ABSTRACT

A FILTER BANK APPROACH TO AUTOMATE VESSEL EXTRACTION WITH APPLICATIONS

By

Nen Huynh

May 2013

Methods to extract vessel networks in medical images have been in high demand for its applications to health risk predictions. For example, vessel enhancement of retinal images has shown promises in diagnosing diabetes. Within the existing literature, multiscale vessel enhancement stands out as one of the best for its accuracy, speed, and simplicity. But like many vessel extraction techniques, the efficacy of the method is greatly hindered in the presence of noise, lighting variations, and decreased resolution. This deficiency is presents itself in retinal images and are particularly pronounced in digital photographs of human placenta.

Retinal images have a been popular data set of testing vessel extraction methods because of its simplicity in anatomical structure yet high hopes in diagnosing conditions such as diabetic retinopathy and glaucoma. Thus, the thesis will focus on the application of vessel extraction methods on retinal images. Specifically, we focus on the DRIVE and STARE database.

Also, recent placental pathology evidence has contributed to current understanding of causes of low birth weight and preterm birth, each has been linked to increased risk of later neurodevelopmental disorders. Among various factors that cause such disorders, the vessel network on the placenta has been

hypothesized to offer the most clue in bridging that connection. Herein lies the most essential step of the blood vessel extraction, which has only been done manually through a laborious process.

Motivated by its ability to handle curvilinear structures, we propose the use of directional filter banks to further enhance the results obtained from the multiscale method. Validating experiments will be performed on a private database that is made available by the Placental Analytics, LLC.

It will be shown that for retinal images, the directional filter bank approach significantly improves the performance over the well-known multiscale vessel enhancement method. However, the directional filter bank approach are comparable to multiscale vessel enhancement on placentas.

A FILTER BANK APPROACH TO AUTOMATE VESSEL EXTRACTION WITH APPLICATIONS

A THESIS

Presented to the Department of Mathematics and Statistics

California State University, Long Beach

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Applied Mathematics

Committee Members:

Jen-Mei Chang, Ph.D. (Chair)
Eun Heui Kim, Ph.D.
William Ziemer, Ph.D.

College Designee:

Robert Mena, Ph.D.

By Nen Huynh

B.S. Mathematics, 2008, University of California, Irvine

May 2013

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

ACKNOWLEDGEMENTS

I would like to thank research partner Marylin Vazquez who contributed much to this thesis. Also, I would like to point out the hard work of Almoussa et al. [12] along with the UCLA REU 2010 program for the initial work on placentas that we have built our work on. A special thank also goes to the work of Hoover et al. [1] and Staal et al. [3] for providing the retinal images for our tests and Placental Analytics for the placental images. A big thank goes to Dr. Jen-Mei Chang for providing opportunities that greatly benefited my academic career. Without her, I would not have enjoyed applied mathematics as much as I do now.

I am grateful for the correspondence with author Phan Truc who provided the diamond and hourglass code for study. Also, I appreciate the works of Dirk-Jan Kroon and David Young. Dirk-Jan Kroon of Focal Machine Vision en Optical Systems posted the code for multiscale vessel enhancement, which we have used it extensively. David Young of University of Sussex provided a fast code for convolution based on singular value decomposition. It greatly increased the speed at which our codes ran.

I thank my parents for providing a supportive environment for my talents to flourish. I am grateful for my friends, such as long time friend Jennifer Meyers-Giddings, who draw me out of my bubble. Finally, I thank California State University, Long Beach and its Mathematics Department for their hard work in offering many great learning opportunities. The time I have spent here has been invaluable. And by extension, I thank the American college system for giving me so much yet still rewarded me financially for my accomplishments. Thank you all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER 1

INTRODUCTION

The breakthroughs in imaging technology such as MRIs, sonograms, and even digital cameras have created a wealth of data for the medical field. With such data, there have been a motivation to use such tools for diagnosis. Currently, the paradigm for medical imaging is focused on improving the technology so that medical personnel can more accurately diagnose a patient. However, this paradigm is limited by time and expenses which motivates the automation of some of the diagnosis process. Such motivation has partially fueled the growth of the image processing field. Its goal is to process the images so that there is a higher medically relevant quality so that a medical technician may diagnose more accurately. Specifically, this involves the reduction of noise of an image, removing irrelevant features, and highlighting relevant features. One such relevant feature this thesis will focus on are the veins and arteries of a two-dimensional image. Together, they are called vessels and the process of visually locating such vessels is called vessel extraction. Finding the location of vessels provides a measurement for shape, coloration, size, and spatial distribution of vessels. Such statistical features are useful in the medical diagnosis process.

There is an enormous amount of literature on methods of extracting vessels. They include matched filter response [1], gradient vector fields [2], ridge based methods [3], and hybrid methods [4]. We refer to [5] for a survey of vessel extraction methods. One of the popular methods is multiscale vessel enhancement [6], which will be discussed in chapter 2. The focus of this thesis will be the

directional filter bank modification of multiscale vessel enhancement method [7], called decimation-free directional filter bank (DDFB) vessel enhancement.

To compare the different vessel extraction methods, retinal image databases STARE [1] and DRIVE [3] are used. A placenta data set provided by Placental Analytics LLC is also tested on. This is to test whether directional filter banks is generally better than multiscale vessel enhancement or just on retinal images. Section 1.2 will discuss the retinal image databases in detail and section 1.3 will discuss the placental image databases in detail.

A brief introduction to image processing techniques is provided in section 1.1. For the remainder of chapter 1, we will mention the prepocessing steps that are performed on the mentioned database images. With the exception of homomorphic filtering, these preprocessing steps will only be briefly mentioned and a reference will be provided for those interested in them.

Chapter 2 will explore the necessary parts of differential geometry to lay the necessary groundwork for directional filter banks in chapter 3. Its highlight will be a popular method of vessel extraction called multiscale vessel enhancement [6], with application to retinal images. To evaluate its performance, an accuracy measure will be introduced.

Chapter 3 will focus on the theory of directional filter banks as it pertains to vessel enhancement. Along the way, concepts from signal processing with regards to filtering will be explained. The chapter will conclude with the application of directional filter banks on retinal images and placentas. Metrics for comparing results between multiscale vessel enhancement and directional filter banks will also be introduced.

An Introduction to Image Processing Techniques

An $m \times n$ dimensional grayscale image of dimension can mathematically be represented as a function $I : \{1, 2, ..., M\} \times \{1, 2, ..., N\} \to \mathbb{R}$. I is can be extended using the discrete symmetric extension $\tilde{I} : \mathbb{Z}^2 \to \mathbb{R}$ where
$\tilde{I}(n_1, n_2) = I(s(n_1; M), s(n_2; N))$ and

$$s(n; N) = \begin{cases} n - \lceil \frac{n}{N} \rceil N & \text{if } \lceil \frac{n}{N} \rceil \text{ is even} \\ N + 1 - \left(n - \lceil \frac{n}{N} \rceil N\right) & \text{if } \lceil \frac{n}{N} \rceil \text{ is odd} \end{cases} \tag{1.1}$$

For notational convenience, we will implicitly refer to an image $I : \mathbb{Z}^2 \to \mathbb{R}$, whose domain is $\mathbb{Z}^2$, to be the discrete symmetric extension of
$I : \{1, 2, ..., M\} \times \{1, 2, ..., N\} \to \mathbb{R}$, whose domain is $\{1, 2, ..., M\} \times \{1, 2, ..., N\}$.

For a color image, the digital representation involves the three color channels red, green, and blue. The mathematical representation is then a function $I = (R, G, B) : \mathbb{Z}^2 \to \mathbb{R}^3$ where $R : \mathbb{Z}^2 \to \mathbb{R}$ is the red channel, $G : \mathbb{Z}^2 \to \mathbb{R}$ is the green channel, and $B : \mathbb{Z}^2 \to \mathbb{R}$ is the blue channel. And similarly, R, G, and B are discrete symmetric extensions of a corresponing function with domain $\{1, 2, ..., M\} \times \{1, 2, ..., N\}$.

A square bracket and curly bracket will be used for operators such as differential operators ($L[I]$) and Fourier transform ($\mathfrak{F}\{f\}$). The composition of f and g is denoted by $f \circ g$. The inner product of $\mathbf{x}$ and $\mathbf{y}$ is defined $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y}$ along with its respective norm $\|\mathbf{x}\|_2 = \langle \mathbf{x}, \mathbf{x} \rangle$.

Convolution for functions $f, g : \mathbb{R} \to \mathbb{R}$ is defined as

$$(f * g)(t) = \int f(s) g(t - s) ds$$

while functions with two-dimensional domains $f, g : \mathbb{R}^2 \to \mathbb{R}$ is defined as

$$(f * g)(x, y) = \iint f(u, v) g(x - u, y - v) dv du.$$

For integer-domain functions $f, g : \{1, ..., N\} \to \mathbb{R}$ and $f, g : \{1, ..., M\} \times \{1, ..., N\} \to \mathbb{R}$, (discrete symmetric) convolution is, instead, defined as

$$(f * g)(n) = \sum_{i=1}^{N} \tilde{f}(i)\tilde{g}(n-i) \text{ and } (f * g)(i,j) = \sum_{i=1}^{M}\sum_{j=1}^{N} \tilde{f}(i,j)\tilde{g}(m-i, n-j),$$

respectively, where $\tilde{f}, \tilde{g}$ are the discrete symmetric extension of f, g.

Finally, the process of thresholding will be used throughout this thesis. Suppose we have $I : \mathbb{Z}^2 \to \mathbb{R}$. Then a thresholding of I by t is a function $T : \mathbb{Z}^2 \to \{0, 1\}$ such that

$$T(x) = \begin{cases} 1 & \text{if } I(x) > t \\ 0 & \text{otherwise.} \end{cases} \tag{1.2}$$

Retinal Image Database

As mentioned in [3], the use of retinal images has contributed to diagnostics of diabetic retinopathy [8], retinal vein occlusion [9], hypertension [10], and glaucoma [11]. One of the important features in retinal images is the vessel. The DRIVE and STARE databases are popular data sets to test on. Their popularity is due to their containing mainly visible vessels and clinically shown diagnostics potential.

The DRIVE database [3] contains 40 retinal images evenly divided into the training and test data sets. It is also called the Utrecht database since it is collected at University Medical Center Utrecht, The Netherlands. For each of the training images, there is a corresponding hand trace ground truth image to test the accuracy of the vessel extraction. And for each of the testing images, there are two corresponding hand traces to test the accuracy of the vessel extraction. Because the retinal images are taken with a background which is not relevant to

the extraction result, the background can be removed with a mask that is also provided in the database.

The STARE database [1] contains 20 retinal images and for each image, there are two corresponding hand traces, drawn by Hoover (which will be denoted as AH) and Kouznetsova (which will be denoted as VK). The VK hand traces are more detailed then the AH hand trace, creating an extra challenge for vessel extraction. The STARE database is also called Hoover database for its association with Adam Hoover. The retinal images were collected from the Shiley Eye Center at University of California, San Diego, and Veterans Administration Medical Center in San Diego. Table 1 shows the retinal image numbers for the STARE image and a corresponding indices later used in bar graphs.

A mask is not provided in the data set; however, it can be manually created by picking a good threshold in the red channel because in the red channel, the background is dark compared to the retina. Precisely, $M : \mathbb{Z}^2 \to \{0, 1\}$ is a mask such that

$$M(x, y) = \begin{cases} 1 & \text{if } R(x, y) > \text{threshold} \\ 0 & \text{otherwise} \end{cases} \tag{1.3}$$

where $R(x, y)$ is the red channel of a color image. Table 1 shows the specified thresholds. We mention that some of the STARE images have different threshold values because they may have been captured under different lighting conditions, causing some images to be slightly brighter or dark than others. Hence, different threshold values are chosen.

TABLE 1. Thresholds for Creating Masks in STARE

STARE index	1	2	3	4	5	6	7	8	9	10
Retinal image	1	2	3	4	5	44	77	81	82	139
Threshold	50	40	60	33	50	70	50	50	50	50
STARE number	11	12	13	14	15	16	17	18	19	20
Retinal image	162	163	235	236	239	240	255	291	319	324
Threshold	50	50	50	50	50	40	50	50	50	50

Placental Image Database

We will see that the directional filter bank approach works well on retinal images. It is then natural to ask if it will work on general vessel images. To attempt an answer, we will also test on placenta photographs provided by Placenta Analytics, LLC. The placenta is an organ that regulates nutrient uptake, waste elimination, and gas exchange between the fetus and the mother. In theory, the health of the placenta is then correlated with the health of the fetus. Thus, researchers at Placenta Analytics study the potential of using the placenta to predict the health of the fetus after birth.

To test the researchers' theory, they organized a set of approximately 3200 placental images is obtained by photographing washed placentas for study. Out of those, 330 has a trace of the boundary of the placenta plate and a simple hand trace is drawn to identify the location of the vessels by a trained pathologist. Of the 330, we choose 16 to test the directional filter bank approach. Table 2 shows the list of placentas and the corresponding indices later used in bar graphs.

TABLE 2. Placental Images Used in This Study

Placental index	1	2	3	4	5	6	7	8
Placental image	1973	2041	2095	2141	4561	2666	2743	2744
Placental index	9	10	11	12	13	14	15	16
Placental image	2753	2772	2774	2777	2946	3321	3340	3355

The placentas have regions with discolorations and each placenta has a distinct color profile causing simple extraction methods such as thresholding to fail. Also, the placenta has a lot of vessels with large vessels and small vessels many times overlapping with each other. Even with the best vessel extraction method, this causes an enormous problem in accuracy. However, the possibility of having a prediction for the health of a fetus encourages further research.

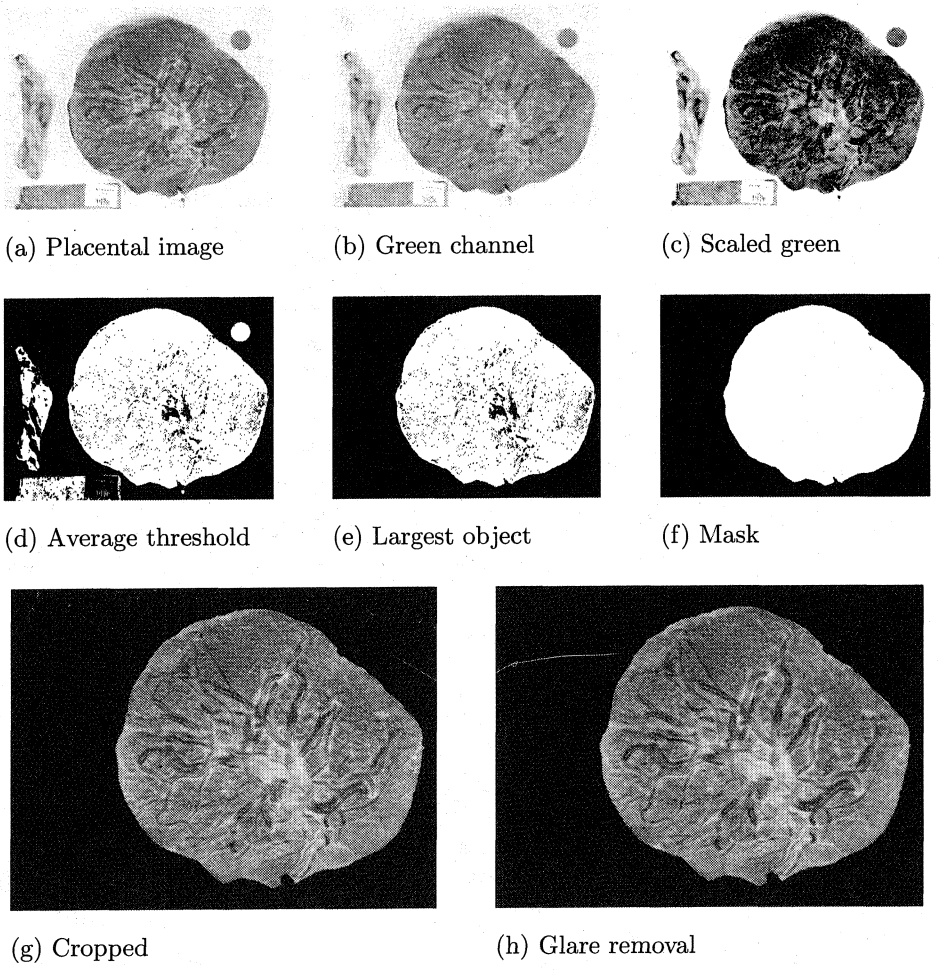
At the present, the only journal-published work on the vessel extraction of such data set has been from [12]. They have shown that the vessels are difficult due to the discoloration on the surface of the placenta. They provide a neural network approach using 5 features: the gradient magnitude, gradient angle, wide-line detector [13], Steger detector [14], and their own modified road detector. They report some success in vessel extraction.

Another notable work comes from [15]. It shows that for placental images, a ridgelet filter applied on the multiscale vessel enhancement thresholded result produces a significant performance compared to just the multiscale vessel enhancement and neural network. However, [15] is still a work in progress so no conclusion can yet be drawn.

Preprocessing

The placental images require preprocessing before the vessel extraction method to be performed. Figure 1 shows the steps in cropping the placental images starting with a raw placental image (Figure 1a). The green channel is isolated (Figure 1b) and the brightness of the green channel is scaled to highlight the brightest and darkest parts of the image (Figure 1c). A threshold is performed with the average brightness intensity as the t value in equation (1.3) (Figure 1d). From the threshold, the placenta is isolated (Figure 1e). Using the process of filling in and morphological erosion, the mask is constructed (Figure 1f). From the mask, the placenta can be isolated by cropping (Figure 1g). Finally, deglaring is performed to remove bright spots on the placenta surface (Figure 1h). Deglaring is performed in the manner of [12], which is a simplified version of [16]. This involves a combination of thresholding, top-hat filtering, and dilation performed to identify the glare region. The region is then replaced using Laplace's equation.

The vessel enhancement methods discussed in this thesis will use the large change in brightness intensity in an image. The boundary between the black background and the foreground causes a sudden change which will be interpreted as a vessel when the vessel extraction algorithms are performed on the images. For placental images, such effect can be ignored using the mask but for retinal images, the actual vessels can be negatively identified if they are too close to such boundary. To negate this for retinal images, the sudden change from foreground to background will be removed by foreground diffusion using the heat equation; the foreground color on the boundary is diffused through the background. The masks for each of the images are usually a few pixels off of the actual boundary between the foreground and background so a few (specifically five pixels) of the boundary are removed from the foreground and relabeled as the background before the



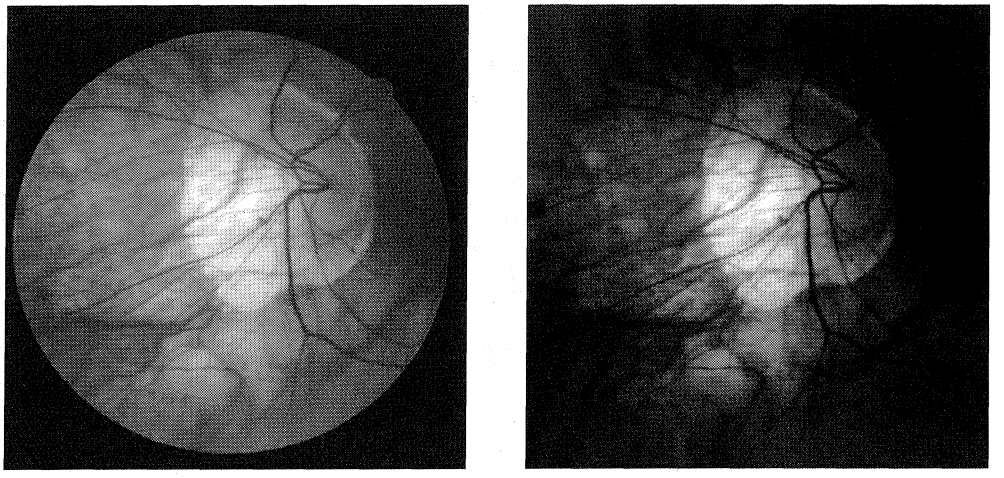
(a) Placental image (b) Green channel (c) Scaled green

(d) Average threshold (e) Largest object (f) Mask

(g) Cropped (h) Glare removal

FIGURE 1. Placental image preprocessing.

diffusion is applied. Figure 2 shows an example of diffusion on a retinal image from the DRIVE database.



(a) Image 34 of DRIVE database (b) Diffused version of image 34

FIGURE 2. An example of diffusion using the heat equation.

Homomorphic Filtering

Images, especially medical images, may suffer from uneven lighting. To deal with this, homomorphic filtering is an option to fix the lighting of such image.

In the simple image formation model [17], a light source is shined on a surface with intensity $i(x, y) : \mathbb{Z}^2 \to [0, \infty)$ and the light source is reflected into a light capturing device, for example a camera, which captures a portion of reflection, $r(x, y) : \mathbb{Z}^2 \to [0, 1]$, of the light. Hence, the image can be modeled as the product

$$I(x, y) = i(x, y)r(x, y). \tag{1.4}$$

In this model, illumination has slow variations so it has low frequency, that is $|\mathfrak{F}\{i\}(\omega)|$ is small for large $|\omega|$ and $|\mathfrak{F}\{i\}(\omega)|$ is large for small $|\omega|$. And reflection has much of the variations present in the image itself so it has a high frequency if

the image has high frequency, that is $|\mathfrak{F}\{r\}(\omega)|$ is small for small $|\omega|$ and $|\mathfrak{F}\{r\}(\omega)|$ is large for large $|\omega|$.

The purpose of homomorphic filtering is to scale the illumination so that it is approximately constant. Typically, the illumination and reflection functions are not known, hence equation (1.4) can only serve as a model. However, the fact that illumination and reflection have opposing frequency information for the image provides a way to scale the illumination.

To perform the scaling, the Fourier transform on the log of the image is performed:

$$\mathfrak{F}\{\log I\} = \mathfrak{F}\{\log i\} + \mathfrak{F}\{\log r\}.$$

Given a filter $H(\omega) : \mathbb{R}^2 \to \mathbb{C}$, we can scale the Fourier transform with

$$T(\omega) := H(\omega)\mathfrak{F}\{\log I\}(\omega) = H(\omega)\mathfrak{F}\{\log i\}(\omega) + H(\omega)\mathfrak{F}\{\log r\}(\omega).$$

The resulting image is obtained by a log and inverse Fourier transform:

$$I' = \exp(\mathfrak{F}^{-1}\{T\}).$$

The filter $H(\omega)$ is called a homomorphic filter function. Its purpose is to scale the illumination while preserving reflection. An example of H looks like the curve in Figure 3. Notice that $H(\omega)$ is small for low $|\omega|$ and approximately constant for large. When multiplied with $\mathfrak{F}\{\log i\}(\omega)$, the low frequency will be dampened so the illumination term of the new image I' is approximately constant. When multiplied with $\mathfrak{F}\{\log r\}(\omega)$, the high frequency will approximately be the same so the reflection term of the new image I' is approximately the same as the original image I. Thus, in theory, homomorphic fixes the illumination to a constant and remove the uneven lighting.

TABLE 3. A List of Commonly Used High-Pass Filters

Ideal	Butterworth	Gaussian
$h(\omega) = \begin{cases} 0 & \text{if } \|\omega\| \leq D_0 \\ 1 & \text{if } \|\omega\| > D_0 \end{cases}$	$h(\omega) = \frac{1}{1+\left[D_0^2/\|\omega\|\right]^{2n}}.$	$h(\omega) = 1 - e^{-\|\omega\|_2^2/(2D_0^2)}$

Without going into too much detail, we only mention that the desired homomorphic filter is constructed by

$$H(\omega) = (\alpha_H - \alpha_L)h(\omega) + \alpha_L$$

where h is often one of three high-pass filters given in table 3 for some constants $D_0 > 0, n \in \mathbb{Z}$. We refer to [17] for the details.

Figure 4 shows two examples of the effect of homomorphic filtering using parameters: Butterworth, $D_0 = 50, \alpha_L = 0.1, \alpha_H = 1.0$. Notice the removal of the different brightness of the vessel background and the bright spot in the middel, also called an occlusion. The use of homomorphic filtering will be mainly used to remove lighting so that the transition from a bright spot to a dark spot (as seen in Figure 4c with a bright center and darker region around it) is not falsely identified as a vessel.

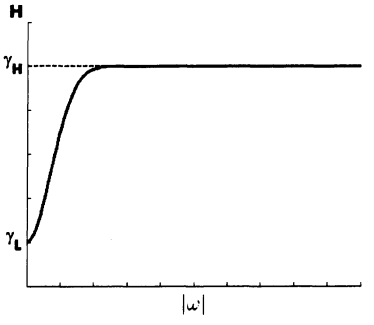
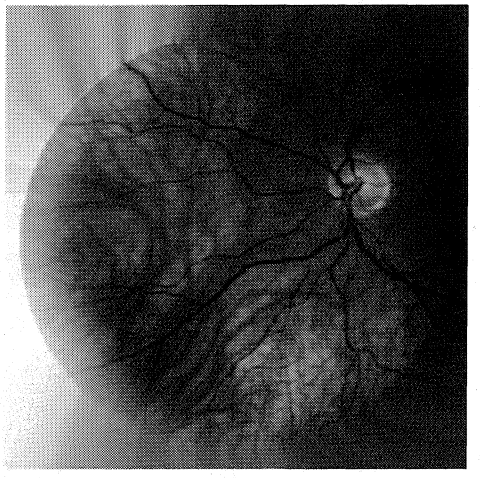
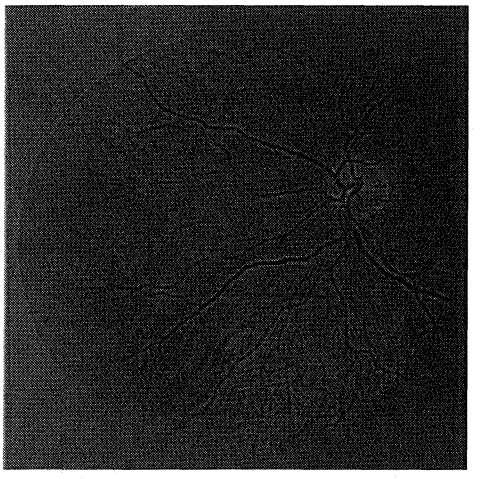


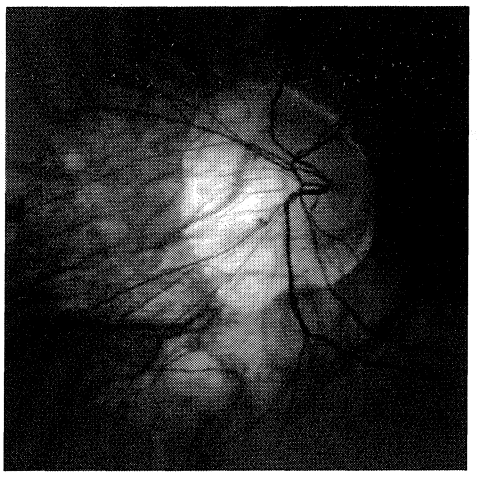
FIGURE 3. An example of a homomorphic filter function.



(a) The (diffused) DRIVE image 23



(b) Homomorphic filtering images of (a)



(c) The (diffused) DRIVE image 34

(d) Homomorphic filtering images of (c)

FIGURE 4. Example images filtered by the homomorphic filter.

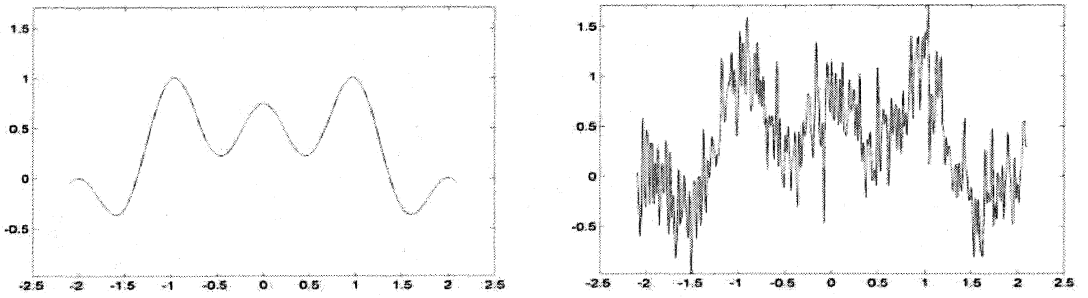
CHAPTER 2

MULTISCALE VESSEL ENHANCEMENT

Much success in vessel extraction has come from multiscale vessel enhancement [6]. By representing an image as a 2D function and solely studying the eigenvalues of its Hessian, multiscale vessel enhancement's efficiency, simplicity, and accuracy has allowed it to still be considered as one of the best vessel extraction methods. Here, an introduction to multiscale differentiation will be provided. The Hessian is then defined from the multiscale differential operators. From such Hessian, multiscale vessel enhancement from [6] is defined and a demonstration of its accuracy on retinal images will be presented. The chapter will conclude with a brief discussion on the limitations of multiscale vessel enhancement.

Multiscale Differential Operators

We start with a look at the one-dimensional signal $S : \mathbb{Z} \to \mathbb{R}$ to introduce multiscale differentiation. The signal contains features that may be studied from differential properties. For example, a relatively flat region may be identified as a region with a relatively small derivative and a parabolic region may be identified by a change of sign in the first derivative and a uniformly positive/negative second derivative. However, numerical methods such as finite difference schemes for differential operators are highly susceptible to noise which are present in typical signals.



(a) The original signal (b) The noisy signal

FIGURE 5. A 1D signal and its noisy counterpart.

As an example, consider a signal

$$Y(x) = \psi_2(x) - \psi_1(x) + 8\psi_{1/2}(x) \text{ where } \psi_a(x) = \begin{cases} e^{-\frac{1}{a^2-x^2}} & \text{if } |x| < a \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

and a noisy signal

$$S(x) = Y(x) + \eta(x) \tag{2.2}$$

where η is Gaussian white noise with signal-to-noise ratio of 10 and sampled at a rate of $h = 0.03$ units ($x \in 0.03\mathbb{Z}$). Figure 5a shows the function $Y(x)$ and 5(b) shows the noisy signal.

