance between each points in a graph represented by
weight
%of the edge between them. Then find a minimum spanning tree of the
graph
%which represent how the points form a vessel. The summation of
weight in
%the tree is the length of the vessel

[row col]=size(vesselPoints);
vlgraph = zeros(row);
for i=1:row
    for j=1:i
        if(i~=j)
            vlgraph(i,j) =
distance2D(vesselPoints(i,1),vesselPoints(i,2),vesselPoints(j,1),vess
elPoints(j,2));

```

```

        end
    end
end

%view(biograph(vlgraph, [], 'ShowArrows','off', 'ShowWeights','on'))
v1sparse = sparse(vlgraph);
[ST,pred] = graphminspantree(v1sparse);

length = sum(sum(full(ST)));

```

distance2D.m

```

function d = distance2D(x1,y1,x2,y2)
%This function find distance between 2 points in 2D
%Input 4 number x and y of 2 points output the distance

d = sqrt((x1-x2)^2 + (y1-y2)^2);
return;

```

draw_clusteredVessels.m

```

function draw_clusteredVessels(imname)
    %% drawing the cluster of vessel by different colors

    %% Define all path
    vessel_path = 'C:\Users\pongsatel\Documents\MATLAB\output\';

    % define color array
    color_array = ['r', 'g', 'b','c','m','y','k'];
    imageNo = strcat(imname, '.mat');
    disp('** Load Data');
    vessels = loadVessels(vessel_path, imageNo);

    % Merge sub vessels that have connection
    fig = figure;
    for m = 1: length(vessels)
        eV = vessels(1,m);           % access matrix of vessels
        eVx = eV.M(:,1)';           % access x-coordinates
    end

```

```

    eVy = eV.M(:,2)';           % access y-coordinates
    mod_value = mod(m, 7);
    if (mod_value == 0)
        mod_value = 7;
    end

    hold on;
    colorTxt = strcat('.', color_array(mod_value));
    plot(eVx, eVy, colorTxt, 'LineWidth', 1, 'Tag', num2str(m));
end
grid off;
axis([1 700 1 605]);
hold off;
fileTxt = strcat(vessel_path,imname);
fileTxt = strcat(fileTxt, '.tif');
saveas(fig, fileTxt, 'tif');
fprintf('finish\n');
end

function vessels = loadVessels(vesselPath, imageNumber)

    fileName = strcat(vesselPath, imageNumber);
    disp(['Starting to load vessels from file: ', fileName]);
    tic

    v=load(fileName);
    vesselsInFile = v.mainCluster;
    dimensionOfVessels = size(vesselsInFile);
    numberOfVessels = vesselsInFile(dimensionOfVessels(1), 1);

    for vesselNumber=1:numberOfVessels
        rows = find(vesselsInFile(:,1)== vesselNumber);
        x = vesselsInFile(rows, 3);
        y = vesselsInFile(rows, 2);
        M = [x y];

        % Exclude duplicates:
        M = unique(M, 'rows');
        vessels(vesselNumber).M = M;

```

```
end
toc
disp(['Finished loading ', num2str(numberOfVessels), ' vessels
from file ', fileName, '.']);
end
```



Appendix B

Rotational 2D Vessel Projection MATLAB source code

ARfastOD.m

```
function time = ARfastOD(debug, set, imnum, startingAngle)
    [absImgPath, outPath] = getFilePaths(set, imnum, startingAngle);
    if strcmp(absImgPath, '')==1
        time = 0;
        return;
    end
    img = imread(absImgPath);
    tic
    [row col] = size(img);
    side = max(row, col)/3;
    ratio = 300/side;
    img = imresize(img, ratio);
    cimg = img;
    img = img(:,:,2);

    % make the image a square image.
    [maxy, maxx] = size(img);
    difxy = abs(maxx-maxy);
    side = max(maxx, maxy);
    sqImg = zeros(side, 'uint8');
    addSp = round(difxy/2);
    if (maxx > maxy)
        if (mod(difxy, 2) == 0)
            sqImg([addSp+1:side-addSp], :) = img;
        else
            sqImg([addSp:side-addSp], :) = img;
        end
    else
        if (mod(difxy, 2) == 0)
            sqImg(:, [addSp+1:side-addSp]) = img;
        else
            sqImg(:, [addSp:side-addSp]) = img;
        end
    end
end
```

