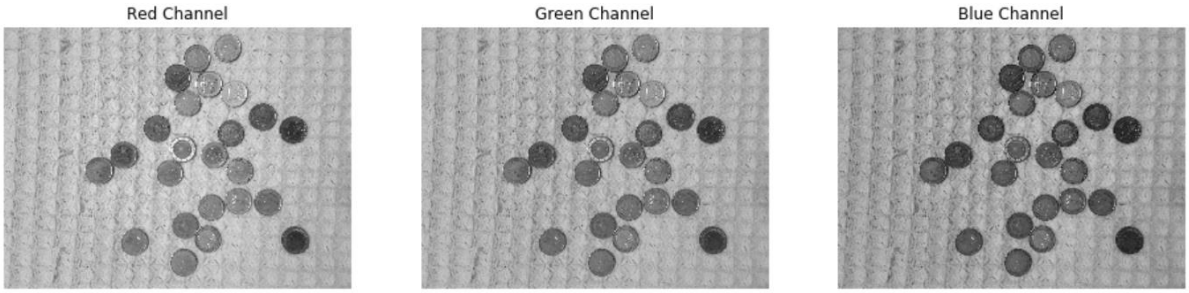
Image Analysis Individual Assignment
Marco Wedemeyer (SNR: 2001451, ANR: 105157)

