Biomedical Image Processing -Final Project: Analysis of Photoreceptors & Retina Diseases

Paul Vu
301169550
April 21st 2017

SFU

Project Overview

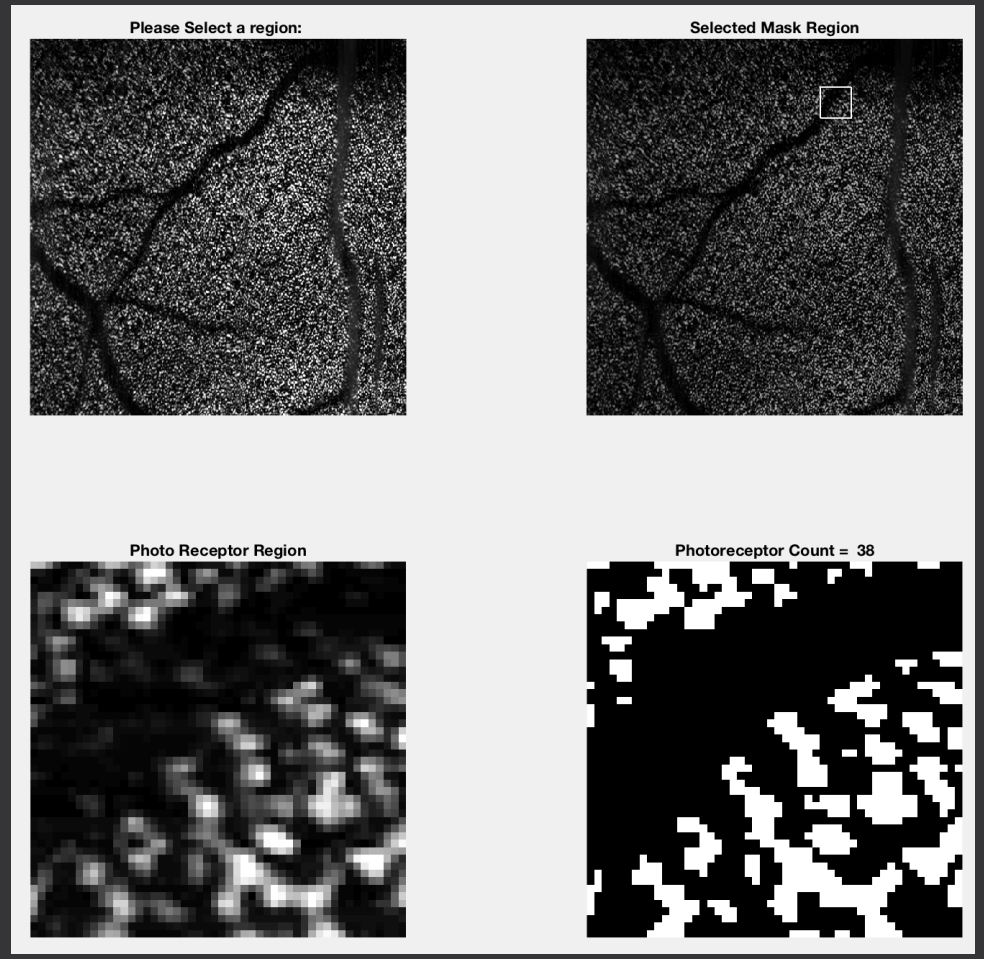
With Optical Coherence Tomography (OCT) scanning systems we are able to obtain 3D images of the retina non-invasively as shown in Fig. 1 below. Here, we can see that each little white dot is a single photoreceptor in the retina. The goal of this project is to design a software & algorithm for doctors to use to easily count these photoreceptors in a given region. This is important because retina's with damaged photoreceptors will have regions with low localization of photoreceptors, therefore having the ability to quantify how many receptors are in a given region can improve diagnostics of diseases within the eye.

Fig.1 An image taken from an OCT scanning system

Project Overview

In this document, we will review two key features of our program.

1. Quantifying photoreceptors of any region on an OCT scan
2. Auto detect possible disease regions of a retina's OCT scan



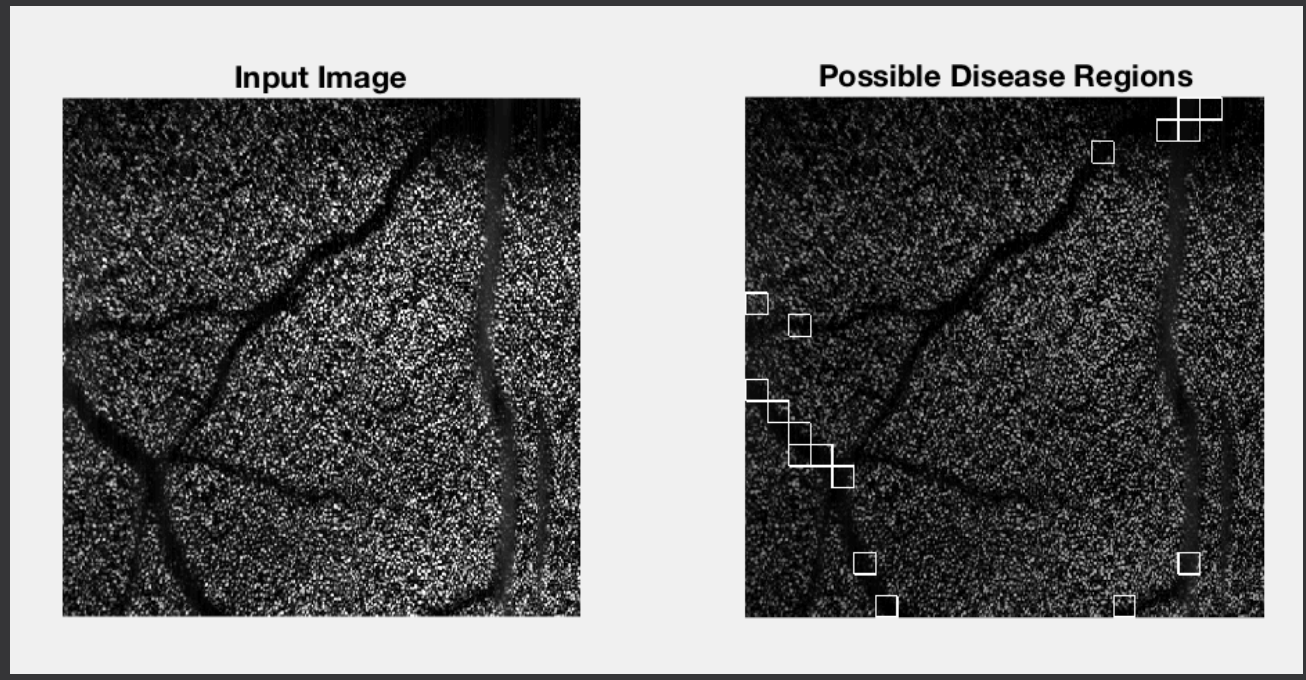
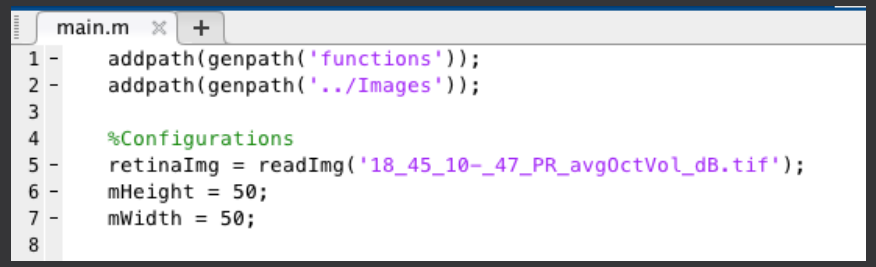


Fig. 2 The two main features of our program, photoreceptor counting and disease detection

Generating a Photoreceptor Mask Region

To begin the program, doctor's must supply the photoreceptor image and specify the mask height and mask width within the "main.m" file as shown in Fig. 3. From here, doctors will run the main script and will be prompted to select a region on the supplied image. By clicking on the region, our program will generate the mask and display the selected region which will be measured as shown below. Note, all test performed were based on a 50x50 mask region.



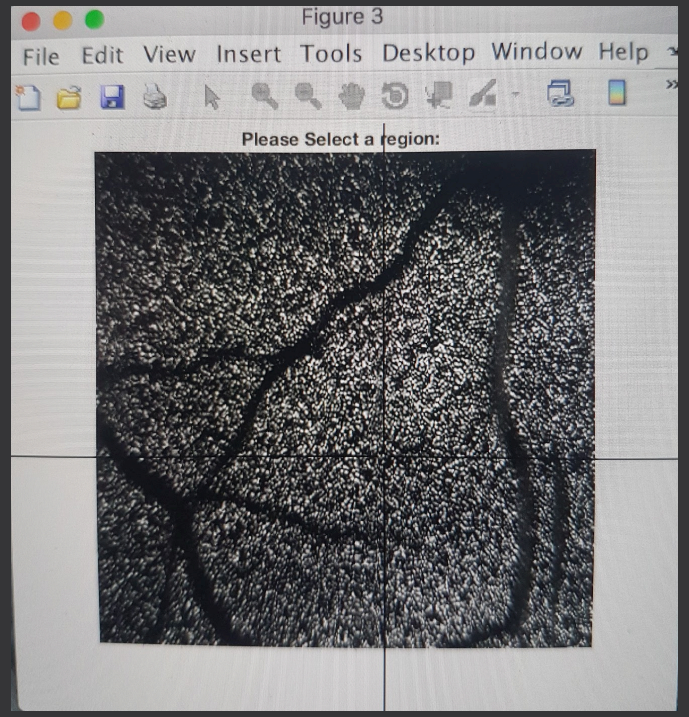


Fig. 3 Based on the specified mask size, a mask will be generated where all the photoreceptors within the selected mask region will be calculated. Here, a 50x50 mask is specified and drawn.

