Automatic Retinal Vessel Segmentation

Introduction

Scope

There have been much research done on retinal vessel segmentation that has yield impressive accuracies on datasets such as STARE, DRIVE and CHASE[1]. For example, DeepVessel has achieved between 94% and 95% accuracies on the above datasets[2].

As a course assignment, we won't aim to further these studies. Instead, we'll study and experiment through referencing these modern imaging techniques to build a vessel segmentation algorithm from scratch. Afterwards, we'll discuss how it works and what we've learnt through this project.

Programming Language

Here, we chose **Python** for its continual, open-source community support and familiarity to the team. We've picked out 2 important image processing libraries, OpenCV2 & Scikit-Image.

Data

We have

- 32 test images
- 1 train image (with ground truth)

Overfitting Caveat

As there is only 1 train image provided, we're prone to customise our pipeline to maximise train accuracy of this 1 instance. By doing so, we fail to consider the pipeline generalisability to unseen data. This is known as overfitting: where an algorithm is too biassed to the train data, such that it fails to consider variation in unseen data.

To prevent this, we avoid:

- models requiring huge training data to converge, such as Deep Neural Networks where it's attempted in some papers using larger datasets according to a review[3].
- assumptions that are too specific to the train data, such as the colour of the background & size of the retina within the image.

---

[1] https://www.medicmind.tech/retinal-image-databases

[2] Fu, H., Xu, Y., Lin, S., Kee Wong, D.W., Liu, J. (2016). DeepVessel: Retinal Vessel Segmentation via Deep Learning and Conditional Random Field. In: Ourselin, S., Joskowicz, L., Sabuncu, M., Unal, G., Wells, W. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016. MICCAI 2016. Lecture Notes in Computer Science(), vol 9901. Springer, Cham. https://doi.org/10.1007/978-3-319-46723-8_16

[3] Panda, N. R., & Sahoo, A. K. (2022). A Detailed Systematic Review on Retinal Image Segmentation Methods. *Journal of digital imaging*, 10.1007/s10278-022-00640-9. Advance online publication. https://doi.org/10.1007/s10278-022-00640-9

Exploratory Analysis

Analysis of Retina Images

The aim of the project is to segment retina blood vessels from a retinal image. **Figure 1** below illustrates the main regions of interest within the image.

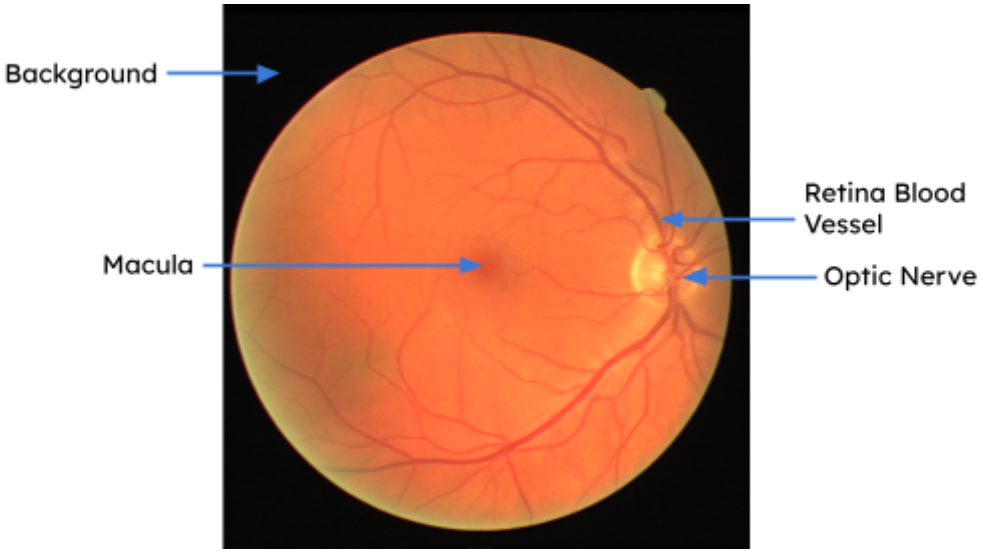


Figure 1: Train Image

As observed, the target retina blood vessel isn't trivial to segment due to various reasons:

1) Thinner vessels are difficult to differentiate from noise
2) The **Optic Nerve** creates false edges, which leads to false positive vessels
3) The **Macula** dims the image, which makes it difficult to yield thin vessels in the centroid
4) The retina is not rectangular, a step to cut out the circular retina foreground is necessary

Analysis of Ground Truth

Figure 2: Train Image (Left), Train Ground Truth (Right)