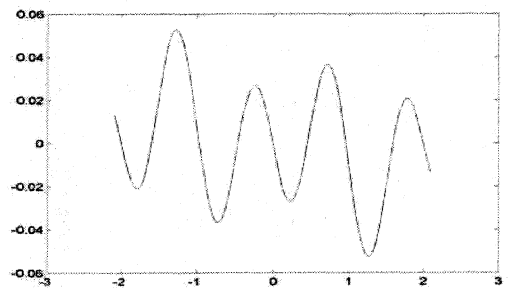
Using finite differences for derivatives, we get

$$S'(x) \approx \frac{S(x+h) - S(x-h)}{2h} \text{ and } S''(x) \approx \frac{S(x+h) - 2S(x) + S(x-h)}{h^2}.$$

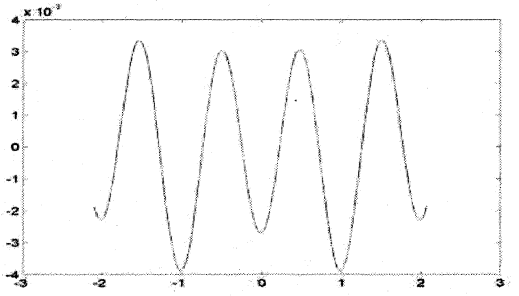
Figure 6 shows that obtaining the first and second derivatives this way will poorly represent the true derivatives in the presence of noise.

To remedy this issue, the signal is first convolved with the Gaussian kernel:

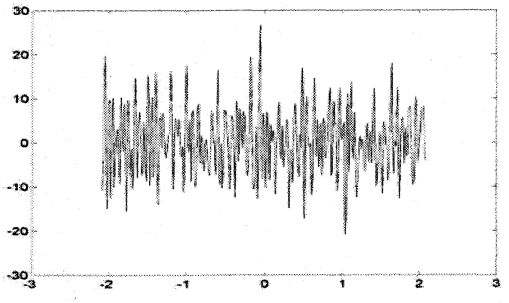
$$S_\sigma(\cdot) = G^\sigma * S(\cdot) \tag{2.3}$$



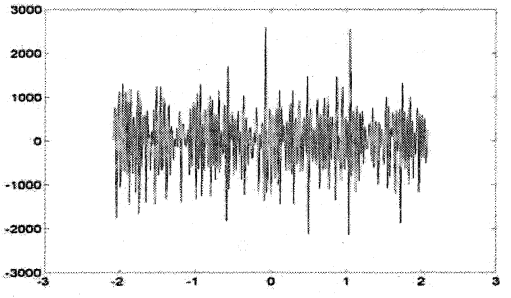
(a) actual first derivative of signal



(b) actual second derivative of signal



(c) finite difference first derivative



(d) finite difference second derivative

FIGURE 6. The impact of noise on finite difference.

where $G^{\sigma}(\mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}^D} e^{-\frac{\|\mathbf{x}\|^2}{2\sigma^2}}$ is the Gaussian kernel and where D is the dimension of x. $D = 1$ for signals and $D = 2$ for images.

Figure 7 shows the signal in Figure 5b that is denoised using such convolution process with different σ values. Notice that $\sigma = 16$ smooths the signal well enough to sufficiently recover the original signal (the smoothed signal is approximately a scalar multiple of the original signal). Also, as σ increases, the small scale details are increasingly smoothed over. Hence, a small σ removes some level of noise and large σ removes much of the noise along with the small scale structures. This allows the separate processing of small scale and large scale structures. This effect can be seen in Figures 7c-e. Notice the large σ removes the

small concavity changes, allowing for easier study of the larger interval with consistent concavity.

Figure 8 shows an example on a section of a retinal image and the varying σ-values of smoothing. Notice as σ increases, the image gets blurrier. For example, $\sigma = 2$ smooths the image so a more appealing image occurs. A $\sigma = 4$ removes most of the noise while keeping most of the important vessel features. For $\sigma = 8$, only the large scale vessels are noticeable, allowing the processing of only large scale vessels. And for $\sigma = 16$, much of the features are too blurry for comprehension.

It is now possible to compute the derivatives of the signal using, for example, finite difference on the convolved signal. However, the derivatives can be calculated using the property

$$\frac{\partial^n}{\partial x^n}[G^\sigma * S] = \left[\frac{\partial^n}{\partial x^n} G^\sigma\right] * S.$$

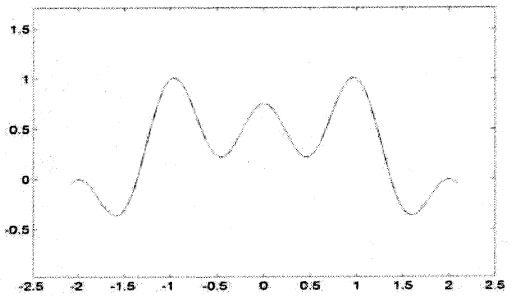
Thus, the multiscale version of differential operators on a signal may be defined as the convolution of the signal with the differential operator applied on the Gaussian kernel with a specified σ. For notational convenience, we define the first and second convolution derivatives to be

$$S' = \sigma^\gamma G_x^\sigma * S \tag{2.4}$$

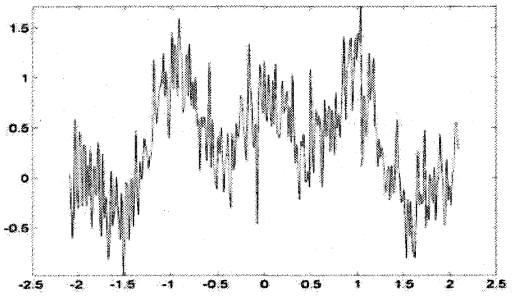
and

$$S'' = \sigma^\gamma G_{xx}^\sigma * S, \tag{2.5}$$

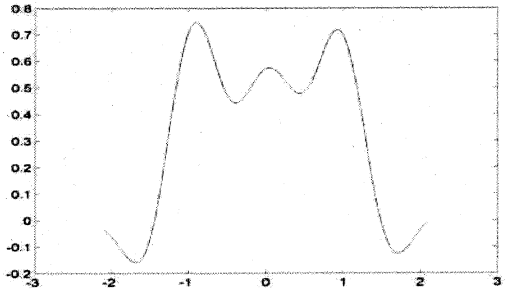
for some fixed scale σ, where $*$ is the convolution operator and the derivative normalization γ introduced by Lindeberg [18]. γ is used to scale the response of differential operators at multiple scales σ. A detailed treatment of the use of γ is found in Lindeberg [18]. To be consistent with [6], γ is fixed to 0 for multiscale



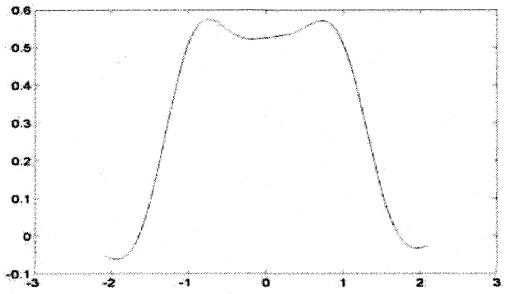
(a) Original signal



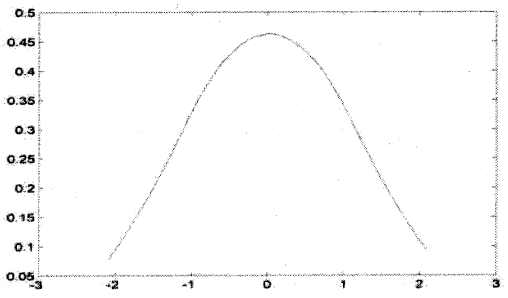
(b) Noisy signal



(c) S_{16}

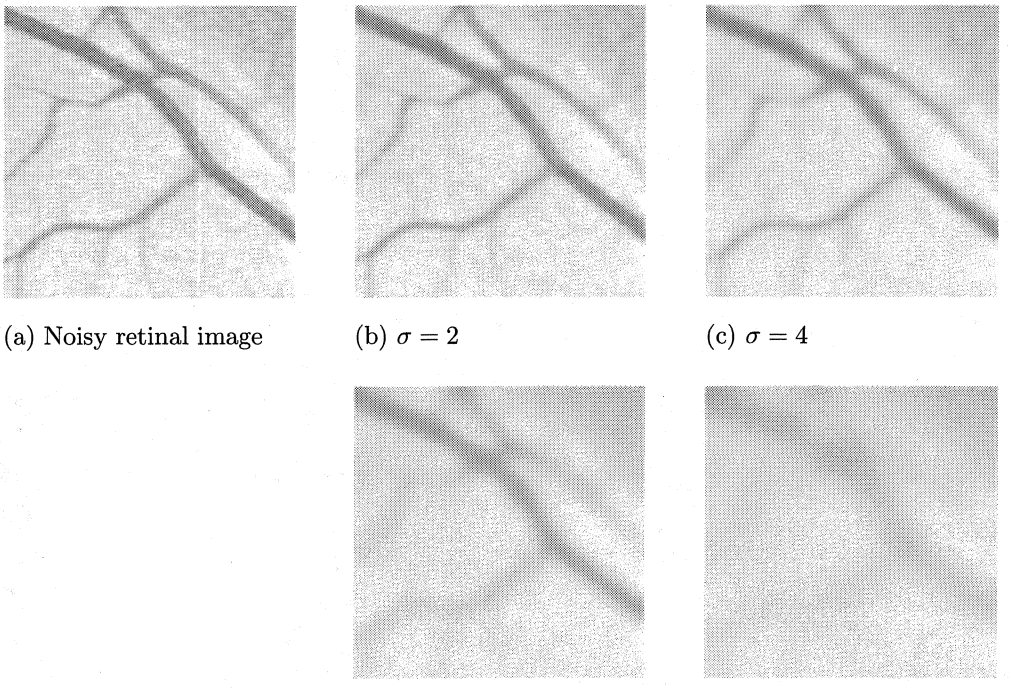


(d) S_{26}



(e) S_{64}

FIGURE 7. Noisy signal convolved with varying σ-valued Gaussian functions.



(a) Noisy retinal image (b) $\sigma = 2$ (c) $\sigma = 4$

(d) $\sigma = 8$ (e) $\sigma = 16$

FIGURE 8. Retinal Image Smoothed using Varying σ-valued Gaussian Functions.

vessel enhancement and to be consistent with [7], γ is fixed to 1 for decimation-free directional filter bank vessel enhancement.

Vessel Extraction with the Hessian

Consider an image $I : \mathbb{Z}^2 \to \mathbb{R}$ (i.e. grayscale image) and denote any $\mathbf{x} = (x, y) \in \mathbb{Z}^2$ as a pixel. For multiscale analysis of images, we define the 2D versions of the Gaussian kernel and the associated multiscale derivatives as follows.

original: $G^\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$

first derivatives: $G^\sigma_x(x,y) = -\frac{x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$, $G^\sigma_y(x,y) = -\frac{y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$

second derivatives: $G^\sigma_{xx}(x,y) = \frac{x^2-\sigma^2}{2\pi\sigma^6} e^{-\frac{x^2+y^2}{2\sigma^2}}$, $G^\sigma_{yy}(x,y) = \frac{y^2-\sigma^2}{2\pi\sigma^6} e^{-\frac{x^2+y^2}{2\sigma^2}}$

mixed derivatives: $G^\sigma_{xy}(x,y) = \frac{xy}{2\pi\sigma^6} e^{-\frac{x^2+y^2}{2\sigma^2}}$

Consequently, we have the derivatives for images

$$I_x = \sigma^\gamma G^\sigma_x * I, \qquad I_{xx} = \sigma^\gamma G^\sigma_{xx} * I, \qquad I_{xy} = \sigma^\gamma G^\sigma_{xy} * I$$

$$I_y = \sigma^\gamma G^\sigma_y * I, \qquad I_{xy} = \sigma^\gamma G^\sigma_{xy} * I, \qquad I_{yy} = \sigma^\gamma G^\sigma_{yy} * I$$

In general, for a differential operator L, $L[I] = \sigma^\gamma L[G^\sigma] * I$. The convolution Hessian is similarly defined as

$$\mathbf{H} = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{yx} & I_{yy} \end{bmatrix}. \tag{2.6}$$

The Hessian provides the curvature information for an image. Suppose a pixel $\mathbf{x_0}$ is in a vessel of an image I. Then there is a direction with almost zero curvature, $\mathbf{Hx} \approx 0$ for some $\mathbf{x}$, and the orthogonal direction has a relatively large curvature. $\mathbf{x}$ (or its orthogonal direction) is then the minimization of $\mathbf{Hx}$ (or maximization for the orthogonal direction). Theorem 2.1 shows that $\mathbf{x}$ is the eigenvector of the Hessian matrix and its eigenvalue is the amount of curvature since the Hessian is symmetric.

Figure 9 shows a pictorial example of this concept. It shows a vessel in blue with its blood-flow direction $\mathbf{v_1}$ and the orthogonal direction $\mathbf{v_2}$. Along $\mathbf{v_1}$, there

is no curvature ($H\mathbf{v_1} = 0$) and along $\mathbf{v_2}$, there is maximum curvature. $\mathbf{v_1}$ and $\mathbf{v_2}$ are the eigenvectors of the Hessian H at the start of the arrow in the Figure 9. Also, the corresponding eigenvalue λ_1 of $\mathbf{v_1}$ is equal to 0 and the corresponding eigenvalue λ_2 of $\mathbf{v_2}$ is equal to the maximum curvature.

Theorem 2.1. *Let A be an $m \times n$ matrix and $C^n = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_2 = 1\}$. Then*

$$\sqrt{\mu_1} = \min_{\mathbf{x} \in C^n} \|A\mathbf{x}\|_2, \qquad \mathbf{u_1} = \operatorname*{argmin}_{\mathbf{x} \in C^n} \|A\mathbf{x}\|_2$$

$$\sqrt{\mu_2} = \max_{\mathbf{x} \in C^n} \|A\mathbf{x}\|_2, \qquad \mathbf{u_2} = \operatorname*{argmax}_{\mathbf{x} \in C^n} \|A\mathbf{x}\|_2$$

where $\mu_i, \mathbf{u_i}$ are the eigenvalues and eigenvectors, respectively, of matrix A^TA.

Also, if A is symmetric, then

$$|\lambda_1| = \min_{\mathbf{x} \in C^n} \|A\mathbf{x}\|_2, \qquad \mathbf{v_1} = \operatorname*{argmin}_{\mathbf{x} \in C^n} \|A\mathbf{x}\|_2$$

$$|\lambda_2| = \max_{\mathbf{x} \in C^n} \|A\mathbf{x}\|_2, \qquad \mathbf{v_2} = \operatorname*{argmax}_{\mathbf{x} \in C^n} \|A\mathbf{x}\|_2$$

where $\lambda_i, \mathbf{v_i}$ are the eigenvalues and eigenvectors, respectively, of matrix A.

Proof. The maximizer/minimizer $\mathbf{x}$ of $\|A\mathbf{x}\|_2$ is a critical point of the function

$$f(\mathbf{x}, \mu) = \langle A\mathbf{x}, A\mathbf{x} \rangle - \mu(\langle \mathbf{x}, \mathbf{x} \rangle - 1)$$

where $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T\mathbf{y}$ and μ is a Lagrange multiplier.

Calculating the partial derivative of f,

$$\begin{aligned} 0 &= \partial_i f(\mathbf{x}, \mu) = (\partial_i \mathbf{x}^T)A^TA\mathbf{x} + \mathbf{x}^TA^TA(\partial_i \mathbf{x}) - \mu(\partial_i \mathbf{x}^T)\mathbf{x} - \mu\mathbf{x}^T(\partial_i \mathbf{x}) \\ &= x_i\mathbf{e}_i^T(A^TA\mathbf{x} - \mu\mathbf{x}) + (\mathbf{x}^TA^TA - \mu\mathbf{x}^T)x_i\mathbf{e}_i \\ &= x_i\mathbf{e}_i^T(A^TA\mathbf{x} - \mu\mathbf{x}) + x_i\left[\mathbf{e}_i^T(A^TA\mathbf{x} - \mu\mathbf{x})\right]^T \end{aligned}$$

where $\mathbf{x} = (x_1, x_2, ..., x_n)$. So $A^TA\mathbf{x} - \mu\mathbf{x} = 0$. And

$$0 = \partial_\mu f(\mathbf{x}, \mu) = \langle \mathbf{x}, \mathbf{x} \rangle - 1 = \|\mathbf{x}\|_2^2 - 1$$

so $\|\mathbf{x}\|_2^2 = 1$. Hence, μ and $\mathbf{x}$ is an eigenvalue and eigenvector pair of A^TA. From the definition of the $\|\cdot\|_2$ norm,

$$\begin{aligned} \|A\mathbf{x}\|_2^2 &= \langle A\mathbf{x}, A\mathbf{x}\rangle = \mathbf{x}^T A^T A\mathbf{x} \\ &= \mu\mathbf{x}^T\mathbf{x} = \mu\|x\|_2^2 = \mu. \end{aligned}$$

So

$$\sqrt{\mu_1} = \min_{\mathbf{x}\in C^n} \|A\mathbf{x}\|_2, \qquad \sqrt{\mu_2} = \max_{\mathbf{x}\in C^n} \|A\mathbf{x}\|_2$$

for some eigenvalues μ_1, μ_2 of A^TA.

For the case where A is symmetric, $A^2\mathbf{x} - \mu\mathbf{x} = 0$. If λ and $\mathbf{v}$ is an eigenvalue and eigenvector pair of A, then $A^2\mathbf{v} = \lambda^2\mathbf{v}$. So $\lambda_i^2 = \mu_i$ and $\mathbf{v}_i = \mathbf{u}_i$ for some eigenvalues λ_i and eigenvectors $\mathbf{v}_i$ for matrix A. Thus,

$$|\lambda_1| = \min_{\mathbf{x}\in C^n} \|A\mathbf{x}\|_2, \qquad |\lambda_2| = \max_{\mathbf{x}\in C^n} \|A\mathbf{x}\|_2$$

for some eigenvalues λ_1, λ_2 of A.

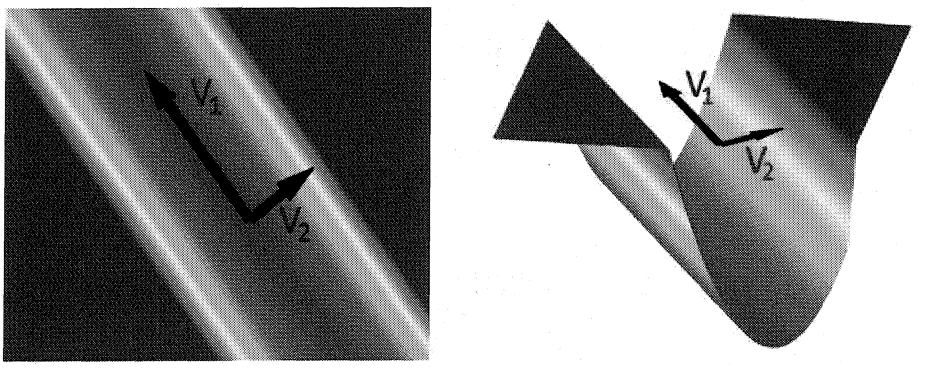
□

Ordering the two eigenvalues so that $|\lambda_1| \leq |\lambda_2|$, we define the ratio

$$R = \frac{\lambda_1}{\lambda_2} \tag{2.7}$$

as the blobness measure. On a vessel location, $\lambda_1 \approx 0$ and λ_2 is large in magnitude. Thus, the smaller R is, the more likely the pixel $\mathbf{x}$ is a part of a vessel.

But background pixels tend to be noisy so the eigenvalues of the Hessian for such pixels are small. For example, if $\lambda_1 = 0.001$, $\lambda_2 = 0.010$ then $R = 0.1$ so because R is small, we may identify the pixel with such Hessian eigenvalues to be a part of a vessel. However, the eigenvalues are small so the pixel neighborhood is locally flat. Thus, the pixel is not a part of a vessel. To account for this



(a) Artificial vessel viewed from the top with the direction with least and most curvature

(b) Artificial vessel viewed from the side with the direction with least and most curvature

FIGURE 9. An example of a vessel with its eigenvectors.

discrepancy, the second order structureness measure is used:

$$S = \sqrt{\lambda_1^2 + \lambda_2^2}. \tag{2.8}$$

S is small when the image is flat (low contrast) in the neighborhood of a specified pixel. Otherwise, the image has high contrast features. Thus, for a pixel to be considered as part of a vessel, R should be small and S should be large.

Rather than having two separate measures, the vesselness measure is defined to be

$$f^\sigma = e^{-\frac{R^2}{2\beta^2}} \left(1 - e^{-\frac{S^2}{2c^2}}\right)$$

where the parameters β and c are the scaling parameters for R^2 and S^2, respectively. Notice that to account for both a small R and large S, f^σ is large for small R and large S. Also, notice that f^σ is also dependent on the parameter σ because the multiscale version of the partial derivatives on the image depends on σ.

Because images tend to have either dark vessels with light background contrast (dark-on-light vessels) or light vessels with dark background contrast

(light-on-dark vessels), the sign of the curvature (which is also the sign of the second eigenvalue) must be taken into account. So instead, we define the vesselness measure as

$$v(\mathbf{x};\sigma) = \begin{cases} e^{-\frac{R^2}{2\beta^2}}\left(1 - e^{-\frac{S^2}{2c^2}}\right) & \text{if } \lambda_2 < 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

for light-on-dark vessel images and

$$v(\mathbf{x};\sigma) = \begin{cases} e^{-\frac{R^2}{2\beta^2}}\left(1 - e^{-\frac{S^2}{2c^2}}\right) & \text{if } \lambda_2 > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

for dark-on-light vessel images.

Finally, the vessel enhancement result is

$$v(\mathbf{x}) = \max_{\sigma>0} v(\mathbf{x};\sigma). \tag{2.11}$$

It is not complete without mentioning that a 3-D version of multiscale vessel enhancement is also available and detailed in [6].

Results on Retinal Images

With multiscale vessel enhancement defined, we are ready to look at its result on retinal images. An example of multiscale vessel enhancement on the STARE and DRIVE database can be seen in Figures 10 and 11, respectively. Figures 10(a) and 11(a) show a color image of the STARE and DRIVE sample. Typical features of a retinal image include a dark center due to low illumination of the back of the eye when the image is taken, a vessel center where the vessels branch out, and a bright circle around the vessel center called an occlusion.

Figures 10b and 11b shows the green channel of the color image. For multiscale vessel to be used, a grayscale image $I : \mathbb{Z}^2 \to \mathbb{R}$ is needed as input. But color images are functions $C = (R, G, B) : \mathbb{Z}^2 \to \mathbb{R}^3$ where R is the red channel, G

is the green channel, and B is the blue channel. This situation can be solved by converting it into a grayscale by using a linear combination $I := a_1R + a_2G + a_3B$ or selecting a single channel. The red channel is usually too saturated due to retinal images being mostly red in color. Vessels tend to have low intensity in the green channel while nonvessels tend to have higher intensity. And the blue channel tend to be too low in contrast be useful. Thus, the green channel is the best channel for multiscale vessel enhancement to be applied on and a linear combination will not work well.

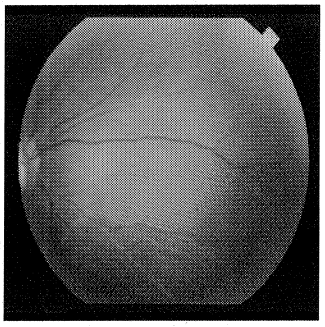
The definition of the vessel enhancement $v(\mathbf{x})$ evaluates $v(\mathbf{x};\sigma)$ for all positive values of σ. However, it is not practical to do so and there is a bias for identifying large vessels over small vessels when using large σ values. Thus, a small set of values for σ is chosen instead. We choose the optimal case of $\sigma \in \{1, 3, 5\}$ along with $\beta = 0.75$ and $\gamma = 15$. The choice of σ comes from the size of the vessels which varies about 1 to 5 pixels while β and γ are chosen from frequently used values for them.

Figure 10 and 11 shows the result of multiscale vessel enhancement. Figure 10a and 11a show the color retinal image. Figure 10b and 11b shows the green channel. Figures 10c shows the AH hand trace, 10d shows the VK hand trace and 11c show a human tracing of the vessels for comparison. Figure 10e and 11d shows the multiscale vessel enhancement result. The vessels are identified well using multiscale vessel enhancement but some of the small vessels are not identified.

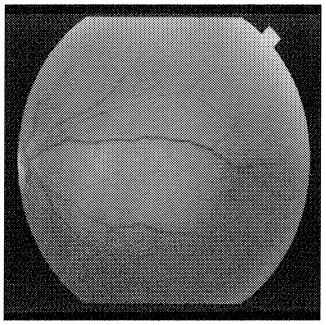
Note that the enhancement result does not directly identify vessels. To identify and extract the vessels, a threshold t must be chosen in some way:

$$\mathbf{x} \text{ is defined to be a vessel location if } v(\mathbf{x}) > t. \tag{2.12}$$

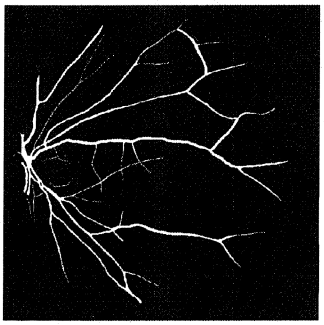
Too low of a threshold will result in too many nonvessels being identified as vessels



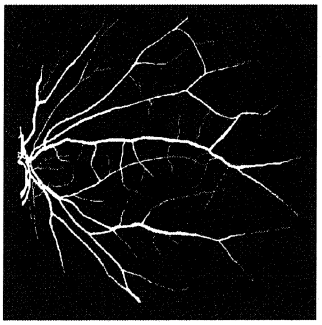
(a) The color image



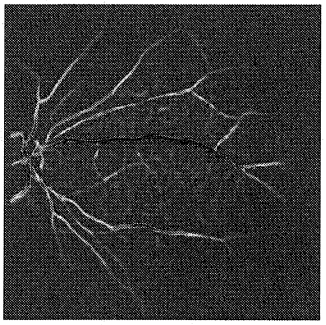
(b) The green channel of (a)



(c) AH hand trace

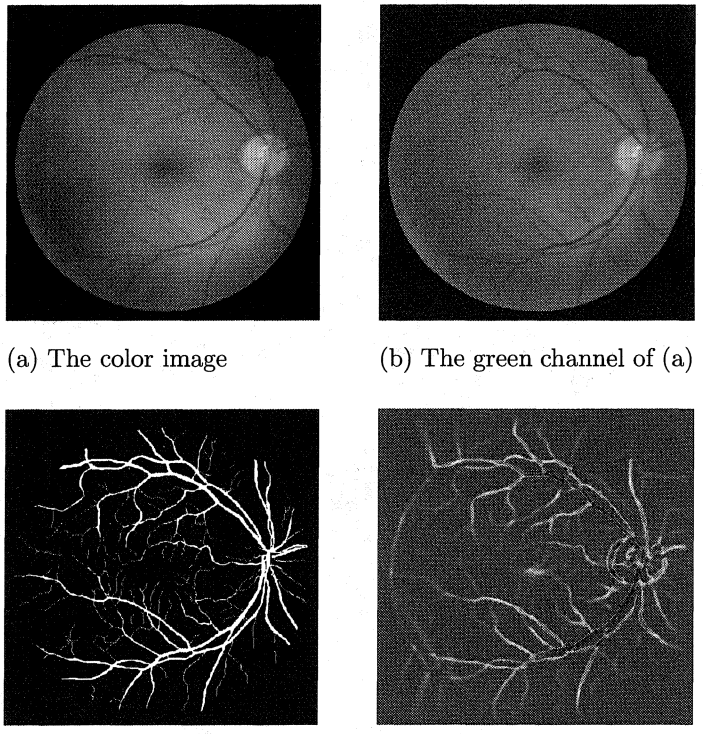


(d) VK hand trace



(e) The result of MVE

FIGURE 10. Multiscale vessel enhancement (MVE) on STARE retinal image 291.



(a) The color image

(b) The green channel of (a)

(c) Hand trace by a human observer

(d) The result of MVE

FIGURE 11. Multiscale vessel enhancement (MVE) on DRIVE retinal image 10.

(high false positive) and too high of a threshold will result in many vessels being identified as nonvessels (low true negative).

One way to choose an appropriate threshold is to perform multiscale vessel enhancement on a set of vessel images and find the best threshold that maximizes the accuracy of vessel identification. Here, accuracy is measured by

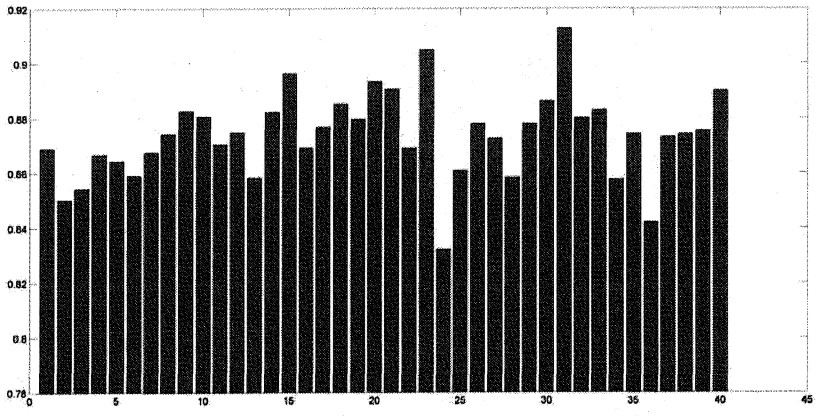
$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Number of pixels}}. \tag{2.13}$$

Note that true positive and true negative dependents on the threshold value so accuracy can be thought of as a function of the threshold.

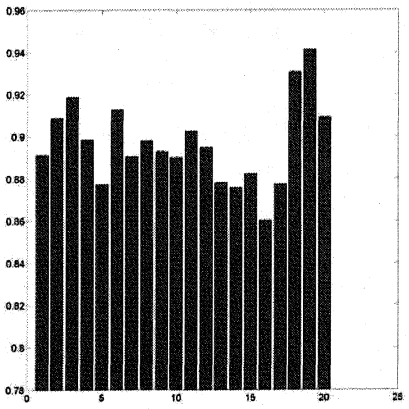
Figure 12 has a bar plot of all the maximum accuracies, as measured by equation (2.13), for each of the images for the DRIVE and STARE databases. It shows that for the DRIVE database and STARE database compared to the AH hand trace, multiscale vessel enhancement does a good job when the threshold is properly chosen. Since the VK hand traces are more detailed in that it the small vessels are more prominent, the accuracy is lower due to the background noise interfering with the identification of such small vessels.