Determine Threshold Value for Segmentation

To determine which value to use for our segmentation threshold we must plot a histogram and check for two class regions (photoreceptor region, non-photoreceptor region). If they exist, we can simply select the threshold value to be a value in between these two histogram peak values. An ideal histogram is shown below in Fig. 4. On the following slide, we plot the histogram of the entire photoreceptor image to determine if two classes of pixels exist.

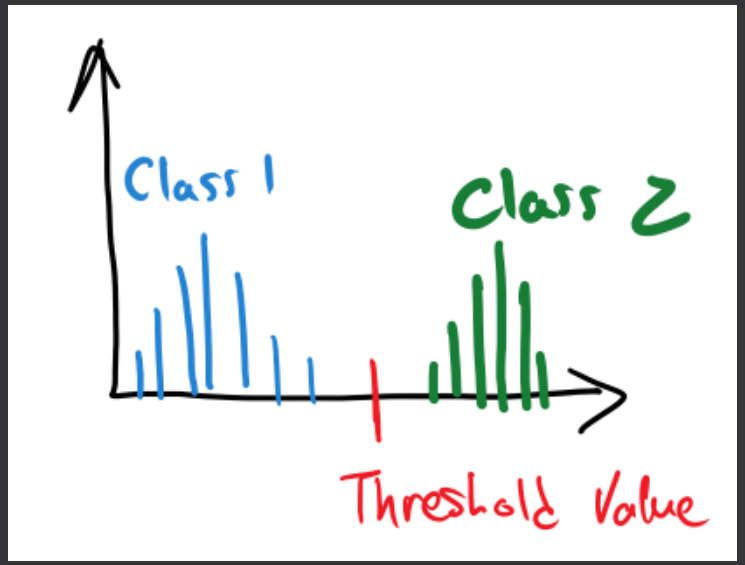


Fig. 4 An ideal histogram of intensity values where there are two classes of pixel intensities in an image, class 1 (non-photoreceptor pixels) and class 2 (photoreceptor pixels).

Determine Threshold Value for Segmentation

As shown in Fig 5. a histogram of intensity values from 0 – 255 is plotted based on the photoreceptor image provided by the OCT scanner. We can see that the histogram does not exactly ideal, however we can still see two classes of pixels shown by the two peaks. Thus an ideal threshold value would exist in between these two peaks. The most optimal value to choose as a threshold would be the intensity value right before the second peak, which would be ~ 70.

Peak 1 (non-photoreceptor intensities)

Peak 2 (photoreceptor intensities)

Ideal Threshold Intensity

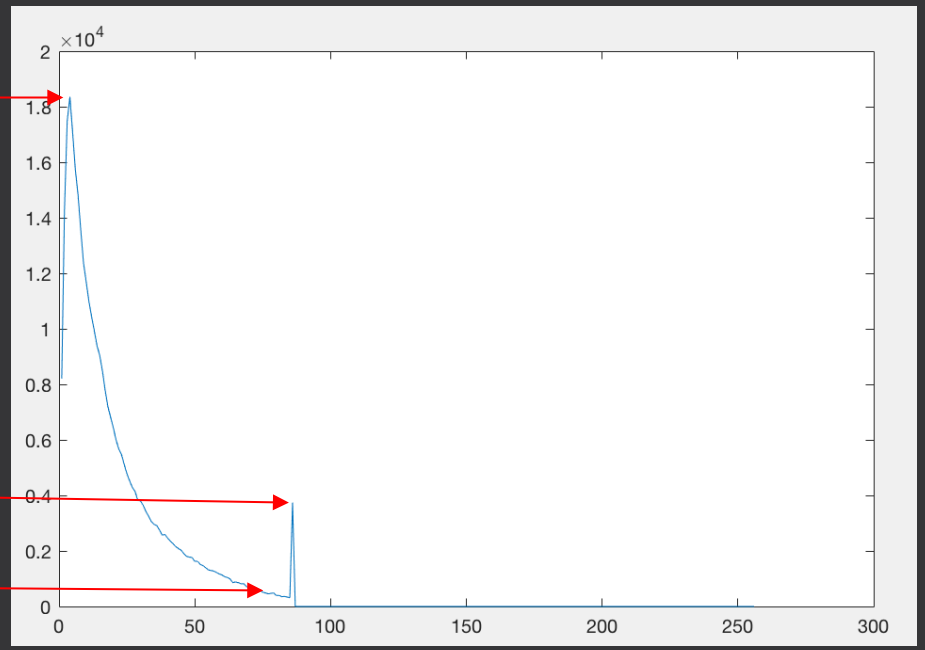


Fig. 5 Histogram plot of intensity values based on the provided photoreceptor image

Determine Threshold Value for Segmentation

Since we determined that the ideal threshold value is ~70, can we assume that this is true for all cases? Probably not, since various OCT scans of the retina will have varying noise & different people could have different photoreceptor intensities during an OCT scan. Thus we can not hardcode this value into an algorithm. We must determine an algorithm to choose a varying threshold value based various histogram plots.

Ideal Threshold Intensity

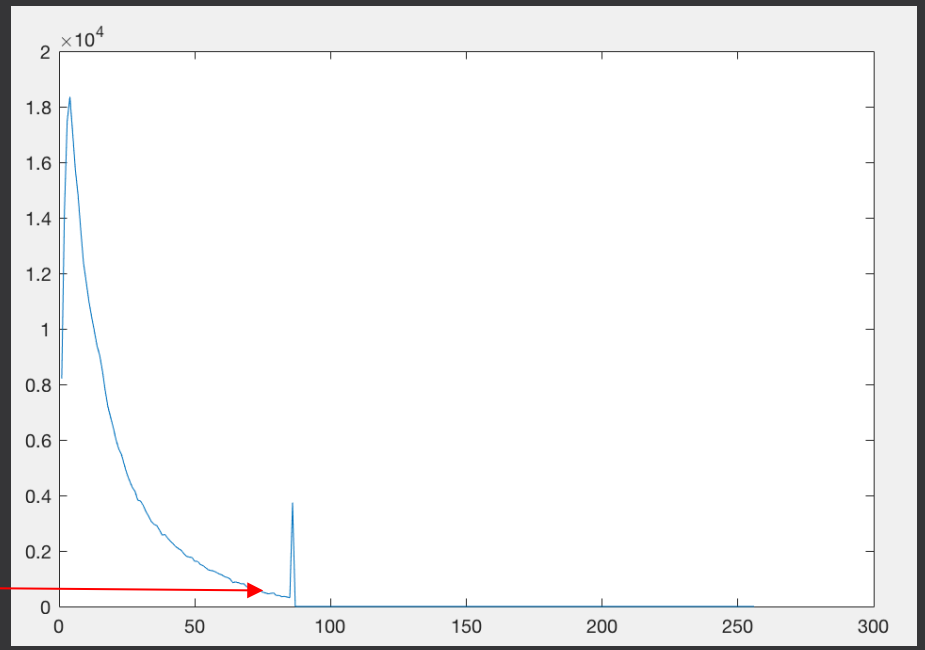


Fig. 5 Histogram plot of intensity values based on the provided photoreceptor image

Threshold Segmentation without Histogram

One idea is to divide the image into photoreceptor pixels and non-photoreceptor pixels based on the strongest intensity value in a region. Since each receptor is a “white dot” we first find the brightest “white dot” and based on this value we assume that a receptor should have at least half this value.

If a pixel had an intensity value lower than half this threshold value, it probably isn’t a photoreceptor (set the pixel value to 0). For all other pixels we set it to 1 therefore the only values in our output image are either 0 or 1. The results of this segmentation are shown below in Fig. 6.