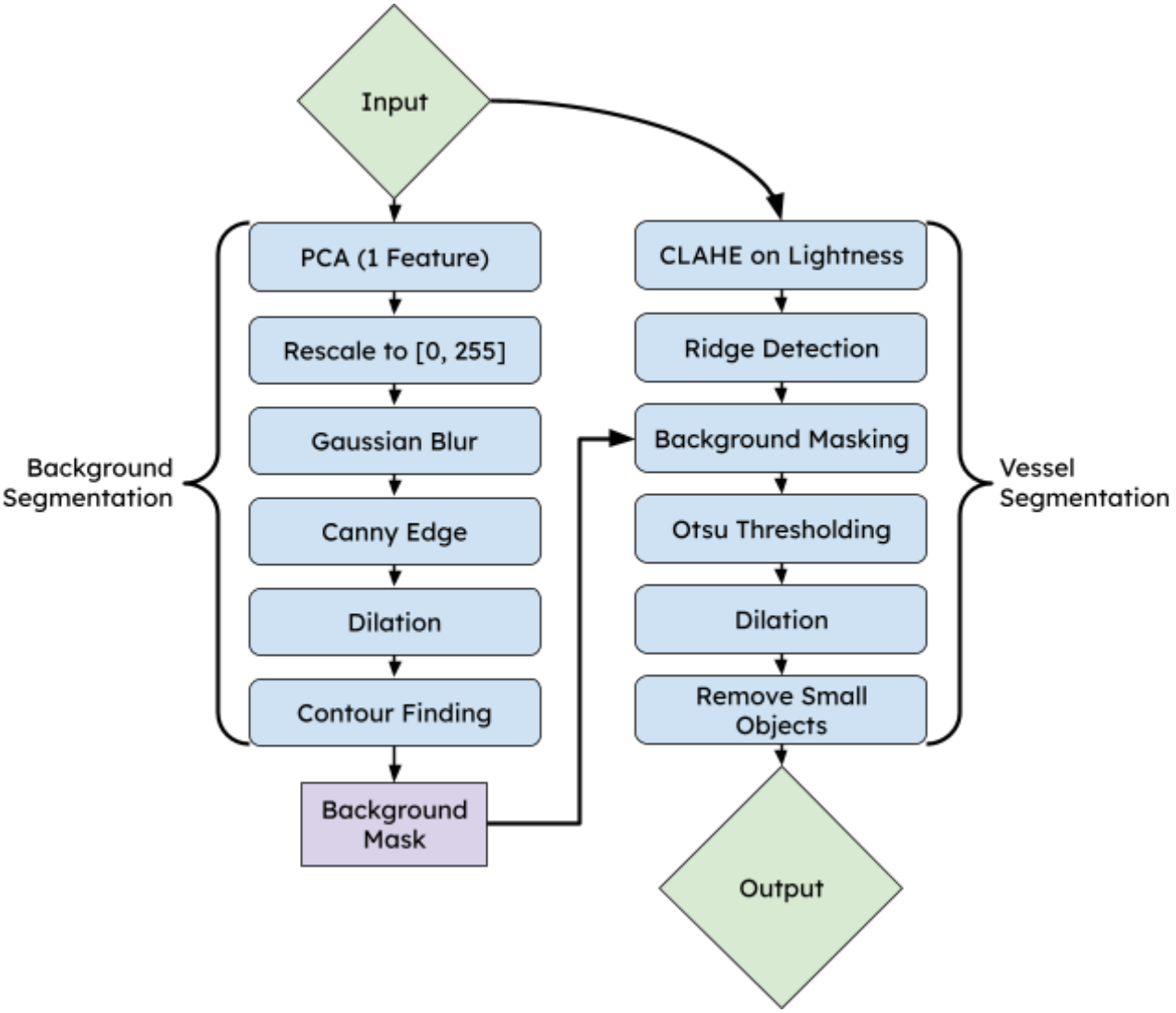
The ground truth, annotated by an expert, maps out all of the vessels in a boolean image.
We observe high precision of thinner vessels, which our algorithm may not achieve, though erring to the side of caution of overfitting may be desirable for generalisation.

Methodology

There's two parts in segmenting the retinal vessels:

1) Background Segmentation
2) Vessel Segmentation

We illustrate the workflow below.



Background Segmentation

Analysis

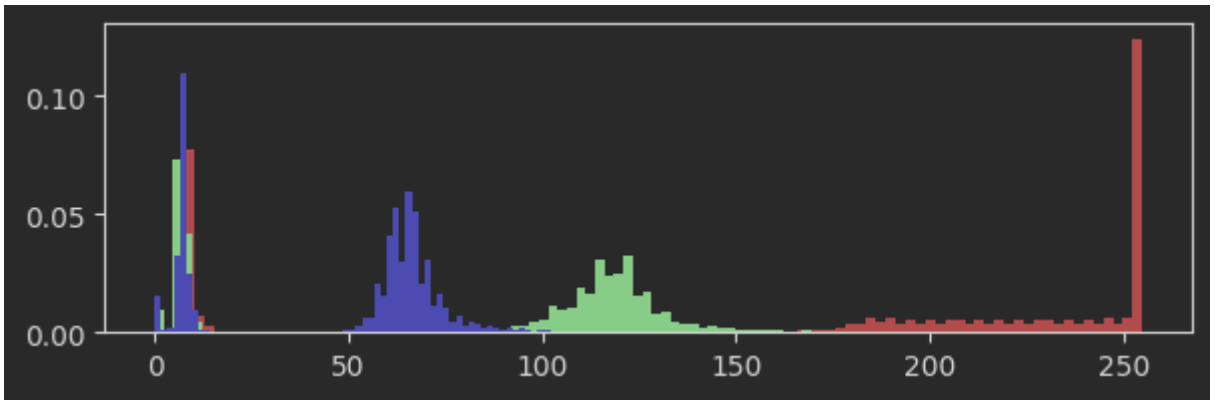


Figure 3: Image RGB Histogram Spread

We observe that the background, the leftmost peaks, is clearly separated from the retina's peaks

After experimentation, we opted for a morphological technique for background segmentation instead of:

- Thresholding
    - Because it's not generalisable to images with white backgrounds.
    - Furthermore, if the background were white, we'll remove retinal information in the red channel.
- K-Means
    - Because K-means does not respect spatial relationships. Thus, if the background was white in our example, we'll remove info from the red channel.

Canny Edge Detection

Canny Edge Detection[4] is an algorithm to detect edges within an image.

Briefly described:

- Canny calculates the Sobel derivatives in the X & Y direction.
    - These 2 derivatives are combined in 2D space to give a magnitude & angle for each pixel
- Non-Maximal Suppression is then used to remove non-local maximas along the angle of their derivative mentioned earlier.
- Hysteresis Thresholding is used to connect sure-edges with edges slightly below threshold to avoid broken edges.

One caveat of Canny Edge is the variable Hysteresis Thresholds, where optimum values vary among images. Fortunately, we can automatically determine them through Otsu's Threshold as long as the

[4] Canny, John. "A computational approach to edge detection." *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986): 679-698.

image has 2 histogram extremums.[5] Using the lower bound as half of Otsu's Threshold and upper as the Otsu's Threshold.
After applying Canny Edge detection, we find the maximum contour (by area) to finally segment the image as shown below

