Multi-Scale Vessel Extraction Using Curvilinear Filter-Matching Applied to Digital Photographs of Human Placentas

Marilyn Vazquez
Advisor: Dr. Jen-Mei Chang
Department of Mathematics and Statistics
California State University, Long Beach

Abstract

Current medical interest in the placenta has inspired the work of vascular network extraction on placenta images. The focus of this research is to develop an automated program that detects vessels in these images. The placenta's irregular surface and variation in vessel coloration are some of the various challenges to extract these vessels. The multi-scale filter, based on eigenvalues of second derivatives, has been shown to be successful in identifying vessels of varying sizes on medical images. However, with the multi-scale frame work, the placenta's rough and irregular surface is also detected as part of the vessel network; therefore, we propose a new filter, a special form of the Ridgelet filter, in combination with the multi-scale filter to further enhance extraction results. This filtering process has been tried on a 181-by-181 placenta patch and a whole placenta image of size 1600-by-1200. The results on the whole placenta image have been compared to tracings of the vessel structure for the same image, the results of the multi-scale filter on its own, and the previous work on placenta images. These experiments show that the proposed filtering process has improved the multi-scale filter results and the previous work done in placentas.

1 Introduction

The placenta is a very important organ for the fetus since it provides nutrients and discards waste through the mother's blood [3]. Therefore, it is not a surprise that recent studies, such as [9], suggest that features of the placenta influence the growth of the fetus which can then affect the weight of the newborns. This is of special concern since low birth weight can lead to many difficulties for the child, such as high blood pressure within the first ten years of life [11]. Also, in [3] it is reported that there is a significant relationship between the shape of the placenta and two important health factors for newborns: the birth weight and gestational age. These findings have raised interest in the exploitation of measurable placenta characteristics to better understand diseases in newborns and possibly young adults.

Since placentas provide an inflow of nutrients and outflow of waste and in [3] it is explained that shape influenced by uterine environment and vessel growth, an important characteristic to analyze is the pervasiveness of the placental vascular network [10]. Since placenta research is still at its infancy, the only existing work in placental vessel extraction was done with the neural network approach in 2010 [1]. This research leads to very promising results, but it was done using a machine-learning technique that requires human guidance to train the computer to recognize the vessels in the images making this a very time consuming approach. In our study, we seek to develop an automated program that extracts vessel structures from digital

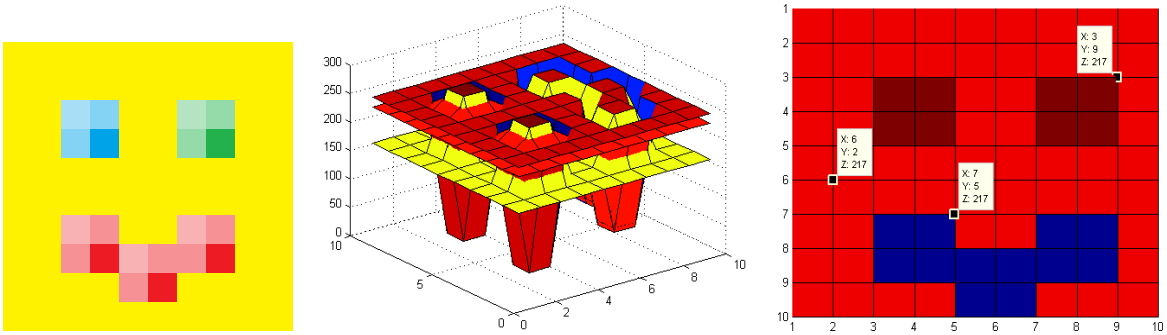
(a) (b) (c)

Figure 1: Placenta images. Each placenta varies in texture, color, and shape even within itself. Note the variations in vessel color, which are circled, in placentas (a) and (c). Also note the smoother surface on the left of placenta (b) compared to its right side.

photos of placental surfaces. The challenge faced when extracting any placental vessel network lies in the varying texture, color, and shape of placentas, which can be seen in Figure 1. Note the differences from one placenta to another and the variation in color and texture within the same placenta.

For implementation purposes, an image is defined in the following form: $I = (R, G, B) : \mathbb{Z}^2 \to \mathbb{R}^3$, where $R, G, B : \mathbb{Z}^2 \to \mathbb{R}$. Figure 2 illustrates how an image like Figure 2(a) can be decomposed into three channels and how the intensity values can be treated as the z-coordinate value, seen in Figure 2(b)-(c). Conventionally, R picks up the "red", G picks up the "green", and B picks up the "blue" of the image. All our implementations use the green channel of the image, such as Figure 2(c), for better contrast since placentas are red most of the time, and thus using the red channel does not highlight the vessels but the placenta surface. Hence, from now on, $I(x, y)$ will denote the green channel of the image function I and x, y the pixel location. For example, from Figure 2(c) $I(6, 2) = 217$ meaning that the pixel located at the position $(6, 2)$ has a "green" intensity value of 217. When an image channel is represented in this form it can be referred to as the intensity map of the image.

In our research, we used the linear filtering approach which is widely used in image processing due to its effectiveness and efficiency. A linear filtering process uses a window that moves throughout the whole image and assigns weighted values to each pixel in that window to describe the centered pixel. This process can be viewed as a convolution of the image $I(x, y)$ with a filter function $f(x, y)$ and the moving window as a convolution mask [8]. An example of a convolution mask centered at (i, j) is given in Figure 3. Notice how the convolution mask moves with respect to its center.



(a) (b) (c)

Figure 2: (a) An image, (b) its three channel decomposition, (c) the green channel, which is the one used for the implementations of this paper to create a better contrast, with some pixels labeled.

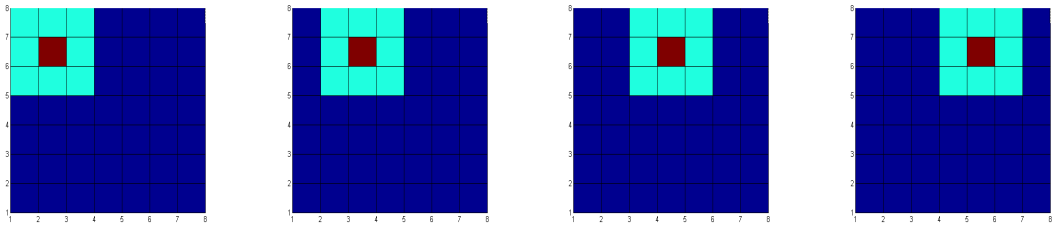


Figure 3: Convolution masks in light blue with red centers. These convolution masks move throughout the image as illustrated here.

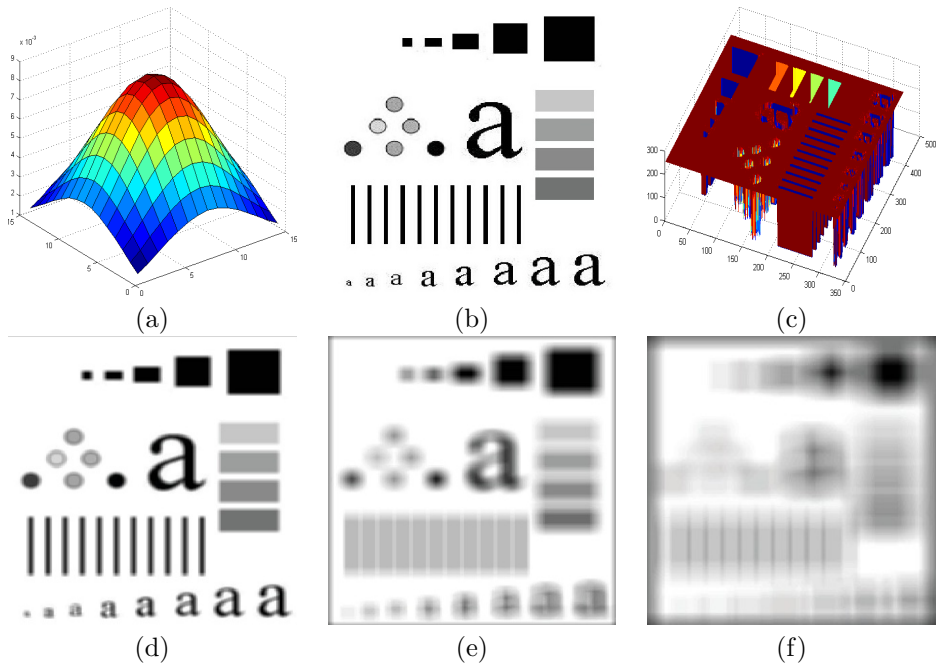


Figure 4: Gaussian filtering is a process that smoothes out images. The level of smoothing depends on the size of the filter window. (a) Gaussian filter of 15 by 15. (b) An example. (c) Intensity map of (b). (d)-(f) Image (a) convolved with the Gaussian filter of square window size 5, 15, and 35, respectively.