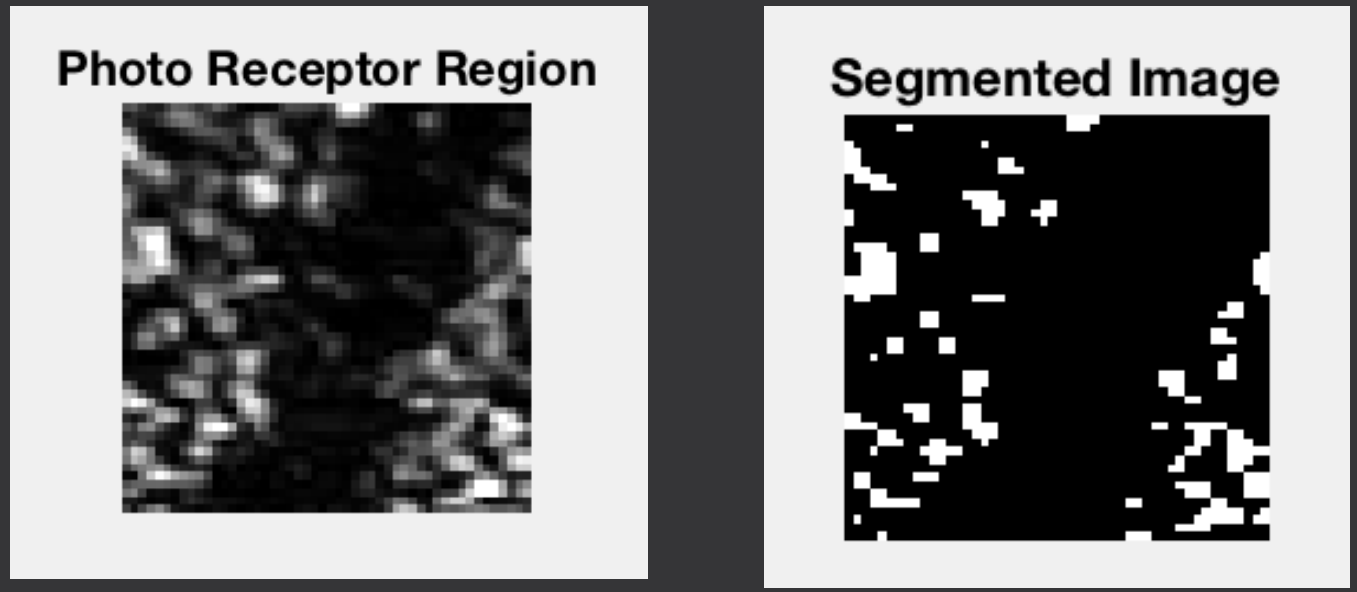


Fig. 6 A 50x50 photoreceptor region which is segmented based on thresholding using the 0.5*highest pixel intensity value of this region.

Threshold Segmentation Analysis

By using the highest intensity value as a threshold we filtered out some photoreceptors which is to be expected since there will always be some data lost when introducing a filter. However, we note that quite a few photoreceptors were lost and thus our threshold value should be lower in order to preserve more information.

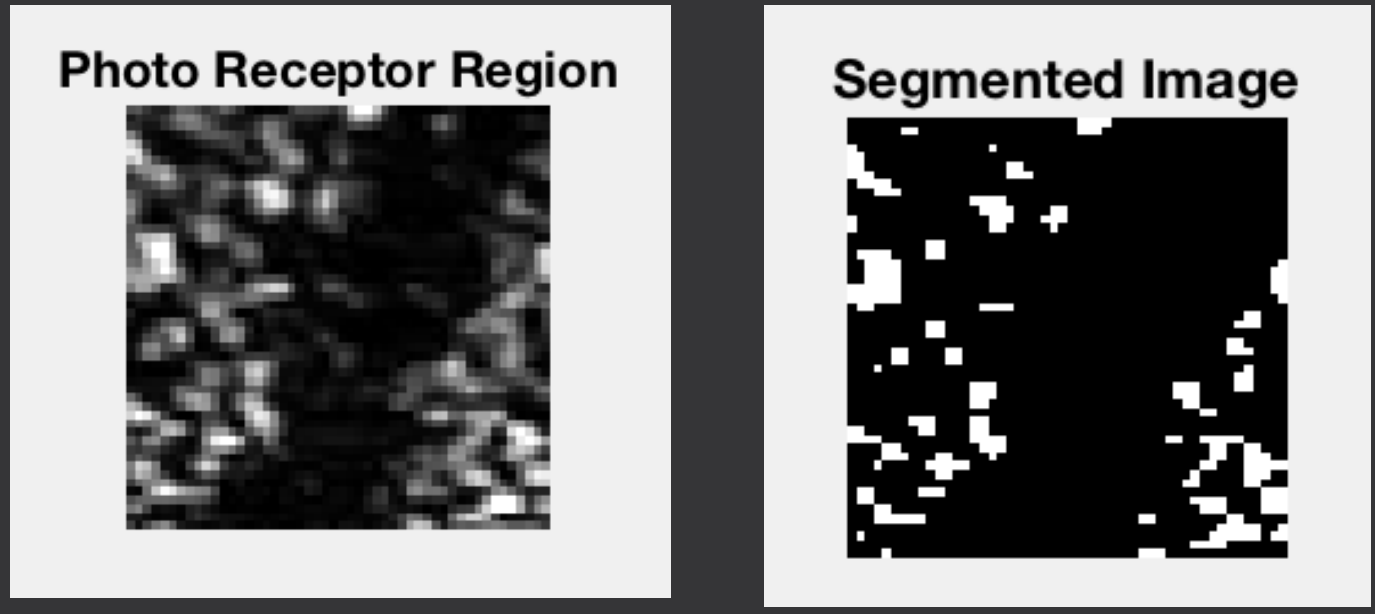


Fig. 6 A 50x50 photoreceptor region which is segmented based on thresholding using the highest pixel intensity value of this region.

Photoreceptor Image Analysis

In analyzing the previous results, there are a couple things to note.

Firstly, we must select a threshold value carefully such that we do not lose too much information when segmenting.

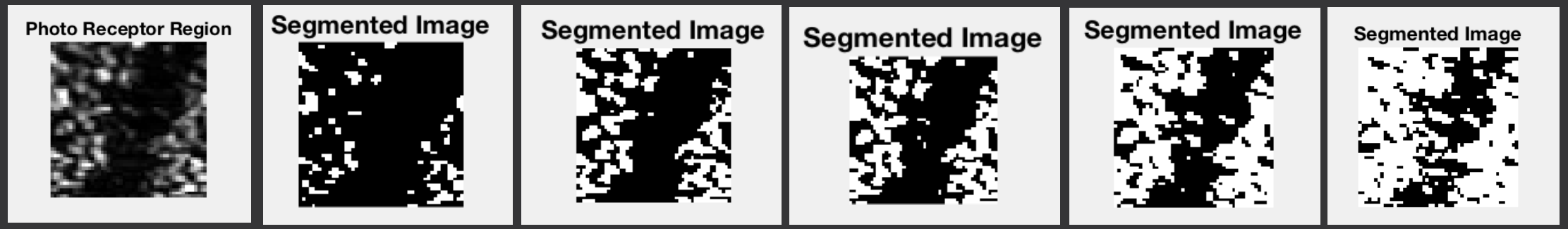
Secondly, not all photo receptors share the same intensity value. In Fig. 7 we can clearly see not all white dots are as bright as others. Thus, our threshold value should ideally be the smallest intensity value of a photoreceptor.

Thus, it is important to keep these issues in mind when developing an algorithm.

Fig. 7 An image taken from an OCT scanning system

Estimation Threshold value

We see that segmenting the image based on half of the highest intensity value found in the region results in large data lost. Therefore, we must use a smaller threshold value to preserve data. Ideally, this value would be the smallest intensity value of a photoreceptor. We manually evaluated various threshold values based on the largest intensity value with varying scales and illustrate our results in Fig. 8 below.



Original P =50% P =25% P =20% P =10% P =5%

Fig. 8 Varying the threshold value used in segmentation. The above results shows the threshold value based on a percentage of the largest intensity value found in that region. For example, with a max intensity value of 0.7 and P =50% we would set all pixels of a value of 0.35 or greater as a photoreceptor, all values less than this threshold is set to 0.

Estimation Threshold value

From these results we can see that a large threshold value such as P = 50% will result in large data lost from the original image. However, too small of a threshold such as P = 5% will see neighboring photoreceptors cluster together due to the noise of the image.

Thus, there is a trade-off as we decrease the threshold value we preserve more photoreceptor data (high sensitivity) at the cost of inaccurate clustering of nearby photoreceptors (low specificity). In both cases, these will impact the number of photoreceptors our algorithm will count.

Thus the ideal threshold is somewhere around the middle of these two extremes, for example P = 25%. At this threshold value, there is a little data lost with minimal inaccurate receptor clustering due to noise. We understand that this is subjective from image to image and give the doctor the ability to change this threshold value in the "main.m" function under configuration.

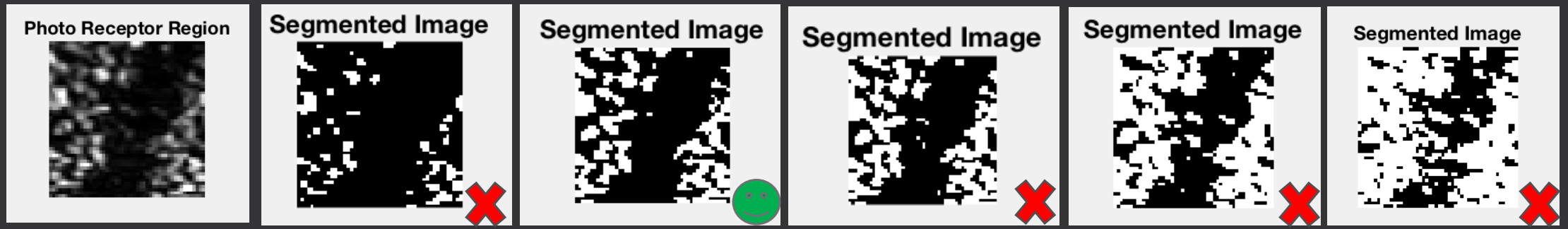


Fig. 8 Varying the threshold value used in segmentation.

