Vessel Detection using Multi-level Thresholding and Pruning in Retinal Images

Logan Bissonnette

***Abstract*—In this paper, a simple verification based solution for vessel detection and segmentation using multithresholding is reintroduced an analyzed based on computational performance as well as the specificity and sensitivity of the resulting segmentation. Results are compared to another implementation of the algorithm.**

I. Introduction

RETINAL blood vessel segmentations are important for the automatic screening and detection of diseases such as glaucoma and diabetic retinopathy. Segmentation by hand is a time consuming process which requires a trained professional. An automated process provides an inexpensive solution which allows the doctor or technician resume other work activities instead of spending time segmenting images. [1]

Global thresholds are effectively used in images where the objects that need to be detected have a consistant colour, or are a colour with a strong contrast to the background of the image. Global thresholds do not work well in images which contain objects of varying colour or intensity, contain objects which show a weak contrast with the background, or contain nonuniform illumination; which is often the situation with retinal images. [2]

This paper analyzes an approach first introduced by Jiang and Mojon in [2]. The proposed method takes a single coloured retinal image, thresholds it at various levels, removes anything which is not part of a vessel, then recombines the images to produce a single segmented image.

II. Related Work

This implementation in this paper is based on the algorithm proposed by Jiang and Mojon in [2]. Other multithreshold examples inglude O'Gorman, who uses a method based on the histogram of the image [3]. Many other papers look into the problem of blood vessel segmentation in the retina, including Selvathi and Lalitha who propose using the Gabor Wavelet responses at different scales [4], Cheng Lee and Che Ku who propose using a line detector to classify pixels [5].

Logan Bissonnette is with the Department of Software Engineering, Victoria, Canada, currently working towards an Undergraduate Degree
E-mail: lbiss@engr.uvic.ca

Manuscript received August 1, 2013. This work was supported in part by the Image Sciences Institute, who provided an image database for the project.

III. Approach

A. Image Binarization and Thresholding

The first step of the algorithm is the production of a binarized image, which will later be verified and pruned. Since the red channel is often saturated, and the blue channel often empty, the green channel provides the best contrast, and should be used for the rest of the algorithm [2]. The green channel of the image is extracted, and a global threshold is used to produce a binary image.

B. Pruning and Verification Operations

The pruning and verification operations the critical steps in the algorithm, as they remove all of the non-vessel data for the image. If the pruning operation removes too much data at each step, the resulting image may be missing vessels, depending on the number of iterations used. By contrast, if the pruning operation leaves data for non-vessel objects, the resulting image will contain unwanted noise.

The pruning operation works by uses various verification methods to deduce whether it is likely that an object is part of a vessel. If the object fails verification, then it is removed. Application-dependant verfiers can be used if the algorithm is not being applied to vessel detection. For vessel detection, Jiang proposes four verifiers.

The first two verifiers are based on two variables, d and ϕ, both of which are introduced in [6]. We model the blood vessels as being thin curvilinear bands in our image; these two verifiers used to detect the likelihood that a particular pixel in the image is a pixel in the center of a curvilinear band.

These values are found by calculating for each candidate pixel p (i.e. pixel below our threshold value in the thresholded image). The algorithm is best described using psudo code;

For each p in image
  For each n in the 4-neighborhood of p (N_p)
    Find e_p, the coordinates of the nearest background pixel of p
    Find e_n, the coordinates of the nearest background pixel of n
    Calculate d_n, the euclidean distance from e_p to e_n
    Calculate ϕ_n, the angle produced by connecting pixel p with e_p and e_n
$d = \max_{n \in N_p}(d_n)$
$\phi = \max_{n \in N_p}(\phi_n)$

A visual representation of d and ϕ can be found in Fig. **2.**

A more mathematical description of the operation is as follows; where N_p represents the set of four pixels in the 4-neighbourhood of p, and e_p and e_n represent the closest background pixel to p and n

$$d = \max_{n \in N_p}\left(\|\overline{e_p e_n}\|\right)$$

$$\phi = \max_{n \in N_p}\left(angle\left(\overline{pe_p}, \overline{pe_n}\right)\right)$$

For the purposes of efficiency, the coordinates of e_p and should be computed only once for each binarized image and stored in a table or matrix before any calculations of d or ϕ are done; the values are guarenteed to be used for each pixel below our threshold. A fast way of computing this matrix can be found in [7], other pixel based mothods work as well.