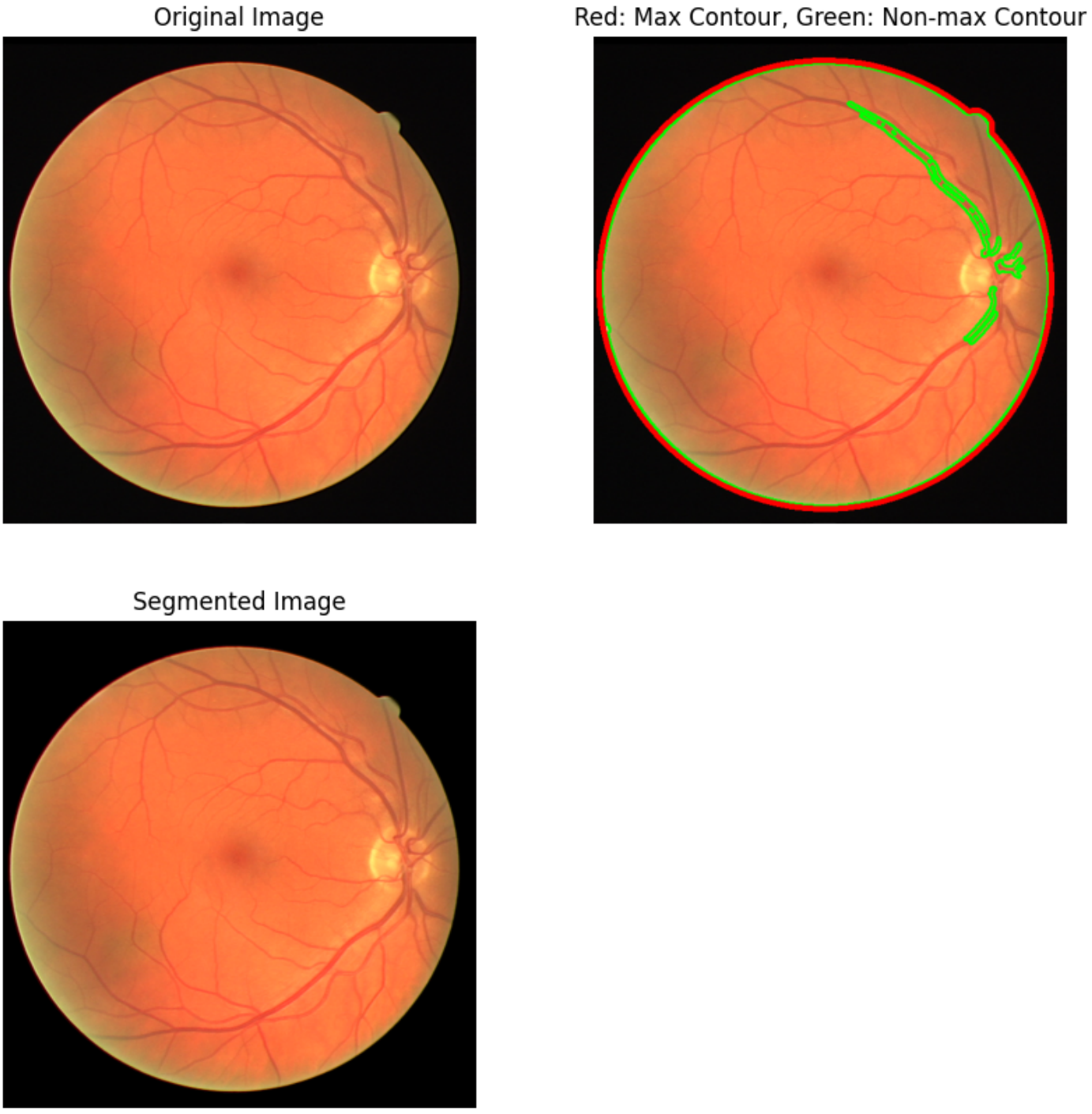


Figure 4: Canny Edge Segmented Image

Pre & Post-processing for Canny Edge

However, Canny Edge, as is, is not robust, thus failing to produce a clear segmentation above. Some processing was needed before we applied Canny Edge.

[5] Fang, Mei, Guangxue Yue, and Qingcang Yu. "The study on an application of otsu method in canny operator." *Proceedings. The 2009 International Symposium on Information Processing (ISIP 2009)*. Academy Publisher, 2009.

- Principal Component Analysis (PCA) was used for dimensionality reduction of the channels from 3 (RGB) to 1. PCA is a deterministic algorithm to yield new variables that maximises variance, which increases the foreground-background separation, thus higher contrast.
    - Canny Edge only works with grayscale, thus dimensionality reduction is necessary.
- The output of PCA is rescaled linearly to [0, 256) for Canny Edge
- Gaussian Blur is used to reduce false high-frequency edges created by noise

Note that Canny Edge returns a boolean image of the edges found.

Post-processing:

- After Canny Edge detection, dilation is used to close broken edges
- Contours are then found through border following[6]
- The largest contour (by area) is then used for segmentation.

[6] Suzuki, Satoshi. "Topological structural analysis of digitized binary images by border following." *Computer vision, graphics, and image processing* 30.1 (1985): 32-46.

Vessel Segmentation

After segmentation of the background, we preprocess the vessels for thresholding.

Contrast Limited Adaptive Histogram Equalisation (CLAHE)

Adaptive Histogram Equalisation, which equalises histograms by each pixel's neighbourhood[7]. However, this algorithm is prone to amplify noise on near-constant regions. CLAHE fixes this by limiting how much contrast can be amplified through a clipping value.

CLAHE only works on 1 dimension of the image, thus, we apply it onto the Lightness dimension in the L*a*b* dimension.

Using CLAHE on our Lightness dimension, we observe that the vessels are amplified.