A concrete example of a filter is the Gaussian filter. For images, the 2D Gaussian filter function is defined in the following way:

$$G(x,y) = \mathrm{e}^{-\frac{1}{2}\left(\frac{x^2}{\sigma_1^2}+\frac{y^2}{\sigma_2^2}\right)}, \tag{1}$$

where σ_1 and σ_2 can be thought of as the "width" and "length" of the filter, respectively. For convenience, let $w = \frac{1}{\sigma_1}$ and $l = \frac{1}{\sigma_2}$. An example of this Gaussian filter is shown in Figure 4(a). If Figure 4(b) with intensity map shown in Figure 4(c) is taken as the image function $I(x,y)$, and then it is discretely convolved with Gaussian filters of squared sizes 5, 15, and 35, then Figures 4(d), 4(e), and 4(f) will be returned, respectively. This Gaussian filter function is essential in the construction of the Frangi and Ridgelet filters discussed later.

Another important component of our research is the ridgelet. The continuous ridgelet transform, as discussed in [4] and [5], is a signal analysis approach designed to pick up the lines in images. To achieve this, the line singularities are mapped to point singularities using the Randon transform, then the point singularities can then be handled with the well known wavelets. Our developed ridgelet does not use the signal analysis but has the same line detection and directional idea, and thus the name of the special form of the ridgelet. Figure 5 shows different ridgelet examples, which can later be compared to our developed

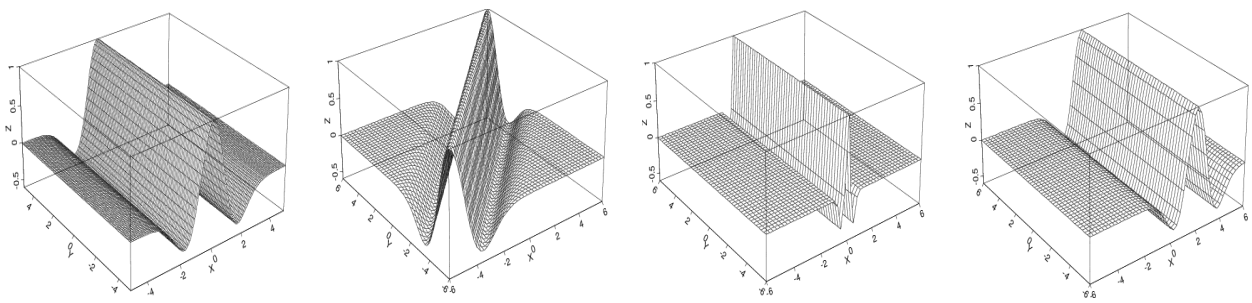


Figure 5: Different Ridgelet examples.

filter.

The following section describes the method of our research, which include the multi-scale method referred to as the Frangi approach in our discussion, the proposed filter, and how these two were combined. Section 3 then gives empirical results, and finally section 4 concludes our findings. Acknowledgements are given in section 5, and section 6 includes the MATLAB codes used for the implementation of this research.

2 The Method

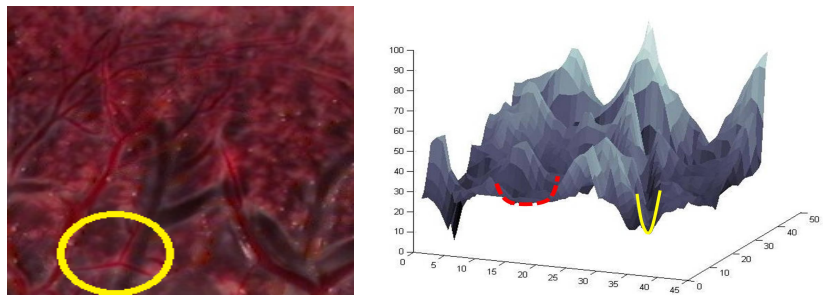
2.1 The Frangi Filter

The following discussion briefly describes the Frangi method as presented in [7]. In this article, the authors explain that vessels occupy a small area of the total area being analyzed in the images and either are lighter than their background or darker than their background. This last characteristic allowed the authors to develop a function that finds the likelihood that a pixel is part of a vessel because this contrast of intensity values is depicted in $I(x, y)$ as valleys or peaks. An example of such is seen in the yellow highlighted region in Figure 6(b), which corresponds to the vessel circled in Figure 6(a).

Recall that the intensity values of the image $I(x, y)$ can be thought as a discrete function. Thus, the Hessian, a 2-by-2 matrix containing the second partial derivatives of a function, can be calculated at each pixel as follows:

$$H(x, y) = \begin{pmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} \\ \frac{\partial^2 I}{\partial y \partial x} & \frac{\partial^2 I}{\partial y^2} \end{pmatrix} \tag{2}$$

Note that the Hessian contains all the second order information of the image needed for each pixel. This also means that just like the image, the Hessian is also a discrete function. The Hessian can be approximated



(a) (b)

Figure 6: Placentas vary in coloring, which appear as valleys and peaks in the intensity map. (a) A placenta patch, and its (b) intensity map rotated to magnify intensity values. Notice that a vessel, highlighted in yellow, and noise, highlighted in red, cannot be distinguished easily.

to a continuous function using the Gaussian filter and the differentiation property of convolution in the following manner:

$$H(x,y) \approx G \star \begin{pmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} \\ \frac{\partial^2 I}{\partial y \partial x} & \frac{\partial^2 I}{\partial y^2} \end{pmatrix} = \begin{pmatrix} \frac{\partial^2 G}{\partial x^2} & \frac{\partial^2 G}{\partial x \partial y} \\ \frac{\partial^2 G}{\partial y \partial x} & \frac{\partial^2 G}{\partial y^2} \end{pmatrix} \star I(x,y), \tag{3}$$

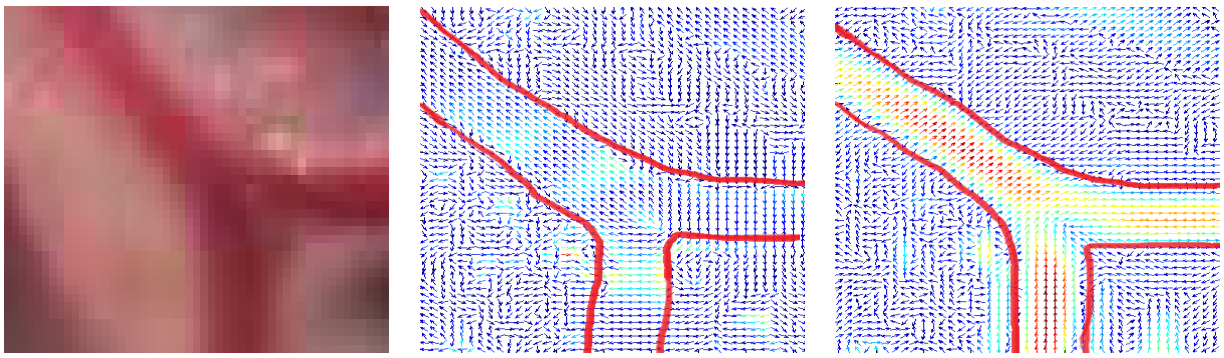
where $\star$ is the usual convolution.

In order to extract information from the Hessian matrix about contrast and direction, its eigenvalues are calculated. For simplicity, let $|\lambda_1| \leq |\lambda_2|$ denote the two eigenvalues of the Hessian matrix and $\mathbf{u}_1, \mathbf{u}_2$ the corresponding eigenvectors. Since λ_1 is the eigenvalue of smallest magnitude, it corresponds to the eigenvector, $\mathbf{u}_1$, pointing in direction of smallest curvature, and λ_2 corresponds to the eigenvector, $\mathbf{u}_2$, pointing in direction of the largest curvature. For vessel pixels this means that $\mathbf{u}_1$ points towards the direction that the vessel is going, $\lambda_1 \approx 0$, $\mathbf{u}_2$ points towards the edge of the vessel, and λ_2 is large in magnitude. In Figure 7(b), $\mathbf{u}_1$ for each pixel in a small region of the placenta is shown, and in Figure 7(c) $\mathbf{u}_2$ for the same region is shown. The red lines in each figure are for reference since they outline the approximate location of the vessel. Also notice from Figure 7(a) how the noisy background of the placenta affects the eigenvectors in Figure 7(b)-(c) since these are numerically approximated.