Limitations of Multiscale Vessel Enhancement

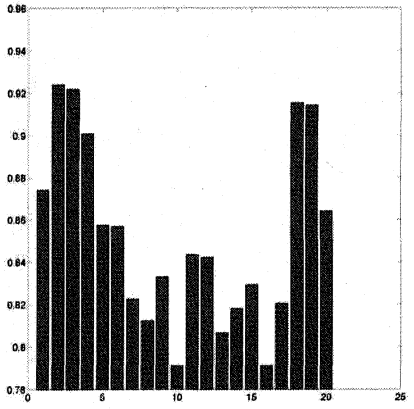
Although Gaussian smoothing allows the use of differential operators on noisy images, the calculation of the Hessian is still highly affected by noise because the Hessian is a second-order differential operator. To minimize the noise, σ can be increased so the amount of smoothing is increased. However, the details, such as small vessels, are then lost as seen in Figure 8. Also, even with a large σ, the noise may still have negative effect on the smoothed image. For example, Figure 13 shows that the noise causes slight changes in the intensity value on the circle. The slight change, in turn greatly effects the eigenvalues of the Hessian.



(a) 40 retinal images in the DRIVE database

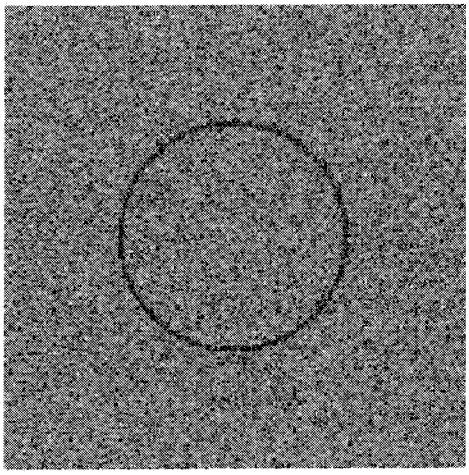


(b) 20 retinal images in the STARE database as compared to AH hand trace

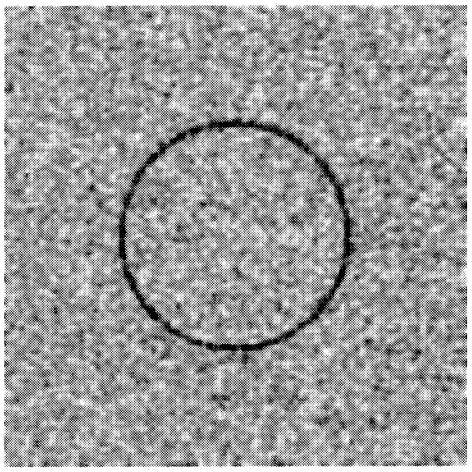


(c) 20 retinal images in the STARE database as compared to VK hand trace

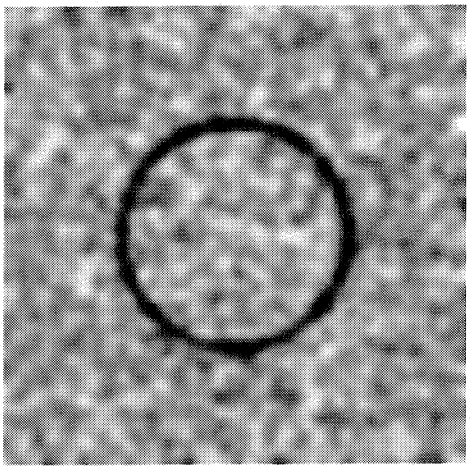
FIGURE 12. The maximum accuracy of retinal images.



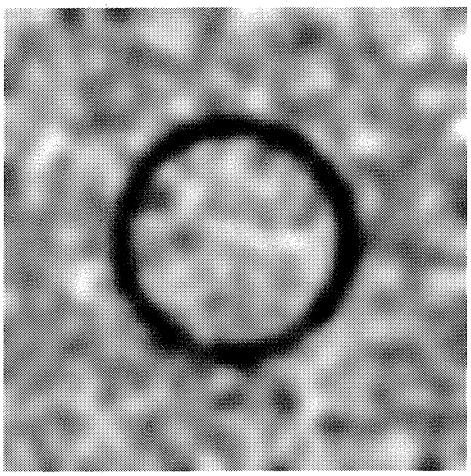
(a) Noisy Image



(b) $\sigma = 1$



(c) $\sigma = 3$



(d) $\sigma = 5$

FIGURE 13. The noisy image smoothed with different σ's.

CHAPTER 3

DIRECTIONAL FILTER BANK VESSEL ENHANCEMENT

As shown in section 2.4, the background noise causes the Hessian, when the scale σ is small, to be inaccurate and hence, small scale vessels are less accurately identified. To decrease such limitation, Truc et al. [7] proposes a decimation-free directional filter bank (DDFB) method which they claimed to be less impacted by noise when calculating the Hessian. They report an average of 5.83% increase in accuracy compared to multiscale vessel enhancement on the DRIVE database of retinal images.

Directional filter bank (DFB) was first proposed by Bamberger and Smith [19]. The theory of directional filter bank is to decompose an image into multiple direction images, each of which contains direction-specific features. Later, Park et al. [20, 21] designed its modern version of directional filter bank to fix the visual distortion present in the original directional filter bank (DFB).

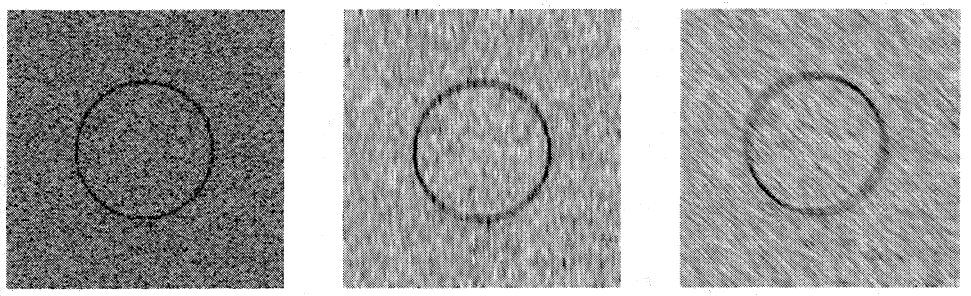
Directional filter bank (DFB) was first applied to image and video compression [22]. Compression was motivated by the theory that in an image, direction-specific information contained much of the information needed to represent an image. The sum of the dimensions of each of the directional images must be less than or equal to the dimension of the original image. Truc et al. [7], instead, uses a decimation-free directional filter bank (DDFB) in that each of the directional images is of the same dimension as the original image. Although this increases the amount of memory used to store the directional images, it can be individually processed and combined for vessel enhancement.

We start with the principle idea of Decimation-free Directional Filter Bank (DDFB) vessel enhancement. In order to perform such enhancement, wedge filters must be constructed. Section 3.2, shows the construction of the wedge filters. But before the filter construction, the discrete-time Fourier transform is first defined along with the discrete convolution theorem. Section 3.3 explains the theory of why it works using the spatial and frequency domain perspective of filter construction. Fron the theory, the diamond filter is the first filter constructed in section 3.4. This allows the wedge filters to be constructed in section 3.5.

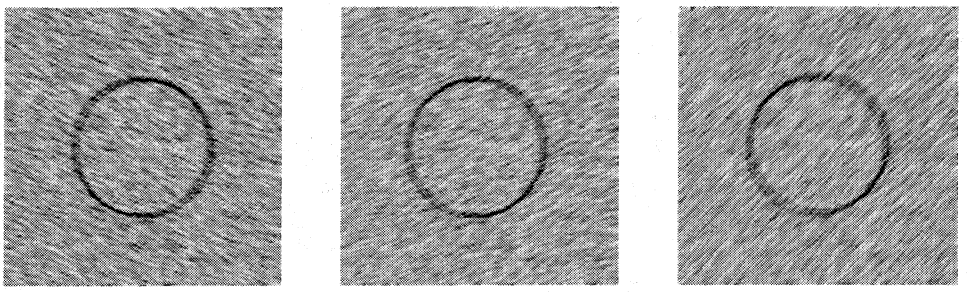
With the wedge filter constructed, the decimation-free directional filter bank (DDFB) vessel enhancement is then defined in section 3.6. Finally, section 3.7 shows the result of DDFB vessel enhancement compared with multiscale vessel enhancement (MVE) for retinal images. Section 3.8 will conclude this chapter with the application vessel extraction methods on placental images. A detailed comparison of MVE, DDFB and DDFB with homomorphic filtering is performed on the retinal images. Also, a comparison with some of the contemporary vessel extraction methods is provided for retinal images. For placental images, a comparison is also made with a neural networks based vessel extraction [12].

Decimation-Free Directional Filter Bank (DDFB)

Given an image I, the decimation-free directional filter bank (DDFB) method of vessel enhancement involves splitting such image into multiple direction images I_i, each of which contains only features that are oriented to a specific direction. Figure 14 shows an example; the portion of the circle of the same direction as θ is enhanced so that in such portion, is uniformly the same in intensity. Specifically, Figure 14a is the noisy circle image. Figure 14b has the noise now face upward, causing the top and bottom of the circle to be blurry but the left and right to be clearly shown. Figure 14c has a clear portion of the circle



(a) Noisy Image (b) $\theta = \frac{\pi}{32}$ (c) $\theta = \frac{7\pi}{32}$



(d) $\theta = \frac{13\pi}{32}$ (e) $\theta = \frac{19\pi}{32}$ (f) $\theta = \frac{25\pi}{32}$

FIGURE 14. The Noisy image enhanced in specified directions.

in the top-right and bottom-left of the circle while being blurry on the other portions. Figures 14c-f similarly have clear regions and opposing blurry regions whose location is dependent of the angle ϕ. Notice that when compared to Gaussian smoothing seen in Figure 13, the circle does not increase in width.

These direction-specific feature images can be constructed by convolving with a set of filters, called wedge filters, $\{F_i\}_{i=1}^n$ where $F_i : \mathbb{Z}^2 \to \mathbb{R}$ is a set of filters with direction $\theta_i = \frac{(2i-1)\pi}{2n}$.

A directional image is defined to be:

$$I_i = F_i * I.$$

The directional images are then individually vessel enhanced and the maximum

response is the directional filter bank vessel enhancement result:

$$v(\mathbf{x}) = \max_{1 \le i \le n} v_i(\mathbf{x})$$

where $v_i(\mathbf{x})$ are the later discussed individual vessel enhancements. Figure 15 shows a diagram of this process.

Constructing the Filters

The set of filters $\{F_i\}_{i=1}^n$ used to obtain the directional images I_i can be obtained through the discrete convolution theorem:

Theorem 3.1. *Let* $\mathbf{x}, \mathbf{y} : \{1, 2, ..., M\} \times \{1, 2, ..., N\} \to \mathbb{R}$, *then*

$$\mathbf{x} * \mathbf{y} = \mathfrak{F}^{-1}[\mathfrak{F}\{\mathbf{x}\} \cdot \mathfrak{F}\{\mathbf{y}\}]$$

where the discrete-time Fourier transform ($\mathfrak{F}$ *or DTFT) is defined as*

$$C(k,l) := \mathfrak{F}\{I\}(k,l) = \sum_{n=1}^{N} \sum_{m=1}^{M} I(m,n) e^{-2\pi i \left(\frac{mk}{M} + \frac{nl}{N}\right)}$$

and the inverse discrete Fourier transform ($\mathfrak{F}^{-1}$*) is*

$$I(m,n) = \mathfrak{F}^{-1}\{C\}(m,n) = \frac{1}{MN} \sum_{l=1}^{N} \sum_{k=1}^{M} C(k,l) e^{2\pi i \left(\frac{km}{M} + \frac{ln}{N}\right)}.$$

The directional image, I_i, has features oriented in a specific direction θ_i so its DTFT will contain high intensity in a wedge-shaped section of the transform, seen in Figure 16a-b.

Using appropriate angled wedge filter multiplied with the DTFT of the line image, the line features can be separately studied. Figure 16c-d shows an example of this. Here, the line image has a DTFT of dots lined up in two directions. Using two wedge filters angled in the direction as the two dotted lines, we can isolate each of the stripes. To do so, the DTFT is multiplied with the appropriate wedge filter and the inverse DTFT is applied. Similarly, the discrete convolution theorem

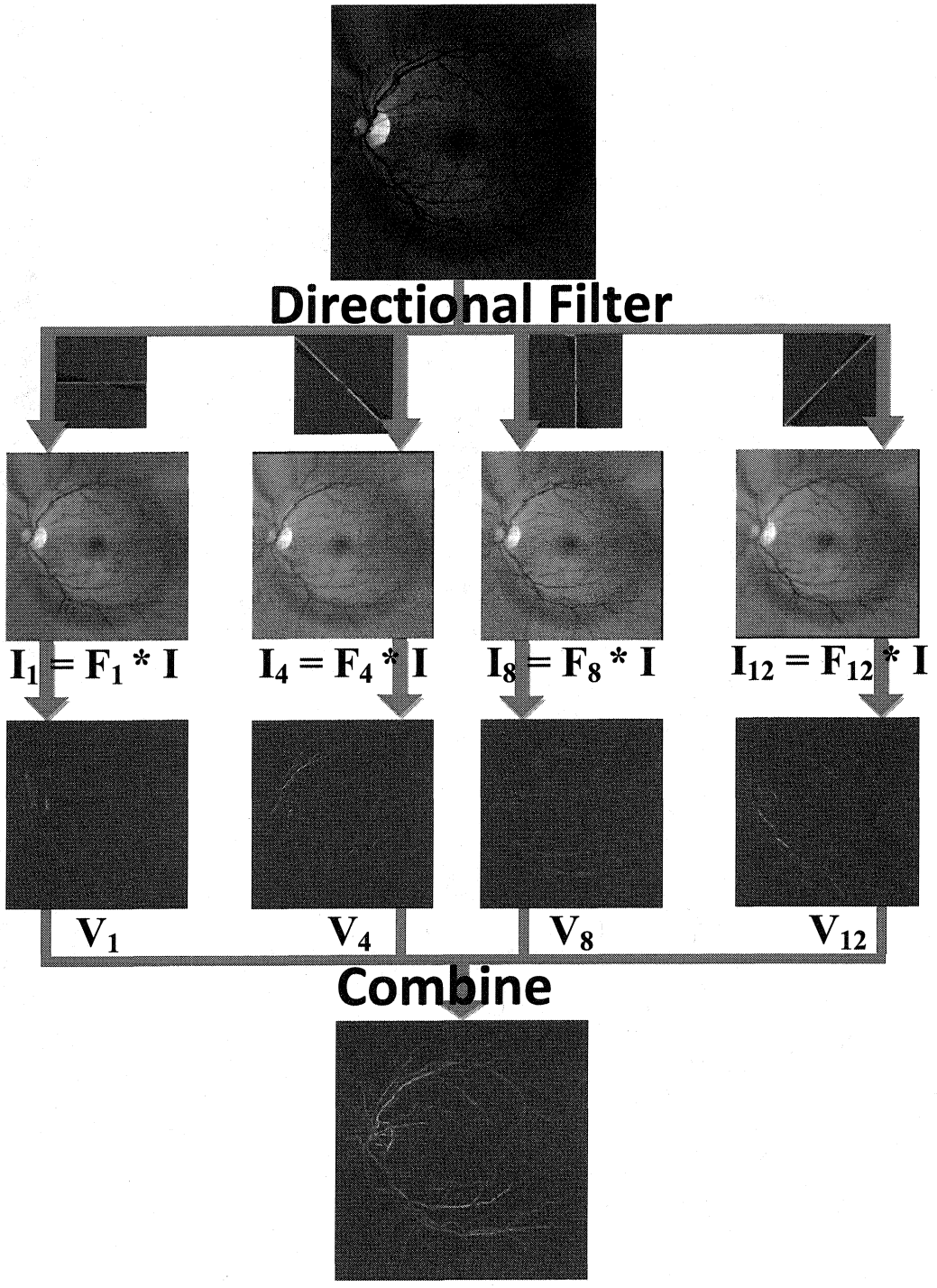
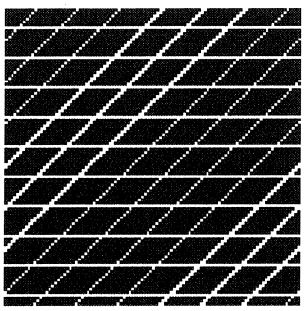
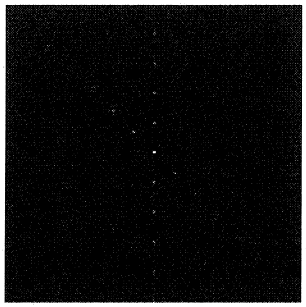


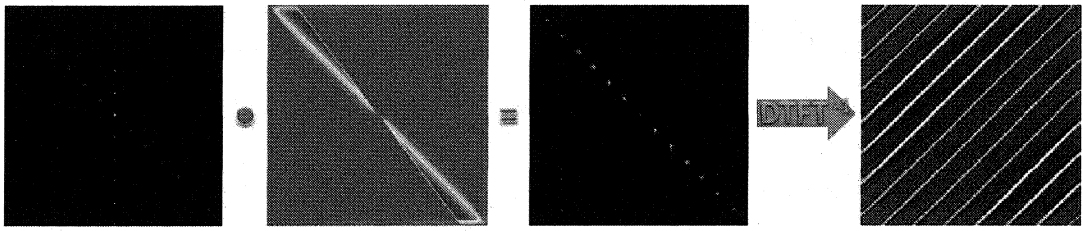
FIGURE 15. A diagram of directional filter bank vessel enhancement.



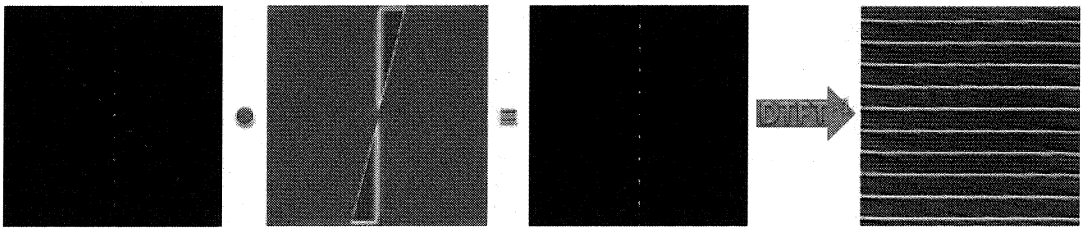
(a) Artificial image



(b) DTFT of image



(c) Product of the DTFT and $\frac{9\pi}{32}$ wedge filter. The inverse DTFT of such product.



(d) Product of the DTFT and $\frac{17\pi}{32}$ wedge filter. The inverse DTFT of such product.

FIGURE 16. A demonstration of wedge filter multiplication.

states that the result can be obtained by convolving the image in Figure 16a with the inverse DTFT of the wedge filters.

One of the methods to construct the wedge filters comes from initially constructing a filter whose DTFT is a diamond filter D. Section 3.4 will show a construction of the diamond filter and section 3.5 will complete the construction of the wedge filters.

Before we can proceed to construct these filters, the theory behind the directional filter banks construction will be covered in section 3.3.

Spatial and Frequency domain

Two domains are artificially labeled: the vessel image and filters are considered to belong in the spatial domain while their respective discrete-time Fourier transform (DTFT) are considered to belong in the frequency domain. The transforms $\mathfrak{F}$ and $\mathfrak{F}^{-1}$ can be considered as a linear mapping between these two domains. Here, we will establish some of the properties between these two domains that will be necessary for the wedge filter construction.

The spatial domain in the discrete case has the domain $\mathbb{Z}^2$ while the frequency domain has the domain $\mathbb{R}^2$ (or an equally-spaced subset of it for the discrete case). For images, the domain is $\{1,2,...,M\} \times \{1,2,...,N\}$ but a function extension can be performed on the image so that its domain is $\mathbb{Z}^2$. This extension can be done by either assigning zero to the domain outside of $\{1,2,...,M\} \times \{1,2,...,N\}$, by cycling the image

$$\tilde{\mathbf{I}}(i,j) = \mathbf{I}\left(i - M\lfloor\frac{i}{M}\rfloor, j - N\lfloor\frac{j}{N}\rfloor\right) \text{ for all } (i,j) \in \mathbb{Z}^2$$

or by reflection

$$\tilde{\mathbf{I}}(i,j) = \mathbf{I}(s(i;M), s(j;N)) \text{ for all } (i,j) \in \mathbb{Z}^2$$

where $s(n;N)$ was defined in equation (1.1). To be consistent with signal processing literature, the reflection extension is used.

Functions in the frequency domain are of period 2π so the functions can be completely studied in the domain $[-\pi,\pi]^2$. The spatial domain functions are real-valued in this thesis because the functions of interest are images and filters, the frequency domain functions in general are complex-valued. Hence, most of the figures involving the frequency domain functions seen in this thesis are the modulus of such functions.

Definition 3.1. *Let $f:\mathbb{R}^2\to\mathbb{C}$ s.t. $f(\omega_1,\omega_2)=\mathfrak{F}\{I\}(\omega_1,\omega_2)$. x-axis modulation of f is defined as $f(\omega_1+\pi,\omega_2)$, y-axis modulation of f is defined as $f(\omega_1,\omega_2+\pi)$, and xy-axis modulation of f is defined as $f(\omega_1+\pi,\omega_2+\pi)$.*

Definition 3.2. *Let $A_1:\mathbb{Z}^2\to\mathbb{Z}^2$ s.t. $A_1(n_1,n_2)=(-1)^{n_1}=e^{-\pi i n_1}$, $A_2:\mathbb{Z}^2\to\mathbb{Z}^2$ s.t. $A_2(n_1,n_2)=(-1)^{n_2}=e^{-\pi i n_2}$, and $A_{12}:\mathbb{Z}^2\to\mathbb{Z}^2$ s.t. $A_{12}(n_1,n_2)=(-1)^{n_1+n_2}=e^{-\pi i(n_1+n_2)}$. A_1, A_2, and A_{12} is defined as the alternating functions.*

Lemma 3.2. *Modulation in the frequency domain is equivalent to multiplication by the appropriate alternating function in the spatial domain.*

Proof.

$$\begin{aligned}\mathfrak{F}\{A_1 I\}(\omega_1,\omega_2) &= \sum_{(n_1,n_2)\in\mathbb{Z}^2} A_1(n_1,n_2)I(n_1,n_2)e^{-i(n_1\omega_1+n_2\omega_2)}\\ &= \sum_{(n_1,n_2)\in\mathbb{Z}^2} I(n_1,n_2)e^{-i(n_1\omega_1+n_2\omega_2)-\pi i n_1}\\ &= \sum_{(n_1,n_2)\in\mathbb{Z}^2} I(n_1,n_2)e^{-i(n_1(\omega_1+\pi)+n_2\omega_2)} = f(\omega_1+\pi,\omega_2).\end{aligned}$$

Similarly $\mathfrak{F}\{A_2 I\}(\omega_1, \omega_2) = f(\omega_1, \omega_2 + \pi)$, and

$$\begin{aligned}
\mathfrak{F}\{A_{12} I\}(\omega_1, \omega_2) &= \sum_{(n_1,n_2)\in\mathbb{Z}^2} A_{12}(n_1, n_2) I(n_1, n_2) e^{-i(n_1\omega_1 + n_2\omega_2)} \\
&= \sum_{(n_1,n_2)\in\mathbb{Z}^2} I(n_1, n_2) e^{-i(n_1\omega_1 + n_2\omega_2) - \pi i(n_1+n_2)} \\
&= \sum_{(n_1,n_2)\in\mathbb{Z}^2} I(n_1, n_2) e^{-i(n_1(\omega_1+\pi) + n_2(\omega_2+\pi))} = f(\omega_1 + \pi, \omega_2 + \pi).
\end{aligned}$$

□

Theorem 3.3. *Let $M \in \mathbb{Z}^{2\times 2}$ be a nonsingular matrix of integers $M : \mathbb{Z}^2 \to \mathbb{Z}^2$, also called a resampling matrix. Then the DTFT of the composition $I \circ M$ is*

$$\mathfrak{F}\{I \circ M\}(\omega) = f((M^{-1})^T \omega) \det(M^{-1})$$

Proof. Let $\mathbf{n} = (n_1, n_2), \omega = (\omega_1, \omega_2)$. Then

$$\mathfrak{F}\{I \circ M\}(\omega_1, \omega_2) = \sum_{(n_1,n_2)\in\mathbb{Z}^2} I \circ M(n_1, n_2) e^{-i(n_1\omega_1 + n_2\omega_2)} = \sum_{\mathbf{n}\in\mathbb{Z}^2} I \circ M(\mathbf{n}) e^{-i\mathbf{n}\cdot\omega}. \quad (3.1)$$

Recall the multivariate change of variable theorem [23],

$$\int_{\phi(U)} g(\mathbf{v}) d\mathbf{v} = \int_U g \circ \phi(\mathbf{u}) |\det(\mathbf{D}\phi)(\mathbf{u})| d\mathbf{u}.$$

where $U \subseteq \mathbb{R}^n$ is an open set, $\phi : U \to \mathbb{R}^n$ injective differentiable function with continuous partial derivatives and nonzero Jacobian on U, $\mathbf{D}\phi$ is the Jacobian matrix containing the partial derivatives of ϕ, and f is a continuous real-valued, compactly supported function with its support contained in $\phi(U)$.

A discrete version of mentioned theorem can be derived:

$$\sum_{\mathbf{n}\in\phi(\mathbb{Z}^2)} g(\mathbf{n}) = \sum_{\mathbf{n}\in\mathbb{Z}^2} g \circ \phi(\mathbf{n}) |\det(\mathbf{D}\phi(\mathbf{n}))|.$$

Using the linear transform $\phi(\mathbf{n}) = M^{-1}\mathbf{n}$ on expression (3.1),

$$\begin{aligned}
\mathfrak{F}\{I \circ M\}(\omega) &= \sum_{\mathbf{n}\in\mathbb{Z}^2} I(\mathbf{n}) e^{-i(M^{-1}\mathbf{n})\cdot\omega} |\det(M^{-1})| = \sum_{\mathbf{n}\in\mathbb{Z}^2} I(\mathbf{n}) e^{-i(M^{-1}\mathbf{n})^T\omega} |\det(M^{-1})| \\
&= \sum_{\mathbf{n}\in\mathbb{Z}^2} I(\mathbf{n}) e^{-i(\mathbf{n})^T[(M^{-1})^T\omega]} |\det(M^{-1})| = \sum_{\mathbf{n}\in\mathbb{Z}^2} I(\mathbf{n}) e^{-i\mathbf{n}\cdot[(M^{-1})^T\omega]} |\det(M^{-1})| \\
&= f((M^{-1})^T\omega)|\det(M^{-1})|.
\end{aligned}$$

□

Definition 3.3. *For a resampling matrix $M \in \mathbb{Z}^{2\times 2}$, downsampling is defined as*

$$\mathbf{I}_M(\mathbf{n}) = \mathbf{I}(M\mathbf{n}), \tag{3.2}$$

and upsampling is defined as

$$\mathbf{I}^M(\mathbf{n}) = \begin{cases} \mathbf{I}(M^{-1}\mathbf{n}) & \text{if } \mathbf{n} \in M[\mathbb{Z}^2] \\ 0 & \textit{otherwise} \end{cases} \tag{3.3}$$

Lemma 3.4. *Downsampling in the spatial domain corresponds to upsampling in the frequency domain and vice versa.*

Proof. It can be easily proven from 3.3. For downsampling,

$$\begin{aligned}
\mathfrak{F}\{I_M\}(\omega) &= \mathfrak{F}\{I \circ M\}(\omega) = f((M^{-1})^T\omega)|\det(M^{-1})| \\
&= f((M^T)^{-1}\omega)|\det(M^{-1})| = |\det(M^{-1})| \cdot f^{M^T}(\omega)
\end{aligned}$$

and for upsampling,

$$\begin{aligned}
\mathfrak{F}\{I^M\}(\omega) &= \mathfrak{F}\{I \circ M^{-1}\}(\omega) = f(M^T\omega)|\det(M)| \\
&= |\det(M)| \cdot f_{M^T}(\omega)
\end{aligned}$$

□

Because $\mathfrak{F}\{I \circ M\}(\omega) = f((M^{-1})^T\omega)|\det(M^{-1})|$, downsampling in the spatial domain corresponds to upsampling in the frequency domain and vice versa.

Table 4 sums up these properties.

TABLE 4. The Spatial-Frequency Duality

	Spatial Domain $I(n_1, n_2)$	Frequency Domain $f(\omega_1, \omega_2)$
Convolution Theorem	$(I_1 * I_2)(n_1, n_2)$	$f_1(\omega_1, \omega_2) \cdot f_2(\omega_1, \omega_2)$
Modulation in x-axis	$(-1)^{n_1} I(n_1, n_2)$	$f(\omega_1 + \pi, \omega_2)$
Modulation in y-axis	$(-1)^{n_2} I(n_1, n_2)$	$f(\omega_1, \omega_2 + \pi)$
Modulation in xy-axis	$(-1)^{n_1+n_2} I(n_1, n_2)$	$f(\omega_1 + \pi, \omega_2 + \pi)$
Linear Transform	$I \circ M(n_1, n_2)$	$f((M^{-1})^T\omega)\|\det(M^{-1})\|)$
Resampling	$I_M(n_1, n_2)$	$\|\det(M^{-1})\| \cdot f^{M^T}(\omega_1, \omega_2)$
	$I^M(n_1, n_2)$	$\|\det(M)\| \cdot f_{M^T}(\omega_1, \omega_2)$

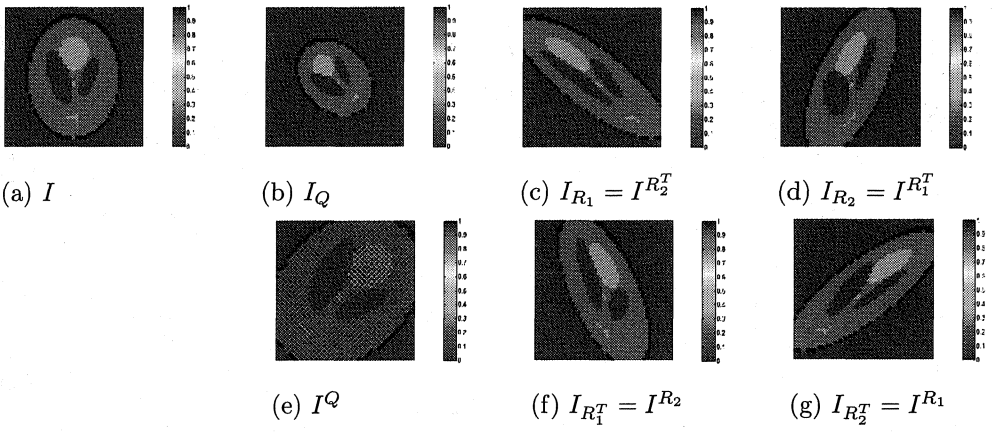
In this thesis, three resampling matrices will be used. The quincunx matrix[1]

$$Q = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix},$$

and skew matrices

$$R_1 = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \text{ and } R_2 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

[1]We may have used $Q = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$ for the quincunx matrix.