Estimation Vs Ideal Threshold Value

Interestingly, we see that 25% of 256 gives an intensity value of 63.75. This is a good approximation to the ideal threshold value which was 70. However, due to a lack of image samples we are unsure if this approximation is true for all OCT scans. Thus we conclude that our program will allow doctors to use either the approximation method or the ideal threshold method which can be specified in configurations section of main.m.

Note that there is a trade-off here. In using the threshold estimation method based on a percentage of intensity value, doctors can use our program more quickly since it requires no user input. However, for more accurate results doctors must manually select a threshold value which will take longer and require some training in selecting a proper threshold.

Threshold Estimation

Ideal Threshold

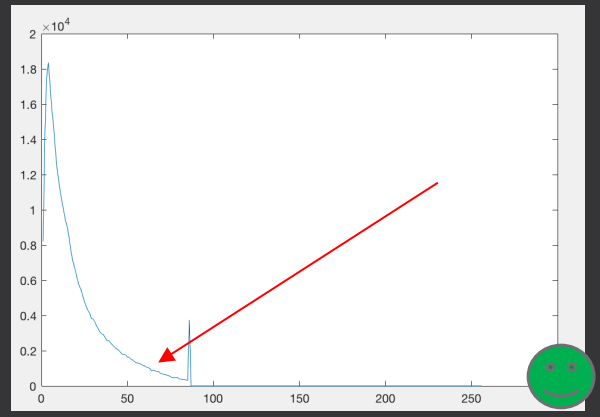


Fig. 9 With P = 25%, an estimated threshold intensity is .25*255 = 63.75. While the Ideal threshold determined from a histogram plot is approximately 70.

Manually selecting Ideal Threshold Value

The figure below illustrate how a doctor would manually select an ideal threshold value should they choose.

```
%-------------------------------------------------------------------------------------
%-------------------------------------------------------------------------------------
%Configurations


%Counting photoreceptors of a specific region configurations
retinaImg = readImg('18_45_10-_47_PR_avgOctVol_dB.tif');
mHeight = 50;
mWidth = 50;
%Uncomment below to allow for ideal threshold value based on user selection
%thresholdValue = 0.25; %uncomment this to use an estimated version of segmentation threshold
thresholdValue = idealThresholdCalc(retinaImg); %uncomment this to use an ideal version of segmentation threshold


%Disease detection configurations
disease_mask_height = 25;
disease_mask_width = 25;
%-------------------------------------------------------------------------------------
%-------------------------------------------------------------------------------------
```

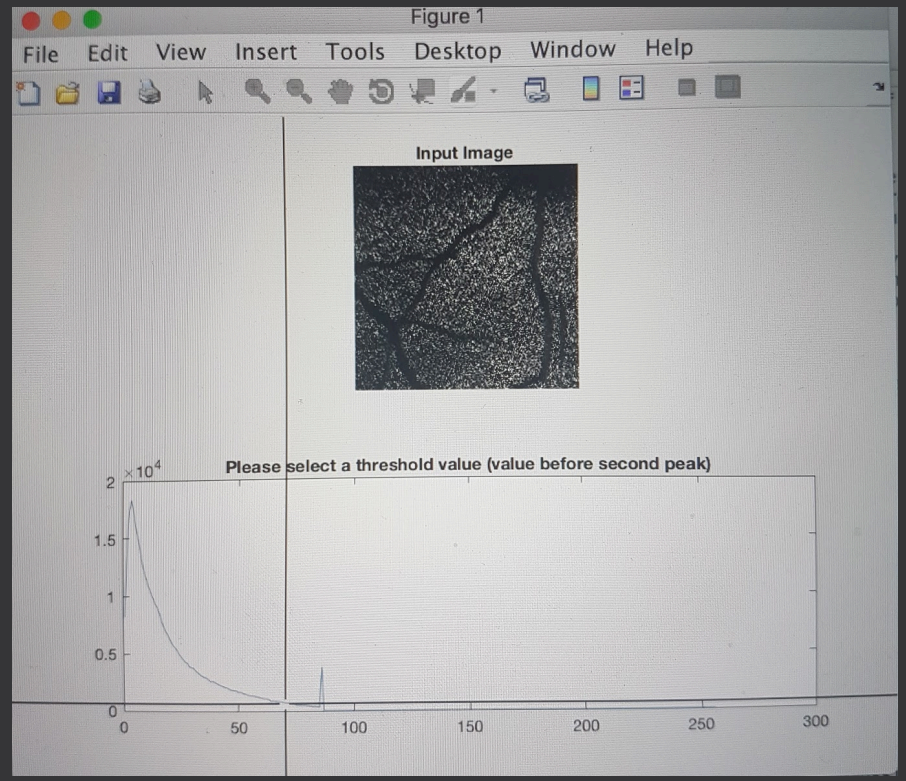


Fig. 10 The user interface provided to doctors should they choose an ideal threshold value. Note that the user must uncomment the specified line in main.m under configurations to allow this feature, otherwise the program will use an estimated threshold value of P=0.25*(largest intensity value)

Threshold Value : Otsu's method

Otsu's method is an alternative way in determining a threshold value for segmentation. MATLAB has a built in function called "graythresh()" which calculates a threshold value based on Otsu's method. It's results are show in Fig. 11. In the below example, the threshold intensity value was determined to be 29.98 or 11.76%. As mentioned in the previous slides, an ideal threshold value is closer to 70 or 25% for the same image.

We can see that although the method is effective, it suffers from clustering nearby photoreceptors into a giant photoreceptor. This means that the Otsu's threshold value is likely too small which is confirmed when we compare the obtained 29.98 threshold vs ideal threshold of 70. Thus, we choose not to use Otsu's method in our algorithm.

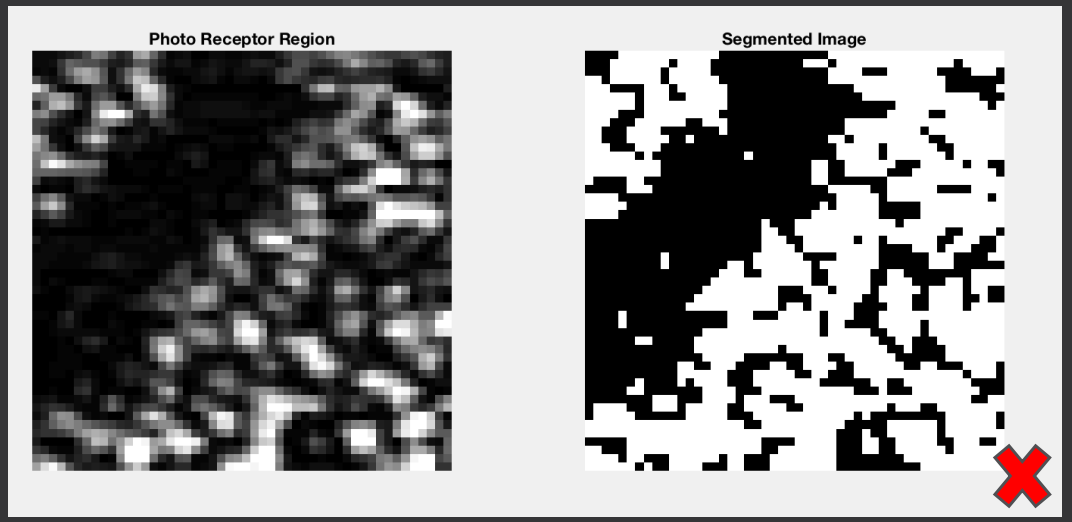


Fig. 11 Otsu Method Thresholding using graythresh() provided by MATLAB

Counting Photoreceptors Method 1:

Now that we have our segmented image (via estimation threshold or ideal threshold), we will attempt to develop an algorithm to count the number of photoreceptors. In our first method, we simply count the number of high intensity pixels in the image since each pixel must represent a photoreceptor. Our results are shown below.

Total Photoreceptors = 270

Fig. 12 Results of counting photoreceptors by counting high intensity pixels

Counting Photoreceptors Method 1 Analysis:

Clearly, there are not 270 photoreceptors in the image below. We discovered that not all receptors are the same size. This is because each receptor could have varying size based on how many high intensity values are clustered together.

Secondly, it is important to note that not every white dot is a single white pixel, photo receptors could be a cluster of 3,4,5 or even 10 or more pixels clustered together with a contiguous high intensity value. Thus, since these receptors have varying sizes we cannot just count by pixel intensities, this would lead to an over estimation as our results clearly showed!

Total Photoreceptors = 270

Fig. 12 Results of counting photoreceptors by counting high intensity pixels

Counting Photoreceptors Method 2

Since this is a binary image (composed of only 0 and 1 intensity values) we can use MATLAB's built in function "bwconncomp" which returns the number of connected components of a binary image based on either 4-connectivity or 8-connectvity.