```

        end
    end
    maxx = side;
    maxy = side;
    img = sqImg;
    maxThick = 6;

    img = imrotate(img,startingAngle,'bilinear','crop');
    [bwImg,filteredImg] = seg(img,maxThick);

    % class Answer
    Answer.xy=[0 0];
    Answer.angle=0;
    Answer.meanVar = [];
    Answer.diff = [];
    answer(1) = Answer;

    score = intmax();
    ansCount = 0;
    cAngle = 0;
    ODR = 20;
    step = 5;
    baseBwImg = bwImg;
    baseImg = img;
    outImg = zeros(maxy,maxx);

    for i=0:180/step
        angle = i*step;
        bwImg = imrotate(baseBwImg,angle,'bilinear','crop');
        img = imrotate(baseImg,angle,'bilinear','crop');
        outImg = imrotate(outImg,angle,'bilinear','crop');

        [height,width] = size(bwImg);
        [x,diff,bwH,bwV] = fastXPos(bwImg,maxThick);

        imGray = img;
        [y,meanVar] = fastYPos(imGray,x,ODR,maxThick);
        mid = maxy/2;
        % Draw circle at OD

```



```

        outSingle = zeros(maxy,maxx);
        if(y > height/4 && y<3*height/4)
            ansCount = ansCount+1;
            outImg = drawCircle(outImg,1,x,y,ODR);
        end

        answer(i+1).xy = [x y];
        answer(i+1).angle = angle;
        answer(i+1).meanVar = meanVar;
        answer(i+1).diff = diff;

        outImg = imrotate(outImg,-1*angle,'bilinear','crop');
    end

    % find brightest spot in outImg which is OD
    [I,J] = findBrightestSpot(outImg);
    OD = [I,J];
    time = toc;

    ref = [width/2 height/2];
    ODref = [OD(1)-ref(1) ref(2)-OD(2)]; %correct
    [THETA,RHO] = cart2pol(ODref(1),ODref(2));
    minDist = intmax();

    %find the answer that is closest to the OD to find the correct
    angle.
    for i=0:180/step
        [THETA,RHO] = cart2pol(ODref(1),ODref(2));
        THETA = THETA+degtorad(answer(i+1).angle);
        [ODrotated(1),ODrotated(2)] = pol2cart(THETA,RHO);
        ODrotated = [ODrotated(1)+ref(1) ref(2)-ODrotated(2)];
        dist =
        distance2D(answer(i+1).xy(1),answer(i+1).xy(2),ODrotated(1),ODrotated
        (2));
        if(dist < minDist)
            ansIndex = i+1;
            ansOD = answer(i+1).xy;
            minDist = dist;
        end
    end

```

```

end

if(debug == 1)
    bwImg = baseBwImg;
    img = baseImg;
    [x,diff,bwH,bwV] = fastXPos(bwImg,maxThick);
    [y,meanVar] = fastYPos(imGray,x,ODR,maxThick);
    figure('units','normalized','outerposition',[0 0 1 1]),
    subplot(2, 3, 1);
        imshow(bwImg);
        title('Vessel image')
    subplot(2, 3, 4);
        imshow(filteredImg);
        title('top-hat filtered image')
    subplot(2, 3, 6);
        imshow(img);
        hold on
        ansOD = OD;
        plot(ansOD(1),ansOD(2),'wx','MarkerSize',30);
        plot(ansOD(1),[0:height],'r','LineWidth',maxThick);
        plot([0:width],ansOD(2),'r','LineWidth',maxThick);
        hold off
        axis ij
        title(['Result ' int2str(imnum) ' at '
int2str(answer(ansIndex).angle) ' degree'])
    subplot(2, 3, 5);
        imshow(outImg,[]);
        hold on
        plot(OD(1),OD(2),'rx','MarkerSize',30);
        hold off
        title(['int2str(ansCount) '/' int2str(180/step) ' result
candidates, best score = ' int2str(max(max(outImg)))])
    subplot(2, 3, 2);
        imshow(bwH);
        title('Horizontal Vessel')
    subplot(2, 3, 3);
        imshow(bwV);
        title('Vertical Vessel')
else

```