(a) I (b) I_Q (c) $I_{R_1} = I^{R_2^T}$ (d) $I_{R_2} = I^{R_1^T}$

(e) I^Q (f) $I_{R_1^T} = I^{R_2}$ (g) $I_{R_2^T} = I^{R_1}$

FIGURE 17. A (phantom) image with different samplings.

Notice that for the quincunx matrix, $(Q^{-1})^T = \frac{1}{2}Q$, $\det(Q) = 2$, and $\det(Q^{-1}) = \frac{1}{2}$. Thus,

$$\mathfrak{F}\{I_Q\}(\omega) = \frac{1}{2}f((Q^T)^{-1}\omega) = \frac{1}{2}f^{Q^T}(\omega),$$

and

$$\mathfrak{F}\{I^Q\}(\omega) = 2f(Q^T\omega) = 2f_{Q^T}(\omega).$$

For the skewing matrices, $(R_1^{-1})^T = R_2$, $(R_2^{-1})^T = R_1$, $((R_1^T)^{-1})^T = R_2^T$ and $((R_2^T)^{-1})^T = R_1^T$. Thus, $\mathfrak{F}\{I_{R_i}\}(\omega) = f^{R_i^T}(\omega)$, $\mathfrak{F}\{I^{R_i}\}(\omega) = f_{R_i}(\omega)$, $\mathfrak{F}\{I_{R_i^T}\}(\omega) = f^{R_i}(\omega)$, and $\mathfrak{F}\{I^{R_i^T}\}(\omega) = f_{R_i^T}(\omega)$.

Figure 17 shows the effects of upsampling and downsampling on an image. Notice that the quincunx downsampling is a 45° counterclockwise rotation and a 25% decrease in size while quincunx upsampling is a $-45°$ rotation and a 25% increase in size. The skewing matrices function as their name imply; it shifts the image in the direction of one of the axis. In this case, R_i and R_i^T skew the image in the x-axis and y-axis.

Finite Impulse Filter

There are multiple ways to construct the wedge filters. The simplest method involves the binary function $b : \mathbb{R}^2 \to \{0, 1\}$ so that $b(x) = 1$ if x is in the wedge and $b(x) = 0$ if x is outside the wedge. The wedge filter function would then be the inverse DTFT of b. However, because such wedge filter will not have a compact support and computer systems work well with discrete functions with compact support, such construction is ill-posed. Hence, having compact support is preferred.

In terms of signal processing, a function has finite impulse response (FIR) if it has compact support. Otherwise, the function has infinite impulse response (IIR). Because our construction of the wedges requires convolution, modulation, and resampling, it is necessary to have FIR (or IIR) filters remain FIR (or IIR) after such operation. Fortunately, using the definition of DTFT, it is trivial to prove that

Property 1: If f and g are FIR (or IIR) then $f * g$ is FIR (or IIR),

Property 2: if f is FIR (or IIR) then the (spatial domain equivalent of) modulation of f is FIR (or IIR), and

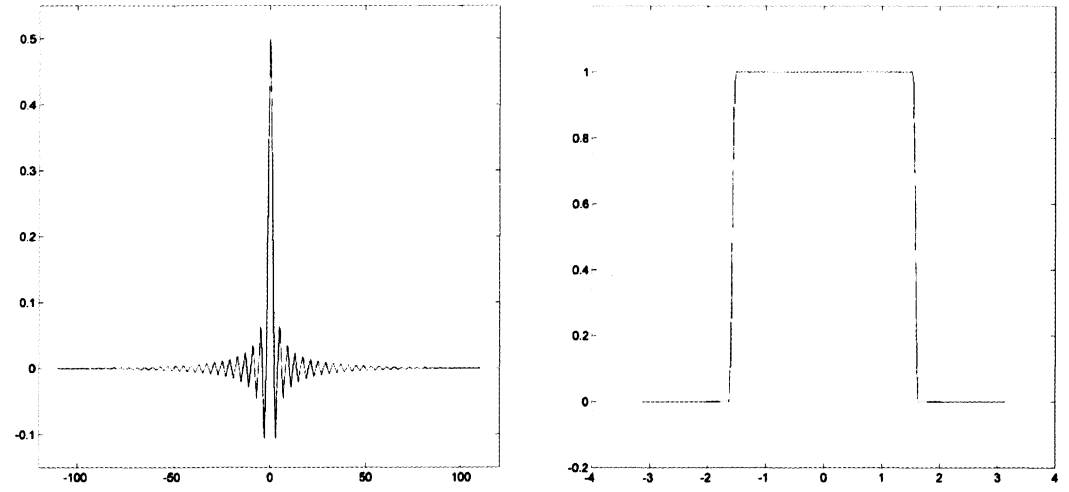
Property 3: if f is FIR (or IIR) then the downsampling and upsampling of f is FIR (or IIR).

We are now ready to construct filter functions.

Diamond Filter

The diamond construction starts with a one-dimensional filter $\tilde{f} : \mathbb{Z} \to \mathbb{R}$ constructed from [24][2]. The function $\tilde{f}$ is specifically designed so it is FIR and

[2]The one-dimensional filter is implemented in MATLAB's firl function.



(a) The signal (in spatial domain)

(b) The modulus of the DTFT of the signal (frequency domain)

FIGURE 18. The signal $\tilde{f} : \mathbb{Z} \to \mathbb{R}$ used to construct the diamond filter.

that the modulus of its DTFT ($|\mathfrak{F}^{-1}\{\tilde{f}\}|$) approximates $\chi_{[-\pi/2,\pi/2]}$ where χ_A is the indicator function of subset A. Figure 18 shows the signal with its DTFT. Notice that Figure 18b approximates $\chi_{[-\pi/2,\pi/2]}$ well. The background detail about filter $\tilde{f}$ is beyond the scope of this thesis. We refer to [24] and [25] for more details.

The square filter is then constructed: $S_q : \mathbb{Z}^2 \to \mathbb{R}$ such that $S_q(i,j) = \tilde{f}(i) \cdot \tilde{f}(j)$. We can see why the magnitude of the DTFT of the filter is square-shaped from the fact that

$$|\mathfrak{F}\{S_q\}(\omega_1,\omega_2)| = |\mathfrak{F}\{\tilde{f}(\cdot)\tilde{f}(\cdot)\}(\omega_1,\omega_2)| = |\mathfrak{F}\{\tilde{f}\}(\omega_1)| \cdot |\mathfrak{F}\{\tilde{f}\}(\omega_2)|$$
$$\approx \chi_{[-\frac{\pi}{2},\frac{\pi}{2}]}(\omega_1)\chi_{[-\frac{\pi}{2},\frac{\pi}{2}]}(\omega_2) = \chi_{[-\frac{\pi}{2},\frac{\pi}{2}]^2}(\omega_1,\omega_2)$$

A corner filter is then constructed by xy-axis modulation $C : \mathbb{Z}^2 \to \mathbb{R}$ such that $C(i,j) = (-1)^{i+j} S_q(i,j)$.

So if $A_{12} : \mathbb{R}^2 \to \mathbb{R}$ such that $A_{12}(i,j) = (-1)^{i+j}$, then from the modulation in xy-axis property,

$$|\mathfrak{F}\{C\}(\omega_1,\omega_2)| = |\mathfrak{F}\{A_{12}\cdot S_q\}(\omega_1,\omega_2)| = \mathfrak{F}\{S_q\}(\omega_1+\pi,\omega_2+\pi)|$$
$$\approx \chi_{[-\frac{\pi}{2},\frac{\pi}{2}]}(\omega_1)\chi_{[-\frac{\pi}{2},\frac{\pi}{2}]}(\omega_2) = \chi_{([-1,-\frac{\pi}{2}]\cup[\frac{\pi}{2},1])^2}(\omega_1,\omega_2)$$

Combined, with the square filter, the checkerboard filter is created by defining

$$C_r : \mathbb{Z}^2 \to \mathbb{R} \text{ such that } C_r(i,j) = S_q(i,j) + C(i,j).$$

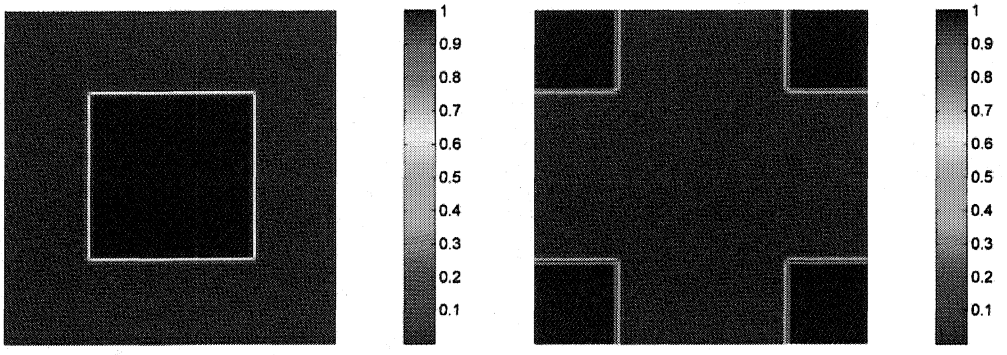
Finally, the diamond filter can be constructed by quincunx downsampling (which is upsampling in the frequency domain):

$$D : \mathbb{Z}^2 \to \mathbb{R} \text{ s.t. } D(i,j) = C_r\left(Q\begin{bmatrix} i \\ j \end{bmatrix}\right) = C_r(i-j, i+j).$$

Figure 19 shows each of these filters in progression. Notice that we start with an FIR filter function $\tilde{f}$ to form a square filter function S_q, which is then also FIR. Then through modulation performed by multiplying with an alternating function, another FIR filter function, C, is defined. Another FIR filter function C_r is defined from the summation of two FIR filter functions. Finally, the diamond filter is formed from a downsampling. From lemma 3.4 and property 3, the diamond filter is then an FIR filter function.

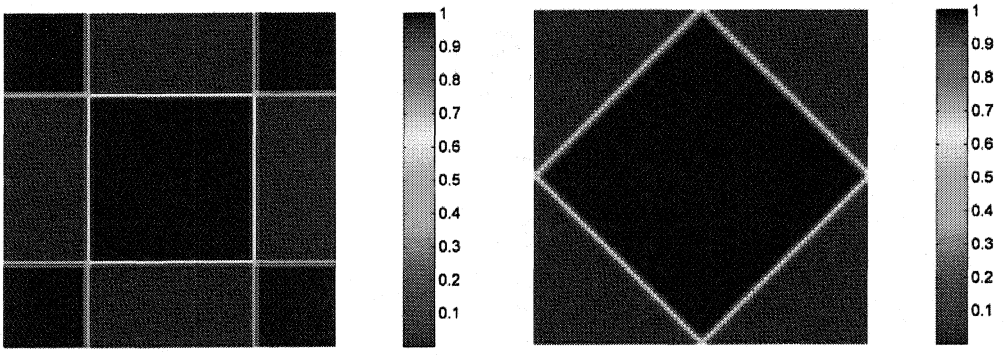
Constructing the Wedge Filters

The wedge filters are constructed from a diamond filter $D(\omega_1,\omega_2)$. Figure 20 shows the first few steps of the construction. It starts with the previously constructed diamond filter. Two hourglass filters are constructed by modulation: $H_0(i,j) := (-1)^i D(i,j)$ and $H_1(i,j) := (-1)^j D(i,j)$. The two hourglass filters form the first level of filters $\mathbb{F}_1 = \{H_0, H_1\}$. The second level of wedge filters are formed by quincunx upsampling the hourglass filters and convolving with the



(a) Square filter

(b) Corner filter



(c) Checkerboard filter

(d) Diamond filter

FIGURE 19. The magnitude of the DTFT, $|\mathfrak{F}\{\cdot\}|$, of each of the filters.

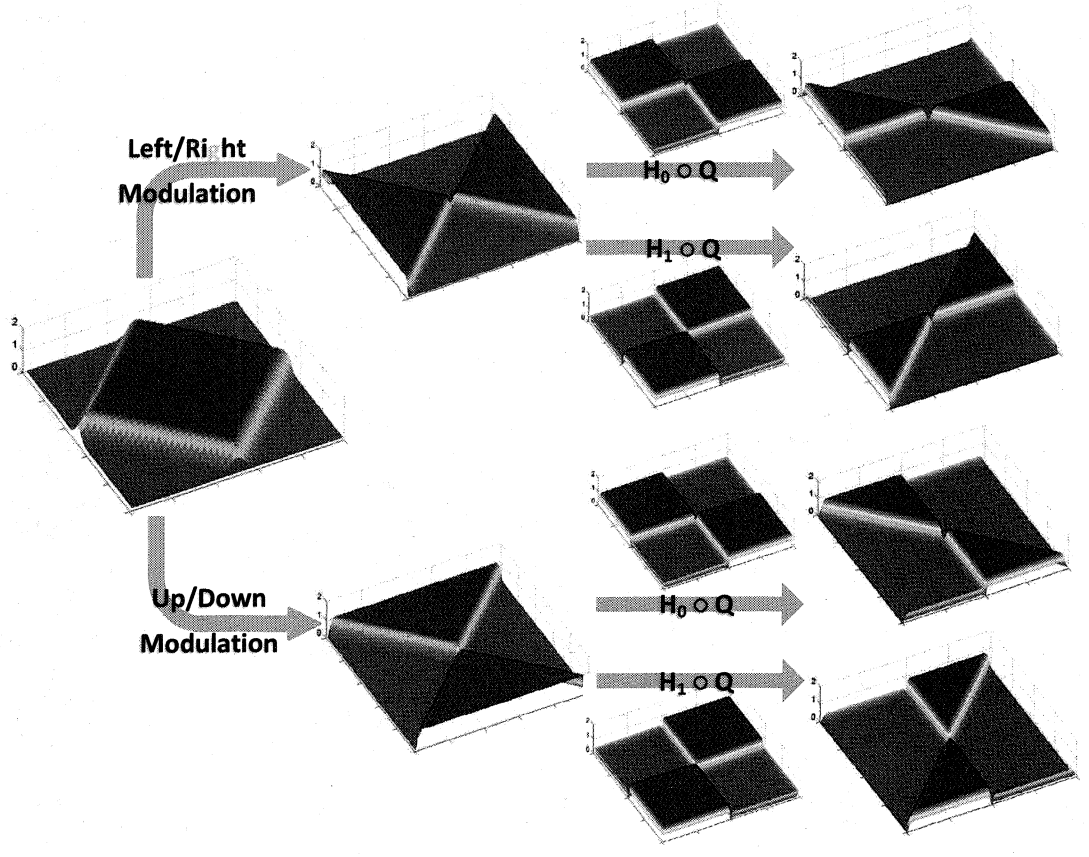


FIGURE 20. The diagram of the constructing of wedge filters up to 2 levels.

hourglass filters. For notational convenience, fg will mean composition $f \circ g$. The second level of wedge filters consist of the collection

$$\mathbb{F}_2 = \{H_0 * H_1 Q, H_1 * H_1 Q, H_1 * H_0 Q, H_0 * H_0 Q\}.$$

The third level of wedge filters is formed by convolving the elements of $\mathbb{F}_2$ with filters of the form $H_i R_j Q$ and $H_i R_j^T Q$ where $i \in \{0, 1\}$ and $j \in \{1, 2\}$. The

third level of wedge filters are then

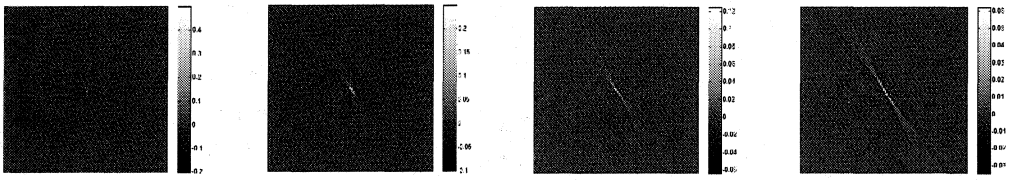
$$\begin{aligned}\mathbb{F}_3 = \{&H_0 * H_1Q * H_0R_1Q, H_0 * H_1Q * H_1R_1Q,\\ &H_1 * H_1Q * H_0R_1^TQ, H_1 * H_1Q * H_1R_1^TQ,\\ &H_1 * H_0Q * H_1R_2Q, H_1 * H_0Q * H_0R_2Q,\\ &H_0 * H_0Q * H_1R_2^TQ, H_0 * H_0Q * H_0R_2^TQ\}.\end{aligned}$$

The fourth level of wedge filters is formed by convolving the elements of $\mathbb{F}_3$ with filters of the form $H_iR_jQR_kQQ$ and transpose versions of R_j, R_k where $i \in \{0,1\}$ and $j,k \in \{1,2\}$.

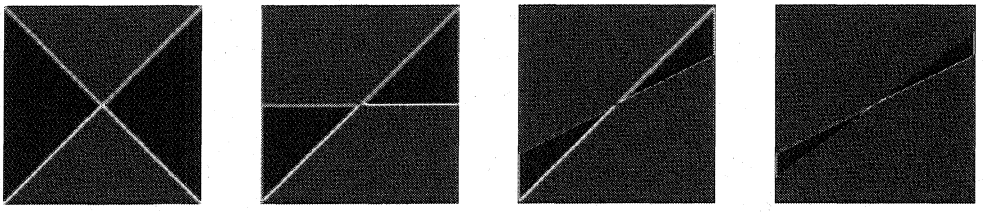
$$\begin{aligned}\mathbb{F}_4 = \{&H_0 * H_1Q * H_0R_1Q * H_1R_2^TQR_1QQ, \quad H_0 * H_1Q * H_0R_1Q * H_0R_2^TQR_1QQ,\\ &H_0 * H_1Q * H_1R_1QH_0R_2^TQR_1QQ, \quad H_0 * H_1Q * H_1R_1Q * H_1R_1QR_1QQ,\\ &H_1 * H_1Q * H_0R_1^TQ * H_1R_2^TQR_1^TQQ, \quad H_1 * H_1Q * H_0R_1^TQ * H_0R_2^TQR_1^TQQ,\\ &H_1 * H_1Q * H_1R_1^TQ * H_0R_1QR_1^TQQ, \quad H_1 * H_1Q * H_1R_1^TQ * H_1R_1QR_1^TQQ,\\ &H_1 * H_0Q * H_1R_2Q * H_0R_1^TQR_2QQ, \quad H_1 * H_0Q * H_1R_2Q * H_1R_1^TQR_2QQ,\\ &H_1 * H_0Q * H_0R_2QH_1R_2QR_2QQ, \quad H_1 * H_0Q * H_0R_2Q * H_0R_2QR_2QQ,\\ &H_0 * H_0Q * H_0R_2^TQ * H_0R_1^TQR_2^TQQ, \quad H_0 * H_0Q * H_1R_2^TQ * H_1R_1^TQR_2^TQQ,\\ &H_0 * H_0Q * H_0R_2^TQ * H_1R_2QR_2^TQQ, \quad H_0 * H_0Q * H_0R_2^TQ * H_0R_2QR_2^TQQ\}.\end{aligned}$$

Higher levels can be defined similarily; however, levels that are higher than four levels do not improve accuracy performance by much while requiring much more computational time [7]. For the construction of higher level wedge filters, we refer interested readers to [26] for details.

As seen from Figure 20 and Figure 21, the wedges in each level are subsections of the previous wedges. Figure 20a shows one of the hourglass filter function shown as a surface plot. Notice because it is FIR, the nonzero values are



(a) H_0 (b) A filter from F_2 (c) A filter from F_3 (d) A filter from F_4



(e) DTFT of filter H_0 (f) DTFT of filter (b) (g) DTFT of filter (c) (h) DTFT of filter (d)

FIGURE 21. A sample of the different wedge filters and its DTFT.

self-contained in the center of the plot. Figure 21e shows the DTFT of such hourglass filter. Similarly, some of its subsections are shown along with its DTFT. Notice that as the wedge becomes thinner in the frequency domain, its support elongates.

DDFB-based Vessel Enhancement

With the wedge filters, the directional images can now be obtained as $I_i := f_i * I$, where f_i are the unique filters in $\mathbb{F}_4$. Each I_i contains sections of vessels oriented in a direction θ_i. The next step for DDFB vessel enhancement is to individually vessel enhance each of the I_i's. Hence, the eigenvalues of the Hessian is needed similar to multiscale vessel enhancement. Truc et al. [7] reports that if these directional images are aligned so that the vessels are instead oriented in the x-axis direction, the Hessian is more accurate because pixels are arranged

on a grid and such grid has an x-axis and y-axis bias when performing differential operations.

To align the vessels to the x-axis, we start with the Hessian

$$\mathbf{H}_i(x,y) = \begin{bmatrix} \frac{\partial^2 I_i}{\partial x^2}(x,y) & \frac{\partial^2 I_i}{\partial x \partial y}(x,y) \\ \frac{\partial^2 I_i}{\partial x \partial y}(x,y) & \frac{\partial^2 I_i}{\partial y^2}(x,y) \end{bmatrix} \tag{3.4}$$

and use the substitution

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix}.$$

(x', y') is then the coordinate with the vessels aligned to the x-axis for directional image I_i. The (x', y') coordinate Hessian is then

$$\mathbf{H}_i : = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 I_i}{\partial x'^2} & \frac{\partial^2 I_i}{\partial x' \partial y'} \\ \frac{\partial^2 I_i}{\partial x' \partial y'} & \frac{\partial^2 I_i}{\partial y'^2} \end{bmatrix} \tag{3.5}$$

$$= \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix}^T \begin{bmatrix} \frac{\partial^2 I_i}{\partial x^2} & \frac{\partial^2 I_i}{\partial x \partial y} \\ \frac{\partial^2 I_i}{\partial x \partial y} & \frac{\partial^2 I_i}{\partial y^2} \end{bmatrix} \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix}. \tag{3.6}$$

Equation (3.6) is true since if $\begin{bmatrix} x \\ y \end{bmatrix} = A \begin{bmatrix} x' \\ y' \end{bmatrix}$, then $\begin{bmatrix} \partial x' \\ \partial y' \end{bmatrix} = A^T \begin{bmatrix} \partial x \\ \partial y \end{bmatrix}$ and $\begin{bmatrix} \partial x' & \partial y' \end{bmatrix} = \begin{bmatrix} \partial x & \partial y \end{bmatrix} A$. Hence,

$$\mathbf{H}_{x',y'} = \begin{bmatrix} \partial x' \\ \partial y' \end{bmatrix} \begin{bmatrix} \partial x' & \partial y' \end{bmatrix} = A^T \mathbf{H}_{x,y} A.$$

With the help of the double-angle formulas: $\sin 2\theta = 2\sin\theta\cos\theta$ and

$\cos 2\theta = 2\cos^2\theta - 1 = 1 - 2\sin^2\theta$, we can simplify the individual h_{ij}'s to

$$h_{11} = \frac{\partial^2 I_i}{\partial x'^2} = \frac{\partial^2 I_i}{\partial x^2}\cos^2\theta_i + \frac{\partial^2 I_i}{\partial x \partial y}\sin 2\theta_i + \frac{\partial^2 I_i}{\partial y^2}\sin^2\theta_i, \tag{3.7}$$

$$h_{22} = \frac{\partial^2 I_i}{\partial y'^2} = \frac{\partial^2 I_i}{\partial x^2}\sin^2\theta_i - \frac{\partial^2 I_i}{\partial x \partial y}\sin 2\theta_i + \frac{\partial^2 I_i}{\partial y^2}\cos^2\theta_i, \text{ and} \tag{3.8}$$

$$h_{12} = h_{21} = \frac{\partial^2 I_i}{\partial x' \partial y'} = -\frac{1}{2}\frac{\partial^2 I_i}{\partial x^2}\sin 2\theta_i + \frac{\partial^2 I_i}{\partial x \partial y}\cos 2\theta_i + \frac{1}{2}\frac{\partial^2 I_i}{\partial y^2}\sin 2\theta_i. \tag{3.9}$$

Because the new coordinate aligns the vessel in the directional image along the x'-axis, the x'-axis direction has the smallest curvature and the y'-axis has the largest curvature for pixels in a vessel. So the eigenvalues of $\mathbf{H}_i$, λ_1 and λ_2, are the diagonals of $\mathbf{H}'_i$. That is $h_{11} = \lambda_1$ and $h_{22} = \lambda_2$, if the vessel is angled in the θ_i direction. Similar to multiscale vessel enhancement, we define $R = \frac{h_{11}}{h_{22}}$ and $S = \sqrt{h_{11}^2 + h_{22}^2}$. From here, we can obtain the vesselness measure $v^{(i)}(\mathbf{x}, \sigma)$ of equation (2.9) or (2.10) for each directional image I_i. Finally, the vessel enhancement response is

$$v(\mathbf{x}) = \max_{\sigma > 0} \sum_{1 \le i \le 16} v^{(i)}(\mathbf{x}, \sigma). \tag{3.10}$$

We must mention that Truc et al. [7] uses the maximum instead of the summation:

$$v(\mathbf{x}) = \max_{\sigma > 0} \max_{1 \le i \le 16} v^{(i)}(\mathbf{x}, \sigma), \tag{3.11}$$

hence, the parameter values chosen in this thesis (table 5) are different from [7]. We choose summation instead because the performance is less impacted by deviations in parameter choice. This is important in the case of placental images in that finding an optimal parameter will be difficult if a slight change in a parameter value will produce completely different results. With the exception of standard deviation later shown in section 3.7, the use of summation rather than maximum does not significantly impact the performance for retinal images.

Truc et al. [7] also proposes the use of homomorphic filtering in between the directional filter bank step and the vessel enhancement step. That is, define I_i' to be the homomorphic filtered result of I_i, the Hessian from (3.4) is instead,

$$\mathbf{H}_i(x,y) = \begin{bmatrix} \frac{\partial^2 I_i'}{\partial x^2}(x,y) & \frac{\partial^2 I_i'}{\partial x \partial y}(x,y) \\ \frac{\partial^2 I_i'}{\partial x \partial y}(x,y) & \frac{\partial^2 I_i'}{\partial y^2}(x,y) \end{bmatrix}, \tag{3.12}$$

and $v(\mathbf{x}, \sigma)$ is defined from the eigenvalues of such Hessian.

Truc et al. [7], claims that DDFB-based vessel enhancement less effected by noise than multiscale vessel enhancement, resulting in better performance and when the optional homomorphic filtering step is included, the performance improves even further. Based on their tests, they report an increase in accuracy for detection of vessels compared to the multiscale vessel enhancement (MVE) on the DRIVE database. We attempt to reproduce such result in section 3.7 along with a test on the STARE database.

Results on Retinal Images

To compare the three methods, multiscale vessel enhancement (MVE), decimation-free directional filter bank (DDFB) vessel enhancement, and DDFB with homomorphic filtering, the parameters used presented in table 5.

TABLE 5. The Parameters Used for Comparing Results

Method	Parameters
MVE:	$\gamma = 0$; $\sigma = 1, 3, 5$; $\beta = 0.75$; $c = 15$.
DDFB:	$\gamma = 1$; $\sigma = 2, 3, 4, 5, 6$; $\beta = 0.75$; $c = 15$.
Homomorphic Filter:	Butterworth filter with $\alpha_L = 0.10$, $\alpha_H = 1.0$, $D_0 = 300$, $n = 2$.

The performance can be visually assessed on three DRIVE images and two STARE images. The chosen retinal images DRIVE image 23, DRIVE image 31, DRIVE image 34, STARE image 2, and STARE image 240 respectively have a cloudy background, a large occlusion, discoloration, bright spots, and bad lighting. These flaws can be used to to assess the limitations of the three vessel enhancement methods. Figures 22-26a show these retinal images along with the green channel in Figures 22-26b, and the hand traces in Figures 22-26c to test. Visually, the vessels on an image can be located from the output of the different methods as seen in Figures 22d-f, 23d-f, 24d-f, 25d-f, and 26d-f. We mention that DDFB with homomorphic filtering (DDFBH) may look as if it performs worst then just DDFB but this is only because of the color scales. A simple fix can be performed by using histogram equalization. Because histogram equalization is not performed in this thesis, we refer to [17] for the details.

From section 2.3, one way to assess the performance of a method comes from comparing the maximum accuracy measured by expression (2.13). But curiously, the maximum accuracy are exactly the same, seen in Figure 27. Thus, the use of accuracy as a way to compare between methods is not appropriate. A further study is warranted for why they are equal.

Even with the accuracy results being inconclusive, we can still visually employ it for comparison. Recall that to calculate such maximum accuracy, a threshold, as in expression (2.12), is sought so that the accuracy is maximized. Another way to compare the results is by looking at the threshold function:

$$\zeta(\mathbf{x}) = \begin{cases} 1 & \text{if } v(\mathbf{x}) > t_{\max} \\ 0 & \text{otherwise} \end{cases}$$

where $t_{\max}$ is the threshold for the maximum accuracy. Figures 22g-i, 23g-i, 24g-i,

25g-i, and 26g-i shows the surface plot threshold function ζ where 0 is in blue and 1 is in red. We can see that for MVE, the best performance for MVE is when almost none of the locations in the image is identified as a nonvessel. That is, its performance is close to a method that incorrectly identifies concludes the image has no vessels. DDFB does identify vessels and DDFB with homomorphic filtering having the most vessels identified compared with the other two methods. Hence, DDFB and DDFB with homomorphic filtering produces a more informative threshold function.

A popular alternative is to use the receiver operating characteristic (ROC) curve. The ROC curve is generated by plotting the true positive rate (fraction of true positives out of the positives) vs. false positive rate (the fraction of false positives out of the negatives). Details can be found in [27]. Figures 22j, 23j, 24j, 25j, and 26j show examples of the ROC curves from MVE vs. DDFB for various retinal images.

A method with high overall accuracy is one that has an ROC curve that is the closest to the constant function $\equiv 1$. That is, the method that has the largest area under the curve (AUC) defined as $\int_0^1 f(x)dx$ where $f(x)$ is the ROC curve.