With this in mind, two measures were defined to measure the anisotropy and contrast of the pixel. They are defined as follows:

$$A = \frac{|\lambda_1|}{|\lambda_2|}, \text{ and } C = \sqrt{\lambda_1^2 + \lambda_2^2}.$$

Note that A will be low for vessel pixels so that the lower A is the more likely it is a vessel pixel. C will



(a) (b) (c)

Figure 7: (a) Vessel in placenta image. (b)-(c) Eigenvectors corresponding to the smallest magnitude and larger magnitude, respectively. Note that the direction of the eigenvector $\mathbf{u}_1$ in (b) is the same as the vessel path and the direction for the eigenvector $\mathbf{u}_2$ in (c) points towards the side of the vessel.

be low if both the eigenvalues are small for the lack of contrast so that the larger C is the more likely it is a vessel.

For images where the vessels are darker than their background, meaning that the vessels are valleys, the curvature will be negative so $\lambda_2 < 0$. Hence, the developed likelihood function, or "vesselness equation," is:

$$F(\cdot) = \begin{cases} 0 & \text{if } \lambda_2 > 0, \\ e^{-\frac{A^2}{2\alpha^2}} \left(1 - e^{\frac{C^2}{2\beta^2}}\right) & \text{otherwise,} \end{cases} \tag{4}$$

where α and β are parameters to adjust A and C, respectively. If there is a high probability that the pixel being analyzed is part of a vessel, then a high output value, also called response value, is assigned to that pixel. This process is repeated for every pixel and the response values are the output.

The drawback of applying this filter on placenta images is that it detects background noise as part of the vessel structure because of the non-uniform texture of the placenta. Figure 6(b) shows the intensity values of the placenta patch in Figure 6(a). Note how the region highlighted in yellow corresponds to a vessel, and the region highlighted in red is not a vessel; however, when Frangi method is applied to Figure 6(a), the Frangi response is about the same for some non-vessel pixels and some vessel pixels because they are both "valley" structures. This is seen in Table 2.1 where actual Frangi reponse values are shown. From these values it is clear that some vessel pixels cannot be differentiated from non-vessel pixels.

Table 1: Actual Frangi response values

x	y	$I(x,y)$	vessel/nonvessel	Frangi response
115	128	22	vessel	0
159	161	13	non-vessel	0.03983
334	319	15	non-vessel	0.1477
147	395	20	vessel	0.0228
303	428	32	vessel	0

2.2 The Special Form of the Ridgelet

Since the Frangi method on its own is not enough to identify only the vessel pixels in the images, we propose a new filtering process. The first step in the filtering process is to enlarge the original image to three times its original size using a bi-cubic interpolation E by a factor of r. This will further approximate the discrete image function I to a continuous function and will also give us a larger data with which we can work. Loosely speaking, if we call the process of calculating F a Frangi filtering process, then F is applied to the re-sized image and only pixels that have a response value higher than zero, meaning that they are possible vessel pixels, are considered. That is, a pixel (X, Y) is a vessel pixel if $F\{E_r\,(I(X,Y))\} > 0$. To simplify notations, let us define a binary response:

$$B(X,Y) = \begin{cases} 1 & \text{if } F\{E_r\,(I(X,Y))\} > 0, \\ 0 & \text{otherwise.} \end{cases}$$

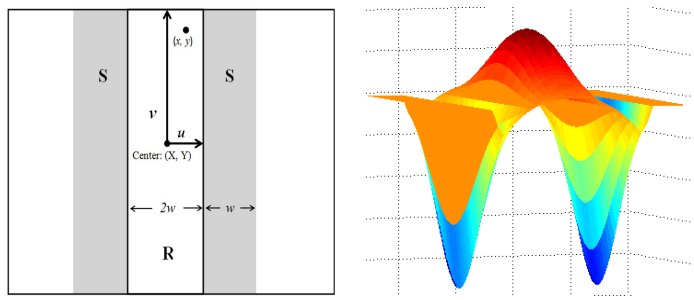
A pixel (X, Y) inside of $B^{-1}\{1\}$ has a high likelihood to be a vessel pixel, but because Frangi identifies some background noise as vessel pixels, further considerations are needed. On the images, vessels are locally linear and have long structures. Hence, we developed a filter that (1) highlights the linear structures, (2) penalizes surrounding pixels, and (3) weights the importance of the surrounding pixels for implementation purposes according to how close they are to the one being analyzed. In other words if the pixel $(X, Y) \in B^{-1}\{1\}$ is part of a linear structure of width $2w$, then there are two orthogonal vectors $u, v \in \mathbb{R}^2$ with $||u|| = w$ and $||v||>w$ such that:

(1) $R = \{r\mathbf{u} + t\mathbf{v} + (X,Y) \mid -1 < r, t < 1\} \subseteq B^{-1}\{1\}$

(2) $S = \{r\mathbf{u} + t\mathbf{v} + (X,Y) \mid -1 < t < 1, 1 < |r| < 2\} \subseteq B^{-1}\{0\}$

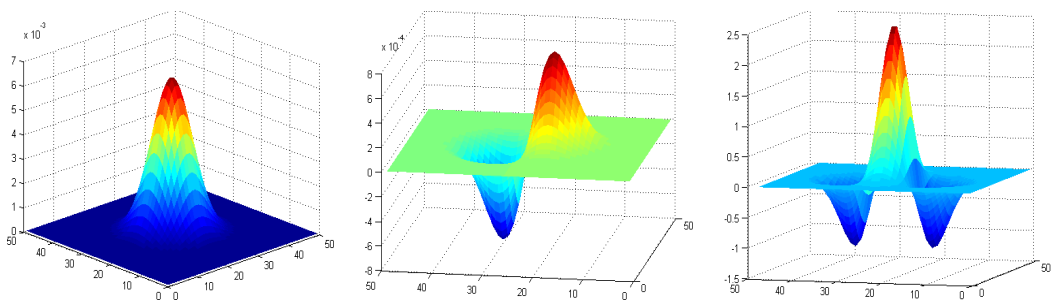
(3) points in $R \cup S$ are inversely weighed based on their distances to pixel (X, Y)

Figure 8(a) provides a visualization of the conditions. Note that our proposed filter is a special form of



(a) (b)

Figure 8: Our proposed filter works by detecting the linear structures and penalizing the surrounding non-vessel pixels. (a) A diagram of the elements of the criteria that our filter tries to satisfy. The (X, Y) is the pixel being analyzed, R is the vessel region, and w is the width of the filter. Since (X, Y) is inside R, applying the Ridgelet filter will return a high response value. (b) The Ridgelet filter.



(a) (b) (c)

Figure 9: (a) Squared Gaussian filter of size 50, (b) the first partial derivative of the filter, and (c) negative of the second partial derivative of the filter.

the Ridgelet because of criterion (1). Figure 8(b) shows a filter that satisfies all 3 criteria, which clearly resembles the Ridgelet mentioned earlier.

To meet the first two criterion, it is possible to use a filter approach that rewards pixels in R and penalizes those in S. Because the second derivative of the Gaussian filter has a structure with a centered peak, valleys in the side, and a leveling out effect in the further away pixels, all seen in Figure 9(c), it can be used to meet the criteria. Figure 9(a) shows the Gaussian filter and Figure 9(b) the first derivative of the Gaussian for completeness.

The actual function for the second derivative of the 2D Gaussian is:

$$-\frac{\partial^2 G}{\partial x^2} = \left(1 - w^2 x^2\right) \exp\left(-\frac{1}{2}(w^2 x^2 + l^2 y^2)\right) \tag{5}$$

With Equation (5), a large weight will be given to surrounding pixels, a negative weight will be given

to pixels in region S, and pixels that are too far away will be dismissed by giving them a weight of zero. However, for the distribution to be smooth and the width of consideration to be $2w$, the term w^2x^2 in Equation (5) is replaced with $\frac{3}{4-w^2x^2}$. This way, as $wx \to \pm 2$, $\mathrm{e}^{\left(-\frac{1}{2}\left(\frac{3}{4-w^2x^2}+l^2y^2\right)\right)} \to 0$ and we have a smooth distribution. Hence, our proposed Ridgelet function is:

$$\Psi(x,y) = \begin{cases} \frac{1-w^2x^2}{4-w^2x^2}\mathrm{e}^{\left(-\frac{1}{2}\left(\frac{3}{4-w^2x^2}+l^2y^2\right)\right)} & \text{if } |x|<2w \\ 0 & \text{if } |x| \geq 2w \end{cases} \tag{6}$$

The filter can be seen in Figure 8(b).