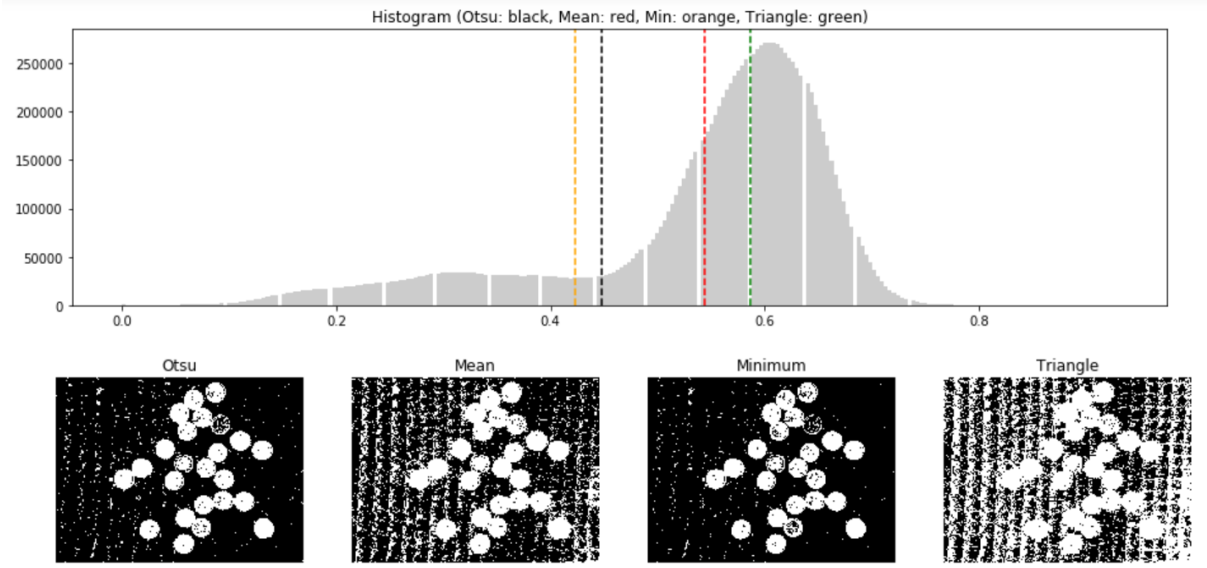
1 Processing an image of coins

1.1 Reading the Image

1.2 Displaying the color channels



1.3 Comparing segmentation methods

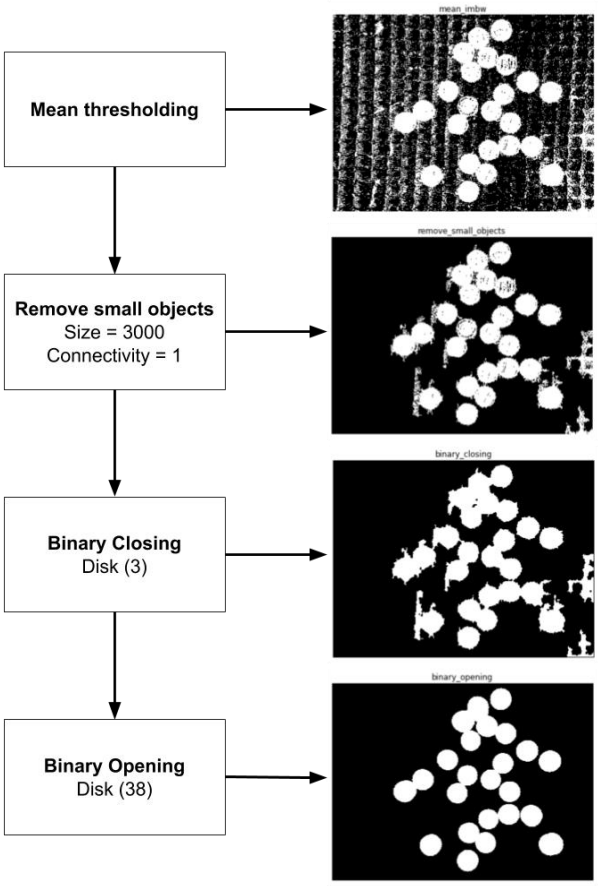


One method of segmenting images is by applying a global threshold. There are many types of global thresholding methods so a choice needs to be made. Above we can see the histogram of the green color

channel of the image, which was chosen for its contrast. Four dashed lines represent the different global thresholds that the Otsu, Mean, Minimum and Triangle methods would set. For the purpose of comparison we will look at Otsu and Mean. In this case the Otsu threshold has removed a lot of noise, however, one of the coins is rather reflective in is thus almost not recognised. The Mean threshold properly captures all of the coins but also includes a lot of noise. A second approach to segmentation is adaptive thresholding.

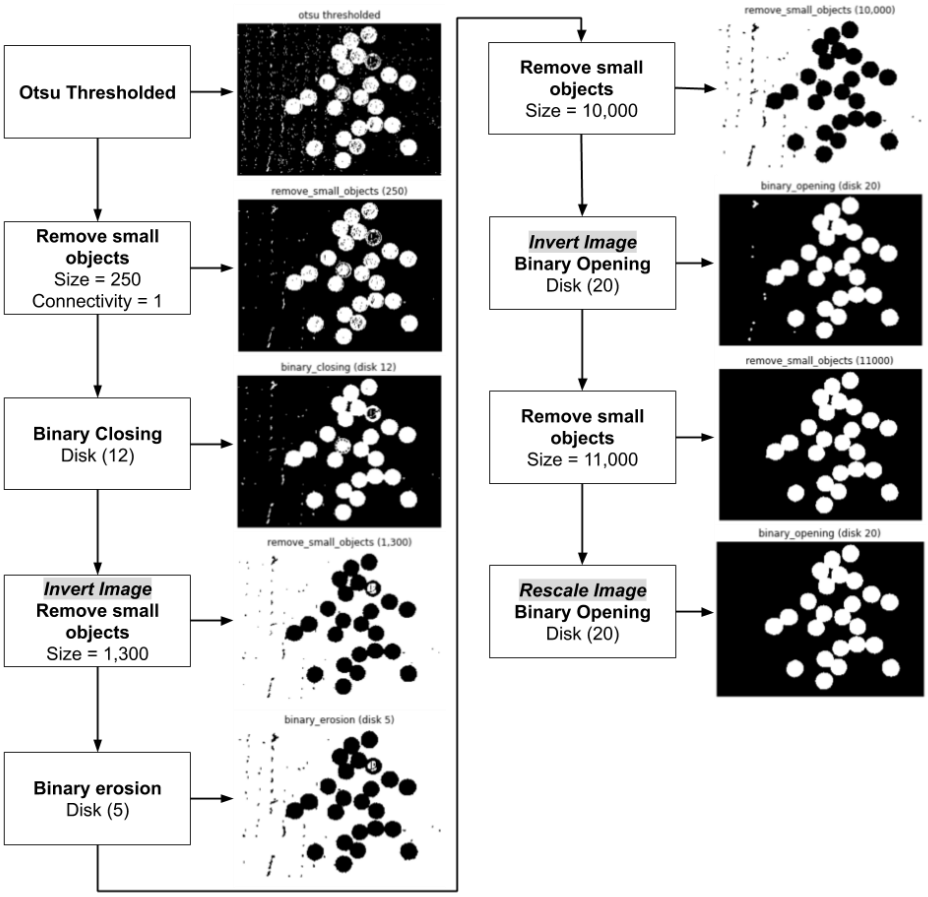
Method 1.1 Mean Thresholding

Firstly we apply the mean thresholding to the green channel. Next we remove as much noise as possible by removing small objects of the size smaller than 3000 pixels. Following this we perform a binary closing with a disk of radius 3. This fills in the coins. Lastly we apply an opening using a disk mask of radius 38. This removes the noise around the coins that do not fit the round shape of the coins. We are left with only the coins marked as 1 and the background as 0. A caveat is that the space between the four touching coins in the top middle of the image is rather small. Due to the noise present in that region the information about the edges was nearly destroyed. The other coins retained their shape rather well.