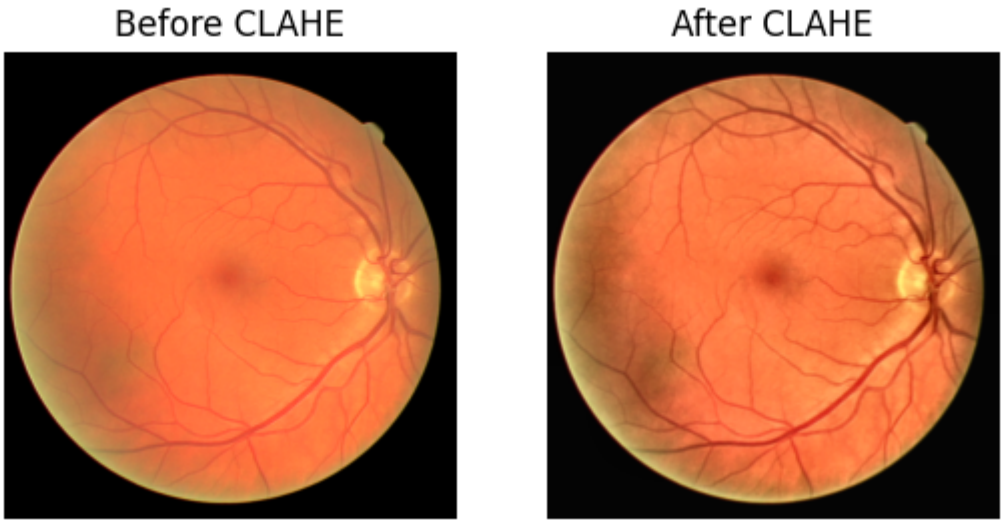
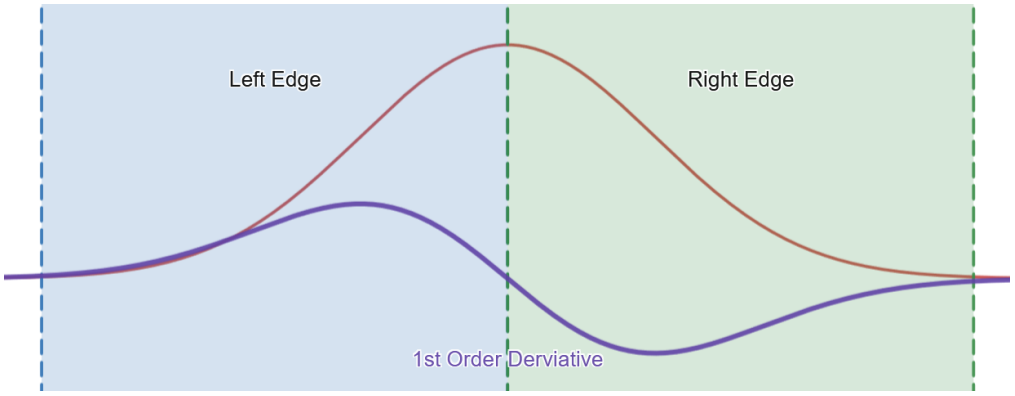


Figure 5: Before & After CLAHE

Ridge Detection

While an edge detector detects edges with a 1st order derivative, the ridge detector uses a 2nd order derivative. Much like how a 1st derivative measures the slope of change, 2nd derivatives measure how fast the slope changes.

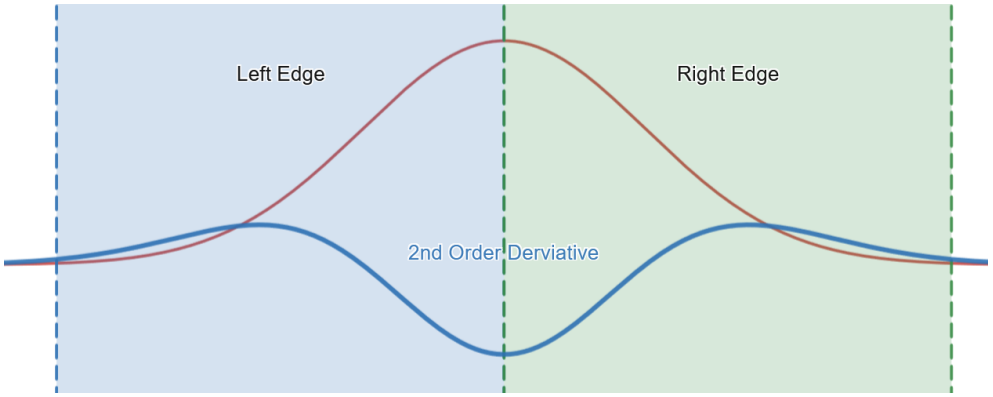
We'll explore the motivation behind using a 2nd order derivative in a 1D cross section:



Taking a cross section of any vein, we observe this level function above.

[7] Hummel, Robert. "Image enhancement by histogram transformation." *Unknown* (1975).

Recall that Canny uses a Sobel Derivative, functionally similar to a 1st order derivative. We yield 1 peak and 1 valley as minimas/maximas. As a result, we yield 2 edges for each vein cross section, which is suboptimal.



Using a 2nd Order Derivative, we yield a clear valley. Though, as a consequence, there may be unwanted minimas/maximas.

Going into technicalities, we firstly calculate the Hessian Matrix: a matrix for 2nd Order derivatives. Then, through matrix decomposition, reduce dimensionality of the matrix into 2 eigenvalues: notably, 1 detecting the strength of each ridge, much like the 2nd order derivative above.

We use OpenCV's RidgeDetector[8], which uses the "main negative eigenvalue of the Hessian matrix."[9]