Notice how our Ridgelet is a heuristic filter since it does not include direction information and vessels can point at different angles. To incorporate direction information in our filter equation, consider a certain direction, $\mathbf{u}$, a rotation function, $T_{\mathbf{u}}$, that maps $\mathbf{u}$ to $(||\mathbf{u}||, 0)$, and the Ridgelet template $W(x,y) = \Psi \circ T_{\mathbf{u}}(x,y)$. In order to take account for many different directions, we created a library of templates W_k corresponding to the collection of n orientations $\frac{\pi}{n}$ radians apart from each other: $\{\mathbf{u}_k = (\cos\theta_k, \sin\theta_k) | \theta_k = \frac{k\pi}{n}\}_{k=1}^{n}$. We then end up with $W_k(x,y) = \Psi \circ T_k(x,y)$, where

$$T_k(x,y) = \begin{bmatrix} \cos\theta_k & -\sin\theta_k \\ \sin\theta_k & \cos\theta_k \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

From these, define the response values $V_k := W_k \star B(x,y)$ at different k values, which will give us a set of k different response outputs. With this function, the following are achieved:

(1) Pixels in region R will return a positive V_k

(2) Pixels in S return negative V_k

(3) Pixels close to the one being analyzed are given a higher weight in the $\Psi(x,y)$ function

In addition, we also have achieved: (4) only pixels that Frangi identified as vessel pixels are considered, since $V_k = 0$ otherwise, and (5) different vessel directions are considered using the W_k function. This shows that we have achieved all our goals. Figure 10 summarizes this section. The first step was to construct the Ridgelet filter with the Ψ function, such as in Figure 11(a), then rotations were added with the T_k templates and hence we get a directory of filters W_k such as Figure 11(a)-(d). Finally, these filters were applied to the Frangi response, as shown in Figure 11(e)-(h).

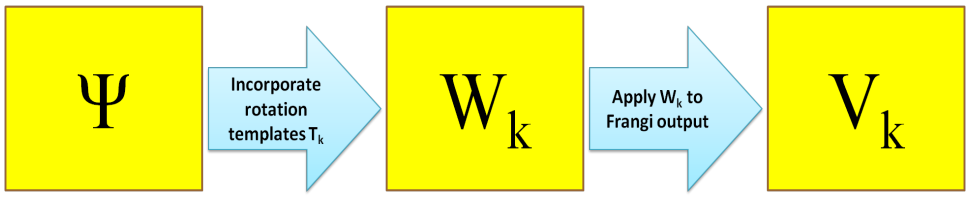


Figure 10: Summary of the development of the Ridgelet filtering process. First the filter function Ψ was found, then rotation templates were incorporated, and then this was applied to the Frangi output.

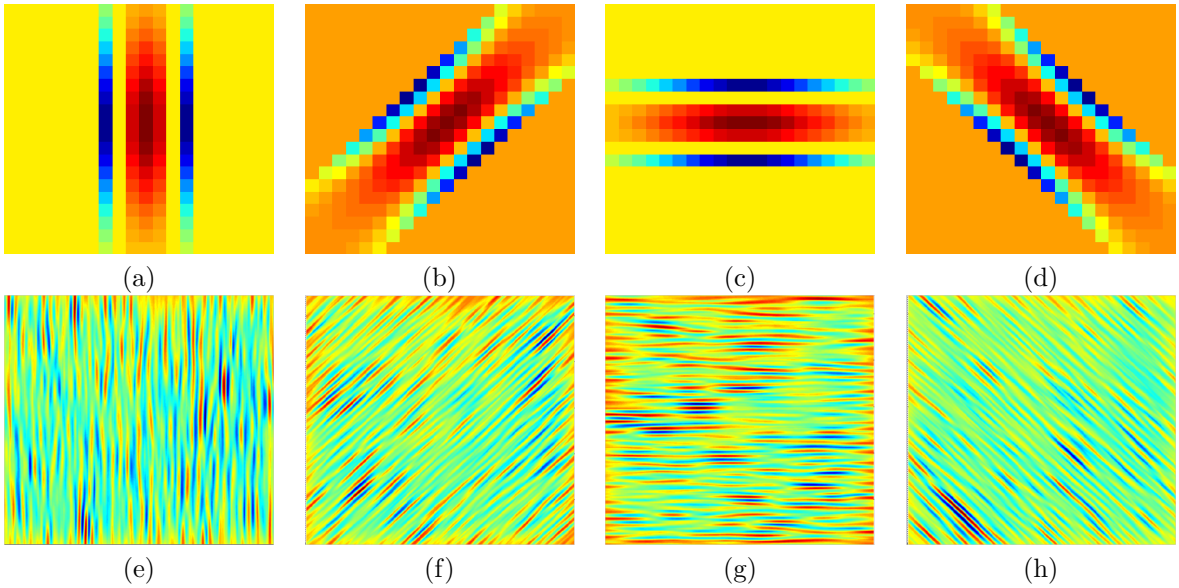


Figure 11: The Ridgelet filters and their response when applied to the Frangi output. Notice that the Ψ function simply gives a filter such as in (a), and when the templates T_k are put in place, the filter directory W_k is assembled with filters of different angles such as (a)-(d). Finally, when these filters are applied to the Frangi output, the responses V_k are such as in (e)-(h).

		True ID	
		True	False
Labeled ID	True	True Positive	False Positive
	False	False Negative	True Negative

2.3 Combining Both Filters and Thresholding

Now that the Ridgelet has helped us locate the most linear regions from the Frangi filter, the next step is to combine the results from these two filters. Recall that after applying $W_k(x, y)$ on $B(x, y)$, there is a collection of responses per pixel, that is $\{V_k\}_{k=1}^{n}$. With this information, each pixel is given the highest response value from the Ridgelet filter so that

$$V(x, y) = \max_{1 \leq k \leq n} V_k.$$

Figure 12(a)-(d) illustrate this for $n = 12$ and $k = 3, 6, 9, 12$. The next step is to divide the Frangi output into components B_i; that is, if neighboring pixels were identified as vessel pixels, they will belong to the same component, or to the same B_i. An example of the Frangi output and its corresponding connected components is given in Figure 12(e) and (f), respectively. In Figure 12(f), each color represent a different component, or B_i. For implementation purposes, we then assign the maximum value of $V(x, y)$ for all $(x, y) \in B_i$ to every pixel in the component B_i and call this new $V(x, y)$ function $V'(x, y)$. To further improve the enhancement, we choose a threshold μ and use it to discard pixels with the new $V(x, y)$ response value below the threshold. Examples of threshold values and their effects in an image are shown in Figures 12(g)-(h). The resulting filtering process identifies pixels as vessel pixels by $\mathcal{V}_\mu = \cup_i Z_i$, where $Z_i = \{B_i | V'(x, y) > \mu\}$. A diagram illustrating the combination process is presented in Figure 13.

3 Empirical Results

To test the effectiveness of the filtering process, we used the confusion matrix to compare our results to the hand tracings of the placenta images done by pathologist. The confusion matrix, also known as the contingency matrix, is a 2×2 matrix used to display the four possible outcomes: true positives (correct classification of vessel pixels), true negatives (correct classification of non-vessel pixels), false negatives (classification of vessel pixels as non-vessel pixels), and false positives (classification of non-vessel pixels as vessel pixels) [6]. The confusion matrix can be constructed in the following form:

By convention, we will refer to the true positive, false positive, false negative, and true negative as

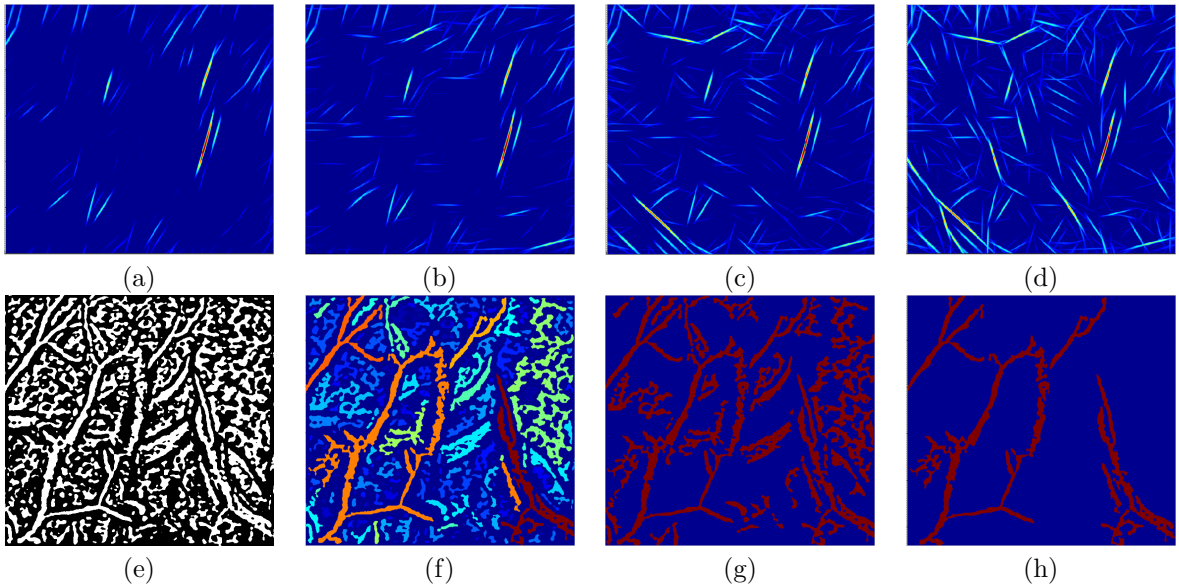


Figure 12: Maximum responses for angles (a) $\frac{\pi}{4}$, (b) $\frac{\pi}{2}$, (c) $\frac{3\pi}{4}$, and (d) π. (e) Frangi response, (f) Frangi separated into components, and thresholds of (g) 20 and (h) 40 applied to the combined result.