Using DRIVE image 23 of Figure 22 as an example, the AUCs are **0.8729** for MVE, **0.9114** for DDFB, and **0.9276** for DDFB with homomorphic filtering. This shows that DDFB vessel enhancement (both with and without the homomorphic filtering) performs very well over MVE. DRIVE image 31 of Figure 23 as an example, the AUCs are **0.8979** for MVE, **0.9323** for DDFB, and **0.9427** for DDFB with homomorphic filter. Thus, even with a large occlusion, DDFB's performance is still high. However, DRIVE image 34 of Figure 24 has AUC **0.8575** for MVE, **0.9003** for DDFB, and **0.8906** for DDFB with homomorphic filter. Hence, vessels in a discolored image is difficult to extract, yet DDFB still

outperforms MVE. However, homomorphic filtering will not generally improve the accuracy performance and that a method to improve lighting may not decrease discoloration.

In the case of STARE image 2, the AUCs are **0.8941** for MVE, **0.9060** for DDFB, and **0.9357** for DDFB with homomorphic filter. We see that for bright spots, the use of only DDFB will not go far but homomorphic filtering creates a large contribution in accuracy. For STARE image 240, the AUCs are **0.9427** for MVE, **0.9413** for DDFB, and **0.9554** for DDFB with homomorphic filter. This shows that lighting has almost no effect on the accuracy of the three methods.

For an overview the results, Figure 28 shows the AUC for each of the retinal images in the DRIVE and STARE databases. We see that for retinal images, DDFB and DDFB with homomophic filtering significantly has a better performance and MVE. Also only STARE retinal images 240 (index 16) and 324 (index 20) compared to hand trace AH has MVE performing better than DDFB. And DDFB with homomorphic filtering consistently outperforms MVE on retinal images. Finally, when comparing Figure 28 with [7]'s table of results, they almost overlap (with the unknown exception of DRIVE image 34). Hence, even though we chose to replace maximum (equation (3.11)) with summation (equation (3.11)) and chose different parameters, the results are essentially the same (with the unknown exception of DRIVE image 34).

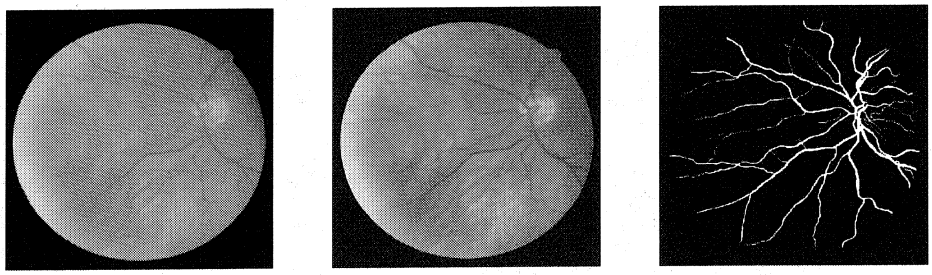
From the five sampled image and the Figure 28 we can sum up the performance of DDFB compared with MVE. In general, DDFB outperforms MVE and the homomorphic filtering adds to DDFB's performance. Specifically, DDFB is better when there is a cloudy background, large occlusion and general discoloration. However, when there are bright spots on the image, DDFB alone

will not improve much and when the only problem is lighting, MVE performs well enough that DDFB does not significantly outperform MVE.

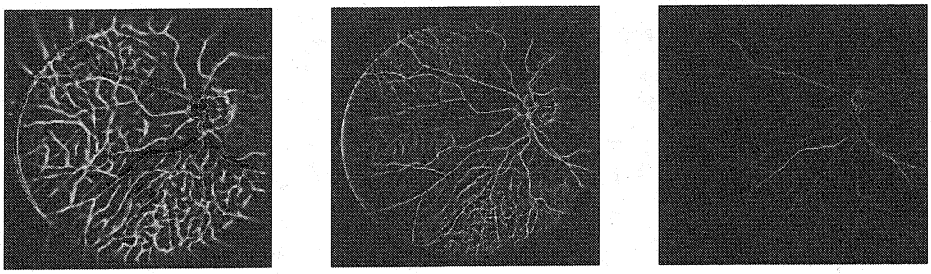
Homomorphic filtering, generally contributes to the increased performance. It increased performance for cloudy background, for large occlusion, for bad lighting, and significantly for bright spots. But negatively effects the discolored retinal images. From the Figure 28, we can see that homomorphic filtering rarely significantly increases DDFB's performance and can even negatively decrease performance. Thus, we can conclude that in the use of DDFB vessel enhancement, the vessel images should first be checked for bad lighting, discolorations, etc. to determine whether or not to consider the use of homomorphic filtering.

Table 6 shows the mean and standard deviation of each the methods for the DRIVE and STARE databases. Decimation-free directional filter bank (DDFB) has a higher AUC than multiscale vessel enhancement. With the use of the homomorphic filter, the AUC can be higher than DDFB without it. In the test of the DRIVE database, there is a 4.6% increase in accuracy. Truc et al. [7] was able to optimize the method enough to obtain a 5.2% to 5.3% improvement and a small standard deviation of 0.0060 compared to our 0.0150. This may be due to the use of maximum (equation (3.11)) instead of summation (equation (3.10)). DDFB compares well with contemporary methods such as Staal et al. [3], M. Niemeijer et al. [28], and X. Jiang et al. [29] which has mean AUCs 0.9520, 0.9294, and 0.9114, respectively.

For the STARE database, the use of DDFB and homomorphic filtering is also an improvement. We can also see that the hand traces will also suffer from accuracy. This is why the results for the hand trace of Dr. Adam Hoover (AH) is higher than Dr. Valentina Kouznetsova (VK). Hence, the quality of the hand trace will effect the test itself. The STARE database is not used as much as the DRIVE

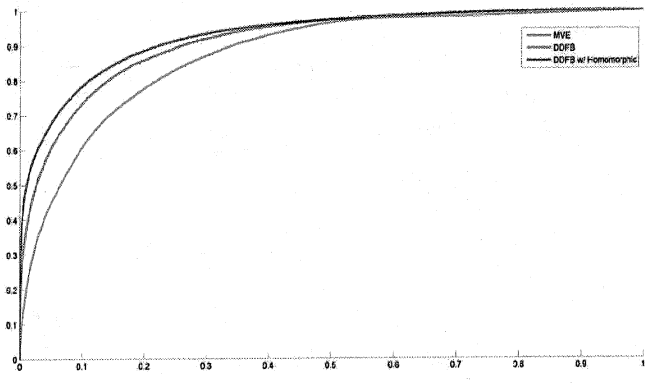


(a) DRIVE image 23 (b) Green channel (c) Hand trace



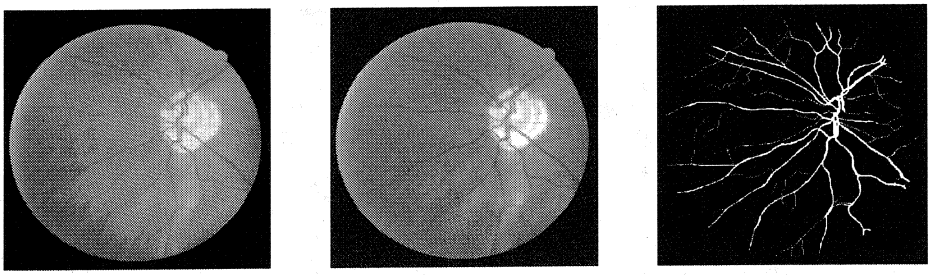
(d) MVE (e) DDFB (f) DDFBH

(g) Threshold MVE (h) Threshold DDFB (i) Threshold DDFBH

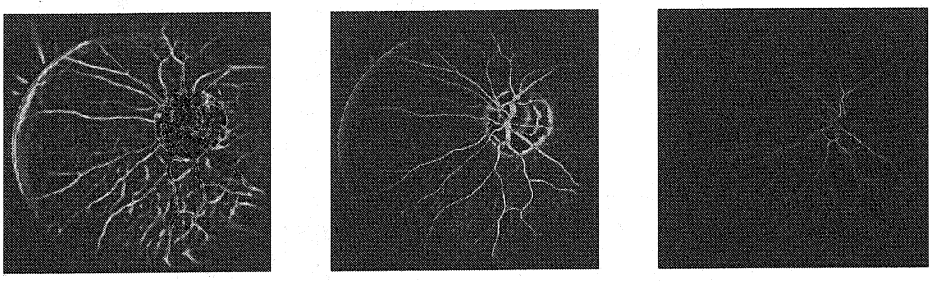


(j) The ROC curve

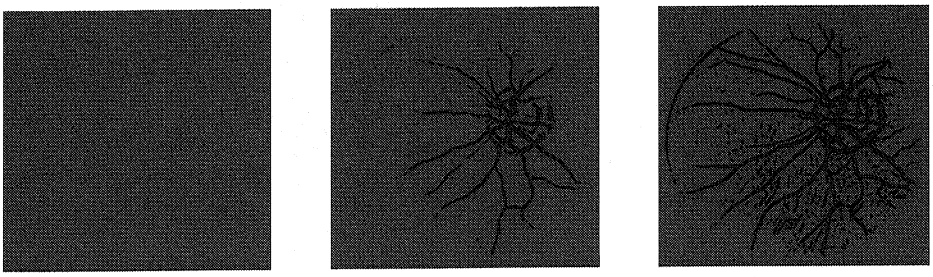
FIGURE 22. DRIVE image 23 tested using MVE and DDFB.



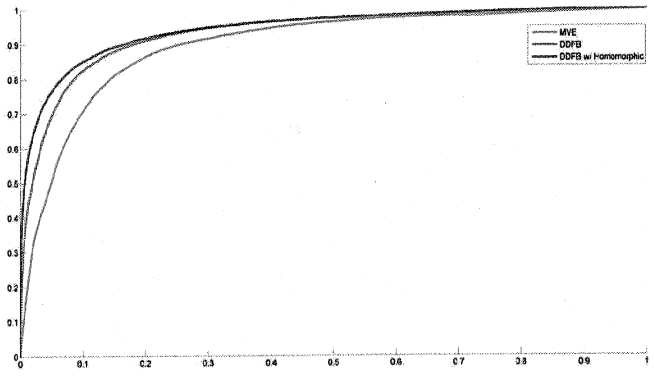
(a) DRIVE image 31 (b) Green channel (c) Hand trace



(d) MVE (e) DDFB (f) DDFBH

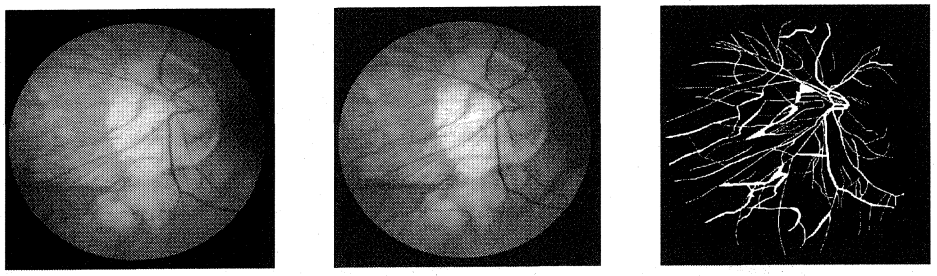


(g) Threshold MVE (h) Threshold DDFB (i) Threshold DDFBH

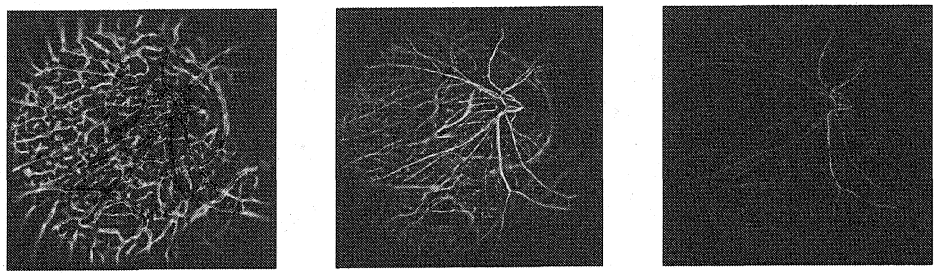


(j) The ROC curve

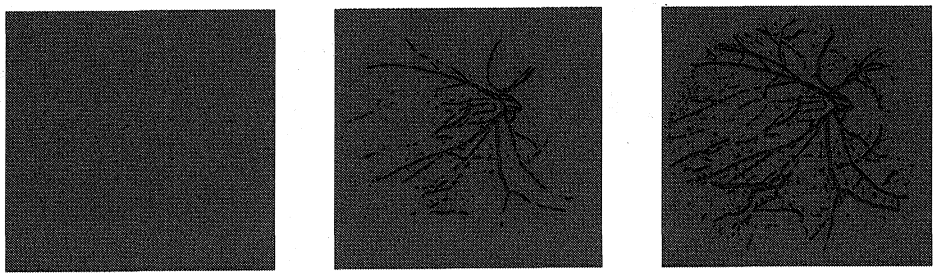
FIGURE 23. DRIVE image 31 tested using MVE and DDFB.



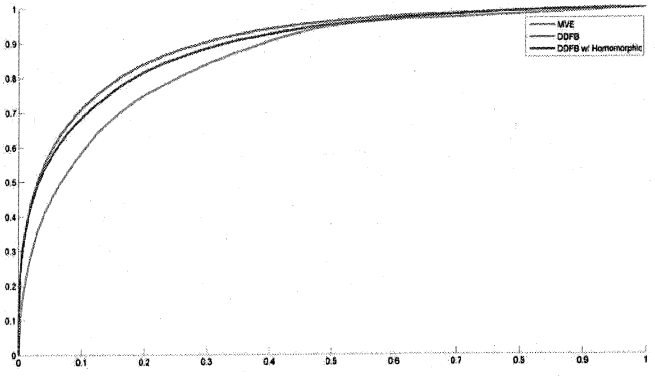
(a) DRIVE image 34 (b) Green channel (c) Hand trace



(d) MVE (e) DDFB (f) DDFBH

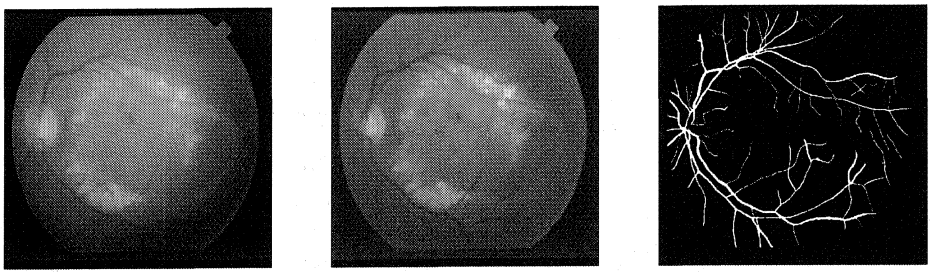


(g) Threshold MVE (h) Threshold DDFB (i) Threshold DDFBH

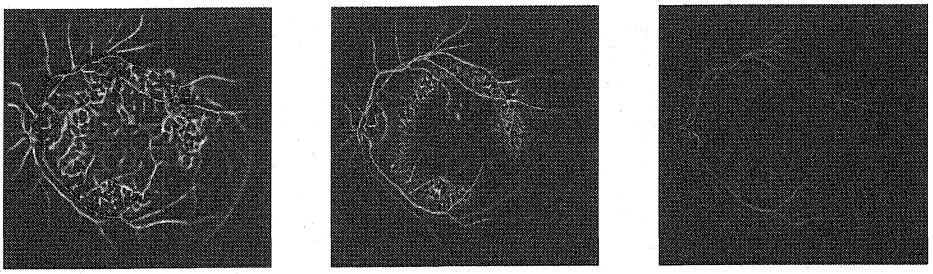


(j) The ROC curve

FIGURE 24. DRIVE image 34 tested using MVE and DDFB.

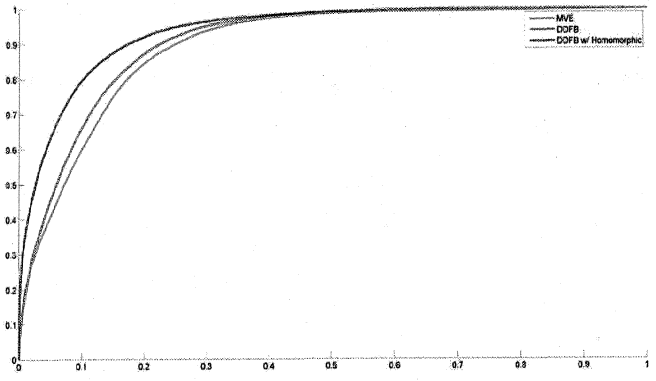


(a) STARE image 2 (b) Green channel (c) Hand trace AH



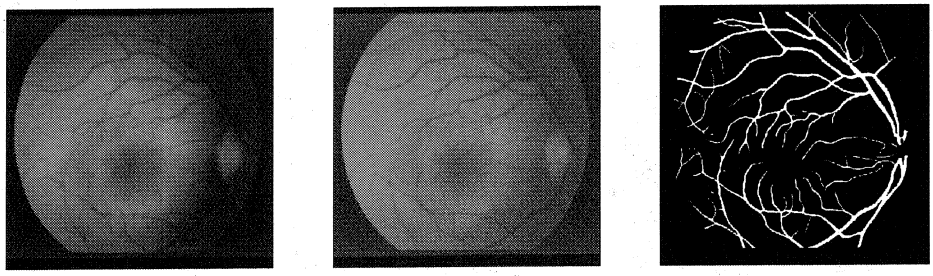
(d) MVE (e) DDFB (f) DDFBH

(g) Threshold MVE (h) Threshold DDFB (i) Threshold DDFBH

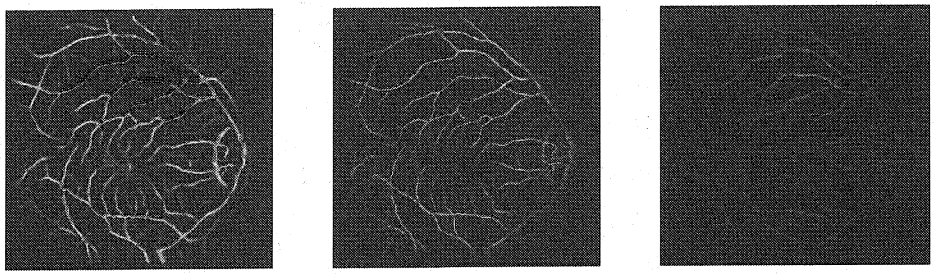


(j) The ROC curve

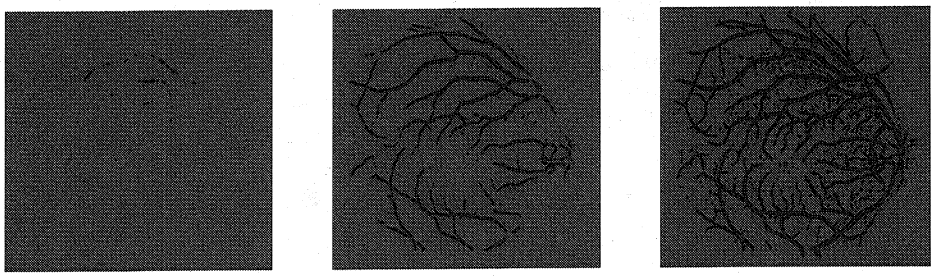
FIGURE 25. STARE image 2 tested using MVE and DDFB.



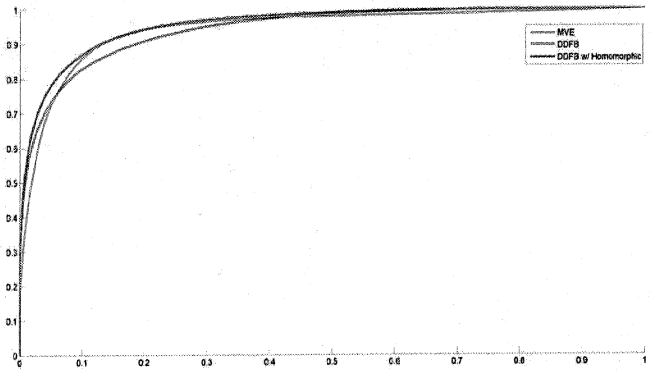
(a) STARE image 240 (b) Green channel (c) Hand trace AH



(d) MVE (e) DDFB (f) DDFBH

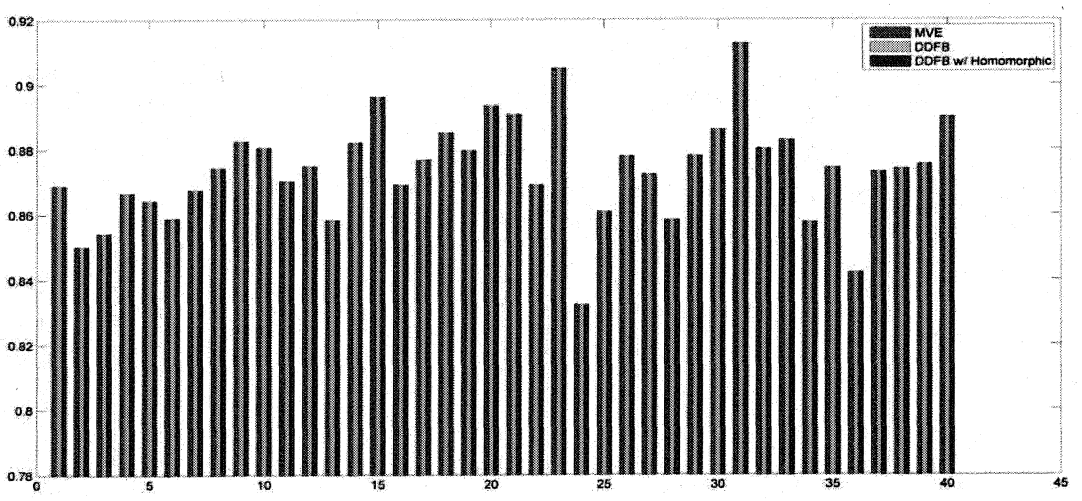


(g) Threshold MVE (h) Threshold DDFB (i) Threshold DDFBH

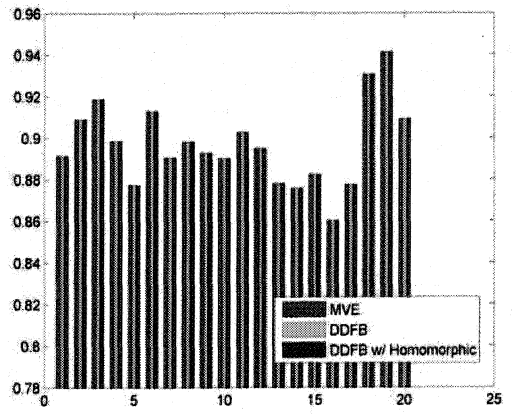


(j) The ROC curve

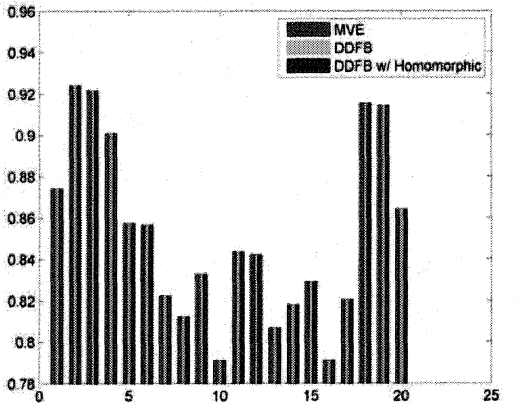
FIGURE 26. STARE image 240 tested using MVE and DDFB.



(a) Maximum accuracy for the DRIVE database

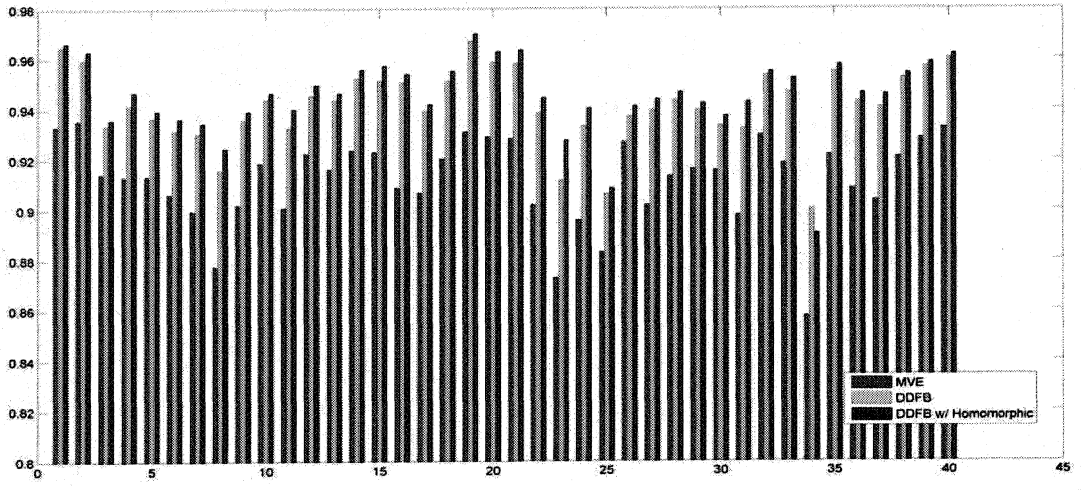


(b) Maximum accuracy for database STARE compared with hand trace AH

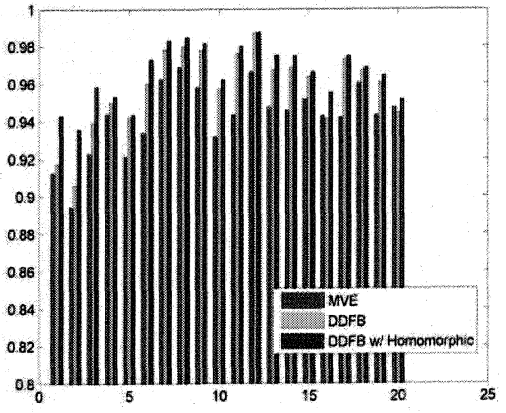


(c) Maximum accuracy for database STARE compared with hand trace VK

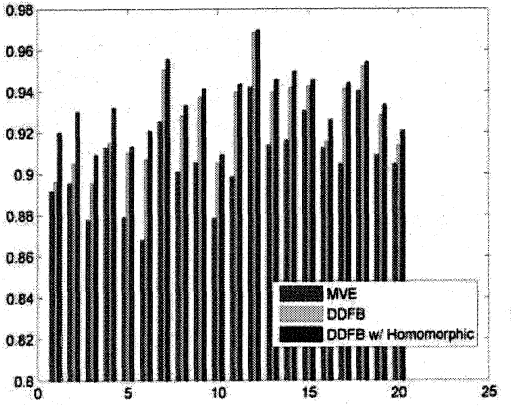
FIGURE 27. The maximum accuracy for each of DRIVE and STARE images.



(a) AUC for the DRIVE database



(b) AUC for database STARE compared with hand trace AH



(c) AUC for database STARE compared with hand trace VK

FIGURE 28. The AUC for each of the images in the DRIVE and STARE databases.

DRIVE	**mean**	**std deviation**
MVE:	$\mu = 0.9117$	$\sigma = 0.0172$
DDFB:	$\mu = 0.9416$	$\sigma = 0.0150$
DDFB with homomorphic:	$\mu = 0.9455$	$\sigma = 0.0151$

STARE with AH trace	**mean**	**std deviation**
MVE:	$\mu = 0.9419$	$\sigma = 0.0187$
DDFB:	$\mu = 0.9579$	$\sigma = 0.0212$
DDFB with homomorphic:	$\mu = 0.9658$	$\sigma = 0.0151$

STARE with VK trace	**mean**	**std deviation**
MVE:	$\mu = 0.9053$	$\sigma = 0.0203$
DDFB:	$\mu = 0.9266$	$\sigma = 0.0205$
DDFB with homomorphic:	$\mu = 0.9348$	$\sigma = 0.0165$

TABLE 6. Mean and Standard Deviation of the Methods on Retinal Databases

database due to it being focused on retinal abnormalities so it is less appealing to be tested on. It is only natural that we compare DDFB with Hoover et al. [1]. However, the evaluation of [1] does not use true positive/false positive accuracy nor area under the curve (AUC) so it cannot be directly measured. The interpretation of accuracy is then interpreted by their example images compared with DDFB's. From it, we conclude that DDFB likely outperforms [1] and potentially, so does MVE.

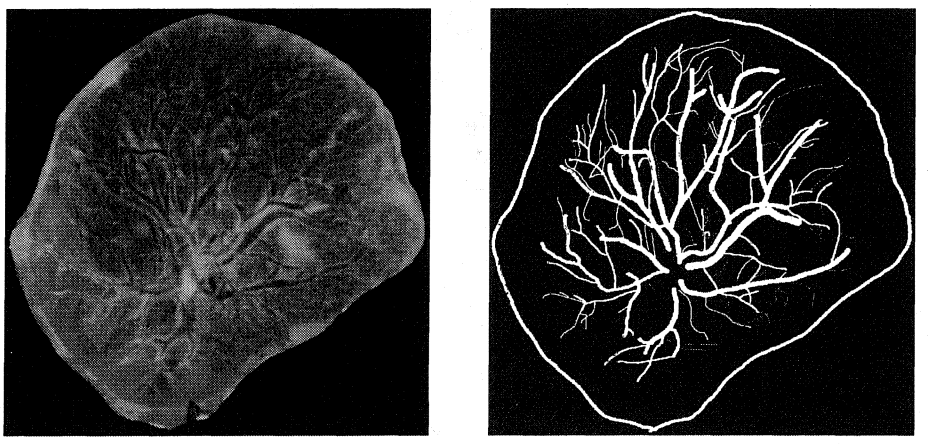
Results on Placental Images

The comparison of the three methods can similarly performed on placental images. However, because there are many more nonvessels than vessels, the AUC of the ROC curve is biased towards rating the performance negatively [30]. The Matthews correlation coefficient (MCC) is used instead:

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP} + \mathrm{FP})(\mathrm{TP} + \mathrm{FN})(\mathrm{TN} + \mathrm{FP})(\mathrm{TN} + \mathrm{FN})}} \tag{3.13}$$

where TP = true positive, TN = true negative, FP = false positive, and FN = false negative. Because TP, TN, TP and FN are dependent on thresholding, MCC is a function of the threshold value. Hence, such function produces a curve called the MCC curve. The area under the curve (AUC) for the MCC curve can be similarly defined. To be consistent with [12], the MCC will be the primary performance metric for placental images.