We are faced with a design choice in determining how to count the photoreceptor's connectivity. 4-connectivity will only join photoreceptor pixels which are connected horizontally and vertically, while 8-connectivity will consider diagonally attached pixels as well. Thus, an 8-connective algorithm will typically count less photoreceptors than a 4 connective algorithm due to the additional diagonal criterion. This is shown more clearly in Fig 13 below.

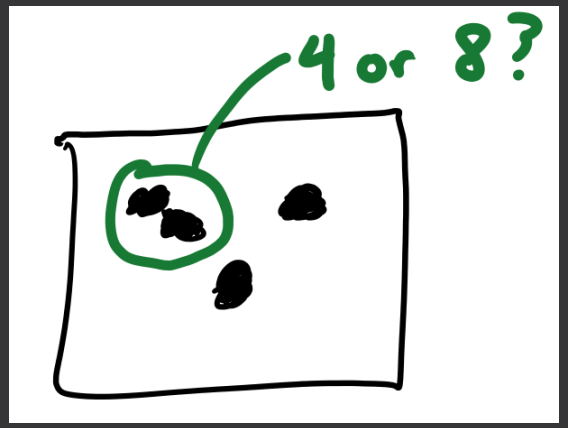


Fig. 13 Using 4-connected counting methods, the above image will count 4 connected objects, while 8-connected method will only count 3 connected objects. This is due to the object which is considered connected diagonally in the 8-connected method, while it is considered as two separate objects in the 4 connected method.

Counting Photoreceptors Method 1 Analysis:

In counting the photoreceptors using bwconncomp with 4-connected specifications we count 36 total photoreceptors, while an 8-connected method counts only 34 total photoreceptors of the below image.

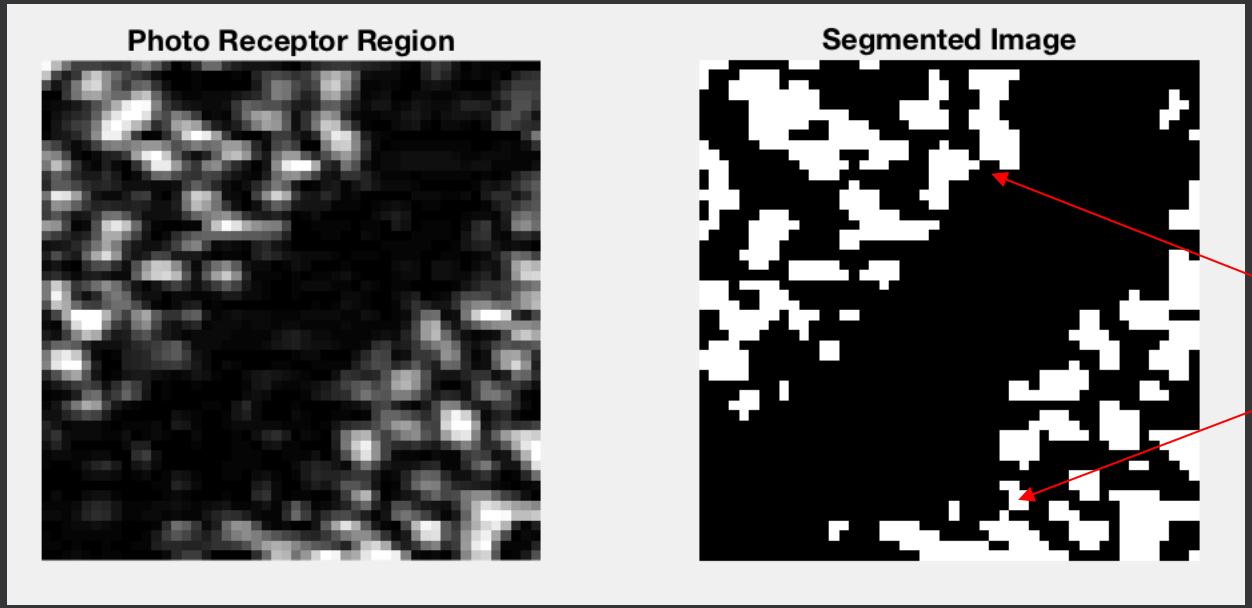


Fig. 14 Illustrates the sample region of photoreceptors being counted. A 4-connected method counts 36 total photoreceptors, while a 8-connected method counts 34 photoreceptors

Counting Photoreceptors Method 1 Analysis:

As a design choice, we believe that a 4-connected method is more accurate in this case since these photoreceptors are likely not connected in real life. Being diagonally connected by a SINGLE pixel is highly unlikely since this would imply a very fine connection in a photoreceptor. A typical photoreceptor could have a diagonal connection but must also be accompanied by a horizontal or vertical connection to establish an obvious connection. Thus, it is justified to ignore diagonally connected criterion and use a 4-connected method.

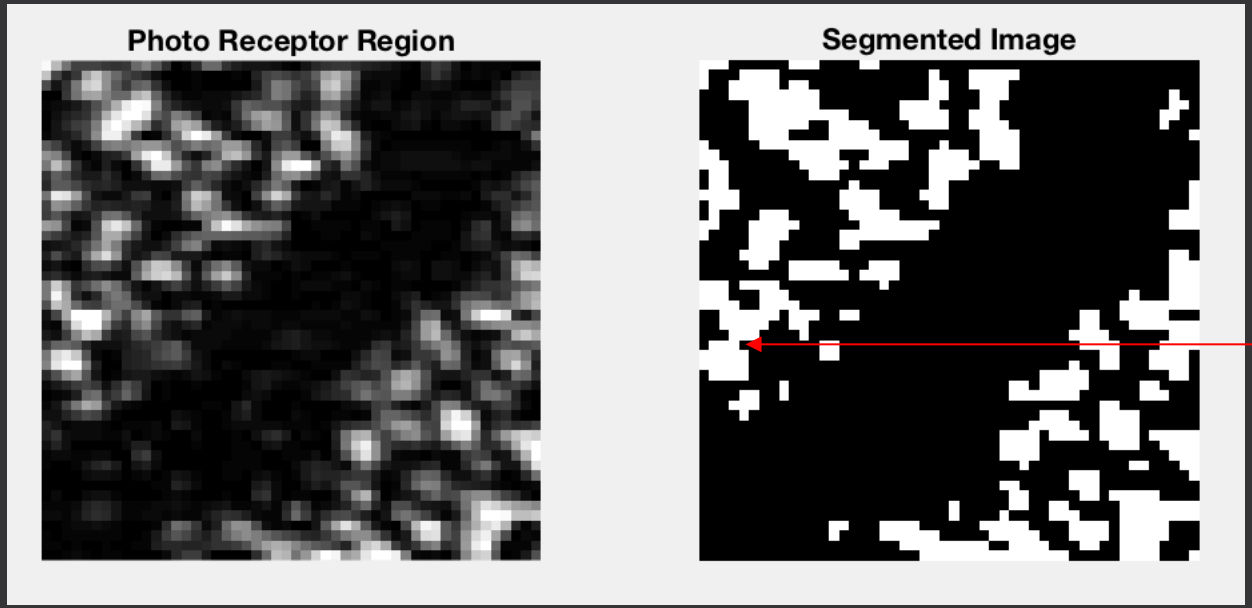
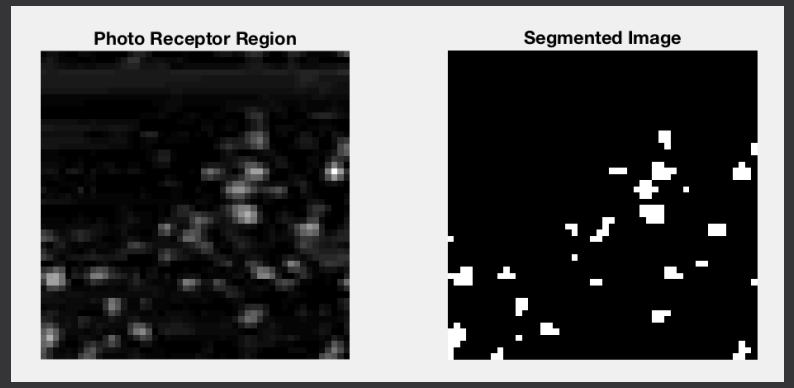


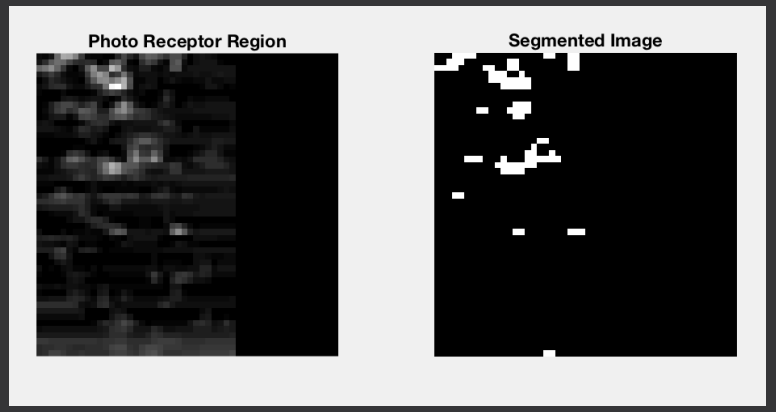
Fig. 14 Illustrates the sample region of photoreceptors being counted. A 4-connected method counts 36 total photoreceptors, while a 8-connected method counts 34 photoreceptors

Various Results:

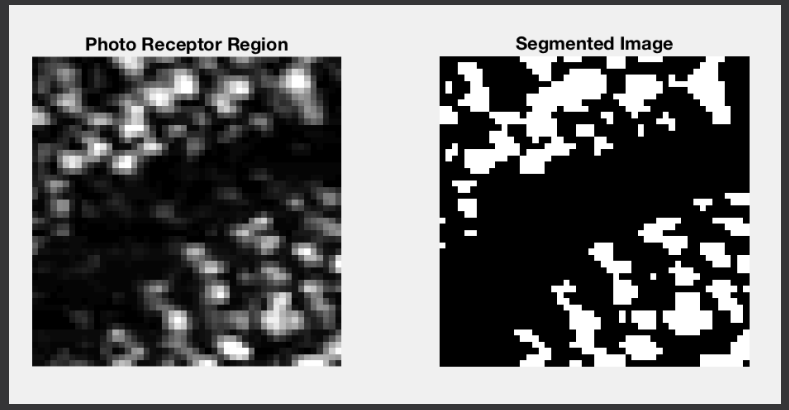
Results for several 50x50 mask regions are shown below.