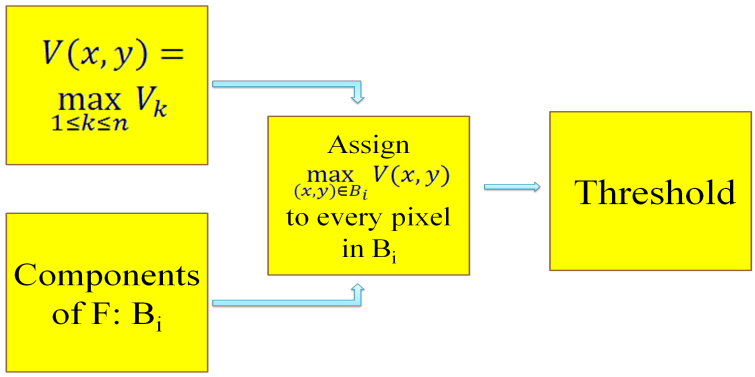


Figure 13: Summary of the combination process. Once the Frangi output is divided into components and $V(x, y)$ is calculated, the maximum $V(x, y)$ value in the each component is assigned to every pixel in that component and then a threshold is placed.

$TP, FP, FN,$ and TN, respectively.

The MCC metric was used to measure how well our Frangi-Ridgelet detects vessel pixels because, unlike other measures, it utilizes all the parts of the confusion matrix giving a more balanced value. The only problem with this metric comes when there are very few false positives and true positives [2]. Our algorithm returns a very high number of true positives, so this should not be a problem. Another reason why the MCC metric was used is to have comparable data to the Neural Networks approach [1], in which the MCC was also used to measure the effectiveness of the method. The MCC metric is defined in the following manner:

$$\text{mcc}(x, y) = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FN)(TP + FP)(TN + FP)(TN + FN)}}$$

The MCC metric is a correlation measure with values between -1 and 1 that for our case will be measuring how related the Frangi-Ridgelet identification of the vessels are to the actual vessel locations. In our case, a MCC value of 0 means that the identification did no better than a random identification of pixels, and anything higher than 0 represents a much better identification of pixels than a random one.

Since the MCC gives an idea of how well the filtering process identified the vessels for the given parameters, it can be plotted against different threshold values to compare the overall accuracy of the Frangi-Ridgelet filtering process. Then, the area under the MCC curve can be calculated to have a single value to compare results. Since a zero value can be thought as random identification and one as perfect identification, the area under a curve of random identification would be zero and area under a curve of perfect identification would be simply the number of threshold values. Since the Neural Networks research returned threshold values go up to 1.01, the x-axis of the MCC curve for our results were scaled down to 1.01 for fair comparison.

After running several tests on the placenta image in Figure 14(a), we concluded that the parameters that give the best overall results, meaning that it returns the MCC curve with the largest area under the curve, are: resizing scale = 3, Frangi scale = 5, Ridgelet width = 28, Ridgelet length = 36, and number of Ridgelet filters = 12. This set of parameters will be later referred as **experiment 1**, and Figure 14(e) shows the result of using these parameters. However, we found that a higher MCC value was achieved with the parameters: resizing scale = 2, Frangi scale = 6, Ridgelet width = 14, Ridgelet length = 35, and number of Ridgelet filters = 10. This new set of parameters will be later referred as **experiment 2**, and the result for these parameters are shown in Figure 14(f).

The Frangi-Ridgelet process in Figures 14(e)-(f) was able to get rid of much of the noise that Frangi picked up and has a much cleaner display. Another visual advantage of our proposed filter is that it picked up some vessels that the Neural Networks did not and the vessels are better connected as shown in the

Table 2: Results

Method	Highest MCC value	Area Under the MCC curve
Neural Networks	0.345	0.22
Frangi	0.2552	0.1216
Experiment 1	0.3539	0.25
Experiment 2	0.3676	0.2328

circled portions of Figure 14(f).

For more objective results, one can look at the area under the MCC curve and from these values it is clear that the process improved the Frangi results and those from the Neural Networks. From the scaled MCC curve, shown in Figure 15(a), the area under it was calculated to be 0.25 and 0.2328 for experiment 1 and experiment 2, respectively. The highest MCC value for experiment 1 was 0.3539 at a threshold value of 148 and for experiment 2 it was 0.3676 at a threshold value of 77. As shown in [1], the highest area under the curve for their results was 0.22 and the highest reported MCC value was 0.345 as indicated in Figure 15(b). For the same image resized to 3 times its original size and Frangi with a scale of 5 applied on this image, the area under its MCC curved with an x-axis scaled down to 1.01 is 0.073 and its highest MCC value 0.1754 at a threshold of 6. If instead the image is resized to 2 times its original size and Frangi with a scale of 6 is applied on it, the area under its MCC curved with an x-axis scaled down to 1.01 is 0.1216 and the highest MCC value is 0.2552 at a threshold of 12. The summary of these numbers can be seen in Table 2.

The disadvantage of the proposed method can be seen visually when its results are compared with the ones from Frangi and Neural Networks, shown in Figure 14(c) and (d) respectively. Notice that our proposed filtering process losses some information. For example, in Figure 14(c), the highlighted vessels that Frangi identified were not identified in Neural Networks or our Frangi-Ridgelet process. Also, in Figure 14(d), the yellow vessels were not detected with our proposed filter. However, as seen from the area under the MCC curve, we know that the overall result is better than the one from Frangi and Neural Networks.

From these results we see that our proposed filter has improved results from the other methods tried on placenta images. On top of that, the proposed Frangi-Ridgelet is able to run in about 2 minutes on a 1600×1200 placenta image, providing us with a new method for general vessel extraction of medical images. The results where obtained using MATLAB in a 64-bit laptop with Intel Core i7 at 1.73 GHz CPU and 6 GB RAM, and the computational time was measured in CPU time.