During the testing of the three methods on placental images, we find that the use of DDFB and DDFB with homomorphic filtering does not significantly improve on MVE. To show this, a simple experiment is proposed. First, placental image 2141, seen in Figure 29, is chosen for the experiment in order to be consistent with [12] and [15]. From it, an optimal (approximate) set of parameters similar to table 5 is sought. There are too many combination of parameter values

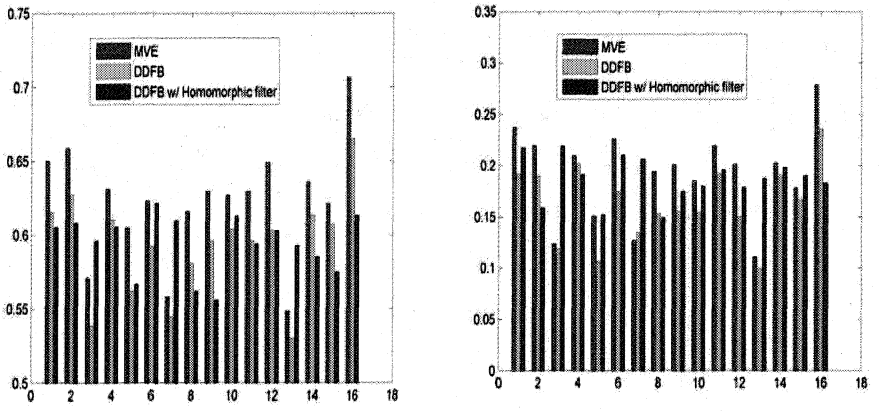


(a) Placenta 2041 (b) Hand Trace

FIGURE 29. Placenta 2141 along with its hand-trace.

to sample through for an optimal set of parameters. Instead, the parameters are picked using simulated annealing [31, 32]. Simulated annealing allows the ability to pick set of parameters with a high AUC for the MCC curve for placenta 2141. Hence, the criterion for simulated annealing is based on maximizing the AUC of the MCC. Then such AUC for each of the methods is compared to each other on each of the 16 placental images. Because an optimal parameter is chosen with minimal human intervention, the comparison is minimally biased towards one of the three methods. So if all three methods perform similarly well with the parameters from simulated annealing, we can say with some level of confidence that DDFB and DDFB with homomorphic filtering have a comparable performance to MVE. Hence, their higher performance over MVE cannot be generalized to placental images.

From simulated annealing, an optimal set of parameters for the three methods is shown in table 7.



(a) The AUC of the ROC curve (b) The AUC of the MCC curve

FIGURE 30. The AUC for the 16 placenta images.

TABLE 7. The Parameters Used for Comparing Results on Placental Images

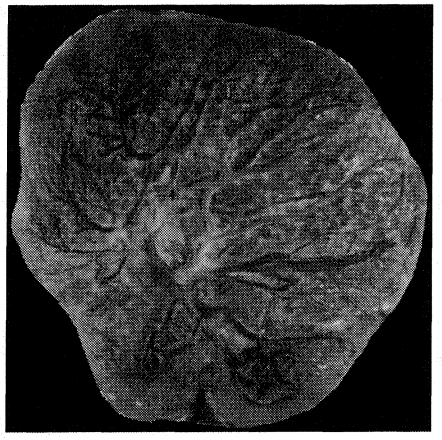
Method	**Parameters**
MVE:	$\gamma = 0$; $\sigma = 3.9, 6.2$; $\beta = 2.6$; $c = 37.5$
DDFB:	$\gamma = 1$; $\sigma = 3.7$; $\beta = 0.4$; $c = 7.9$
DDFB with Homomorphic Filter:	$\gamma = 1$; $\sigma = 1.3$; $\beta = 41.4$; $c = 35$; Butterworth filter with $\alpha_L = 0.5$; $\alpha_H = 1.9$; $D_0 = 228.8$; $n = 2$

Figure 30 shows both the ROC and MCC curves' area under the curve (AUC). We can see that out of the three methods, none of them consistently outperforms its competitors. To see when one method may outperform another, we look at six placental images with specific characteristics: placenta 2041 is more reddish than the others, placenta 2666 have transitioning color background, placenta 2743 has thin vessels, placenta 2777 is darker than the other placentas,

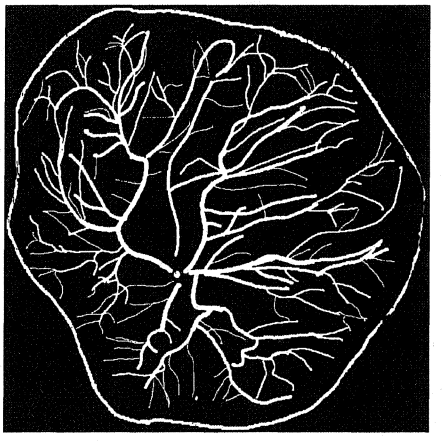
placenta 2946 has thin vessels with low contrast against its background, and placenta 3355 has high contrast between vessels and its background. Figures 31, 32, 33, 34, 35, and 36 show the results for the three methods along with its ROC and MCC curves. Notice that one method is considered quantitatively better than another if it has a higher AUC than the other method. That is, the MCC curve (or ROC curve) of the preferred method is above the other methods' MCC curve (or ROC curve).

From Figure 30, we can conclude some information about what methods may be appropriate to which placental images. As expected, the results on placental images with thin vessels (image 2743, 2946) against placental images (image 2041, 2777, 3355) shows that DDFB work better on small vessels. MVE, on the other hand, are works better on large vessels. For high contrast images such as image 3355, MVE already performs well but DDFB has a better performance on low contrast images such as image 2946. For placental images with transitioning colors such as image 2666, homomorphic filtering gives DDFB a better performance than MVE.

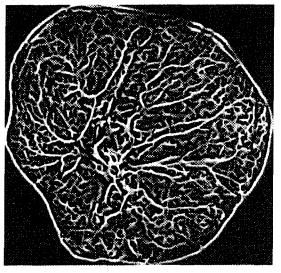
Figures 31, 32, 33, 34, 35, and 36 show that although MVE will in general outperform DDFB quantitatively using the AUC as a measure of accuracy, the enhancement from DDFB tend to be better than MVE. This can be seen from parts c-e of the above figures. However, this may be due to DDFB assigning $\gamma = 1$ and MVE assigning $\gamma = 0$, values which can effect the enhancement result. In terms of homomorphic filtering, it produces a low quality enhancement result. The vessels are less visible compared to the other two methods. Also, there are bright circles on the resulting image likely due the sharp transitions from light to dark that are present on the placental image. For example, on the boundary between the placenta and the dark background, the DDFB with homomorphic filtering



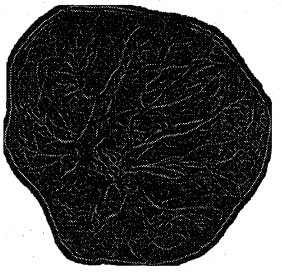
(a) Placenta 2041



(b) Hand Trace

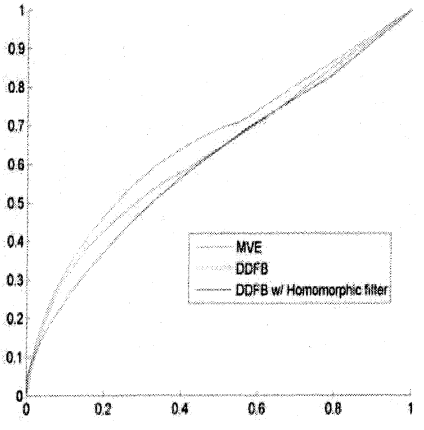


(c) MVE

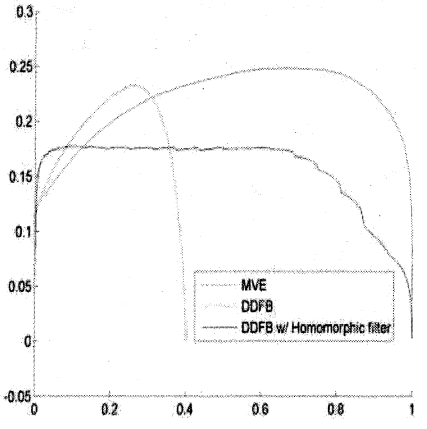


(d) DDFB

(e) DDFBH

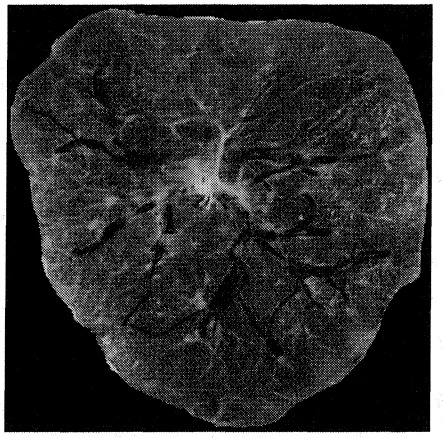


(f) ROC curve

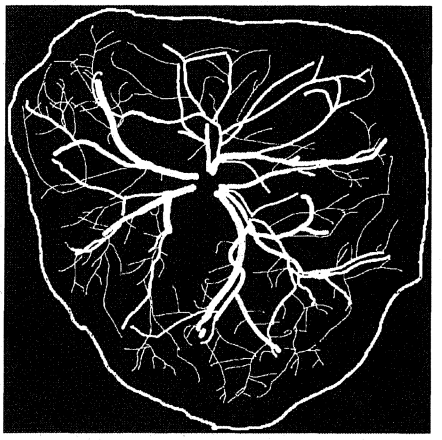


(g) MCC curve

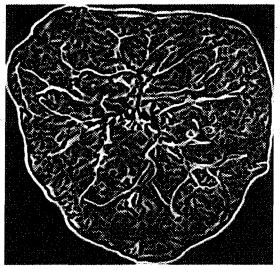
FIGURE 31. Various enhancement results for placenta 2041.



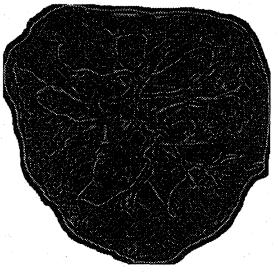
(a) Placenta 2666



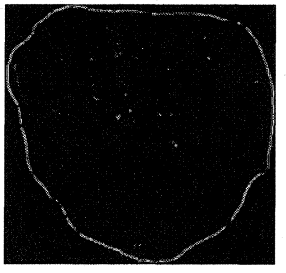
(b) Hand Trace



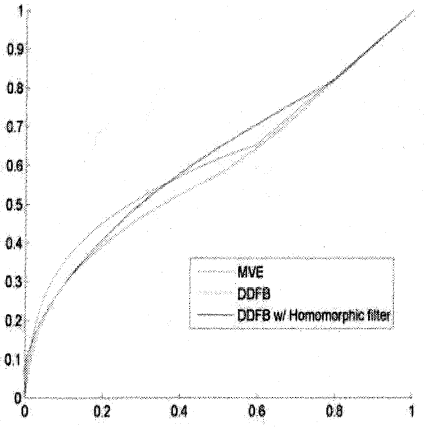
(c) MVE



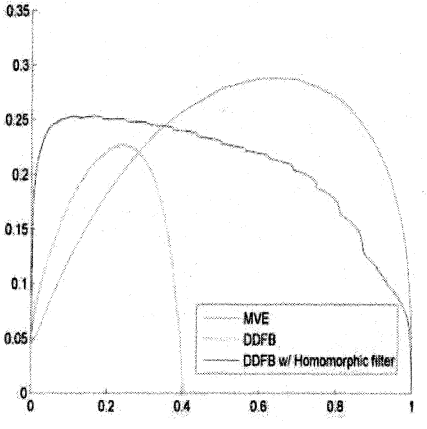
(d) DDFB



(e) DDFBH



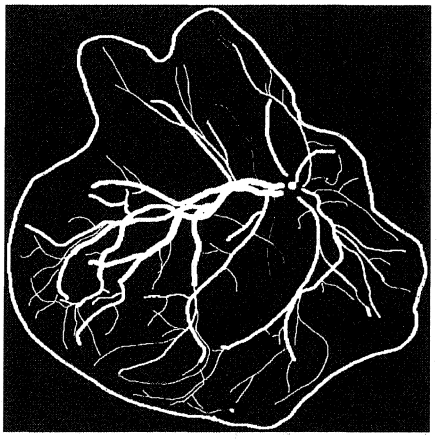
(f) ROC curve



(g) MCC curve

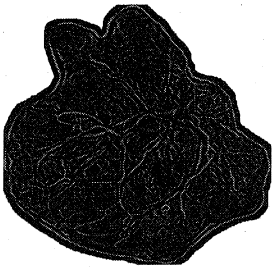
FIGURE 32. Various enhancement results for placenta 2666.

(a) Placenta 2743

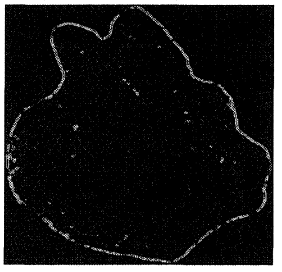


(b) Hand Trace

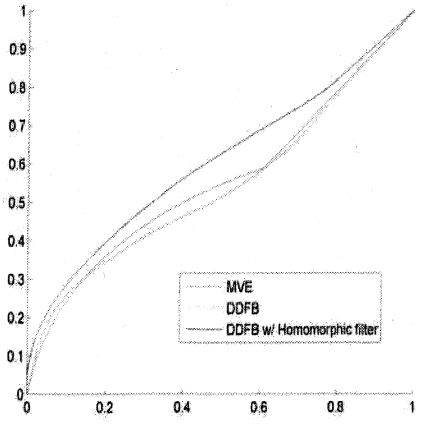
(c) MVE



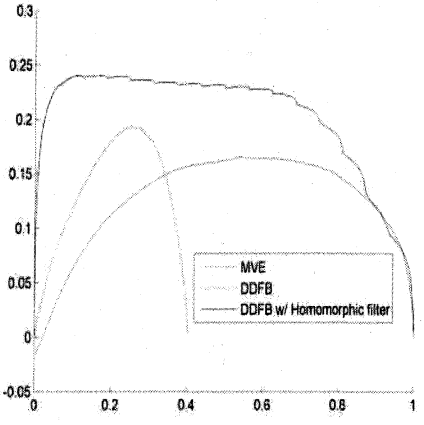
(d) DDFB



(e) DDFBH



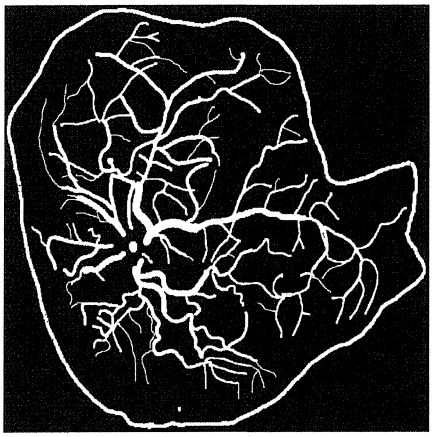
(f) ROC curve



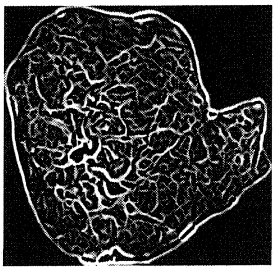
(g) MCC curve

FIGURE 33. Various enhancement results for placenta 2743.

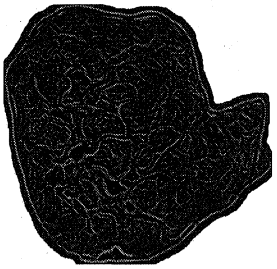
(a) Placenta 2777



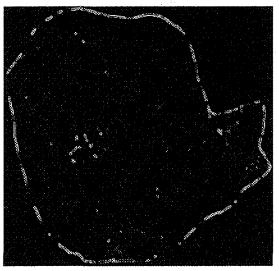
(b) Hand Trace



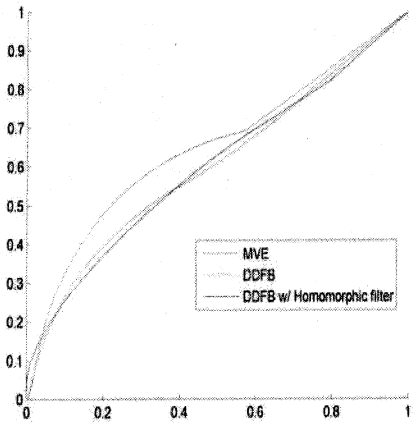
(c) MVE



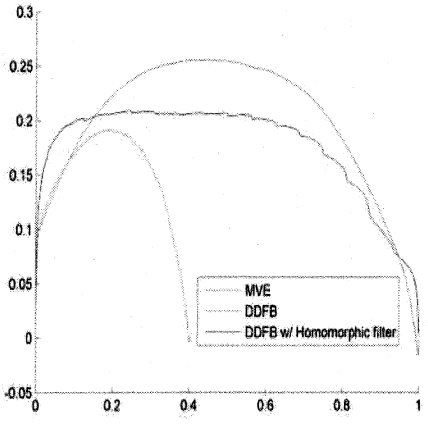
(d) DDFB



(e) DDFBH

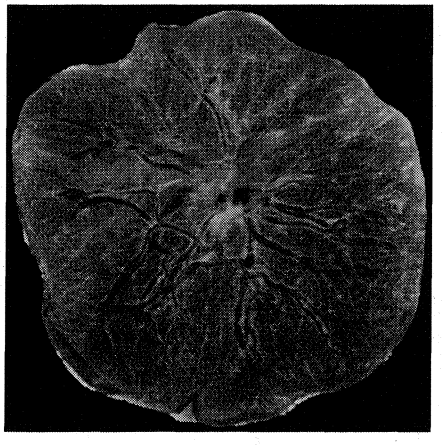


(f) ROC curve

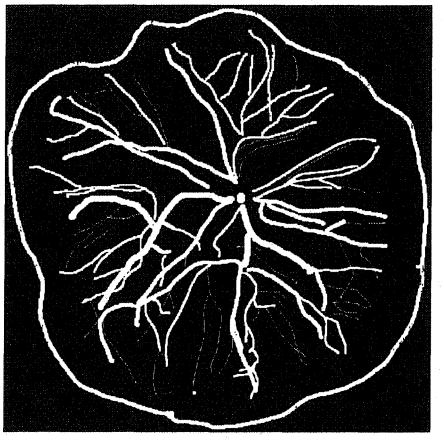


(g) MCC curve

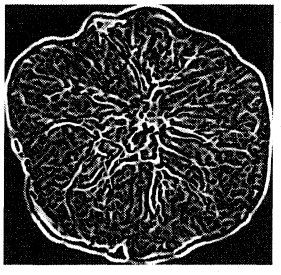
FIGURE 34. Various enhancement results for placenta 2777.



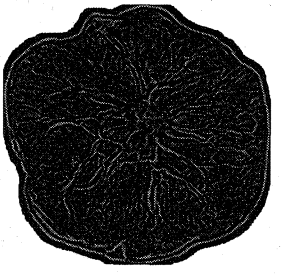
(a) Placenta 2946



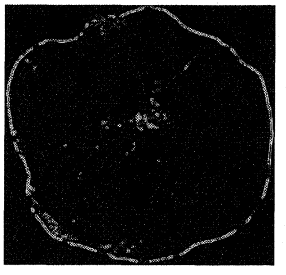
(b) Hand Trace



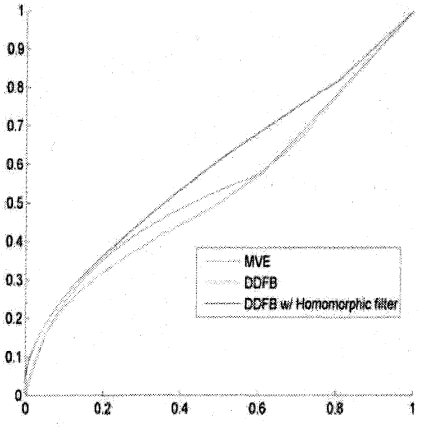
(c) MVE



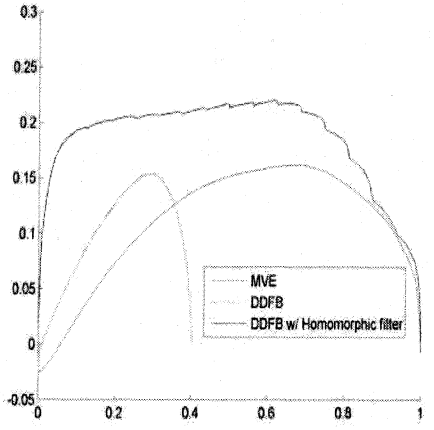
(d) DDFB



(e) DDFBH

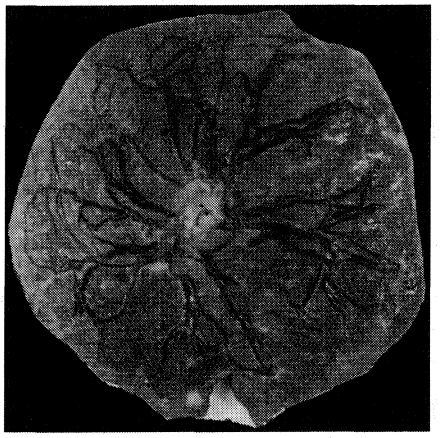


(f) ROC curve

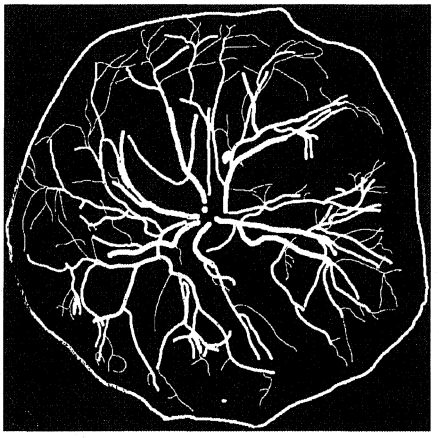


(g) MCC curve

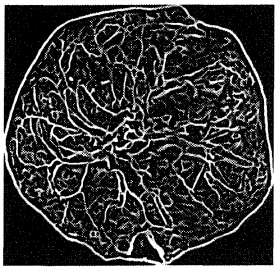
FIGURE 35. Various enhancement results for placenta 2946.



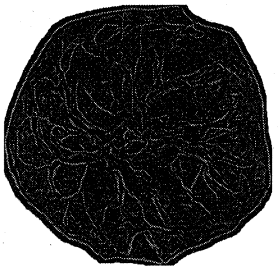
(a) Placenta 3355



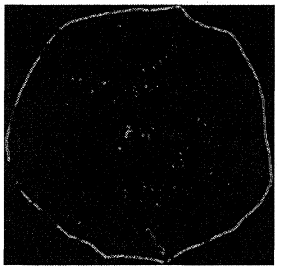
(b) Hand Trace



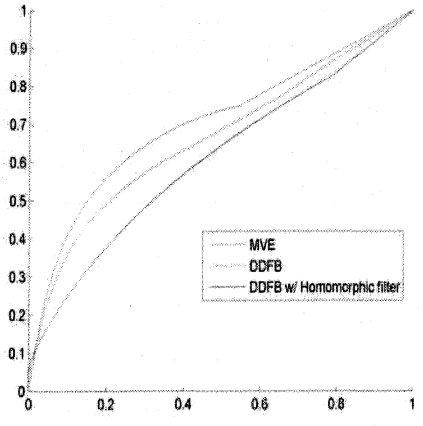
(c) MVE



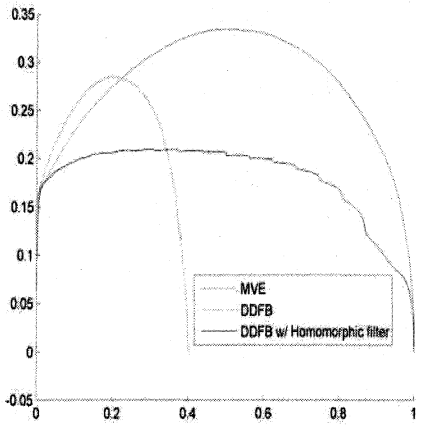
(d) DDFB



(e) DDFBH



(f) ROC curve



(g) MCC curve

FIGURE 36. Various enhancement results for placenta 3355.

produces artificial vessels. One can solve this problem using difussion on the boundary as mentioned in preprocessing. Here, we choose not to perform it for placental images so that we may fairly compare it with the only existing work on

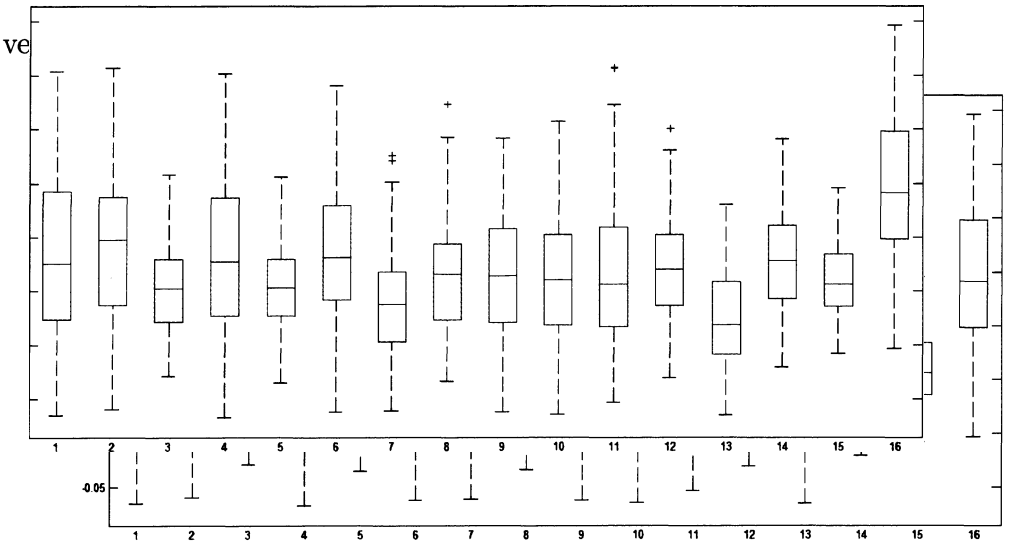


FIGURE 37. Box plot result of neural network vessel extraction.

We note that because [12] requires finding the approximate minimum of a hard-to-minimize performance function using a random number generator, the outcome of [12] will very each time it is performed. To evaluate its performance, the method from [12] is run 400 times. Each of the run provides an enhancement for each of the 16 placental images. From each of those enhancements, an AUC of the MCC curve may evaluate its performance. Hence, for each of the 16 placental images, there is an associated set of 400 AUC values from the 400 runs. Then the mean, median, and standard deviation of these 400 AUCs for each of the placentas may be computed, shown as a box plot in Figure 37. From them, a comparison with MVE, DDFB, and DDFB with homomorphic filtering may be performed. Compared with Figure 30, [12] is shown not to perform well compared with MVE, DDFB, and DDFB with homomorphic filtering. We also look at Chang et al. [15]

for comparison. Chang et al. [15] shows that a match-filtering of MVE has the potential to outperform MVE, DDFB, DDFB with homomorphic filtering, and [12]. As of writing, [15] is still a work in progress so the comparison is limited.

CHAPTER 4

CONCLUSION

We have presented the theory behind multiscale vessel enhancement (MVE) and decimation-free directional filter bank vessel enhancement (DDFB), along with homomorphic filtering, used for vessel extraction. To quantitatively evaluate their performance, we introduced the accuracy measure, the ROC curve along with its AUC, and the MCC curve along with its AUC. Moreover, we have tested them on retinal images and placental images. Along the way, we introduce the image processing (and some signal processing) concepts necessary for understanding this thesis.

In the process of testing, we find that DDFB is well suited for retinal images. But when compared with placental images, the likely reason why retinal images are great for DDFB would its the high contrast between vessels and nonvessels for retinal images, the relatively consistent lighting, and consistent vessel size for retinal images. Also, homomorphic filtering contributes well to DDFB for retinal images for its ability to remove lighting issues. However, when there is a large amount of discoloration in the image, specifically placental images, homomorphic filtering negatively contributes to the performance of DDFB. Thus, the type of vessel image will determine whether homomorphic filtering should be used.

The success of retinal images can be seen by the high AUC value of DDFB with homomorphic filtering along with its contemporaries such as [3]. Also, the

thresholding results of Figures 22i, 23i, 24i, 25i and 26i show that vessel extraction is sufficiently useful for further study such as statistical study.

As for placental images, section 3.8 shows that they are currently very difficult to extract vessel from. At the present, the only journal-published vessel extraction for placental images [12] has a low performance rate compared to an older method [6] and that a relatively new method [7] does not provide a higher performance. This can be attributed to placental images having discolorations, changing colors on vessels, over-saturation of redness, and low contrast. Thus, much more work must still be done for placental images.

APPENDIX: MATLAB CODES

The MATLAB code requires the signal processing toolbox (for fir1.m) and the image processing toolbox (for bwmorph.m, bwdist.m, imfilter.m). Furthermore, code from Dirk-Jan Kroon's "Hessian based Frangi Vesselness filter" and from David Young's "Fast 2-D convolution" are needed and can be found in MATLAB central, file exchange. The scripts need the STARE, DRIVE, and Placental Analytics database of vessel images.