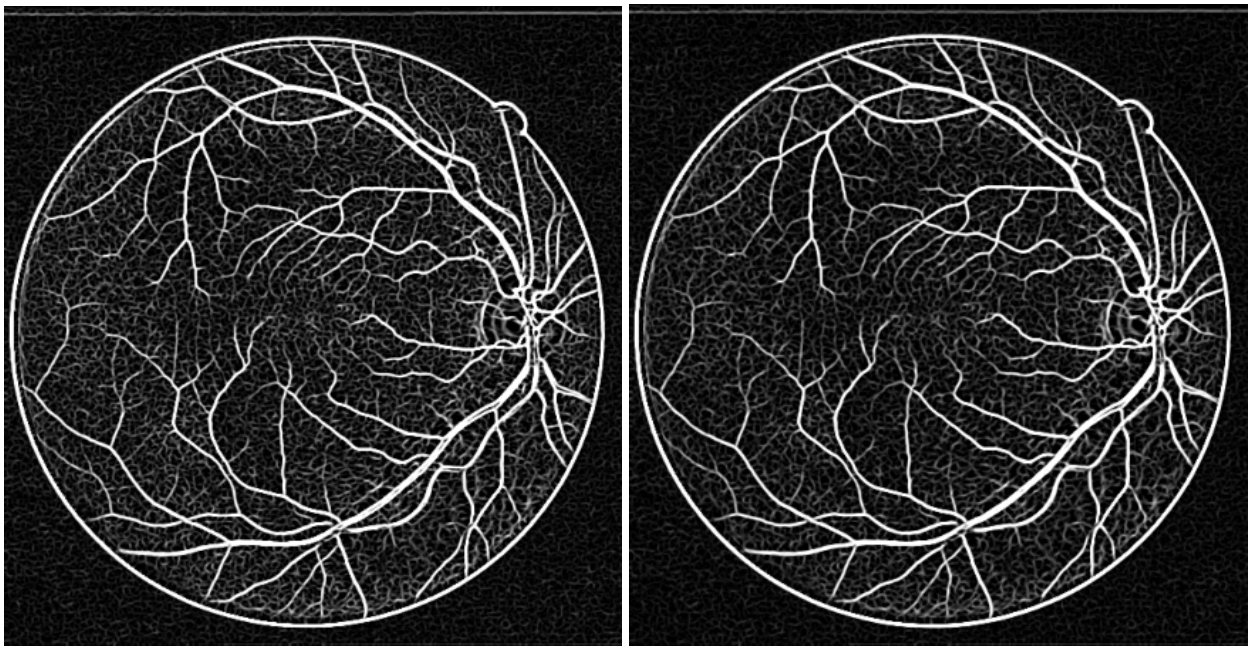


Figure 6: Ridge Filter Result. Without Gaussian Blur (Left), With Pre-Gaussian Blur (Right)

As mentioned, it's prone to unwanted minimas/maximas, thus a Gaussian Blur was used **before** the ridge detection. Its effects are shown in Figure 6.

Notice the outer ring is detected as a ridge, we'll use our Background Segmentation mask to fix that next.

[8] https://docs.opencv.org/3.4/d4/d36/classcv_1_1ximgproc_1_1RidgeDetectionFilter.html
[9] https://reference.wolfram.com/language/ref/RidgeFilter.html

Background Masking

At this stage, we remove the background by multiplying the boolean mask with the image.

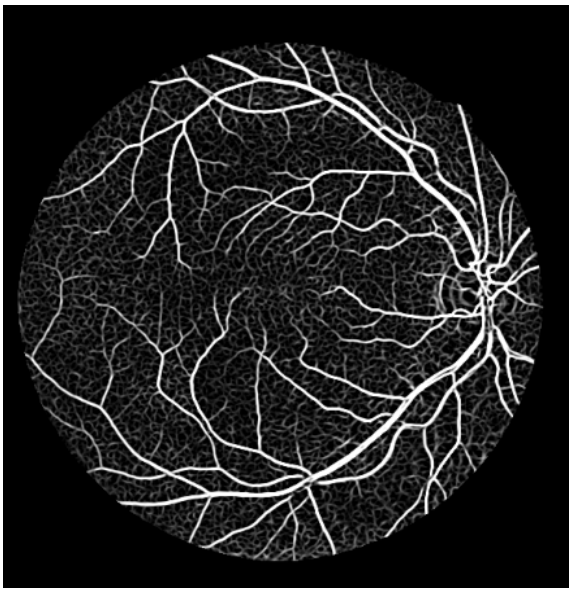


Figure 7: Masking Result

Note that ridges on the edges will still be present if we cropped the vessel foreground early, thus we only did it **after ridge detection**.

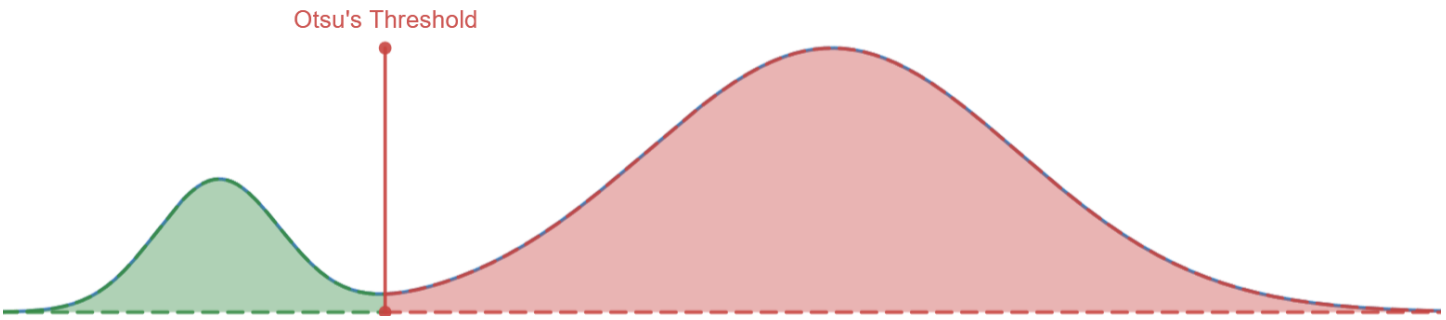
Thresholding

We'll use Otsu's Threshold here to automatically threshold the vessels.

Recall that we used this threshold in Canny Edge, we'll further expand on how it works here.

Otsu's Method (Threshold)[10] works by minimising total per-class weighted variance through iteration. Intuitively, it's finding a threshold where values above it and below it have the least sum class-weighted variance: the variance contribution is proportional to the size of the class.

$$\text{Weighted Variance} = \frac{N}{N+M}\sigma_n^2 + \frac{M}{N+M}\sigma_m^2$$
$$N, M : \text{Class Instances}$$



The weight is the number of instances of each class, thus it'll be proportionally scaled to scenarios of class imbalance like the one above.

[10] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," in IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62-66, Jan. 1979, doi: 10.1109/TSMC.1979.4310076.