Since vessels are modeled as *thin* curvilinear bands, a maximum value of d can be imposed on the image. This maximum value removes any areas which are too large to be vessels. Since this leaves any pixels which are close to any pixel above the threshold value – including any pixels which are noise (shown in Fig. 3.), ϕ is used to remove any pixels which are not crentral to a band. Since pixels close to the center of a curvilinear band usually have large values for ϕ, a minimum value is placed on ϕ, and any pixels below that minimum are pruned.

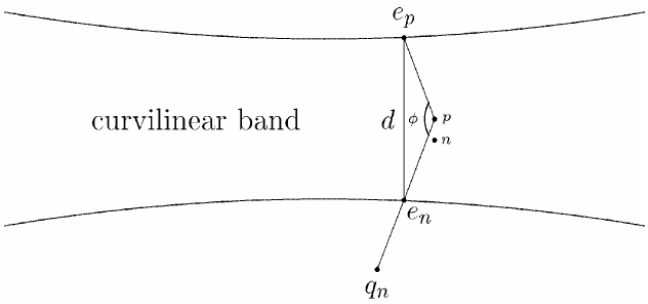

Fig. 2. Values for testing likelihood of pixel being part of a curvilinear band. The value of d is used to prune off any vessel candidates which are part of an object too thick to be a vessel, while ϕ is used to prune off anything which is not a center pixel. Image from **[2]**

Pruning just based on these values will sometimes result in thin, curvilinear structures appearing which show a weak contrast to the surrounding area [2], i.e. a background area with a curvilinear structure which was coincidentally picked up by a threshold. To prune these areas, a third verification metric – contrast – is used.

For candidate pixel p, each pixel n in the 4-neighbourhood N_p of p, and e_n, the pixel nearest background pixel to n, contrast is defined as $c_n = \frac{intensity(e_n)}{intensity(p)}$, and c is defined as the maximum value of c_n for pixel p. A minimum limit on c is enforced to remove any pixels with an insignificant contrast compared to the surrounding pixels. This verification operation is the least significant of all of the verification operations defined in the article, in terms of the number of false positives it removes. Its intended use is as a form of noise reduction [2].

Fig. 3. Example of image which has been verified by only d. The image has many square or rectangular patches which surround a single pixel above the threshold.

The final verification step is a size verification, which results in a significant reduction in noise. With the pruning operations defined above, the resulting image is usually covered in noise as the result of having many small structures pass through the pruning operations – up to this point the only requirements for a pixel are to be part of a thin

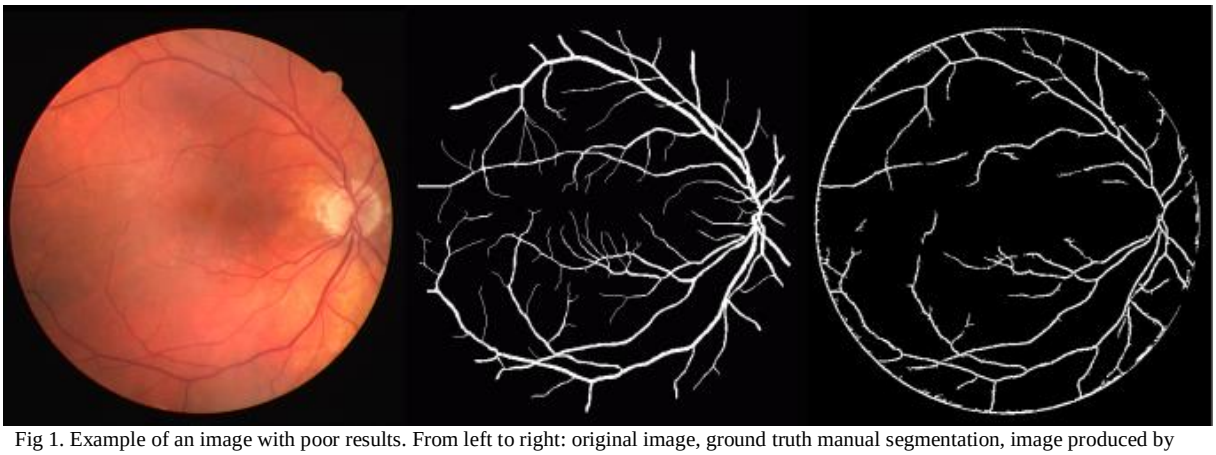

Fig 1. Example of an image with poor results. From left to right: original image, ground truth manual segmentation, image produced by algorithm. Many of the vessels in the original image show a poor contrast to the surrounding pixels (One level of intensity on the green channel) and either do not show up in the binarized (Thresholded) images, or are pruned by contrast pruning.