```

        figure,
        imshow(baseImg, []);
        hold on

plot(OD(1),OD(2),'x','LineWidth',3,'MarkerSize',30,'MarkerEdgeColor',
'b','MarkerFaceColor','r');
        hold off
        saveas(gcf,outPath);
        close
    end

```

seg.m

```

function [imgVessel,g2] = seg(img,maxThick)
    G = img;
    ig = imcomplement(G);
    maxThick = maxThick/2;
    if mod(maxThick,2)==0 maxThick = maxThick+1; end
    maxThick = maxThick*2;

    se = zeros(maxThick);
    se(:,ceil(maxThick/2)) = 1;
    se(ceil(maxThick/2),:) = 1;
    g2 = imtophat(ig,se);
    g2 = imadjust(g2);
    level = graythresh(g2);
    BW = im2bw(g2,level);
    imgVessel = BW;

```

drawCircle.m

```

function outImg = drawCircle(img,grayLevel,x,y,r)
    [imageSizeY imageSizeX] = size(img);
    [columnsInImage rowsInImage] = meshgrid(1:imageSizeX,
1:imageSizeY);
    centerX = x;
    centerY = y;
    radius = r;

```

```

    circlePixels = (rowsInImage - centerY).^2 + (columnsInImage -
centerX).^2 <= radius.^2;
    img(circlePixels) = img(circlePixels)+grayLevel;
    outImg = img;

```

distance2D.m

```

function d = distance2D(x1,y1,x2,y2)
%This function find distance between 2 points in 2D
%Input 4 number x and y of 2 points output the distance
    d = sqrt((x1-x2)^2 + (y1-y2)^2);

```

findBrightestSpot.m

```

function [xOut,yOut] = findBrightestSpot(img)
    brightestVal = max(max(img));
    thresImg = img==brightestVal;

    [row col] = size(thresImg);
    xs = 0;
    ys = 0;
    for i=1:row
        for j=1:col
            if(thresImg(i,j)==1)
                xs = xs+j;
                ys = ys+i;
            end
        end
    end

    xc = xs/sum(sum(thresImg));
    yc = ys/sum(sum(thresImg));

    xOut = xc;
    yOut = yc;

```

fastXPos.m

```
function [x,diff,bwH,bwV] = fastXPos (bwImg,maxThick)
    se1 = strel('line',maxThick,0);
    se2 = strel('line',maxThick,90);
    bwV = imopen(bwImg,se2);
    bwH = imopen(bwImg,se1);
    [height,width] = size(bwImg);
    nOfSlot = width/maxThick;
    V = zeros(1,width);
    H = zeros(1,width);
    D = zeros(1,width);
    for i=maxThick+1:width-maxThick
        slotV = bwV(:, [i-maxThick i+maxThick]);
        slotH = bwH(:, [i-maxThick i+maxThick]);
        V(i)=sum(sum(slotV));
        H(i)=sum(sum(slotH));
        D(i)=V(i)-H(i);
    end

    [M,I] = max(D);

    x = I;
    diff = D;
```

fastYPos.m

```
function [y,meanVar] = fastYPos (imGray,I,ODR,maxThick)
    [height,width] = size(imGray);
    Var = NaN(1,height);
    SV = zeros(1,height);
    meanVar = Var;
    for i=ODR+1:height-ODR-1
        if(I-ODR <= 0) I = ODR+1; end
        if(I+ODR >= width)
            I = width - ODR ;
        end
        wGray = imGray([i-ODR:i+ODR],[I-ODR:I+ODR]);
```

```
wGray = wGray(wGray>25);  
Var(i) = var(double(wGray));  
end  
for i=ODR+1:height-ODR  
    vv = Var(i-ODR:i+ODR);  
    meanVar(i) = mean(vv(~isnan(vv)));  
end  
[M,y] = max(meanVar);
```