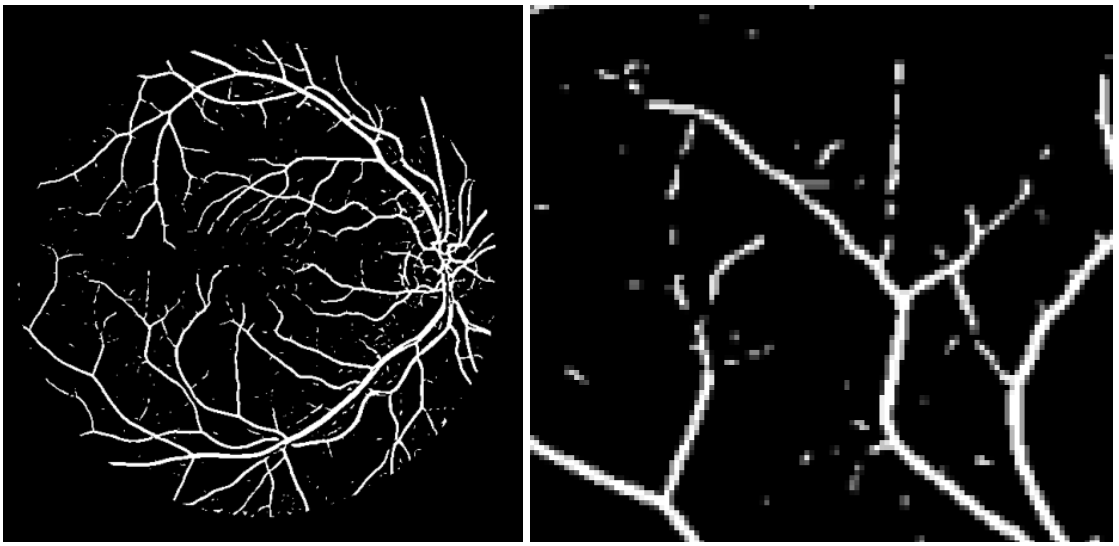


Figure 7: Thresholding Result (Left). Zoomed in on Result (Right)

One may consider removing small disconnected blobs at this stage, however notice on the right of Figure 7, that they may indeed be part of the veins. Thus, we need to connect them somehow.

Dilation

Through dilation, we hope to connect these branches. We used a spherical kernel, where the likelihood of reconnecting broken branches is equal on all angles.

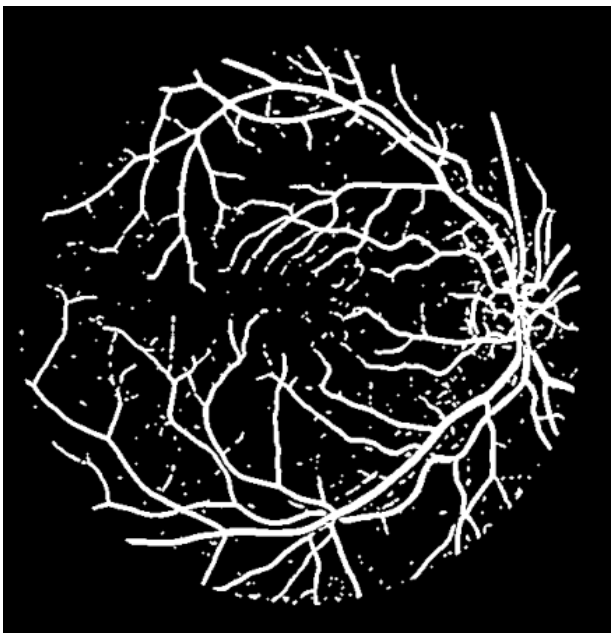


Figure 8: Dilation Result

We found that despite dilation inflating thin veins causing inaccuracy, it still benefited the accuracy metric. We further discuss this problem in the section: **Discussion (Overdilation in Thinner Vessels)**.

Remove Small Objects

Finally, we assume that all small unconnected objects are just noise. We achieve this by removing connected objects of less than 256 pixels.

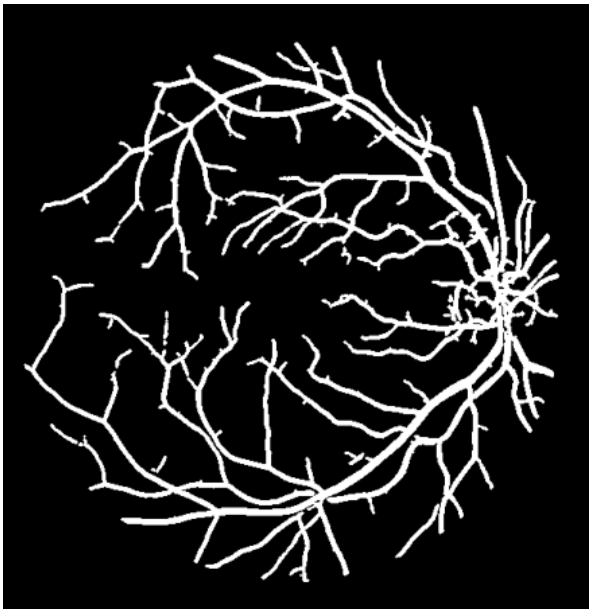


Figure 9: Remove Small Objects Result

Parameter Grid Search

In total, we have 12 hyperparameters to search through.

We took an initial guess of each hyperparameter, then searched for the best neighbourhood hyperparameter. For example, if we had 3 choices for each hyperparameter, we have 3^12 = 531K iterations of the algorithm to find the best combination.

Due to the massive search space, we limited the scope of the search.

Evaluation

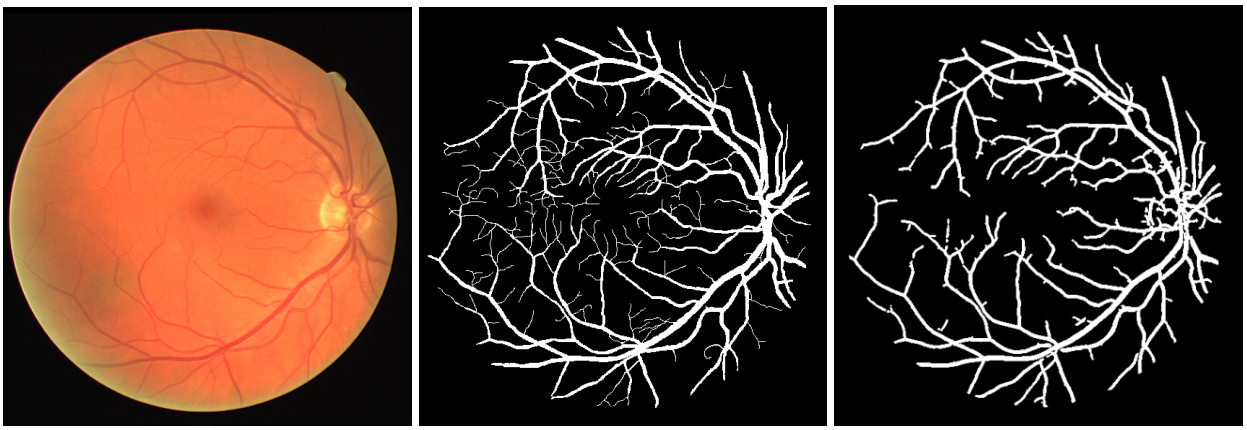


Figure 10: Input, Expected Output & Actual Output

Metrics

Having additional vessels mapped (False Positive) and missing vessels (False Negative) are detrimental to decision making by experts. That's why we consider both precision & recall in the metrics through F1 Score.