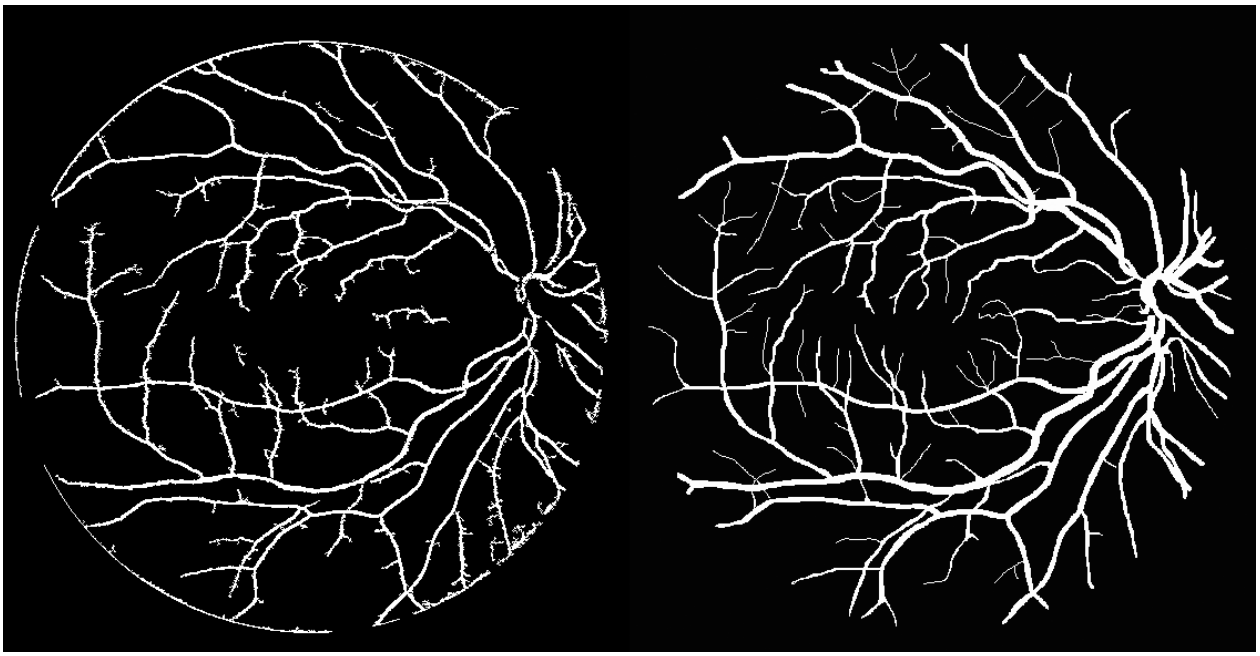

Fig. 4. An example of a segmentation with good results. Left: Result of the algorithm. Right: Ground truth segmentation. Vessels appear thinner than in the ground truth image, and many of the smaller vessels which had low contrast in the original image are missing.

structure, with another nearby pixel having a closest background pixel be a significant angle away. Depending on the algorithm used for finding the nearest background pixel, objects as small as two pixels may satisfy this result, despite not being part of a large curvilinear band.

Size verification removes any objects of size smaller than a given threshold. Fig. **5** shows an example image with and without size verification. Connected component labeling is on an image which has had all other verification steps applied. Any components of size below the threshold are removed. Jiang did not specify whether to use a 4-labelling or an 8-labelling, but either labeling can be effective, as long as the size threshold is adjusted accordingly.

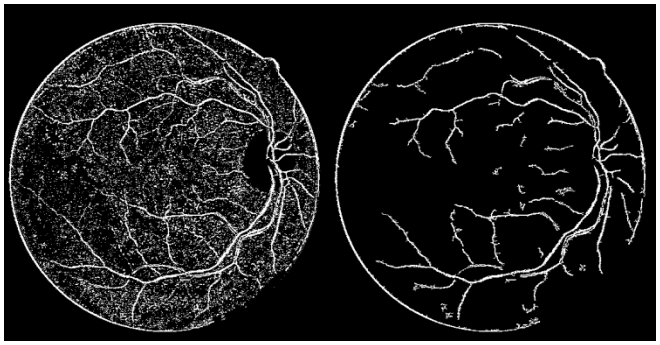

Fig. 5. An example image with and without size verification. Left: No verification, right: with verification. The images used were processed with 5 iterations, as an example. With additional iterations, more noise and vessels would be present.