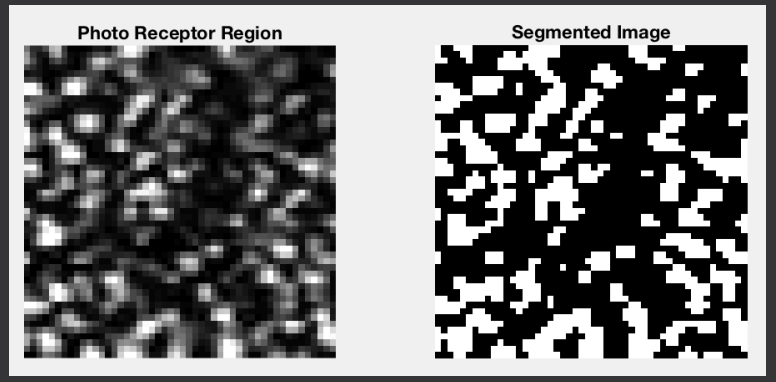
Total Photoreceptors = 26



Total Photoreceptors = 14



Total Photoreceptors = 40



Total Photoreceptors = 59

Disease Detection Feature

Doctors using our program can also analyze a OCT scan of the retina to see if there are any regions with an unusually low number of photoreceptors. These regions could be signs of a disease region of the retina. To use this feature, doctors need only to specify a disease region mask size, for example 25x25. This is specified in the configurations component of “main.m” as shown in the figure below.

```
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%Configurations
%feature 1 Counting photoreceptors of a specific region
retinaImg = readImg('18_45_10-_47_PR_avgOctVol_dB.tif');
mHeight = 50;
mWidth = 50;
thresholdValue = 0.25;
%Uncomment below to allow for ideal threshold value based on user selection
%thresholdValue = idealThresholdCalc(retinaImg);

%feature 2 Disease detection
disease_mask_height = 25;
disease_mask_width = 25;
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
```

Fig 15. Configurations for disease detection require doctors to specify a mask size for partitioning the image in disease detection. The smaller the mask, the smaller the region of disease being searched for.

Disease Detection Algorithm

We first partition the image into grids based on the mask specifications. For a 100x100 mask on a 600x600 size image we will have a partition map of 6x6 as shown below.

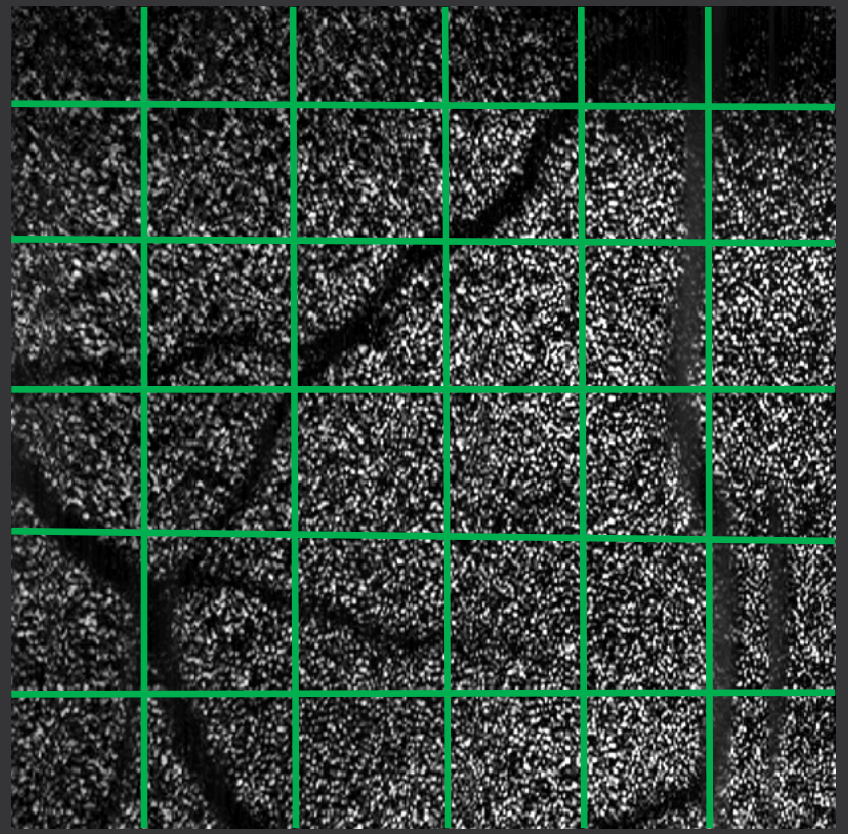


Fig. 16 Sample grid partition based on a user defined mask size

Disease Detection Algorithm

With each partition, we count how many photoreceptors are within each partition using the feature demonstrated earlier in the slides. We keep note of all the photoreceptors found in each partition and take the average found across all regions. We then calculate the standard deviation of photoreceptors in each region.

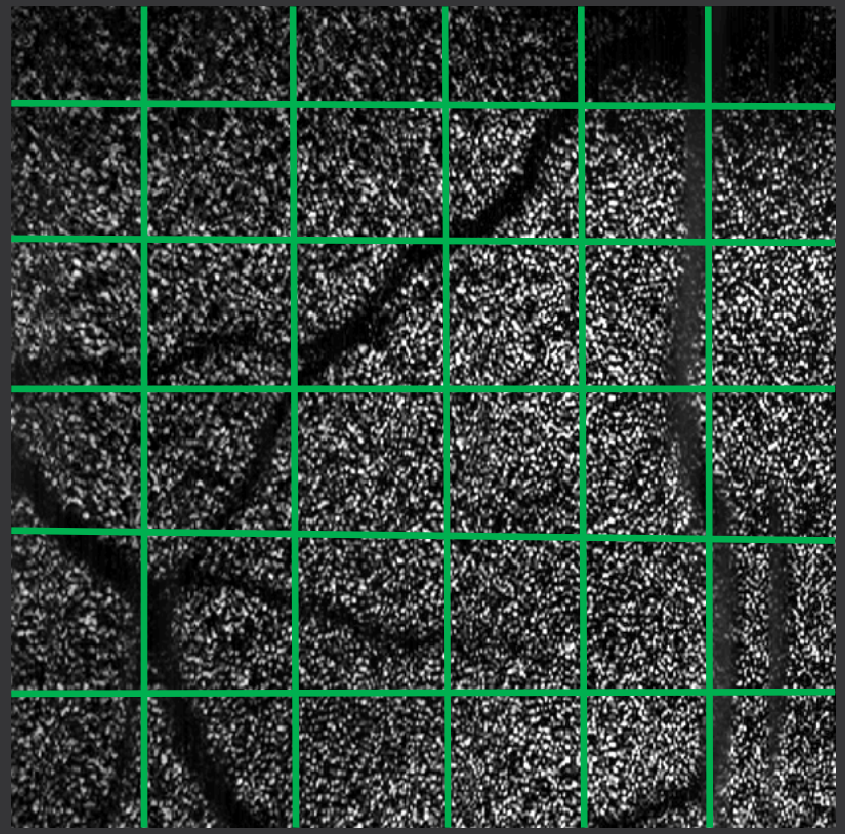
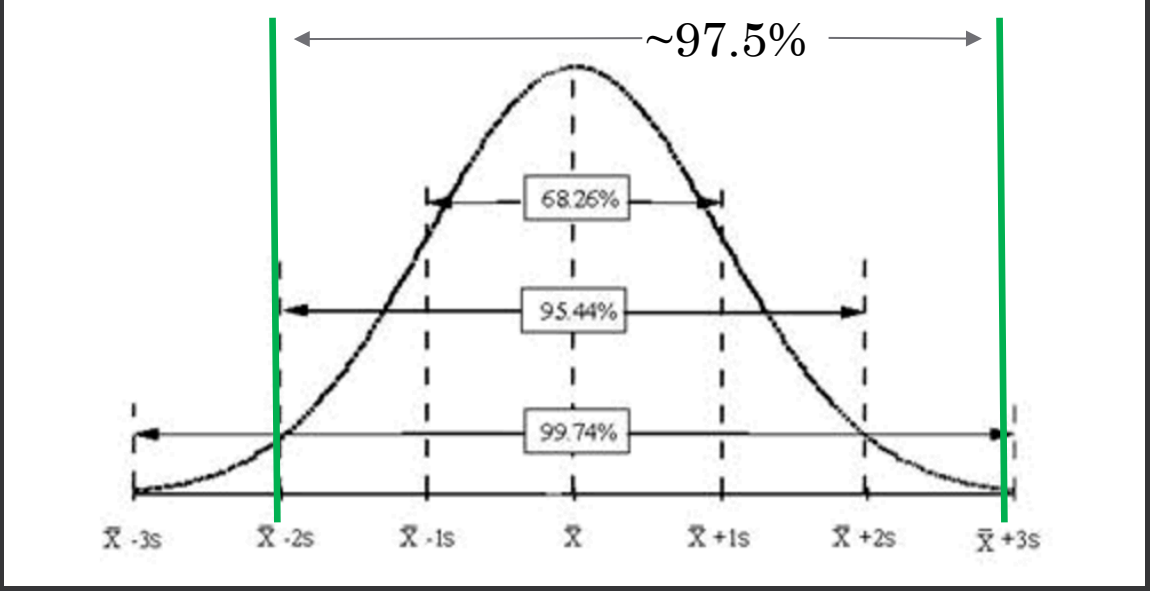