Recall that F1 Score considers both False Positive & False Negative, thus, either mistake will punish the resulting score.

Results

We managed to yield:

F1 Score	Accuracy	Sensitivity (Recall)
80.06%	95.30%	78.75%

Compared to the results of DeepVessel which has achieved 95% on DRIVE and CHASE, this may be promising. However it's biassed because:

- Much of the background inflates accuracy
- Thinner vessels are weighted less than thicker ones, which may misrepresent its importance
- Only 1 training sample is used without validation data.

Discussion

Loss of Info through Blurring

Before ridge segmentation, we performed a 5x5 blur, it may likely have removed crucial information of thinner vessels. Unfortunately, there is a huge overlap in frequency of noise and thin vessels, thus it is difficult to find a balance between the removal of noise and preservation of detail.

Optic Nerve False Edges

Unfortunately, we aren't able to fully remove false edges of the Optic Nerve.
To recap, it's a bright circle near the edge of the retina. See Figure 1.

Figure 11: Close-up of Optic Nerve segmentation result on Expected (Left) and Actual (Right)

Some assumptions must be made to threshold it:

- It's consistently brighter than the rest of the retina
- It's always present

These assumptions aren't confident, thus we refrained from preprocessing it.

Overdilation in Thinner Vessels

We ran into a problem where vessels are broken after thresholding, thus using dilation to reconnect them. This caused a decrease in accuracy through over-dilation in thinner vessels.

However, we still observed an overall improvement in accuracy, this is because thicker veins were in abundance and were too thin, thus driving more benefit when dilation is applied.

Connecting Broken Thresholded Vessels

A better alternative solution would be to extend the stems on the leaves using a rectangular structuring element with respect to the Local Dominant Orientation. However there doesn't seem to be any plug-and-play solution in any Python package.

Figure 12: Proposed Solution

Yielding Thinner Vessels & Overfitting

While adjusting parameters to yield thinner vessels, we see clear signs of overfitting when we run it against the other 31 test images. Signs such as noise being detected as vessels and vice versa. The approach to objectively improving this will require more training data.

Conclusion

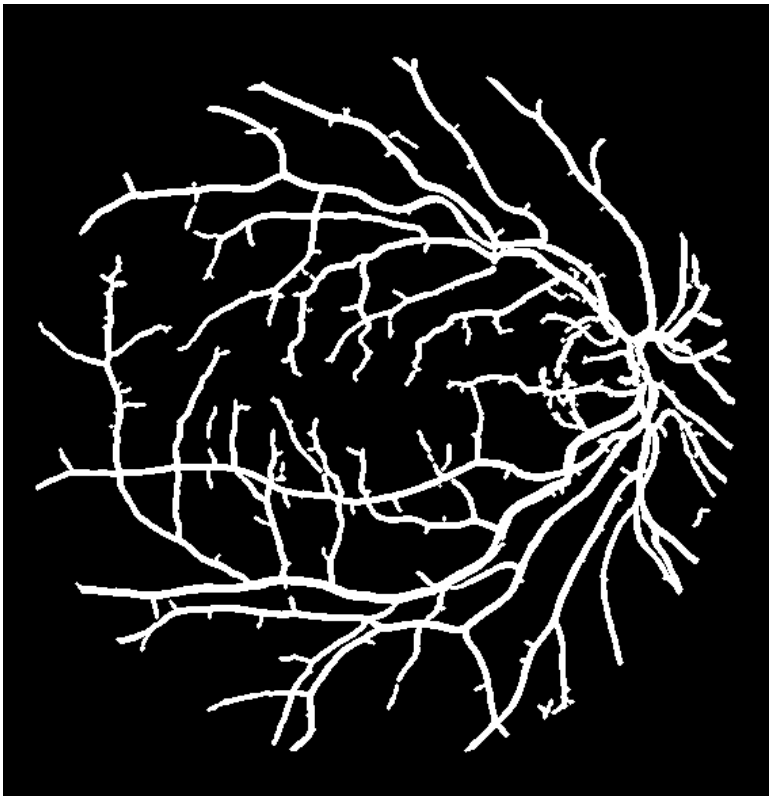
While we have found favourable results in our experiment, it's to be taken with caution as much of the experiment's integrity isn't upheld. Despite that, I've found satisfactory results.

I've also ran the algorithm across other test images to check for overfitting issues in the Annex and it seems like it generalises well.