IV. Experimental Results & Evaluation

The algorithm had troubles with images which contained vessels with low contrast to the background, as was the case in some of the images processed. Fig 1 contains an example of such an image. Some methods, including contrast enhancement by histogram, were attempted to try to resolve this issue, but the image results from preprocessed images were significantly worse, as it negatively affected contrast verification and noise removal.

There is a strict tradeoff between computation time, false positives, and false negatives in this algorithm. Increasing the number of iterations used increasing the sensitivity of the program, and allows stricter verification methods to be used, resulting in more specificity. If specificity is less important, less strict verification methods can be used, resulting in a more sensitive system.

At the extreme end, if the system has verifiers which allow too much through, the result will not be a segmented image, but instead will produce an image where the entire retina is detected as a single vessel.

For the images produced for this paper, the following parameters were used as threshold values:

$d_{threshold} = 9pixels$
$\phi_{threshold} = 135°$
$PSize_{threshold} = 44pixels$
$c_{threshold} = 1.05$

Jiang suggests a static starting and ending threshold in his paper; a manually assisted algorithm was used for determining a start and end threshold. This usually resulted in a starting threshold of approximately 0.2 and ending threshold of approximately 0.6 (Intensity values of 51 and 153 respectively), but results varied between images. In this implementation, images were binarized at a threshold increasing by 2 intensity values at each step – usually this meant approximately 45 iterations.

The implementation of the algorithm used for testing in this paper produced images of a similar quality to those produced by [2], with a few minor differences. Due to the image database being used, the images produced tend to

have a circle around the edge of the retina; this is because there was an area of slightly higher intensity around the edge of the original image, and this area was not masked by the masks provided.

Ten images were used for performance statistics. These images were taken from the Drive Database of retinal images provided by the Image Sciences Institute. Using the thresholds above, the application had a sensitivity of up to 0.606, and a specificity of 0.980 for the same image (shown in Fig. 3). The results had very little noise on them, resulting in a good specificity.

On average, the sensitivity was 0.51 and specificity 0.98. This is comparable to the implementation by [2] which had an average specificity of approximately 0.98 at 0.6 sensitivity (estimated from graphs shown). There are two main factors for the difference in results; firstly the ring around the retina, which was the main cause for false positives (more than half of the false positives for the image); secondly the parameters chosen were stricter than those chosen by Jiang, resulting in less sensitive, more specific images.

A sample of results can be seen in Fig. 6, and show that the results for the segmentations line up with results for the ground truth, and contain very few false positives.

On a desktop computer running windows with a 3.6GHz processor Quad Core Processor, the application took an average of 2 minutes per iteration; early iterations completed quickly taking as short as 10 seconds, while later iterations took up to 10 minutes. A typical image can be completed in 90 minutes. The performance scales approximately linearly with the number of iterations used.

This performance is much worse than what was experienced by Jiang, who found that 30 iterations could be completed in 36 seconds on a server running Linux and a Pentium III at 600MHz.

Performance was largely hindered by MATLAB's performance; using a single core, the application often used less than 8% of the CPU's total power, while a parallel solution running on all four cores ran at 40% after communications overhead. It is expected that there would be more than a 100% performance gain on both the single core and parallel versions if the application were able to make better use of the system hardware.

A lot of work was put into optimizing performance; functional inlining resulted in a 4x speedup in performance, and parallelizing the code resulted in an approximate 2x speedup. The 4x speedup from inlining comes directly from MATLAB's inefficient optimizer.

In contrast to what was found by [2], the most computationally heavy portion of the algorithm was calculating the distance map, rather than the pruning operation. A less efficient version of the distance map