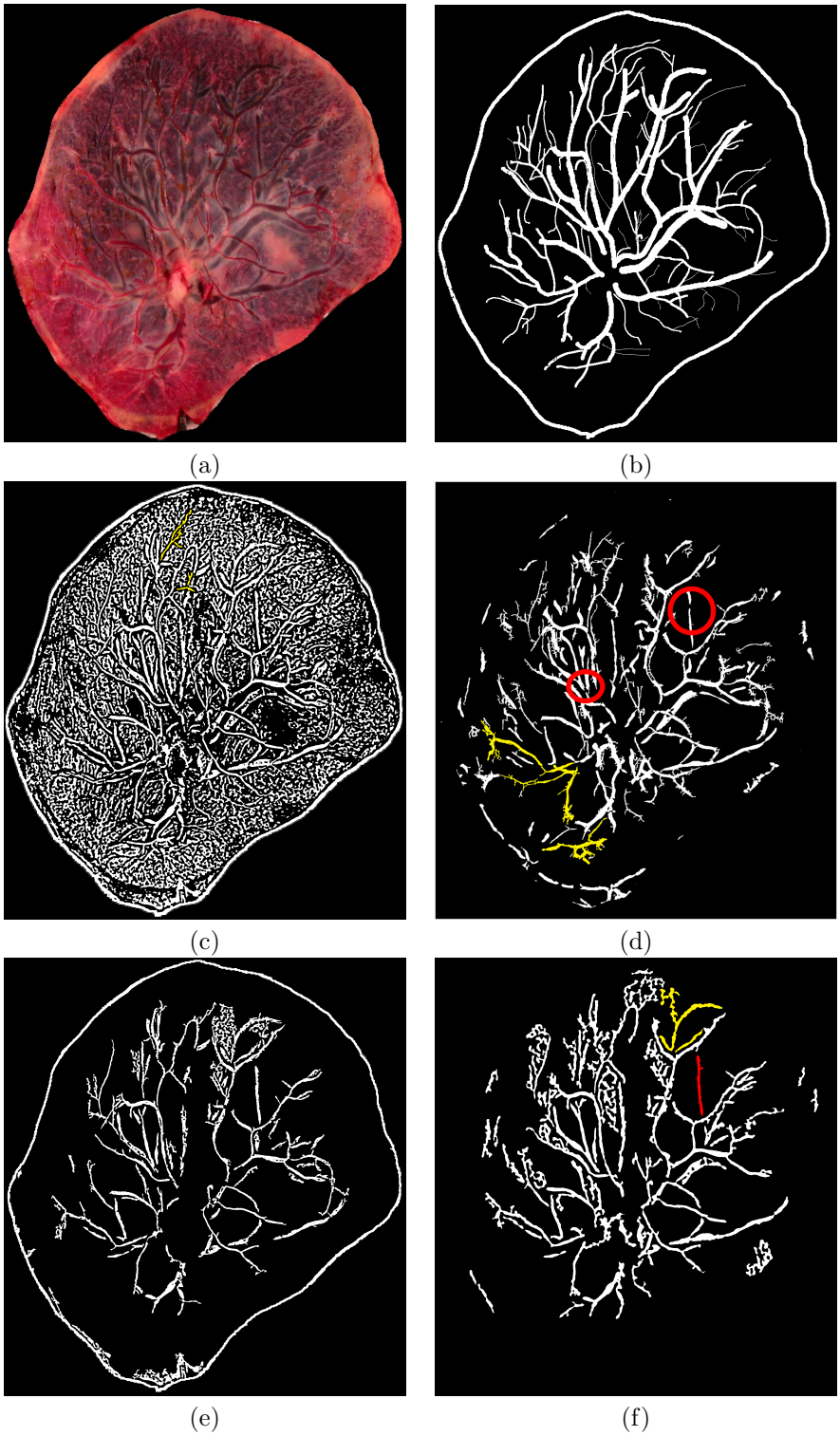
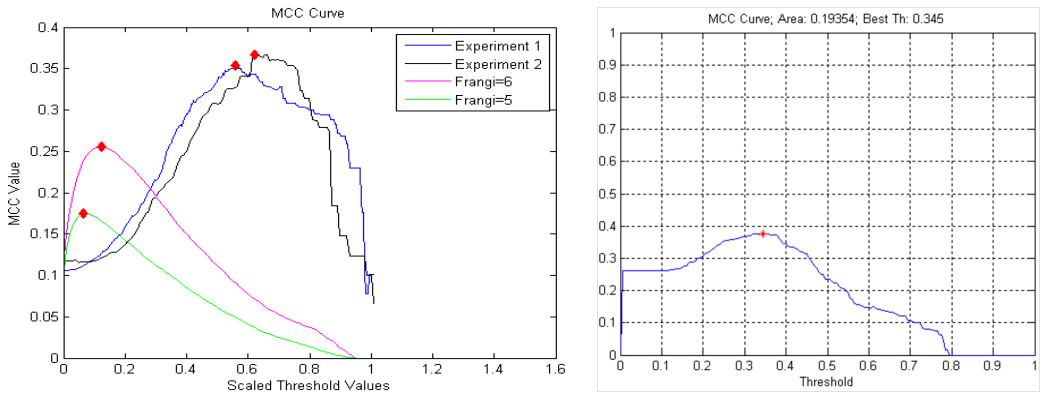


Figure 14: (a) Original image, (b) hand tracing, (c) Frangi results with a threshold of 0, (d) Neural Networks results, (e) Frangi-Ridgelet with resizing = 3, Frangi scale = 5, Ridgelet width = 14, length = 36, number of filters = 12, and (f) Frangi-Ridgelet with resizing = 2, Frangi scale = 6, Ridgelet width = 7, length = 35, number of filters = 10.



(a) (b)

Figure 15: (a) The MCC curves for experiment 1, experiment 2, and Frangi experiments. (b) The Neural Networks results found in [1].

4 Summary

The Frangi-Ridgelet filtering process seeks to identify the vessels on placenta images. The process starts resizing the image and then applying the Frangi filter. Then, this output is enhanced through the application of a Ridgelet function to extract the most linear structures. These two results are then combined and thresholded for better identification. The results were measured in accuracy using the MCC metric. Our experiments show that our proposed filtering process improved the results of the well known multi-scale vessel enhancement and the previous work done in placenta images, the Neural Networks. These are very exciting results taking into account that placenta images are extremely difficult to segment due to their texture.

5 Acknowledgements

I would like to thank my collaborator, Nen Huynh, for all his contribution and patient mentoring, and Dr. Carolyn Salafia from Placental Analytics, LLC for providing the digital photographs of placenta surfaces.