Fig. 16 Sample grid partition based on a user defined mask size

Disease Detection Algorithm

Once we know the average and standard deviation of photoreceptors based on the sample of all partitions we can then calculate a threshold value of photoreceptors in a partition. If a partition has less photoreceptors than the specified threshold, this partition is a possible disease region.

We set the disease threshold to be the difference between the mean and twice the standard deviation. This is because if we assume a normal distribution of photoreceptors in a healthy retina, then approximately 97.5% of photoreceptor partitions should fall within this range as shown in Fig. 17. Any values outside this range is rare (~2.5%), and is assumed to be a possible disease region.



$$Threshold = \mu - 2\sigma$$

$\mu = Mean\ of\ photoreceptors\ in\ a\ set\ of\ partitions$
$\sigma = Standard\ deviation\ of\ photoreceptors\ in\ a\ set\ of partitions$

Fig. 17 A normally distributed data set will contain ~97.5% of the data points within the specified threshold

Disease Detection Results

The below results are shown based on the provided photoreceptor image. These results were obtained using a 25x25 mask with an ideal segmentation threshold value of 0.29. Here, our algorithm draws out possible disease regions. It calculated that the average photoreceptor count was 13.9 based on 625 partitions. The highlighted boxed regions shown in Fig. 18 were regions where photoreceptor count was less than the threshold value, which was 6.1. These regions are well below the average and we can see that the algorithm seems accurate as the black regions (low photoreceptor quantity) of the image are highlighted.

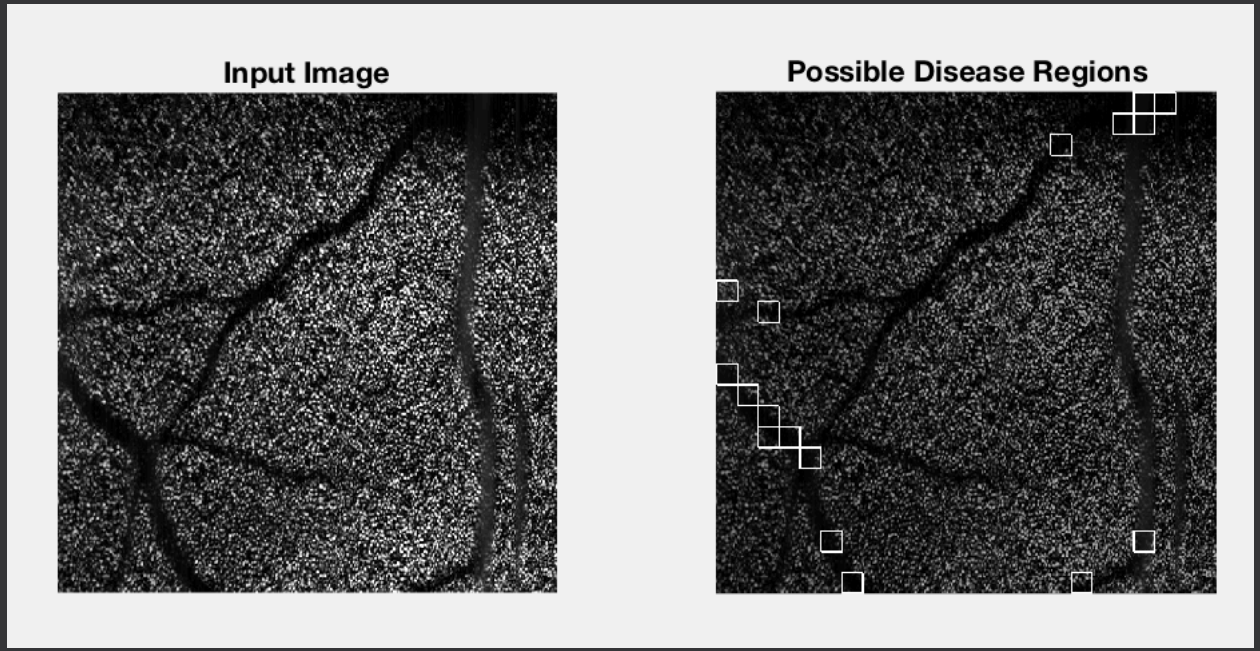


Fig. 18 Results of disease detection algorithm performed on a sample photoreceptor image

Disease Simulations

To solidify the success of our algorithm we created various test images to simulate diseases in the retina. The first simulation image depicts a retina where there is one focal damage at the center with low quantity of photoreceptors. The second simulation depicts a retina with 3 major regions of low quantity photoreceptors. The third image illustrates a uniform reduction in photoreceptors, where every 7^{th} pixel in these regions retain a photoreceptor value. These simulations are shown in Fig. 19 below.

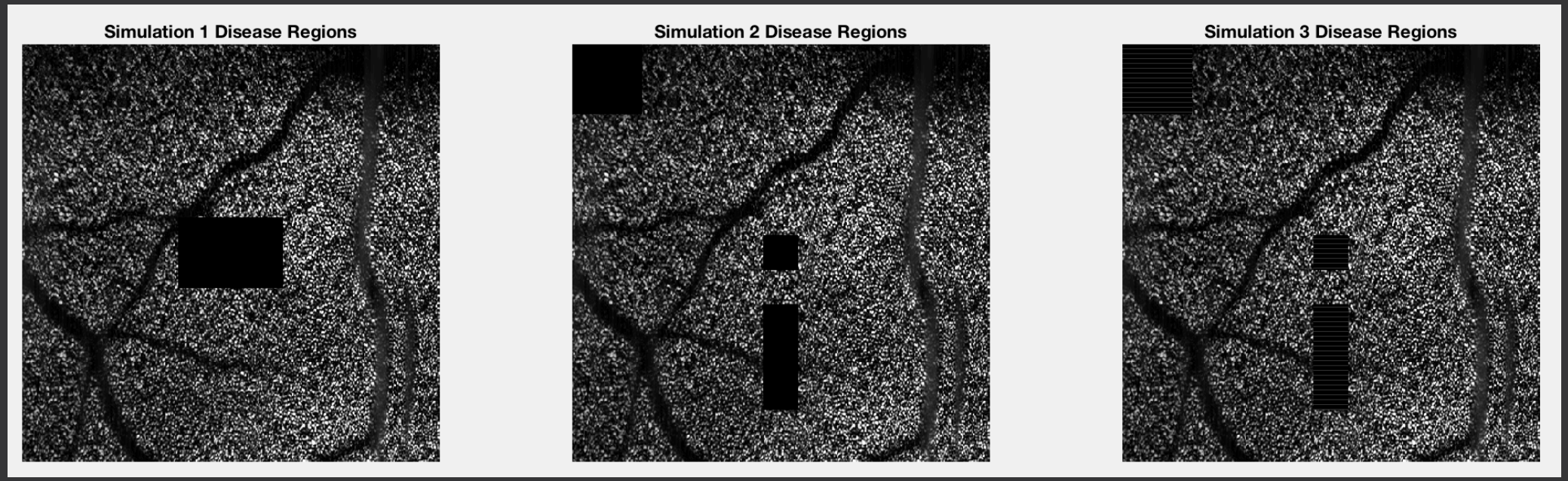


Fig. 19 The three disease simulation images created from diseaseSimulator function.

Disease Simulations Results

The results of our disease detection algorithm is shown below. Our algorithm clearly locates the regions where photoreceptors are low or non existent. This can indicate to doctors that these regions are of interest as they are possible disease regions.