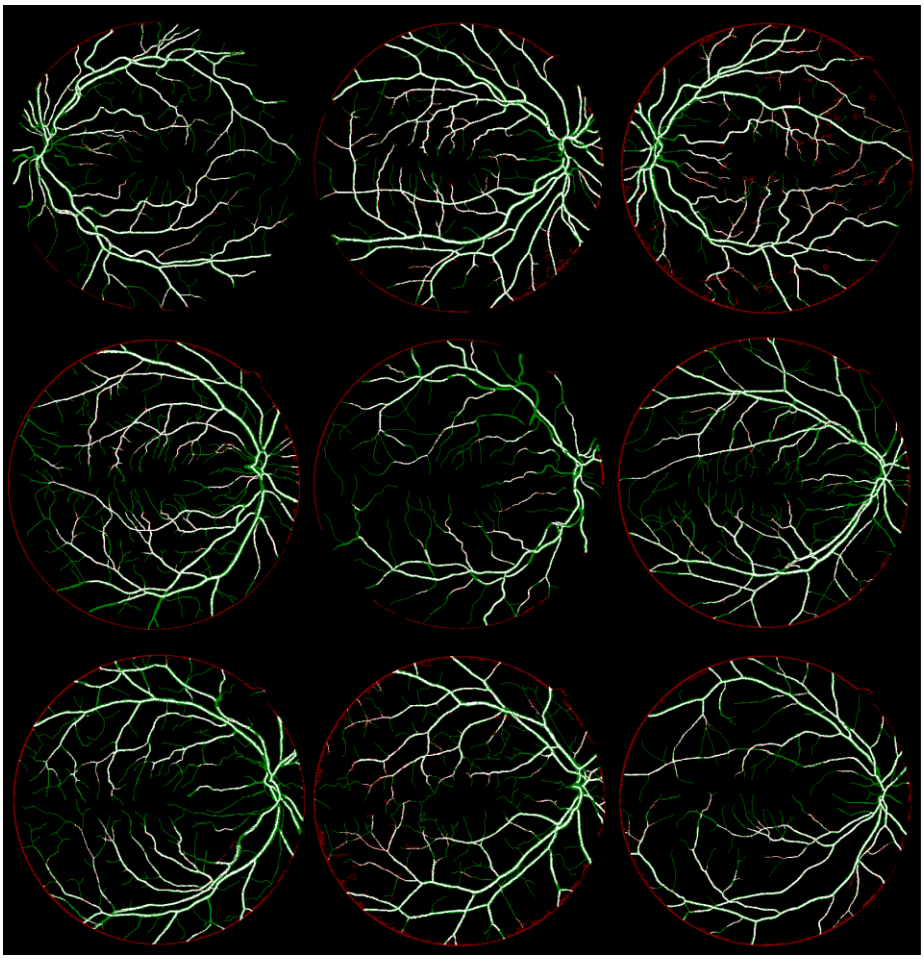

Fig. 6. Sample results for nine different segmented images. Minor vessels are commonly missed by the algorithm. White vessels are true positives which had an overlap with the manual segmentation, red vessels are false positives, and green vessels are false negatives. Because the algorithm produces a skeleton, the segmentation displays the vessels as being smaller than they actually are resulting in a green sheath in these images. Most of the false positives are around the edge of the retina.

algorithm was used in this implementation. The implementation used for testing for this paper used an algorithm whose performance scaled with an upperbound of $O(n^3)$ for an image with n pixels, while the diffusion based algorithm used by Jiang in [2] scaled according to an upperbound of $O(kn^2) = O(n^2)$. This diffusion/wave style algorithm is introduced by Leymarie and Levine in [7]. Although this algorithm would be faster, it would not increase the performance up to the standard set by Jiang, as the verification step is still a significant part of the computation time.

V. Conclusions

In this paper, a method of segmenting blood vessels in the retina by thresholding at multiple levels and pruning off false positives is analyzed. The method is quite intuitive, and leaves much room for adaptation and improvements by introducing or modifying the verification process. The algorithm is simple to implement, yet powerful enough to have uses beyond blood vessel segmentation in the retina.

VI. References

[1] P. Echevarria, T. Miller and J. O'meara, "Blood Vessel Segmentation in Retinal Images," Stanford University, Stanford, 2004.

[2] X. Jiyang and D. Mojon, "Adaptive Local Thresholding by Verification-Based Multithreshold Probing with Application to Vessel Detection in Retinal Imagse," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 1, pp. 131-137, 2003.

[3] L. O'Gorman, "Binarization and Multithresholding of Document Images," *CVGIP: Graphical Models and Image Processing*, vol. 56, no. 6, pp. 494-506, 1994.

[4] D. Selvathi and P. Lalitha Vaishnavi, "Gabor wavelet based blood vessel segmentation in retinal images using kernel classifiers," *Signal Processing, Communication, Computing and Networking Technologies (ICSCCN)*, pp. 830-835, 2011.

[5] C.-C. Lee and S.-C. Ku, "Chien-Cheng Lee; Shih-Che Ku," in *Signal Processing, Communication, Computing and Networking Technologies (ICSCCN)*, Taiwan, 2012.

[6] G. Malandain and S. Fernhdez-Vidal, "Euclidean skeletons," *Image and Vision Computing (ELSEVIER)*, vol. 16, pp. 317-327, 1998.

[7] F. Leymarie and M. Levine, "Fast Raster Scan Distance Propogation on the Discrete Rectangular Lattice," *CVGIP: Image Understanding*, vol. 55, no. 1, pp. 84-94, 1992.