6 Appendix

```
function [frangiwhale,frangi,whale,whaleDir] = FrangiWhaleEff( grayImg,...
                                    resizeScale,frangiScale,isdarklight,...
                                    whaleWidth,whaleLength,whaleNumber,...
                                    verbose)
%FrangiWhaleEff returns a much cleaner version of the Frangi Filter output.
%First, Frangi Filter is applied on an image to identify vessels and a
%result is returned. Then using convolve2, a function made by David Young,
%the Ridgelet filter is applied on the Frangi output and the most linear
%parts of the vessels as identified with Frangi are given higher intensity
%values, which is depicted by red and orange colors, and the less linear
%parts are given lower intensity values, depicted by yellow, green, and
%blue colors. The high intensity parts of the vessels and anything
%connected to them are marked and anything else is discarded returning the
%cleaner version of the Frangi output.
%
%INPUTS:
%       grayImg - the green channel of the original image
%       resizeScale - the enlargement parameter. For example, a 2 will
%                     output twice the original image.
%       frangiScale - the "size" of the vessels to be identified.
%       isdarklight - whether the vessels are darker compared to the
%                     background or viceversa. (true for our experiments)
%       whaleWidth - half the width of the ridgelet filter
%       whaleLength - half the length of the ridgelet filter
%       whaleNumber - the number of filter directions to be checked.
%       verbose - gives information on what process is running and how long
%                 it takes.
%OUTPUTS:
%       frangiwhale - the result of our filtering process
%       frangi - the response value of the Frangi filter
%       whale - the response value of the whale filter
%       whaleDir - the number of the whale filter corresponding to the
%                  highest response value.
%
%Example: Experiment 1
% 1)Input your image in MATLAB and grab the green channel:
```

```
%   I = imread('filename.ext');
%   I = double(I(:,:,2));
% 2)Ready to go! The following code will give you the output for experiment
%   1 before thresholding.
%   [frangiwhale,frangi,whale,whaleDir] = FrangiWhaleEff( I,3,5,true,14,...
%                                                          36,12,verbose)

% Resizing the image and setting up the paramenters for Frangi
grayImg = imresize(grayImg,resizeScale);
options = struct('FrangiScaleRange', [frangiScale frangiScale],...
        'FrangiScaleRatio', 1, 'FrangiBetaOne', 0.5,'FrangiBetaTwo', 15,...
        'verbose',verbose,'BlackWhite',isdarklight);

%Starting timer for Frangi
if verbose
    tic;
end

%Applying Frangi on the image
frangi = FrangiFilter2D(grayImg,options);

%Displaying timer
if verbose
    disp(['Frangi took: ' num2str(toc)]);
end

%Segmenting the Frangi respose and applying the whale filters on this
%response
bw = frangi > 0;
[whale,whaleDir] = WhaleFilterEff(bw,whaleWidth,whaleLength,whaleNumber,...
                                  verbose);
[label,numComp] = bwlabel(bw,4);

%Starting timer for ridgelet
if verbose
    tic;
end

%Finding the maximum ridgelet response for each component
```

```
maxCompValues = zeros(numComp,1);
for i = 1 : size(label,1)
    for j = 1 : size(label,2)
        if label(i,j) > 0 && maxCompValues(label(i,j)) < whale(i,j)
            maxCompValues(label(i,j)) = whale(i,j);
        end
    end
end

%Substitutes each value to the maximum value
frangiwhale = zeros(size(bw));
for i = 1 : size(bw,1)
    for j = 1 : size(bw,2)
        if label(i,j) > 0
            frangiwhale(i,j) = maxCompValues(label(i,j));
        end
    end
end

%Displaying timer
if verbose
    disp(['Getting the components took ' num2str(toc)]);
end

end
```

```
function [outIm,whatScale,Direction] = FrangiFilter2D(I, options)
% This function FRANGIFILTER2D uses the eigenvectors of the Hessian to
% compute the likeliness of an image region to vessels, according
% to the method described by Frangi:2001 (Chapter 2).
%
% [J,Scale,Direction] = FrangiFilter2D(I, Options)
%
% INPUTS
%   I : The input image (vessel image)
%   Options : Struct with input options,
%       .FrangiScaleRange : The range of sigmas used, default [1 8]
%       .FrangiScaleRatio : Step size between sigmas, default 2
```

```
%       .FrangiBetaOne : Frangi correction constant, default 0.5
%       .FrangiBetaTwo : Frangi correction constant, default 15
%       .BlackWhite : Detect black ridges (default) set to true, for
%                         white ridges set to false.
%       .verbose : Show debug information, default true
%
% OUTPUTS
%   J : The vessel enhanced image (pixel is the maximum found in all scales)
%   Scale : Matrix with the scales on which the maximum intensity
%           of every pixel is found
%   Direction : Matrix with directions (angles) of pixels (from minor eigenvector)
%
% Example,
%   I=double(imread ('vessel.png'));
%   Ivessel=FrangiFilter2D(I);
%   figure,
%   subplot(1,2,1), imshow(I,[]);
%   subplot(1,2,2), imshow(Ivessel,[0 0.25]);
%
% Written by Marc Schrijver, 2/11/2001
% Re-Written by D.Kroon University of Twente (May 2009)

defaultoptions = struct('FrangiScaleRange', [1 10], 'FrangiScaleRatio', 2, ...
    'FrangiBetaOne', 0.5, 'FrangiBetaTwo', 15, 'verbose',true,'BlackWhite',true);

% Process inputs
if(¬exist('options','var')),
    options=defaultoptions;
else
    tags = fieldnames(defaultoptions);
    for i=1:length(tags)
         if(¬isfield(options,tags{i})),  options.(tags{i})=defaultoptions.(tags{i}); end
    end
    if(length(tags)≠length(fieldnames(options))),
        warning('FrangiFilter2D:unknownoption','unknown options found');
    end
end

sigmas=options.FrangiScaleRange(1):options.FrangiScaleRatio:options.FrangiScaleRange(2);
```

```
sigmas = sort(sigmas, 'ascend');

beta  = 2*options.FrangiBetaOne^2;
c     = 2*options.FrangiBetaTwo^2;

% Make matrices to store all filterd images
ALLfiltered=zeros([size(I) length(sigmas)]);
ALLangles=zeros([size(I) length(sigmas)]);

% Frangi filter for all sigmas
for i = 1:length(sigmas),
    % Show progress
    if(options.verbose)
        disp(['Current Frangi Filter Sigma: ' num2str(sigmas(i)) ]);
    end

    % Make 2D hessian
    [Dxx,Dxy,Dyy] = Hessian2D(I,sigmas(i));

    % Correct for scale
    Dxx = (sigmas(i)^2)*Dxx;
    Dxy = (sigmas(i)^2)*Dxy;
    Dyy = (sigmas(i)^2)*Dyy;

    % Calculate (abs sorted) eigenvalues and vectors
    [Lambda2,Lambda1,Ix,Iy]=eig2image(Dxx,Dxy,Dyy);

    % Compute the direction of the minor eigenvector
    angles = atan2(Ix,Iy);

    % Compute some similarity measures
    Lambda1(Lambda1==0) = eps;
    Rb = (Lambda2./Lambda1).^2;
    S2 = Lambda1.^2 + Lambda2.^2;

    % Compute the output image
    Ifiltered = exp(-Rb/beta) .*(ones(size(I))-exp(-S2/c));

    % see pp. 45
```

```
    if(options.BlackWhite)
        Ifiltered(Lambda1<0)=0;
    else
        Ifiltered(Lambda1>0)=0;
    end
    % store the results in 3D matrices
    ALLfiltered(:,:,i) = Ifiltered;
    ALLangles(:,:,i) = angles;
end

% Return for every pixel the value of the scale(sigma) with the maximum
% output pixel value
if length(sigmas) > 1,
    [outIm,whatScale] = max(ALLfiltered,[],3);
    outIm = reshape(outIm,size(I));
    if(nargout>1)
        whatScale = reshape(whatScale,size(I));
    end
    if(nargout>2)
        Direction = reshape(ALLangles((1:numel(I))'+(whatScale(:)-1)*numel(I)),size(I));
    end
else
    outIm = reshape(ALLfiltered,size(I));
    if(nargout>1)
            whatScale = ones(size(I));
    end
    if(nargout>2)
        Direction = reshape(ALLangles,size(I));
    end
end
```

```
function [whale,whaleDir] = WhaleFilterEff(bw,whaleWidth,whaleLength,...
                                           whaleNumbers,verbose)
% WhaleFilter uses the function convolve2, made by Daving Young, to create
% linear filters of different angles and when applied in an image, response
% values and the corresponding directions are returned.
%
%INPUTS:
```

```
%       bw -  matrix of logicals
%       whaleWidth - half the width of the ridgelet filter
%       whaleLength - half the length of the ridgelet filter
%       whaleNumbers - the number of filters to be produced
%       verbose - gives information on what process is running and how long
%                 it takes
%OUTPUTS:
%       whale - the maximum response value of the ridgelet filter applied
%               on bw
%       whaleDir - the number of the filter corresponding to the maximum
%               response value of the whale filter on bw

%Setting up
bw = double(bw);
Values = zeros(size(bw,1),size(bw,2),2);
whaleDir = zeros(size(bw,1),size(bw,2));

if verbose
    disp(['Applying ' num2str(whaleNumbers) ' whale filters']);
end

%Makes and applies the filters
for k = 1 : whaleNumbers
    if verbose
        tic;
    end

    Filter = MakeWhaleFilter(whaleWidth,whaleLength,k*pi/whaleNumbers);
    Values(:,:,2) = convolve2(bw,Filter,'same',10^(-7));
    [Values(:,:,1),Temp] = max(Values,[],3);
    whaleDir(Temp>1) = k;

    if verbose
        disp(['Whale filter ' num2str(k) ' took ' num2str(toc) ' secs']);
    end
end

whale = Values(:,:,1);
end
```

```
function [Filter] = MakeWhaleFilter(whaleWidth,whaleLength,angle)
% MakeWhaleFilter creates the ridgelet filters.
%
% INPUTS:
%        whaleWidth - half the width of the ridgelet filter
%        whaleLength - half the length of the ridgelet filter
%        angle - The angle of rotation
% OUTPUT:
%        Filter - the filter with the specified parameters

% Setting up
n = 2*max([whaleWidth whaleLength]);

% Make Coordinate
[x,y] = meshgrid(-n:n,-n:n);

% Rotate
X = x.*cos(angle)-y.*sin(angle);
Y = x.*sin(angle)+y.*cos(angle);

% The filter
Filter = (abs(X) < 2*whaleWidth).*(1-(X/whaleWidth).^2)./...
          (4-(X/whaleWidth).^2).*exp(-1/2*((3./(4-(X/whaleWidth).^2))+...
          (Y/whaleLength).^2));
Filter(isnan(Filter)) = 0;

end
```

```
function y = convolve2(x, m, shape, tol)
%CONVOLVE2 Two dimensional convolution.
%   Y = CONVOLVE2(X, M) performs the 2-D convolution of matrices X and
%   M. If [mx,nx] = size(X) and [mm,nm] = size(M), then size(Y) =
%   [mx+mm-1,nx+nm-1]. Values near the boundaries of the output array are
%   calculated as if X was surrounded by a border of zero values.
%
%   Y = CONVOLVE2(X, M, SHAPE) where SHAPE is a string returns a
%   subsection of the 2-D convolution with size specified by SHAPE:
```

```
%
%       'full'    — (default) returns the full 2—D convolution,
%       'same'    — returns the central part of the convolution
%                   that is the same size as X (using zero padding),
%       'valid'   — returns only those parts of the convolution
%                   that are computed without the zero—padded
%                   edges, size(Y) = [mx—mm+1,nx—nm+1] when
%                   size(X) > size(M),
%       'wrap'    — as for 'same' except that instead of using
%                   zero—padding the input X is taken to wrap round as
%                   on a toroid.
%       'reflect' — as for 'same' except that instead of using
%                   zero—padding the input X is taken to be reflected
%                   at its boundaries.
%
%   CONVOLVE2 is fastest when mx > mm and nx > nm — i.e. the first
%   argument is the input and the second is the mask.
%
%   If the rank of the mask M is low, CONVOLVE2 will decompose it into a
%   sum of outer product masks, each of which is applied efficiently as
%   convolution with a row vector and a column vector, by calling CONV2.
%   The function will often be faster than CONV2 or FILTER2 (in some
%   cases much faster) and will produce the same results as CONV2 to
%   within a small tolerance.
%
%   Y = CONVOLVE2(... , TOL) where TOL is a number in the range 0.0 to
%   1.0 computes the convolution using a reduced—rank approximation to
%   M, provided this will speed up the computation. TOL limits the
%   relative sum—squared error in the effective mask; that is, if the
%   effective mask is E, the error is controlled such that
%
%       sum(sum( (M—E) .* (M—E) ))
%       ——————————————————————————    ≤  TOL
%            sum(sum( M .* M ))
%
%   See also CONV2, FILTER2.

% Copyright David Young, Feb 2002, revised Jan 2005, Jan 2009, Apr 2011