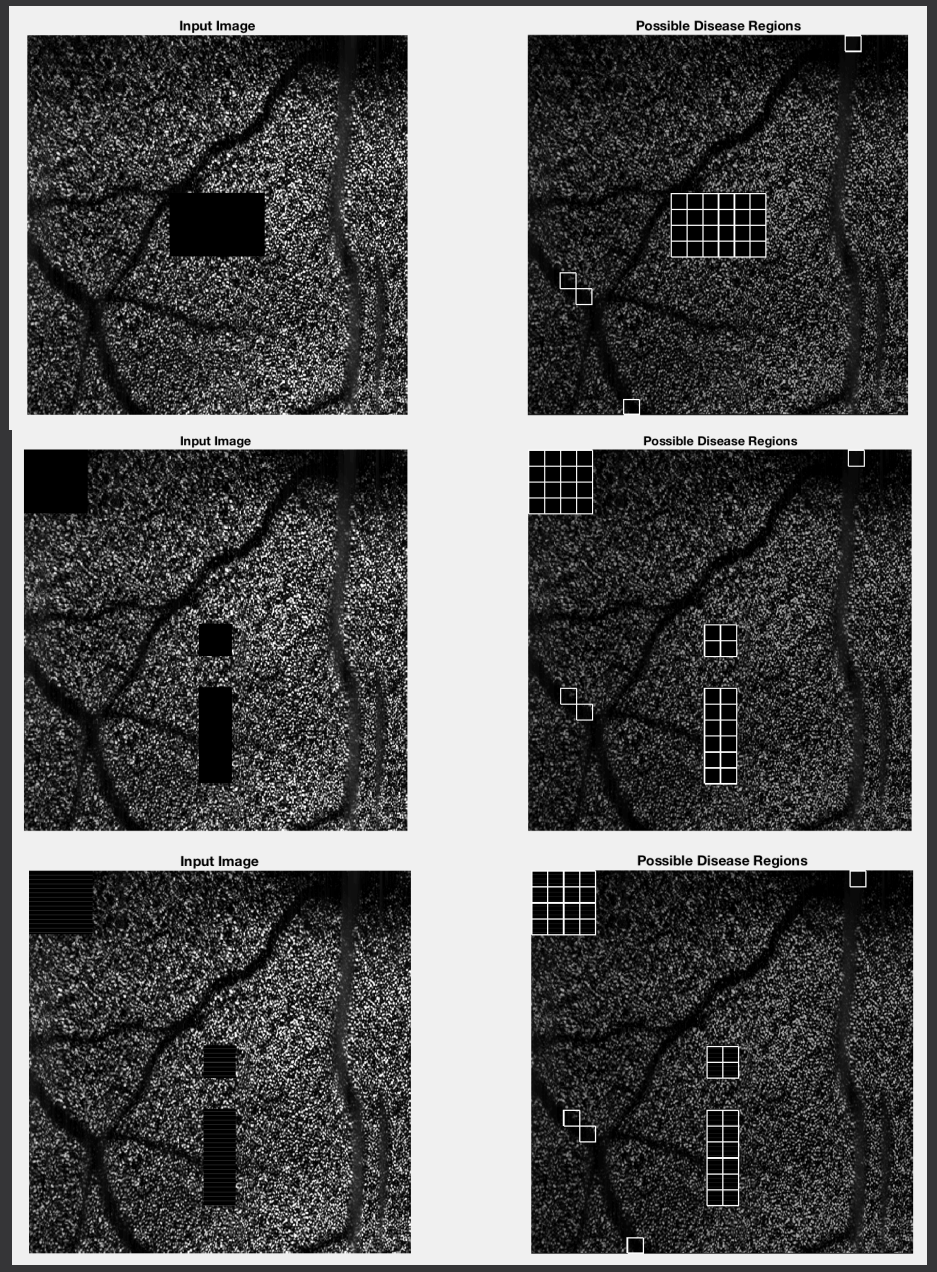
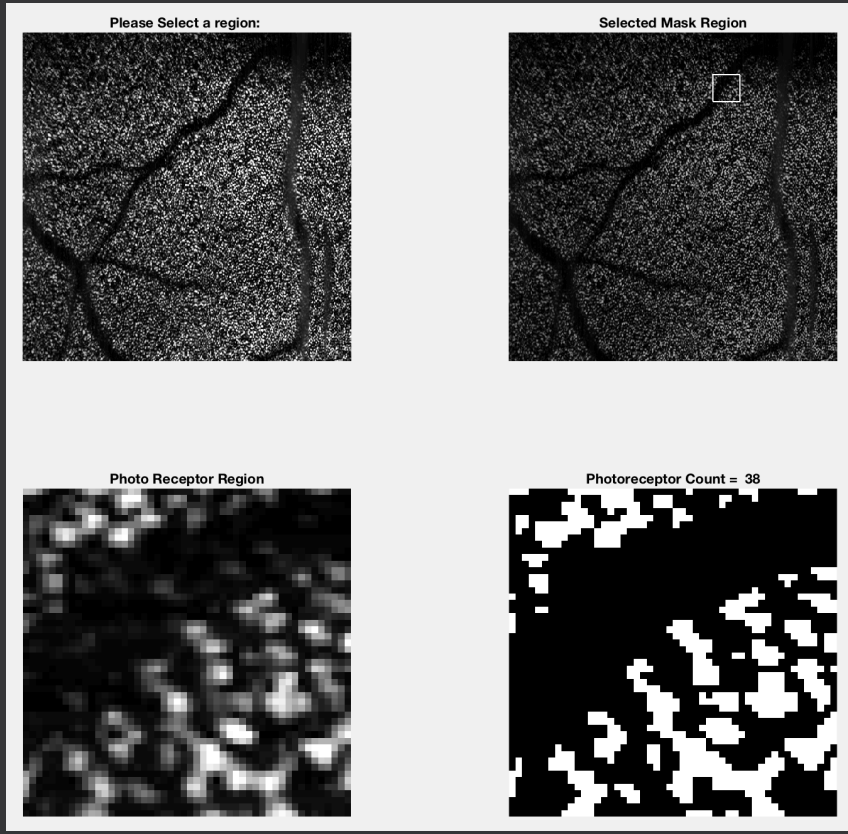


Fig. 20 Results of disease detection algorithm using simulated disease images

Conclusion

Our program allows doctors to dynamically count regions of a photoreceptor OCT scanned image quickly. In addition it allows doctors to quickly parse images for damaged retina regions. These are two useful features which are a product of digital image processing fundamentals.



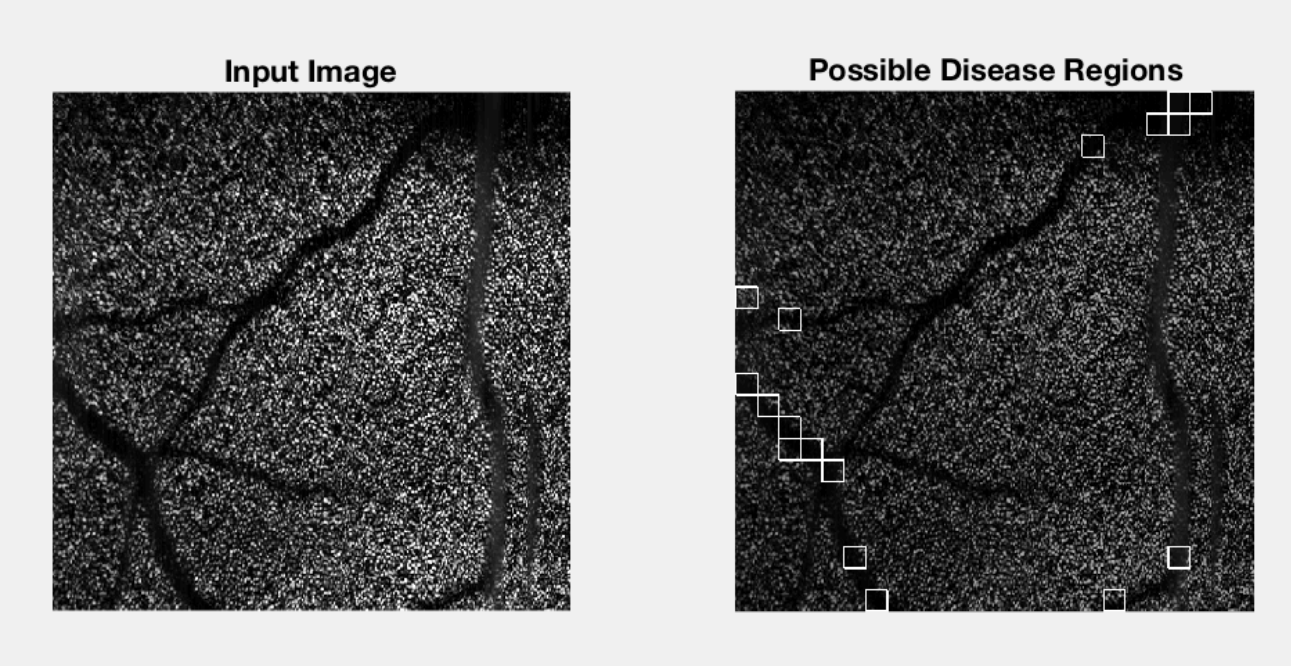


Fig. 21 The above images depicts the two main features of our program, photoreceptor counting of a specified region and disease detection