Preprocessing

```
function [ OutImage ] = BasicHeatInpaint( Image,interior )
% This function performs dissipation of intensity values from
% the boundary to the exterior. The image must be grayscale.
% This function requires the image processing toolbox
% (specifically, bwdist.m).
% INPUT:
%   Image    - grayscale image.
%   interior - a binary marking the location of the interior to
%              dissipate from.

% find the distance from the interior.
dst = bwdist(interior,'cityblock');

% Sort the distance to index from.
[sorted,ind] = sort(dst(:));
ind = ind(sorted > 0);

[I,J] = ind2sub(size(Image),ind);

OutImage = Image;

% Dissipate
for i = 1 : numel(ind)
    x = max(1,I(i)-1):min(size(OutImage,1),I(i)+1);
    y = max(1,J(i)-1):min(size(OutImage,2),J(i)+1);
    mark = double(interior(x,y));

    OutImage(I(i),J(i)) = ...
        sum(sum(OutImage(x,y).*mark))/sum(mark(:));
    interior(I(i),J(i)) = 1;
end

end
```

```
function [ homoI ] = homofilter( I,option )
% This function calculates the accuracies from the
% soft-thresholds S and T is the truth.
% INPUT:
```

```
5  %       I      - The grayscale image.
6  %       option - options for the homomorphic filter.
7  % OUTPUT:
8  %       homoI - The result of homomorphic filter
9  % EXAMPLE:
10 %       im = double(imread('tun.jpg'));
11 %       option = struct('method','Butterworth','D0',1,'n',2,...
12 %                       'alphaL',.0999,'alphaH',1.01);
13
14 % The distance from the center
15 [X,Y] = ndgrid(1-size(I,1)/2:size(I,1)/2,1-size(I,2)/2:size(I,2)/2);
16 abs_omega = sqrt(X.^2+Y.^2);
17
18 % Setup the options if needed
19 if ~exist('option','var') || isempty(option)
20     option = struct('method','Butterworth','n',2,...
21                     'D0',80,'alphaL',0.25,'alphaH',2);
22 end
23
24 % Construct the high-pass filter
25 if strcmp(option.method,'Ideal')
26     H = double(abs_omega > option.D0);
27 elseif strcmp(option.method,'Butterworth')
28     H = 1./(1+(option.D0 ./ abs_omega).^(2*option.n));
29 elseif strcmp(option.method,'Gaussian')
30     H = 1 - exp(-abs_omega.^2 / (2*option.D0^2));
31 else
32     error(['Unknown method: ' option.method]);
33 end
34
35 % The homomorphic filter function
36 H = (option.alphaH-option.alphaL).*H+option.alphaL;
37
38 % Apply the homomorphic filter process
39 freqI = fftshift(fft2(log2(1+I)));
40 homoI = exp(abs(ifft2(ifftshift(H.*freqI))));
41
42 end
```

Decimation-Free Directional Filter Bank

```
1  function [ diamond ] = diamond2D( N )
2  % This function creates the diamond filter. Studied from
3  % Truc et. al.'s code.
4  %   INPUT:
5  %       N       - The order of the diamond filter.
6  %   OUTPUT:
7  %       diamond - the diamond filter output.
8
```

```
% 1-D step (freq domain) function
s = firl(N-1, .5);

% 2D step function
s2D = s'*s;

% Checkerboard (freq domain) function
[X,Y] = ndgrid(1:N,1:N);
checker = (1 + (-1).^(X+Y)).*s2D;

% Downsample in spatial is upsample in frequency
diamond = downsample2D([1 -1; 1 1],checker);

end
```

```
function [ H0,H1 ] = hourglass2D( N,version )
% This function creates the two hourglass filters. Studied from
% Truc et. al.'s code.
%   INPUT:
%       N       - The order of the diamond filter.
%       version - The version of hourglass. OPTIONAL.
%                   v1,v1T,v190,v1-90;v2,v2T,v290,v2-90.
%                 This determines the way the hourglass is
%                 constructed from the diamond filter.

if ~exist('version','var')
    version = 'v1T';
end

%% Construct diamond
D = diamond2D(N);

%% Shift the frequency domain to produce the hourglass from
%% the diamond filter.

if strcmp(version(1:2),'v1')
    H0 = repmat((-1).^(0:N-1),N,1).*D;
    if strcmp(version(3:end),'T')
        H1 = H0';
    elseif strcmp(version(3:end),'90')
        if rem(N,2) == 0
            H1 = rot90([ [H0 zeros(N,1)]; zeros(1,N+1)],1);
            H1 = H1(1:N,1:N);
        else
            H1 = rot90(H0,1);
        end
    elseif strcmp(version(3:end),'-90')
        if rem(N,2) == 0
            H1 = rot90([ [H0 zeros(N,1)]; zeros(1,N+1)],-1);
```

```
            H1 = H1(1:N,1:N);
        else
            H1 = rot90(H0,1);
        end
    else
        error(['Unrecognized version 1 option: ' version(3:end) ...
            '.']);
    end
elseif strcmp(version(1:2),'v2')
    H1 = repmat(((-1).^(0:N-1))',1,N).*D;
    if strcmp(version(3:end),'T')
        H0 = H1';
    elseif strcmp(version(3:end),'90')
        if rem(N,2) == 0
            H0 = rot90([ [H1 zeros(N,1)]; zeros(1,N+1)],1);
            H0 = H0(1:N,1:N);
        else
            H0 = rot90(H1,1);
        end
    elseif strcmp(version(3:end),'-90')
        if rem(N,2) == 0
            H0 = rot90([ [H1 zeros(N,1)]; zeros(1,N+1)],-1);
            H0 = H0(1:N,1:N);
        else
            H0 = rot90(H1,1);
        end
    else
        error(['Unrecognized version 2 option: ' version(3:end) ...
            '.']);
    end
else
    error('Unrecognized version.');
end

end
```

```
function [ Y ] = downsample2D(M,X)
% This function downsamples by matrix M on grayscale image X.
%   INPUT:
%       M - The downsample matrix.
%       X - The grayscale image.
%   OUTPUT:
%       Y - The downsampled image.

% Get the center of the image.
center = floor(size(X)/2)+1;

% Grid the image X with (0,0) being the center.
[i,j] = ndgrid(1-center(1):size(X,1)-center(1),...
```

```
                1-center(2):size(X,2)-center(2));
i = reshape(i,1,[]); j = reshape(j,1,[]);

% Rotate the grid.
sample = M*[i; j];
x = sample(1,:);
y = sample(2,:);

% Find the half indices with zeros.
boolean = rem(x,1) == 0 & rem(y,1) == 0 ...
          & x ≥ 1 - center(1) & x ≤ size(X,1)-center(1) ...
          & y ≥ 1 - center(2) & y ≤ size(X,2)-center(2);

% Only keep the integer indices.
I = sub2ind(size(X),i(boolean) + center(1),j(boolean) + center(2));
J = sub2ind(size(X),x(boolean) + center(1),y(boolean) + center(2));

% Substitute using reindexing.
Y = zeros(size(X));
Y(I) = X(J);
end
```

```
function [ Y ] = upsample2D(M,X,pad)
% This function downsamples by matrix M on grayscale image X.
%   INPUT:
%       M   - The downsample matrix.
%       X   - The grayscale image.
%       pad - Whether or not to pad the unknown. OPTIONAL.
%   OUTPUT:
%       Y - The downsampled image.

% Get the center of the image.
center = floor(size(X)/2)+1;

% Grid the image X with (0,0) being the center.
[i,j] = ndgrid(1-center(1):size(X,1)-center(1),...
               1-center(2):size(X,2)-center(2));
i = reshape(i,1,[]); j = reshape(j,1,[]);

% Rotate the grid.
sample = M*[i; j];
x = sample(1,:);
y = sample(2,:);

if isempty(pad) || ¬pad
    % Find the half indices with zeros.
    boolean = rem(x,1) == 0 & rem(y,1) == 0 ...
          & x ≥ 1 - center(1) & x ≤ size(X,1)-center(1) ...
          & y ≥ 1 - center(2) & y ≤ size(X,2)-center(2);
```

```
28
29      % Only keep the integer indices.
30      I = sub2ind(size(X),i(boolean) + center(1),...
31                          j(boolean) + center(2));
32      J = sub2ind(size(X),x(boolean) + center(1),...
33                          y(boolean) + center(2));
34
35      % Substitute using reindexing.
36      Y = zeros(size(X));
37      Y(J) = X(I);
38  else
39      % Find the half indices with zeros.
40      boolean = rem(x,1) == 0 & rem(y,1) == 0;
41
42      minx = min(x(:)); maxx = max(x(:));
43      miny = min(y(:)); maxy = max(y(:));
44      m = min(minx,miny); M = max(maxx,maxy);
45
46      Y = zeros(M - m + 1);
47
48      % Only keep the integer indices.
49      I = sub2ind(size(X),i(boolean) + center(1),...
50                          j(boolean) + center(2));
51      J = sub2ind(size(Y),x(boolean) - m + 1,...
52                          y(boolean) - m + 1);
53
54      % Substitute using reindexing.
55      Y(J) = X(I);
56  end
57
58  end
```

```
1  function [ Filter ] = CreateDirectionalFilters( N,sz,version )
2  % This function creates the Decimation-free directional filter ...
       bank filters
3  % used in paper: Truc et. al., Vessel enhancement filter using ...
       directional filter bank.
4  % INPUT:
5  %       N        - The diamond filter order.
6  %       sz [m n] - The size of the wedge filters. OPTIONAL.
7  %       version  - The version of hourglass. OPTIONAL.
8  %                  v1,v1T,v190,v1-90;v2,v2T,v290,v2-90.
9  % OUTPUT:
10 %       Filters - an m-by-n-by-16 dimensional matrix with the third
11 %                 dimension indexed for each of the wedge-shaped ...
       %                 filters.
12
13 if N < 0
14    error('N must be a positive integer');
```

```
15 end
16
17 if ¬exist('sz','var') || isempty(sz)
18     sz = 8*(N-1); % nonzero Filter window is 8(N-1), with +8 ...
           extra padded zeros.
19 end
20
21 if ¬exist('version','var')
22     version = 'v1T';
23 end
24
25 % Quincunx
26 Q = [1 -1; 1 1];
27
28 % Sheer matrices
29 R1 = [1 0; -1 1]; R1T = R1';
30 R2 = [1 1;  0 1]; R2T = R2';
31
32 %% Create Level 1 (Hourglass) Filters
33 [ h0,h1 ] = hourglass2D( N,version );
34
35 range = floor([sz sz]/2) + 1 - floor([N N]/2) ...
36         : floor([sz sz]/2) + ceil([N N]/2);
37
38 H0 = zeros(sz);
39 H0(range,range) =  h0;
40
41 H1 = zeros(sz);
42 H1(range,range) =  h1;
43
44 % [ H0,H1 ] = hourglass2D( N );
45
46 %% Create Level 2 Filters
47 H0Q = upsample2D(Q,H0,false);
48 H1Q = upsample2D(Q,H1,false);
49
50 %% Create the 4 Wedges
51 Filter2 = cat(3,...
52               conv2(H0,H0Q,'same'),...
53               conv2(H0,H1Q,'same'),...
54               conv2(H1,H1Q,'same'),...
55               conv2(H1,H0Q,'same'));
56
57 %% Create Level 3 Filters
58 H0R1QQ = upsample2D(R2*Q*Q,H0,false);
59 H1R1QQ = upsample2D(R2*Q*Q,H1,false);
60
61 H0R1TQQ = upsample2D(R2T*Q*Q,H0,false);
62 H1R1TQQ = upsample2D(R2T*Q*Q,H1,false);
63
64 H0R2QQ = upsample2D(R1*Q*Q,H0,false);
```

```
H1R2QQ = upsample2D(R1*Q*Q,H1,false);

H0R2TQQ = upsample2D(R1T*Q*Q,H0,false);
H1R2TQQ = upsample2D(R1T*Q*Q,H1,false);

%% Create the 8 Wedges
Filter3 = cat(3,...
                conv2(Filter2(:,:,1),H0R1QQ,'same'),...
                conv2(Filter2(:,:,1),H1R1QQ,'same'),...
                conv2(Filter2(:,:,2),H1R2TQQ,'same'),...
                conv2(Filter2(:,:,2),H0R2TQQ,'same'),...
                ...
                conv2(Filter2(:,:,3),H1R2QQ,'same'),...
                conv2(Filter2(:,:,3),H0R2QQ,'same'),...
                conv2(Filter2(:,:,4),H0R1TQQ,'same'),...
                conv2(Filter2(:,:,4),H1R1TQQ,'same'));

%% Create Level 4 Filters
H0R1QR1QQ = upsample2D(R2*Q*R2*Q*Q,H0,false);
H1R1QR1QQ = upsample2D(R2*Q*R2*Q*Q,H1,false);

H0R1QR2TQQ = upsample2D(R2*Q*R1T*Q*Q,H0,false);
H1R1QR2TQQ = upsample2D(R2*Q*R1T*Q*Q,H1,false);

% --
H0R2QR2QQ = upsample2D(R1*Q*R1*Q*Q,H0,false);
H1R2QR2QQ = upsample2D(R1*Q*R1*Q*Q,H1,false);

H0R2QR1TQQ = upsample2D(R1*Q*R2T*Q*Q,H0,false);
H1R2QR1TQQ = upsample2D(R1*Q*R2T*Q*Q,H1,false);

% --
H0R1TQR1QQ = upsample2D(R2T*Q*R2*Q*Q,H0,false);
H1R1TQR1QQ = upsample2D(R2T*Q*R2*Q*Q,H1,false);

H0R1TQR2TQQ = upsample2D(R2T*Q*R1T*Q*Q,H0,false);
H1R1TQR2TQQ = upsample2D(R2T*Q*R1T*Q*Q,H1,false);

% --
H0R2TQR2QQ = upsample2D(R1T*Q*R1*Q*Q,H0,false);
H1R2TQR2QQ = upsample2D(R1T*Q*R1*Q*Q,H1,false);

H0R2TQR1TQQ = upsample2D(R1T*Q*R2T*Q*Q,H0,false);
H1R2TQR1TQQ = upsample2D(R1T*Q*R2T*Q*Q,H1,false);

%% Create the 16 Wedges
Filter4 = cat(3,...
                conv2(Filter3(:,:,2),H0R1QR2TQQ,'same'),...
                conv2(Filter3(:,:,2),H1R1QR2TQQ,'same'),...
                conv2(Filter3(:,:,1),H1R1QR1QQ,'same'),...
                conv2(Filter3(:,:,1),H0R1QR1QQ,'same'),...
```

```
                ...
                conv2(Filter3(:,:,8),H0R1TQR2TQQ,'same'),...
                conv2(Filter3(:,:,8),H1R1TQR2TQQ,'same'),...
                conv2(Filter3(:,:,7),H1R1TQR1QQ,'same'),...
                conv2(Filter3(:,:,7),H0R1TQR1QQ,'same'),...
                ...
                conv2(Filter3(:,:,6),H1R2QR1TQQ,'same'),...
                conv2(Filter3(:,:,6),H0R2QR1TQQ,'same'),...
                conv2(Filter3(:,:,5),H0R2QR2QQ,'same'),...
                conv2(Filter3(:,:,5),H1R2QR2QQ,'same'),...
                ...
                conv2(Filter3(:,:,4),H1R2TQR1TQQ,'same'),...
                conv2(Filter3(:,:,4),H0R2TQR1TQQ,'same'),...
                conv2(Filter3(:,:,3),H0R2TQR2QQ,'same'),...
                conv2(Filter3(:,:,3),H1R2TQR2QQ,'same'));...

Filter = Filter4;

end
```

```
function [Dxx,Dxy,Dyy] = Hessian2Dmod(I,sigma,gamma,tol)
%  This function Hessian2 Filters the image with 2nd derivatives
%  of a  Gaussian with parameter Sigma. Modified from D.Kroon
%  University of Twente (June 2009).
%
% [Dxx,Dxy,Dyy] = Hessian2Dmod(I,Sigma);
%
% INPUT:
%   I - The image, class preferable double or single
%   sigma - The sigma of the gaussian kernel used
%
% OUTPUT:
%   Dxx, Dxy, Dyy - The 2nd derivatives.

if nargin < 2
    sigma = 1;
    gamma = 1;
    tol = 10^-3;
elseif nargin == 2
    gamma = 1;
    tol = 10^-3;
elseif nargin == 3
    tol = 10^-3;
end

% Make kernel coordinates
[X,Y] = ndgrid(-round(4*sigma):round(4*sigma));

% Build the gaussian 2nd derivatives filters
```

```
30 DGaussxx = - sigma^gamma * (sigma^2 - ...
       X.^2)/(2*pi*sigma^6).*exp(-(X.^2+Y.^2)/(2*sigma^2));
31 DGaussyy = - sigma^gamma * (sigma^2 - ...
       Y.^2)/(2*pi*sigma^6).*exp(-(X.^2+Y.^2)/(2*sigma^2));
32 DGaussxy = sigma^gamma * ...
       ((X.*Y/(2*pi*sigma^6)).*exp(-(X.^2+Y.^2)/(2*sigma^2)));
33
34 % Find the components of the Hessian using SVD,
35 % which is faster.
36 Dxx = convolve2(I,DGaussxx,'same',tol);
37 Dxy = convolve2(I,DGaussxy,'same',tol);
38 Dyy = convolve2(I,DGaussyy,'same',tol);
39
40 end
```

```
1  function [ directEnhance ] = DirectionalEnhance( ...
       directImg,option )
2  % This function enhances each of the directional vessel images.
3  % INPUT:
4  %   directImg - The cell of directional vessels images.
5  %   option    - Options for the DDFB vessel enhancement.
6  %               .N     - the order of the diamond.
7  %               .sigma - the vector containing the scales.
8  %               .beta  - the beta value for vessel enhancement.
9  %               .c     - the c value for vessel enhancement.
10 %               .LightonDark - whether the vessels are
11 %                              lighter than its background.
12 % OUTPUT:
13 %   directEnhance - a #-#-16 matrix of vessel enhancement on
14 %                   each of the directional images.
15
16 % Number of directional filters
17 n = size(directImg,3);
18
19 % angle of the filters.
20 theta = pi/(2*n) : pi/n : (2*n-1)*pi/(2*n);
21
22 temp = cell(n,1);
23
24 if option.LightonDark
25     for i = 1 : n
26         % Directional vessel enhance on light vessels.
27         temp{i} = DirectionalEnhanceLightonDark(...
28                           directImg(:,:,i),option,theta(i));
29     end
30 else
31     for i = 1 : n
32         % Directional vessel enhance on dark vessels.
33         temp{i} = DirectionalEnhanceDarkonLight(...
```

```
34                              directImg(:,:,i),option,theta(i));
35      end
36  end
37
38  % Combine the directional vessel enhancements.
39  directEnhance = cat(3,temp{:});
40
41  end
42
43
44  function [outImg] = DirectionalEnhanceLightonDark(Img,option,angle)
45  % This function performs the light-on-dark enhancement version.
46
47  outImg = zeros(size(Img));
48  for sigma = option.sigma
49      [Dxx,Dxy,Dyy] = Hessian2Dmod(Img,sigma);
50      h11 = Dxx*cos(angle)^2+Dxy*sin(2*angle)+Dyy*sin(angle)^2;
51      h22 = Dxx*sin(angle)^2-Dxy*sin(2*angle)+Dyy*cos(angle)^2;
52      S = sqrt(h11.^2+h22.^2);
53      R = h11./h22;
54      temp = double(h22 < 0).*(exp(-R.^2/(2*option.beta^2))...
55                               .*(1-exp(-S.^2/(2*option.c^2))));
56      outImg = double(temp > outImg).*(temp - outImg) + outImg;
57  end
58
59  end
60
61
62  function [outImg] = DirectionalEnhanceDarkonLight(Img,option,angle)
63  % This function performs the dark-on-light enhancement version.
64
65  outImg = zeros(size(Img));
66  for sigma = option.sigma
67      [Dxx,Dxy,Dyy] = Hessian2Dmod(Img,sigma);
68      h11 = Dxx*cos(angle)^2+Dxy*sin(2*angle)+Dyy*sin(angle)^2;
69      h22 = Dxx*sin(angle)^2-Dxy*sin(2*angle)+Dyy*cos(angle)^2;
70      S = sqrt(h11.^2+h22.^2);
71      R = h11./h22;
72      temp = double(h22 > 0).*(exp(-(R.^2/(2*option.beta^2)))...
73                               .*(1-exp(-S.^2/(2*option.c^2))));
74      outImg = double(temp > outImg).*(temp - outImg) + outImg;
75  end
76
77  end
```

```
1  function [ enhance ] = DFBMultiscaleEnhance( ...
       I,DDFBoption,midOption )
2  % This function calculates the accuracies from the
3  % soft-thresholds S and T is the truth.
```

```
4  % INPUT:
5  %       I          - Grayscale image.
6  %       DDFBoption - Options for the DDFB vessel enhancement.
7  %                    .N     - the order of the diamond.
8  %                    .sigma - the vector containing the scales.
9  %                    .beta  - the beta value for vessel
10 %                             enhancement.
11 %                    .c     - the c value for vessel enhancement.
12 %                    .LightonDark - whether the vessels are
13 %                                   lighter than its background.
14 %       midOption  - Middle step between DDFB filtering and vessel
15 %                    enhancement options. Optional.
16 %                    .func - The function applied to each of the
17 %                            directional images.
18 %                    ...   - Other parameters used by .func
19 % OUTPUT:
20 %       enhance - The enhancement output.
21
22 %% Setup DDFB options if not provided
23 if ¬exist('DDFBoption','var') || isempty(DDFBoption)
24     DDFBoption = struct('N',3,'sigma', 2 : 1 : 6,...
25                         'beta', 0.75,'c', 15,...
26                         'LightonDark',false);
27 end
28
29 if ¬isfield(DDFBoption,'N')
30     DDFBoption.N = 3;
31 end
32
33 if ¬isfield(DDFBoption,'sigma')
34     DDFBoption.sigma = 2 : 1 : 6;
35 end
36
37 if ¬isfield(DDFBoption,'beta')
38     DDFBoption.beta = 0.75;
39 end
40
41 if ¬isfield(DDFBoption,'c')
42     DDFBoption.c = 15;
43 end
44
45 if ¬isfield(DDFBoption,'LightonDark')
46     DDFBoption.LightonDark = false;
47 end
48
49 %% Directionally filter the image
50 Filter = CreateDirectionalFilters(DDFBoption.N);
51
52 DirL = zeros(size(I,1),size(I,2),16);
53 for k = 1 : 16
54     DirL(:,:,k) = convolve2(I,Filter(:,:,k),'same',0.01);
```

```
55 end
56
57 %% mid-step filter
58 if exist('midOption','var') && ¬isempty(midOption)
59     for k = 1 : 16
60         DirL(:,:,k) = midOption.func(DirL(:,:,k),midOption);
61     end
62 end
63
64 %% Vessel Enhance
65 DirEnhance = DirectionalEnhance(DirL,DDFBoption);
66
67 %% Combine the enhancement
68 % enhance = max(DirEnhance,[],3);
69 enhance = mean(DirEnhance,3);
70
71 end
```

Metrics

```
1  function [ Accy,t ] = SoftAccuracy( S,T )
2  % This function calculates the accuracies from the ...
       soft-thresholds S and T
3  % is the truth.
4  % INPUT:
5  %       S    - The soft threshold. It must be nonnegative.
6  %       T    - The truth (true or false).
7  % OUTPUT:
8  %       Accy - The vector of accuracies.
9  %       t    - The corresponding threshold values.
10 % EXAMPLE:
11 %       N = 1000;
12 %       S = floor(300*rand(1,N));
13 %       T = rand(1,N) > 0.5;
14 %       Accy = SoftAccuracy( S,T );
15 %       Temp = arrayfun(@(t) sum((S>t).*T) + sum((S≤t).*¬T),...
16 %                       [-1 unique(S)])...
17 %                /length(S);
18 %       max(abs(Accy - Temp))
19
20 S = S(:); T = double(T(:));
21
22 N = length(S);
23
24 [SS,I] = sort(S-min(S)+1); TT = T(I);
25
26 t = [ 0; unique(SS) ];
27
28 [b,m,¬] = unique(sort(SS.*TT));
```

```
b = [0; b]; m = [0; m];
TP = FillIn(t,b,N-m);
% FN = sum(double(T)) - TP;

[b,m,¬] = unique(sort(SS.*¬TT));
m = [0; m - m(1)*(b(1) == 0) ]; b = [0; b];
TN = FillIn(t,b,m);
% FP = sum(double(¬T)) - TN;

Accy = (TP + TN)/N;

t = t + min(S) - 1;

end

function [ filled ] = FillIn( long,short,filler )
% Fills non-matching elements.

filled = long;

j = length(short);
for i = length(long) : -1 : 1
    filled(i) = filler(j);

    if long(i) == short(j)
        j = j-1;
    end
end

end
```

```
function [ MCC,t ] = MCCeff( S,T )
% This function calculates the Matthews correlation coefficient
% (MCC) from the soft-thresholds S and T is the truth.
% INPUT:
%       S    - The soft threshold. It must be nonnegative.
%       T    - The truth (true or false).
% OUTPUT:
%       MCC - the vector MCC values.
%       t    - the corresponding threshold values.

S = S(:); T = double(T(:));

N = length(S);

[SS,I] = sort(S-min(S)+1); TT = T(I);

t = [ 0; unique(SS) ];

```

```
[b,m,¬] = unique(sort(SS.*TT));
b = [0; b]; m = [0; m];
TP = FillIn(t,b,N-m);
FN = sum(double(T)) - TP;

[b,m,¬] = unique(sort(SS.*¬TT));
m = [0; m - m(1)*(b(1) == 0) ]; b = [0; b];
TN = FillIn(t,b,m);
FP = sum(double(¬T)) - TN;

MCC = ((TP.*TN) - (FP.*FN))...
        ./sqrt((TP+FN).*(TP+FP).*(TN+FP).*(TN+FN));

t = t + min(S) - 1;

end

function [ filled ] = FillIn( long,short,filler )
% Fills non-matching elements.

filled = long;

j = length(short);
for i = length(long) : -1 : 1
    filled(i) = filler(j);

    if long(i) == short(j)
        j = j-1;
    end
end

end
```