```

```
% Deal with optional arguments
error(nargchk(2,4,nargin));
if nargin < 3
    shape = 'full';    % shape default as for CONV2
    tol = 0;
elseif nargin < 4
    if isnumeric(shape)
        tol = shape;
        shape = 'full';
    else
        tol = 0;
    end
end;

% Set up to do the wrap & reflect operations, not handled by conv2
if ismember(shape, {'wrap' 'reflect'})
    x = extendarr(x, m, shape);
    shape = 'valid';
end

% do the convolution itself
y = doconv(x, m, shape, tol);
end

%-----------------------------------------------------------------------

function y = doconv(x, m, shape, tol)
% Carry out convolution
[mx, nx] = size(x);
[mm, nm] = size(m);

% If the mask is bigger than the input, or it is 1-D already,
% just let CONV2 handle it.
if mm > mx || nm > nx || mm == 1 || nm == 1
    y = conv2(x, m, shape);
else
    % Get svd of mask
    if mm < nm; m = m'; end        % svd(..,0) wants m > n
    [u,s,v] = svd(m, 0);
```

```
    s = diag(s);
    rank = trank(m, s, tol);
    if rank*(mm+nm) < mm*nm         % take advantage of low rank
        if mm < nm;  t = u; u = v; v = t; end  % reverse earlier transpose
        vp = v';
        % For some reason, CONV2(H,C,X) is very slow, so use the normal call
        y = conv2(conv2(x, u(:,1)*s(1), shape), vp(1,:), shape);
        for r = 2:rank
            y = y + conv2(conv2(x, u(:,r)*s(r), shape), vp(r,:), shape);
        end
    else
        if mm < nm; m = m'; end     % reverse earlier transpose
        y = conv2(x, m, shape);
    end
end
end

%-----------------------------------------------------------------------

function r = trank(m, s, tol)
% Approximate rank function - returns rank of matrix that fits given
% matrix to within given relative rms error. Expects original matrix
% and vector of singular values.
if tol < 0 || tol > 1
    error('Tolerance must be in range 0 to 1');
end
if tol == 0             % return estimate of actual rank
    tol = length(m) * max(s) * eps;
    r = sum(s > tol);
else
    ss = s .* s;
    t = (1 - tol) * sum(ss);
    r = 0;
    sm = 0;
    while sm < t
        r = r + 1;
        sm = sm + ss(r);
    end
end
```

```
end

%-----------------------------------------------------------------------

function y = extendarr(x, m, shape)
% Extend x so as to wrap around on both axes, sufficient to allow a
% "valid" convolution with m to return the cyclical convolution.
% We assume mask origin near centre of mask for compatibility with
% "same" option.

[mx, nx] = size(x);
[mm, nm] = size(m);

mo = floor((1+mm)/2); no = floor((1+nm)/2);  % reflected mask origin
ml = mo-1;            nl = no-1;             % mask left/above origin
mr = mm-mo;           nr = nm-no;            % mask right/below origin

if strcmp(shape, 'wrap')
    y = exindex(x, 1-ml:mx+mr, 1-nl:nx+nr, 'circular');
else
    y = exindex(x, 1-ml:mx+mr, 1-nl:nx+nr, 'symmetric');
end

end
```

```
function [CM] = InnerCMc(M,T,experiment)
% InnerCMc returns the confusion matrix for the placenta pixels only for
% the given ground truth, mask, and experiment. That is, only the pixels
% inside of the placenta are counted to construct the confusion matrix.
% The confusion matrix is in the following form:
%
%              True ID
%             T      F
%Labeled  T   TP     FP
%  ID     F   FN     TN
%
%INPUT:     M           - A mask of the placenta.
%           truth       - The ground truth of the vessel location.
```

```
%           experiment  — The algorithm's output.
%OUTPUT:    CM          — The confusion matrix.

% Loading algorithm's output and putting it in the right form
E = double(imread(experiment));
% E = BaW(E);

% Findind the location of the inner pixels
[¬,I] = BoundInnerc(M);

% Setting the stage
[m,n] = size(T);
CM = zeros(2,2);

% Getting the confusion matrix
for i=1:m
    for j =1:n
        % Check point is inside the placenta
        if I(i,j)==1
            % Checking what kind of pixel E(i,j) is and putting it in the
            % right place of the confusion matrix
            [x,y] = checkP(T(i,j),E(i,j));
            CM(x,y) = CM(x,y)+1;
        end
    end
end
```

```
function [B,M] = BoundInnerc(mask)
% BoundInner returns the boundary given a mask of the placenta and a matrix
% with the pixels inside the boundary marked.

%INPUT:     % mask — A mask of the placenta.
%OUTPUT:    % B    — A (0,1)—matrix with ones only in the boundary.
            % M    — A (0,1)—matrix with ones only in the interior of the
%                    boundary.

%EXAMPLE:
% mask =[                  B = [                    M=[
```

```
%       0  0  1 1  0  0          0 0 1 1 0 0          0 0 0 0 0 0
%       0  1  1 1  1  0          0 1 0 0 1 0          0 0 1 1 0 0
%       0  1  1 1  1  0          0 1 0 0 1 0          0 0 1 1 0 0
%       0  0  1 1  0  0]         0 0 1 1 0 0]         0 0 0 0 0 0]


% Get the boundary given a mask of the placenta (This part of the code was
% kindly provided by Amy Mulgrew.)
se = strel('disk',5); %set to 1 for 1pixeled boundary
Me = imerode(mask,se);
B = mask-Me;
B = double(B);
B = BaW(B);

% Get the inside
M = mask - B;
```

```
function [x,y]= checkP(t,e)

%checkP returns the position in the confusion matrix that a pixel belongs
%given the truth and experiment.

% INPUT:  t - If the pixel is a vessel pixel, t=1. Else t=0.
%         e - If the pixel was identified as a vessel pixel, e=1. Else e=0.
% OUTPUT: x - The row position where the pixel belogs.
%         y - The column position where the pixel belogs.

% Example: The pixel was found to be a "false negative," then x=2, y=1.

% CONFUSION MATRIX:
%
%           Ground Truth
%           T      F
%Test   T   TP     FP
%       F   FN     TN

%True Positive
if t == 1 && e==1
```

```
    x = 1;
    y = 1;
%False Negative
elseif t==1 && e==0
    x = 2;
    y = 1;
%True Negative
elseif t==0 && e == 0
    x = 2;
    y = 2;
%False Positive
else
    x = 1;
    y = 2;
end
```

```
function [B] = BaW(A)
% BaW returns a (0,1)-matrix by mapping the positive entries to 1 and
% everything else to 0.

% INPUT:  A - Any matrix
% OUTPUT: B - A (0,1)-matrix

% Example:
% A = [-5.0  1.0   0.2;         B = [0 1 1;
%       3.9  0.5   2.0;              1 1 1;
%       1.0  4.1  -1.0]              1 1 0]

[m,n] = size(A);
B = zeros(m,n);
for i=1:m
    for j=1:n
        if A(i,j) > 0
            B(i,j) = 1;
        else
            B(i,j) = 0;
        end
    end
```

```
end
B = double(B);
```

References

[1] 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 Conference*, 7962, 2011.

[2] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, and Henrik Nielsen. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, 16(5):412–424, 2000.

[3] J.-M. Chang, A. Mulgrew, and C. Salafia. Characterizing placental surface shape with a high-dimensional shape descriptor. *Applied Mathematics*, 3(9):954–968, 2012.

[4] M. N. Do and M. Vetterli. The finite ridgelet transform for image representation. *IEEE Trans. Image Process.*, 12(1):16–28, 2003.

[5] J.M. Fadili and J.-L. Starck. Curvelets and rigdelets. *Encyplopedia of Complexity and Systems Science*, 3:1718–1738, 2009.

[6] T. Fawcett. Roc graphs: Notes and practical considerations for researchers. *Machine Learning*, 31:1–38, 2004.

[7] A. Frangi, W. Niessen, K. Vincken, and M. Viergever. Multiscale vessel enhancement filtering. In *LNCS*, volume 1496, pages 130–137, Germany, 1998. Springer-Verlag.

[8] R. Jain, R. Kasturi, and B. G. Schunck. *Machine Vision*, chapter 4, pages 112–139. McGraw-Hill, Inc., New York, NY, USA, 1995.

[9] S. Pathak, F. Jessop, L. Hook, N. Sebire, and C. Lees. Placental weight, digitally derived placental dimensions at term and their relationship to birth weight. *Journal Of Maternal-Fetal and Neonatal Medicine*, 23(10):1176–1182, 2010.

[10] C. Salafia, D. Misra, M. Yampolsky, A. Charles, and R. Miller. Allometric metabolic scaling and fetal and placental weght. *Placenta*, 30:355–360, 2009.

[11] M. Strufaldi, E. Silva, M. Franco, and R. Puccini. Blood pressure levels in childhood: Probing the relative importance of birth weight and current size. *European Journal Of Pediatrics*, 168(5):619–624, 2009.