Annex

(My) Test Image Result

Test Image Input



Test Image Output

Other Test Images

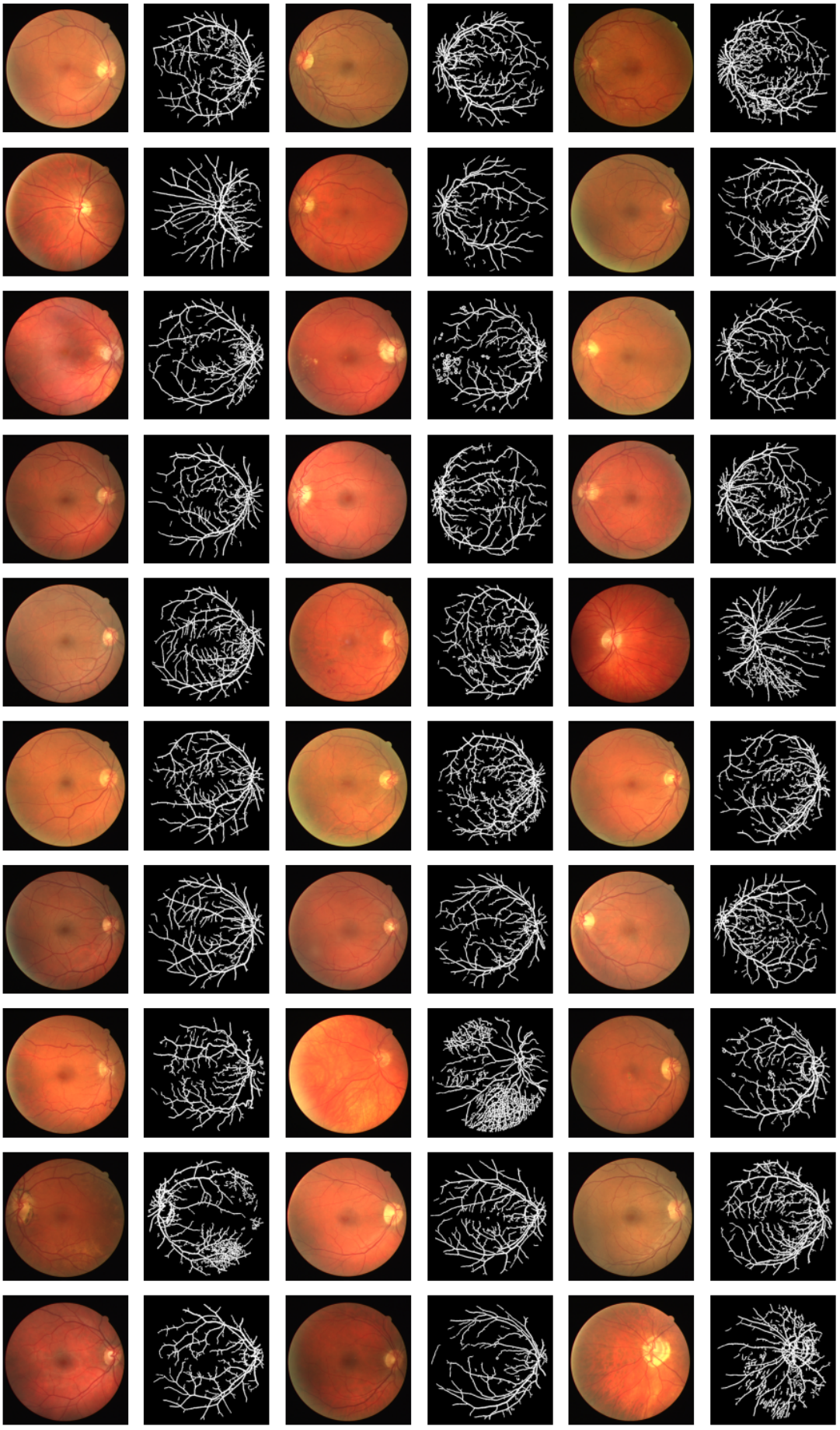
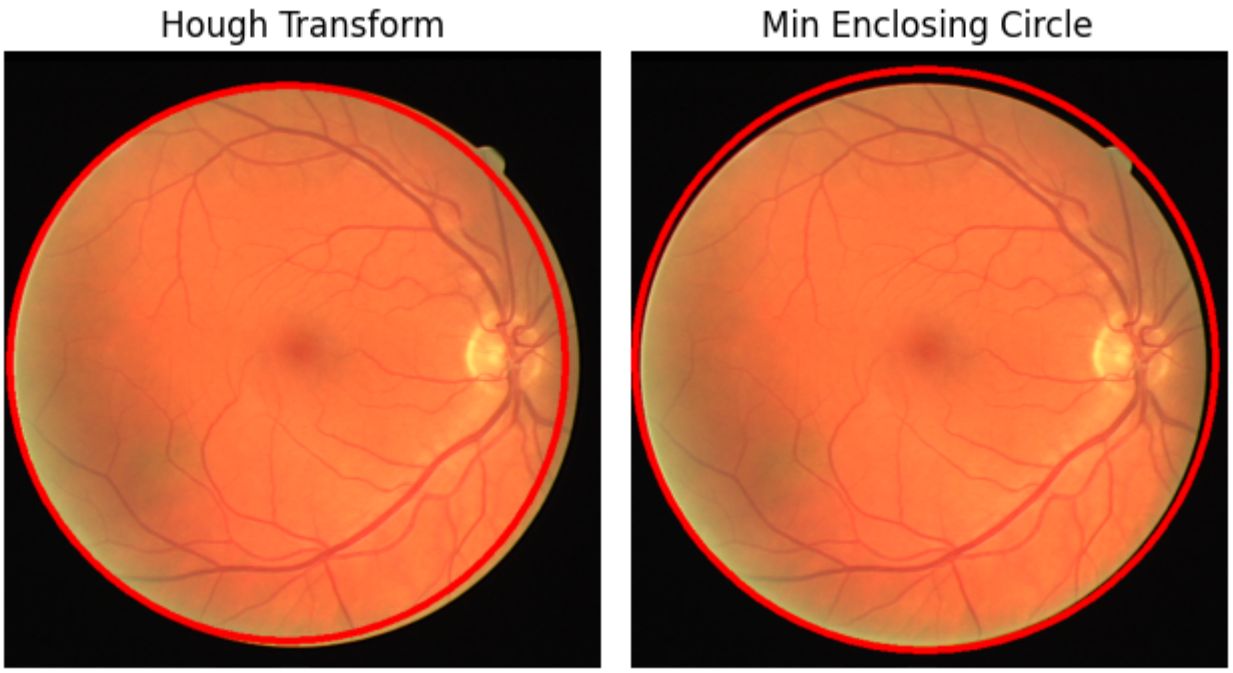


Figure 13: Results of other test images

Alternative Background Segmentation

Before settling on Canny Edge Detection, we tried with the assumption that retinas are always circular. Thus Hough Circle Transform and Minimum Enclosing Circle were used on the filled Canny Edge map:

- **Hough Circle Transform** transforms the boolean image into a likelihood space where a circle of a certain radius and x, y position exists. It then returns the most likely circle.
- **Min Enclosing Circle** tries to enclose the boolean image with the smallest circle.



However, we found that Hough Transform is unpredictable, sometimes intersecting the retina. Minimum Enclosing Circle fails to avoid the “bump” on the top right. Thus we resorted to avoiding that assumption.