Script

```
% RetinalEnhanceScript.m

% Run this script to see bar graph in Figure 28 of thesis
% and the mean of Table 5. To see bar graph in Figure 27,
% change AUC (including variable name AUCplot) to accy
% for part F of this script. To see the standard deviation of ...
   Table 5,
% change part G to call std() instead of mean().

%% A. Setup the parameters

MVEoption = struct('FrangiScaleRange',[1 6],'FrangiScaleRatio',2,...
                        'FrangiBetaOne',0.75,'FrangiBetaTwo',15,...
                        'BlackWhite',true,'verbose',false);
```

```
14
15 DDFBoption = struct('N',3,'sigma',2 : 1 : 6,...
16                                  'beta',0.75,'gamma',15,...
17                                  'LightonDark',false);
18
19 midOption = struct('func',@(I,opt) homofilter(I,opt),...
20                         'method','Butterworth',...
21                         'n',2,'D0',300,'alphaL',0.1,'alphaH',1);
22
23 %% B. DRIVE Vessel extraction database
24 DRIVE_db = cell(40,1);
25 for i = 1 : 9
26     DRIVE_db{i} = struct('I',imread(['DRIVE\test\images\0' ...
           num2str(i) '_test.tif']),...
27                       'mask',imread(['DRIVE\test\mask\0' ...
                             num2str(i) '_test_mask.gif']),...
28                       'trace',imread(['DRIVE\test\1st_manual\0' ...
                             num2str(i) '_manual1.gif']));
29 end
30
31 for i = 10 : 20
32     DRIVE_db{i} = struct('I',imread(['DRIVE\test\images\' ...
           num2str(i) '_test.tif']),...
33                       'mask',imread(['DRIVE\test\mask\' ...
                             num2str(i) '_test_mask.gif']),...
34                       'trace',imread(['DRIVE\test\1st_manual\' ...
                             num2str(i) '_manual1.gif']));
35 end
36
37 for i = 21 : 40
38     DRIVE_db{i} = struct('I',imread(['DRIVE\training\images\' ...
           num2str(i) '_training.tif']),...
39                       'mask',imread(['DRIVE\training\mask\' ...
                             num2str(i) '_training_mask.gif']),...
40                       'trace',imread(['DRIVE\training\1st_manual\' ...
                             num2str(i) '_manual1.gif']));
41 end
42
43 %% C. STARE Vessel extraction database
44 names = {'im0001','im0002','im0003','im0004','im0005',...
45          'im0044','im0077','im0081','im0082','im0139',...
46          'im0162','im0163','im0235','im0236','im0239',...
47          'im0240','im0255','im0291','im0319','im0324'};
48
49 STARE_db = cell(length(names),1);
50
51 for i = 1 : length(names)
52     STARE_db{i} = struct('I',imread(['STARE\Images\' names{i} ...
           '.ppm']),...
53                       'traceAH',imread(['STARE\ah_trace\' ...
                             names{i} '.ah.ppm']),...
```

```
54                          'traceVK',imread(['STARE\vk_trace\' ...
                               names{i} '.vk.ppm']));
55  end
56
57  T = [50 40 60 33 50 70 50*ones(1,9) 40 50*ones(1,4)];
58
59  for i = 1 : length(names)
60      STARE_db{i}.mask = 255*uint8(STARE_db{i}.I(:,:,1) > T(i));
61  end
62
63  %% D. Enhance DRIVE database
64
65  parfor j = 1 : 40
66      disp(['Scanning image ' num2str(j)]);
67
68      % The marking used to identify binary regions
69      mark = max(DRIVE_db{j}.trace(:));
70
71      % Shrink the marked region to accommodate the incorrect mask ...
           provided
72      interior = (DRIVE_db{j}.mask == mark);
73      for i = 1 : 5
74          interior = interior-bwmorph(interior,'remove');
75      end
76
77      % Smooth the boundary to prevent false identification of ...
           boundary as vessel
78      DRIVE_db{j}.diffuse = BasicHeatInpaint( ...
           double(DRIVE_db{j}.I(:,:,2)),interior );
79
80      %%% 1. MVE
81      Temp = struct();
82
83      % Use the Multiscale enhancement
84      Temp.MVEoption = MVEoption;
85
86      [enhance,~,~] = FrangiFilter2D(DRIVE_db{j}.diffuse, ...
           Temp.MVEoption);
87      Temp.enhance = enhance;
88
89      % Get the ROC curve
90      [X,Y,T,AUC] = perfcurve(DRIVE_db{j}.trace(DRIVE_db{j}.mask ...
           == mark),...
91                              enhance(DRIVE_db{j}.mask == mark),mark);
92      Temp.X = X; Temp.Y = Y; Temp.T = T;
93      Temp.AUC = AUC;
94
95      [Accy,t] = SoftAccuracy(enhance(DRIVE_db{j}.mask == mark),...
96                          DRIVE_db{j}.trace(DRIVE_db{j}.mask == mark));
97      [accy,i] = max(Accy);
98      Temp.accy = accy; Temp.thresh = t(i);
```

```
99
100     DRIVE_db{j}.MVE = Temp;
101
102     %%% 2. DDFB
103     Temp = struct();
104
105     Temp.DDFBoption = DDFBoption;
106
107     enhance = ...
            DFBMultiscaleEnhance(DRIVE_db{j}.diffuse,Temp.DDFBoption);
108     Temp.enhance = enhance;
109
110     % Get the ROC curve
111     [X,Y,T,AUC] = perfcurve(DRIVE_db{j}.trace(DRIVE_db{j}.mask ...
            == mark),...
112                             enhance(DRIVE_db{j}.mask == mark),mark);
113     Temp.X = X; Temp.Y = Y; Temp.T = T;
114     Temp.AUC = AUC;
115
116     [Accy,t] = SoftAccuracy(enhance(DRIVE_db{j}.mask == mark),...
117                         DRIVE_db{j}.trace(DRIVE_db{j}.mask == mark));
118     [accy,i] = max(Accy);
119     Temp.accy = accy; Temp.thresh = t(i);
120
121     DRIVE_db{j} = setfield(DRIVE_db{j},...
122                            ['DDFB' num2str(Temp.DDFBoption.N)],...
123                            Temp);
124
125     %%% 3. DDFB homomorphic mid
126     Temp = struct();
127
128     Temp.DDFBoption = DDFBoption;
129
130     Temp.midOption = midOption;
131
132     enhance = DFBMultiscaleEnhance(DRIVE_db{j}.diffuse,...
133                                    Temp.DDFBoption,Temp.midOption);
134     Temp.enhance = enhance;
135
136     % Get the ROC curve
137     [X,Y,T,AUC] = perfcurve(DRIVE_db{j}.trace(DRIVE_db{j}.mask ...
            == mark),...
138                             enhance(DRIVE_db{j}.mask == ...
                                mark),mark);
139     Temp.X = X; Temp.Y = Y; Temp.T = T;
140     Temp.AUC = AUC;
141
142     [Accy,t] = SoftAccuracy(enhance(DRIVE_db{j}.mask == mark),...
143                         DRIVE_db{j}.trace(DRIVE_db{j}.mask == mark));
144     [accy,i] = max(Accy);
145     Temp.accy = accy; Temp.thresh = t(i);
```

```
146
147     DRIVE_db{j} = setfield(DRIVE_db{j},...
148                 ['DDFB' num2str(Temp.DDFBoption.N) 'MidHomo'],Temp);
149
150 end
151
152 %% E. Enhance STARE database
153
154 % AH
155 parfor j = 1 : 20
156     disp(['Scanning image ' num2str(j)]);
157
158     % The marking used to identify binary regions
159     mark = max(STARE_db{j}.traceAH(:));
160
161     % Shrink the marked region to accommodate the incorrect mask ...
          provided
162     interior = (STARE_db{j}.mask == mark);
163     for i = 1 : 5
164         interior = interior-bwmorph(interior,'remove');
165     end
166
167     % Smooth the boundary to prevent false identification of ...
          boundary as vessel
168     STARE_db{j}.diffuse = BasicHeatInpaint( ...
          double(STARE_db{j}.I(:,:,2)),interior );
169
170     %%% 1. MVE
171     Temp = struct();
172
173     % Use the Multiscale enhancement
174     Temp.MVEoption = MVEoption;
175
176     [enhance,¬,¬] = FrangiFilter2D(STARE_db{j}.diffuse, ...
          Temp.MVEoption);
177     Temp.enhance = enhance;
178
179     % Get the ROC curve
180     [X,Y,T,AUC] = perfcurve(STARE_db{j}.traceAH(STARE_db{j}.mask ...
          == mark),...
181                             enhance(STARE_db{j}.mask == mark),mark);
182     Temp.X = X; Temp.Y = Y; Temp.T = T;
183     Temp.AUC = AUC;
184
185     [Accy,t] = SoftAccuracy(enhance(STARE_db{j}.mask == mark),...
186                         STARE_db{j}.traceAH(STARE_db{j}.mask == mark));
187     [accy,i] = max(Accy);
188     Temp.accy = accy; Temp.thresh = t(i);
189
190     STARE_db{j}.MVE_AH = Temp;
191
```

```
192     %%% 2. DDFB
193     Temp = struct();
194
195     Temp.DDFBoption = DDFBoption;
196
197     enhance = ...
           DFBMultiscaleEnhance(STARE_db{j}.diffuse,Temp.DDFBoption);
198     Temp.enhance = enhance;
199
200     % Get the ROC curve
201     [X,Y,T,AUC] = perfcurve(STARE_db{j}.traceAH(STARE_db{j}.mask ...
           == mark),...
202                             enhance(STARE_db{j}.mask == mark),mark);
203     Temp.X = X; Temp.Y = Y; Temp.T = T;
204     Temp.AUC = AUC;
205
206     [Accy,t] = SoftAccuracy(enhance(STARE_db{j}.mask == mark),...
207                       STARE_db{j}.traceAH(STARE_db{j}.mask == mark));
208     [accy,i] = max(Accy);
209     Temp.accy = accy; Temp.thresh = t(i);
210
211     STARE_db{j} = setfield(STARE_db{j},...
212                     ['DDFB' num2str(Temp.DDFBoption.N) '_AH'],Temp);
213
214     %%% 3. DDFB homomorphic mid
215     Temp = struct();
216
217     Temp.DDFBoption = DDFBoption;
218
219     Temp.midOption = midOption;
220
221     enhance = DFBMultiscaleEnhance(STARE_db{j}.diffuse,...
222                                    Temp.DDFBoption,Temp.midOption);
223     Temp.enhance = enhance;
224
225     % Get the ROC curve
226     [X,Y,T,AUC] = perfcurve(STARE_db{j}.traceAH(STARE_db{j}.mask ...
           == mark),...
227                             enhance(STARE_db{j}.mask == mark),mark);
228     Temp.X = X; Temp.Y = Y; Temp.T = T;
229     Temp.AUC = AUC;
230
231     [Accy,t] = SoftAccuracy(enhance(STARE_db{j}.mask == mark),...
232                     STARE_db{j}.traceAH(STARE_db{j}.mask == mark));
233     [accy,i] = max(Accy);
234     Temp.accy = accy; Temp.thresh = t(i);
235
236     STARE_db{j} = setfield(STARE_db{j},...
237             ['DDFB' num2str(Temp.DDFBoption.N) 'MidHomo_AH'],Temp);
238 end
239
```

```
% VK
parfor j = 1 : 20
    disp(['Scanning image ' num2str(j)]);

    % The marking used to identify binary regions
    mark = max(STARE_db{j}.traceVK(:));

    % Shrink the marked region to accommodate the incorrect mask ...
       provided
    interior = (STARE_db{j}.mask == mark);
    for i = 1 : 5
        interior = interior-bwmorph(interior,'remove');
    end

    % Smooth the boundary to prevent false identification of ...
       boundary as vessel
    STARE_db{j}.diffuse = BasicHeatInpaint( ...
       double(STARE_db{j}.I(:,:,2)),interior );

    %%% 1. MVE
    Temp = struct();

    % Use the Multiscale enhancement
    Temp.MVEoption = MVEoption;

    [enhance,¬,¬] = FrangiFilter2D(STARE_db{j}.diffuse, ...
       Temp.MVEoption);
    Temp.enhance = enhance;

    % Get the ROC curve
    [X,Y,T,AUC] = perfcurve(STARE_db{j}.traceVK(STARE_db{j}.mask ...
       == mark),...
                            enhance(STARE_db{j}.mask == mark),mark);
    Temp.X = X; Temp.Y = Y; Temp.T = T;
    Temp.AUC = AUC;

    [Accy,t] = SoftAccuracy(enhance(STARE_db{j}.mask == mark),...
                    STARE_db{j}.traceVK(STARE_db{j}.mask == mark));
    [accy,i] = max(Accy);
    Temp.accy = accy; Temp.thresh = t(i);

    STARE_db{j}.MVE_VK = Temp;

    %%% 2. DDFB
    Temp = struct();

    Temp.DDFBoption = DDFBoption;

    enhance = ...
       DFBMultiscaleEnhance(STARE_db{j}.diffuse,Temp.DDFBoption);
    Temp.enhance = enhance;
```

```
285
286     % Get the ROC curve
287     [X,Y,T,AUC] = perfcurve(STARE_db{j}.traceVK(STARE_db{j}.mask ...
            == mark),...
288                             enhance(STARE_db{j}.mask == mark),mark);
289     Temp.X = X; Temp.Y = Y; Temp.T = T;
290     Temp.AUC = AUC;
291
292     [Accy,t] = SoftAccuracy(enhance(STARE_db{j}.mask == mark),...
293                     STARE_db{j}.traceVK(STARE_db{j}.mask == mark));
294     [accy,i] = max(Accy);
295     Temp.accy = accy; Temp.thresh = t(i);
296
297     STARE_db{j} = setfield(STARE_db{j},...
298                 ['DDFB' num2str(Temp.DDFBoption.N) '_VK'],Temp);
299
300     %%% 3. DDFB homomorphic mid
301     Temp = struct();
302
303     Temp.DDFBoption = DDFBoption;
304
305     Temp.midOption = midOption;
306
307     enhance = DFBMultiscaleEnhance(STARE_db{j}.diffuse,...
308                                    Temp.DDFBoption,Temp.midOption);
309     Temp.enhance = enhance;
310
311     % Get the ROC curve
312     [X,Y,T,AUC] = perfcurve(STARE_db{j}.traceVK(STARE_db{j}.mask ...
            == mark),...
313                             enhance(STARE_db{j}.mask == mark),mark);
314     Temp.X = X; Temp.Y = Y; Temp.T = T;
315     Temp.AUC = AUC;
316
317     [Accy,t] = SoftAccuracy(enhance(STARE_db{j}.mask == mark),...
318                     STARE_db{j}.traceVK(STARE_db{j}.mask == mark));
319     [accy,i] = max(Accy);
320     Temp.accy = accy; Temp.thresh = t(i);
321
322     STARE_db{j} = setfield(STARE_db{j},...
323             ['DDFB' num2str(Temp.DDFBoption.N) 'MidHomo_VK'],Temp);
324 end
325
326 %% F. Bar graph of the ROC AUC
327 % DRIVE
328 AUCplt = [arrayfun(@(i) DRIVE_db{i}.MVE.AUC,1:40); ...
329           arrayfun(@(i) DRIVE_db{i}.DDFB3.AUC,1:40);
330           arrayfun(@(i) DRIVE_db{i}.DDFB3MidHomo.AUC,1:40)]';
331
332 figure('color','white'),
333 bar(AUCplt,'grouped','BaseValue',0.80);
```

```
334 title('DRIVE AUC','fontsize',20);
335 legend('MVE','DDFB','DDFB w/ Homomorphic filter');
336
337 % STARE AH
338 AUCplt = [arrayfun(@(i) STARE_db{i}.MVE_AH.AUC,1:20); ...
339           arrayfun(@(i) STARE_db{i}.DDFB3_AH.AUC,1:20);
340           arrayfun(@(i) STARE_db{i}.DDFB3MidHomo_AH.AUC,1:20)]';
341
342 figure('color','white'),
343 bar(AUCplt,'grouped','BaseValue',0.80);
344 title('STARE AH AUC','fontsize',20);
345 legend('MVE','DDFB','DDFB w/ Homomorphic filter');
346
347
348 % STARE VK
349 AUCplt = [arrayfun(@(i) STARE_db{i}.MVE_VK.AUC,1:20); ...
350           arrayfun(@(i) STARE_db{i}.DDFB3_VK.AUC,1:20);
351           arrayfun(@(i) STARE_db{i}.DDFB3MidHomo_VK.AUC,1:20)]';
352
353 figure('color','white'),
354 bar(AUCplt,'grouped','BaseValue',0.80);
355 title('STARE VK AUC','fontsize',20);
356 legend('MVE','DDFB','DDFB w/ Homomorphic filter');
357
358 %% G. Calculate the mean and standard deviation
359 mnMVE = mean(arrayfun(@(i) DRIVE_db{i}.MVE.AUC,1:40));
360 mnDDFB = mean(arrayfun(@(i) DRIVE_db{i}.DDFB3.AUC,1:40));
361 mnDDFB3MidHomo = mean(arrayfun(@(i) ...
      DRIVE_db{i}.DDFB3MidHomo.AUC,1:40));
```

```
1  % SpecificRetinalScript.m
2
3  % Run script Retinal EnhanceScript.m before this script!
4  % "i" is the retinal image index. For STARE images, change
5  % DRIVE_db to STARE_db, corresponding traceAH/traceVK and
6  % corresponding MVE_AH/MVE_VK, etc.
7  i = 34;
8
9  %% A. Generate the result as seen in Figure 2 and Figure 24
10
11 figure('color','white','name',...
12         ['Retinal Image ' num2str(i)],...
13         'NumberTitle','off'),
14
15 subplot(3,3,1),
16 imagesc(DRIVE_db{i}.I);
17 axis off
18 subplot(3,3,2),
19 imagesc(DRIVE_db{i}.diffuse);
```

```
20 axis off
21 subplot(3,3,3),
22 imagesc(DRIVE_db{i}.trace);
23 axis off
24
25 subplot(3,3,4),
26 imagesc(DRIVE_db{i}.MVE.enhance);
27 axis off
28 subplot(3,3,5),
29 imagesc(DRIVE_db{i}.DDFB3.enhance);
30 axis off
31 subplot(3,3,6),
32 imagesc(DRIVE_db{i}.DDFB3MidHomo.enhance);
33 axis off
34
35 subplot(3,3,7),
36 imagesc(DRIVE_db{i}.MVE.enhance > DRIVE_db{i}.MVE.thresh);
37 axis off
38 subplot(3,3,8),
39 imagesc(DRIVE_db{i}.DDFB3.enhance > DRIVE_db{i}.DDFB3.thresh);
40 axis off
41 subplot(3,3,9),
42 imagesc(DRIVE_db{i}.DDFB3MidHomo.enhance > ...
      DRIVE_db{i}.DDFB3MidHomo.thresh);
43 axis off
44
45 %% B. Generate ROC curve seen in Figure 24
46 figure('color','white','name',...
47         ['Retinal Image ' num2str(i) ' ROC'],...
48         'NumberTitle','off'),
49
50 hold on,
51 plot(DRIVE_db{i}.MVE.X,DRIVE_db{i}.MVE.Y,'g');
52 plot(DRIVE_db{i}.DDFB3.X,DRIVE_db{i}.DDFB3.Y,'r');
53 plot(DRIVE_db{i}.DDFB3MidHomo.X,DRIVE_db{i}.DDFB3MidHomo.Y,'b');
54 hold off
55
56 legend('MVE','DDFB','DDFB w/ Homomorphic');
```

```
1  % PlacentalEnhanceScript.m
2
3  %% Load the placenta images, its masks and its hand traces
4  sel = [1973;2041;2095;2141;2561;2666;2743;2744;...
5         2753;2772;2774;2777;2946;3321;3340;3355];
6
7  Placenta_db = cell(size(sel));
8
9  for j = 1 : length(sel)
10     Placenta = imread(['CD_' num2str(sel(j)) '-FR-FS2.bmp']);
```

```
11      T = imread([num2str(sel(j)) 'Traced.png'],'png');
12      T = double(T > 0);
13      M = imread(['mask_T' num2str(sel(j)) '-FR-FS2.bmp']);
14      M = double(M > 0);
15
16      Placenta_db{j} = ...
            struct('I',double(Placenta),'trace',T,'mask',M);
17  end
18
19  %% Enahance Placenta
20
21  MVEoption = struct('FrangiScaleRange', [3.9 6.2],...
22              'FrangiScaleRatio', 2.3, 'FrangiBetaOne', 2.6, ...
                    'FrangiBetaTwo', 37.5,...
23              'verbose',false,'BlackWhite',true);
24
25  DDFBoption = struct('N',3,'sigma',3.7,...
26                              'beta',0.4,'c',7.9,...
27                              'LightonDark',false);
28
29  DDFBHoption = struct('N',3,'sigma',1.3,...
30                              'beta',41.4,'c',35,...
31                              'LightonDark',false);
32
33  midOption = struct('func',@(I,opt) homofilter(I,opt),...
34                      'method','Butterworth',...
35                      'n',2,'D0',228.8,'alphaL',0.5,'alphaH',1.9);
36
37  parfor j = 1 : numel(sel)
38      disp(['Scanning image ' num2str(j)]);
39
40      % The marking used to identify binary regions
41      mark = max(Placenta_db{j}.trace(:));
42
43      %%% 1. MVE
44      Temp = struct();
45
46      % Use the Multiscale enhancement
47      Temp.MVEoption = MVEoption;
48
49      [enhance,¬,¬] = FrangiFilter2D(Placenta_db{j}.I(:,:,2), ...
            Temp.MVEoption);
50      Temp.enhance = enhance;
51
52      % Get the ROC curve
53      [X,Y,T,AUC] = ...
            perfcurve(Placenta_db{j}.trace(Placenta_db{j}.mask == ...
            mark),...
54                              enhance(Placenta_db{j}.mask == mark),mark);
55      Temp.X = X; Temp.Y = Y; Temp.T = T;
56      Temp.AUC = AUC;
```

```
57
58      % Get the maximum accuracy
59      [Accy,t] = SoftAccuracy(enhance(Placenta_db{j}.mask == mark),...
60                  Placenta_db{j}.trace(Placenta_db{j}.mask == mark));
61      [accy,i] = max(Accy);
62      Temp.accy = accy; Temp.thresh = t(i);
63
64      % Get the MCC curve
65      [ MCC,t ] = MCCeff(enhance(Placenta_db{j}.mask == mark),...
66                  Placenta_db{j}.trace(Placenta_db{j}.mask == mark));
67      Temp.MCC = MCC(2:end-1); Temp.MCCthresh = t(2:end-1);
68      Temp.MCCAUC = trapz(t(2:end-1),MCC(2:end-1));
69
70      Placenta_db{j}.MVE = Temp;
71
72      %%% 2. DDFB
73      Temp = struct();
74
75      Temp.DDFBoption = DDFBoption;
76
77      enhance = ...
           DFBMultiscaleEnhance(Placenta_db{j}.I(:,:,2),Temp.DDFBoption);
78      Temp.enhance = enhance;
79
80      % Get the ROC curve
81      [X,Y,T,AUC] = ...
           perfcurve(Placenta_db{j}.trace(Placenta_db{j}.mask == ...
           mark),...
82                          enhance(Placenta_db{j}.mask == mark),mark);
83      Temp.X = X; Temp.Y = Y; Temp.T = T;
84      Temp.AUC = AUC;
85
86      % Get the maximum accuracy
87      [Accy,t] = SoftAccuracy(enhance(Placenta_db{j}.mask == mark),...
88                  Placenta_db{j}.trace(Placenta_db{j}.mask == mark));
89      [accy,i] = max(Accy);
90      Temp.accy = accy; Temp.thresh = t(i);
91
92      % Get the MCC curve
93      [ MCC,t ] = MCCeff(enhance(Placenta_db{j}.mask == mark),...
94                  Placenta_db{j}.trace(Placenta_db{j}.mask == mark));
95      Temp.MCC = MCC(2:end-1); Temp.MCCthresh = t(2:end-1);
96      Temp.MCCAUC = trapz(t(2:end-1),MCC(2:end-1));
97
98      Placenta_db{j} = setfield(Placenta_db{j},...
99                           ['DDFB' num2str(Temp.DDFBoption.N)],Temp);
100
101     %%% 3. DDFB homomorphic mid
102     Temp = struct();
103
104     Temp.DDFBoption = DDFBHoption;
```

```
105
106     Temp.midOption = midOption;
107
108     enhance = DFBMultiscaleEnhance(Placenta_db{j}.I(:,:,2),...
109                                    Temp.DDFBoption,Temp.midOption);
110     Temp.enhance = enhance;
111
112     % Get the ROC curve
113     [X,Y,T,AUC] = ...
            perfcurve(Placenta_db{j}.trace(Placenta_db{j}.mask == ...
            mark),...
114                         enhance(Placenta_db{j}.mask == mark),mark);
115     Temp.X = X; Temp.Y = Y; Temp.T = T;
116     Temp.AUC = AUC;
117
118     % Get the maximum accuracy
119     [Accy,t] = SoftAccuracy(enhance(Placenta_db{j}.mask == mark),...
120                   Placenta_db{j}.trace(Placenta_db{j}.mask == mark));
121     [accy,i] = max(Accy);
122     Temp.accy = accy; Temp.thresh = t(i);
123
124     % Get the MCC curve
125     [ MCC,t ] = MCCeff(enhance(Placenta_db{j}.mask == mark),...
126                   Placenta_db{j}.trace(Placenta_db{j}.mask == mark));
127     Temp.MCC = MCC(2:end-1); Temp.MCCthresh = t(2:end-1);
128     Temp.MCCAUC = trapz(t(2:end-1),MCC(2:end-1));
129
130     Placenta_db{j} = setfield(Placenta_db{j},...
131             ['DDFB' num2str(Temp.DDFBoption.N) 'MidHomo'],Temp);
132 end
133
134 %% B. Generate ROC curve seen in Figure 30
135
136 AUCplt = [arrayfun(@(i) Placenta_db{i}.MVE.AUC,1:16); ...
137           arrayfun(@(i) Placenta_db{i}.DDFB3.AUC,1:16);
138           arrayfun(@(i) Placenta_db{i}.DDFB3MidHomo.AUC,1:16)]';
139
140 figure('color','white'),
141 bar(AUCplt,'grouped','BaseValue',0.50);
142 title('Placenta ROC AUC','fontsize',20);
143 legend('MVE','DDFB','DDFB w/ Homomorphic filter');
144
145 MCCAUCplt = [arrayfun(@(i) Placenta_db{i}.MVE.MCCAUC,1:16); ...
146              arrayfun(@(i) Placenta_db{i}.DDFB3.MCCAUC,1:16);
147              arrayfun(@(i) ...
                    Placenta_db{i}.DDFB3MidHomo.MCCAUC,1:16)]';
148
149 figure('color','white'),
150 bar(MCCAUCplt,'grouped','BaseValue',0);
151 title('Placenta MCC AUC','fontsize',20);
152 legend('MVE','DDFB','DDFB w/ Homomorphic filter');
```

```
153
154 %% C. Generate ROC curve seen in Figure 31
155 i = 2;
156
157 figure('color','white','name',...
158         ['Retinal Image ' num2str(i)],...
159         'NumberTitle','off'),
160
161 subplot(3,3,1),
162 imagesc(uint8(Placenta_db{i}.I));
163 axis off
164 subplot(3,3,2),
165 imagesc(Placenta_db{i}.trace);
166 axis off
167
168 subplot(3,3,4),
169 imagesc(Placenta_db{i}.MVE.enhance);
170 axis off
171 subplot(3,3,5),
172 imagesc(Placenta_db{i}.DDFB3.enhance);
173 axis off
174 subplot(3,3,6),
175 imagesc(Placenta_db{i}.DDFB3MidHomo.enhance);
176 axis off
177
178 subplot(3,3,7),
179 imagesc(Placenta_db{i}.MVE.enhance > Placenta_db{i}.MVE.thresh);
180 axis off
181 subplot(3,3,8),
182 imagesc(Placenta_db{i}.DDFB3.enhance > Placenta_db{i}.DDFB3.thresh);
183 axis off
184 subplot(3,3,9),
185 imagesc(Placenta_db{i}.DDFB3MidHomo.enhance > ...
       Placenta_db{i}.DDFB3MidHomo.thresh);
186 axis off
187
188
189 figure('color','white','name',...
190         ['Placetal Image ' num2str(sel(i)) ' ROC'],...
191         'NumberTitle','off'),
192 hold on,
193 plot(Placenta_db{i}.MVE.X,Placenta_db{i}.MVE.Y,'g');
194 plot(Placenta_db{i}.DDFB3.X,Placenta_db{i}.DDFB3.Y,'r');
195 plot(Placenta_db{i}.DDFB3MidHomo.X,...
196      Placenta_db{i}.DDFB3MidHomo.Y,'b');
197 hold off
198 legend('MVE','DDFB','DDFB w/ Homomorphic');
199
200 figure('color','white','name',...
201         ['Placetal Image ' num2str(sel(i)) ' MCC'],...
202         'NumberTitle','off'),
```

```
203 hold on,
204 plot(Placenta_db{i}.MVE.MCCthresh,Placenta_db{i}.MVE.MCC,'g');
205 plot(Placenta_db{i}.DDFB3.MCCthresh,Placenta_db{i}.DDFB3.MCC,'r');
206 plot(Placenta_db{i}.DDFB3MidHomo.MCCthresh,...
207      Placenta_db{i}.DDFB3MidHomo.MCC,'b');
208 hold off
209 legend('MVE','DDFB','DDFB w/ Homomorphic');
```

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] A. Hoover, V. Kouznetsova, and M. Goldbaum, "Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response," *IEEE Transactions on Medical Imaging*, vol. 19, no. 3, pp. 203–210, 2000.

[2] C. Bauer and H. Bischof, "A novel approach for detection of tubular objects and its application to medical image analysis," in *Proceedings of the 30th DAGM Symposium on Pattern Recognition*, pp. 163–172, Springer-Verlag, 2008.

[3] J. Staal, M. Abramoff, M. Niemeijer, M. Viergever, and B. Ginneken, "Ridge-based vessel segmentation in color images of the retina," *IEEE Transactions on Medical Imaging*, vol. 23, no. 4, pp. 501–509, 2004.

[4] Y. Yang, S. Huang, and N. Rao, "An automatic hybrid method for retinal blood vessel extraction," *International Journal of Applied Mathematical Computer Science*, vol. 18, no. 3, pp. 399–407, 2008.

[5] C. Kirbas and F. Quek, "Vessel extraction techniques and algorithms: A survey," in *Third IEEE Symposium on Bioinformatics and Bioengineering Proceedings*, pp. 238–245, 2003.

[6] A. Frangi, W. Niessen, K. Vincken, and M. Viergever, "Multiscale vessel enhancement filtering," in *Medical Image Computing and Computer-Assisted Interventation*, pp. 130–137, Springer-Verlag, 1998.

[7] P. Truc, M. Khan, Y.-K. Lee, S. Lee, and T.-S. Kim, "Vessel enhancement filter using directional filter bank," *Computer Vision Image Understand*, vol. 113, no. 1, pp. 101–112, 2009.

[8] A. Cavallerano, J. Cavallerano, P. Katalinic, A. M. Tolson, P. Aiello, L. Aiello, *et al.*, "Use of joslin vision network digital-video nonmydriatic retinal imaging to assess diabetic retinopathy in a clinical program," *Retina*, vol. 23, no. 2, pp. 215–223, 2003.

[9] F. Zana and J. Klein, "A multimodal registration algorithm of eye fundus images using vessels detection and hough transform," *IEEE Transactions on Medical Imaging*, vol. 18, no. 5, pp. 419–428, 1999.

[10] E. Grisan, M. Foracchia, and A. Ruggeri, "A novel method for the automatic grading of retinal vessel tortuosity," *IEEE Transactions on Medical Imaging*, vol. 27, no. 3, pp. 310–319, 2008.

[11] A. Youssif, A. Ghalwash, and A. Ghoneim, "Optic disc detection from normalized digital fundus images by means of a vessels' direction matched filter," *IEEE Transactions on Medical Imaging*, vol. 27, no. 1, pp. 11–18, 2008.

[12] N. Almoussa, B. Dutra, B. Lampe, P. Getreuer, T. Wittman, C. Salafia, and L. Vese, "Automated vasculature extraction from placenta images," *Proceedings of SPIE, Medical Imaging 2011: Image Processing*, vol. 7962, 2011.

[13] L. Liu, D. Zhang, and J. You, "Detecting wide lines using isotropic nonlinear filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 6, pp. 1584–1595, 2007.

[14] C. Steger, "An unbiased detector of curvilinear structures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 2, pp. 113–125, 1998.

[15] J.-M. Chang, N. Huynh, M. Vazquez, and C. Salafia, "Vessel enhancement with multiscale and curvilinear filter matching for placenta images." Submitted 2013.

[16] H. Lange, "Automatic glare removal in reflectance imagery of the uterine cervix," *Proceedings of SPIE*, vol. 5747, pp. 2183–2192, 2005.

[17] R. Gonzalez and R. Woods, *Digital Image Processing.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2nd ed., 1992.

[18] T. Lindeberg, *Scale-Space Theory in Computer Vision.* Kluwer Academic Publishers, 1994.

[19] R. Bamberger and M. Smith, "A filter bank for the directional decomposition of images: theory and design," *IEEE Transactions on Signal Processing*, vol. 40, no. 4, pp. 882–893, 1992.

[20] S.-I. Park, M. Smith, and R. Mersereau, "A new directional filter bank for image analysis and classification," in *IEEE International Conference on Acoustics, Speech, and Signal Processing Proceedings*, vol. 3, pp. 1417–1420, 1999.

[21] S.-I. Park, M. Smith, and R. Mersereau, "Improved structures of maximally decimated directional filter banks for spatial image analysis," *IEEE Transactions on Image Processing*, vol. 13, no. 11, pp. 1424–1431, 2004.

[22] R. Bamberger, "New subband decompositions and coders for image and video compression," in *IEEE International Conference on Acoustics, Speech, and Signal Processing Proceedings*, vol. 3, pp. 217–220, 1992.

[23] W. Rudin, *Real and complex analysis.* New York, NY, USA: McGraw-Hill, Inc., 3rd ed., 1987. Theorem 7.26.

[24] IEEE Acoustics and Speech and Signal Processing Society, *Programs for digital signal processing.* IEEE, 1979. Algorithm 5.2.

[25] W.-K. Chen, *Passive, active, and digital filters.* Boca Raton, FL, USA: CRC Press, Inc., 3rd ed., 2009.

[26] S.-I. Park, *New directional filter banks and their applications in image processing, Ph.d. thesis.* School of Electrical and Computer Engineering, Georgia Institute of Technology, 1999.

[27] Wikipedia, "Receiver operating characteristic — Wikipedia, the free encyclopedia," 2013. [Online; accessed 14-2-2013].

[28] M. Niemeijer, J. Staal, B. Ginneken, M. Loog, and M. Abramoff, "Comparative study of retinal vessel segmentation methods on a new publicly available database," *Proceedings of SPIE*, vol. 5370, no. 1, pp. 648–656, 2004.

[29] X. Jiang and D. Mojon, "Adaptive local thresholding by verification-based multithreshold probing with application to vessel detection in retinal images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 1, pp. 131–137, 2003.

[30] P. Baldi, S. Brunak, Y. Chauvin, C. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: An overview," *Bioinformatics*, vol. 16, no. 5, pp. 412–424, 2000.

[31] V. Cerny, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. 45, pp. 41–51, 1985.

[32] S. Kirkpatrick, "Optimization by simulated annealing: Quantitative studies," *Journal of Statistical Physics*, vol. 34, pp. 975–986, 1984.