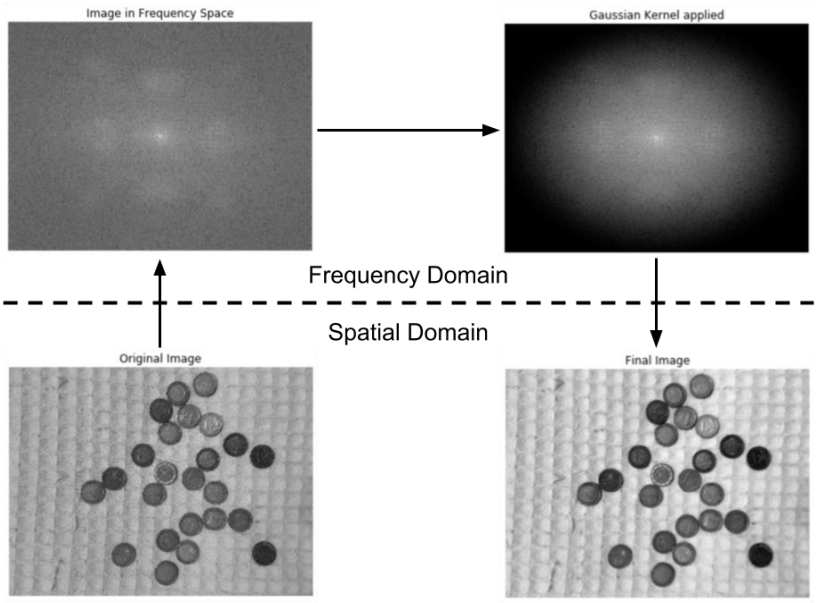
Method 1.2 Otsu Thresholding

The first step is to threshold the green channel using the Otsu method. Next we remove noise by removing objects smaller than 250 pixels. Then we perform a closing with a disk of a radius of 12 to fill up the coins as best as possible. This step can not be as effective as in the previous method as the hole in reflective coin is larger than the gap between the four touching coins. To preserve information of the gap the disk can't be much larger. The next step is to invert the image and remove small objects of size less than 1,300 pixels. This removes the periodic edge wholes of the coin in the center. The following two steps are used to fill in the reflective coin. The first step, erosion separates the inside of the coin. This is possible as the shadow thrown by the imprinted number 1 was set to 0 by the thresholding. This process creates several smaller objects that are each smaller than the gap between the four coins. The second step can now safely remove them by removing objects smaller than 10,000 pixels. Following this we invert the image once again and perform and opening of disk 20 to remove noisy pixels attached to the coins. Next we remove the rest of the noise by removing objects smaller than 11,000 pixels. As the coins are still deformed the following steps try to correct this. As these convolutions would require disks of radius 80 and this would take a long time to calculate, we rescale the image to ¼ of its original size. The last step is to open the image with a disk of radius 20.

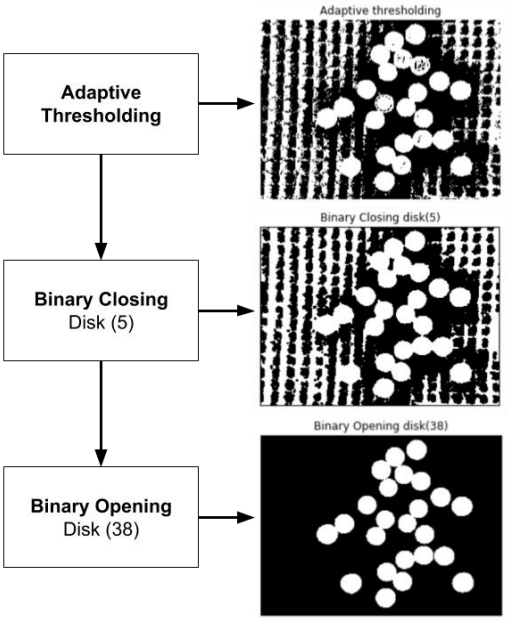


Method 2 Adaptive Thresholding

In order to make the adaptive thresholding more successful we first blur the image using Gaussian kernel in the frequency domain.



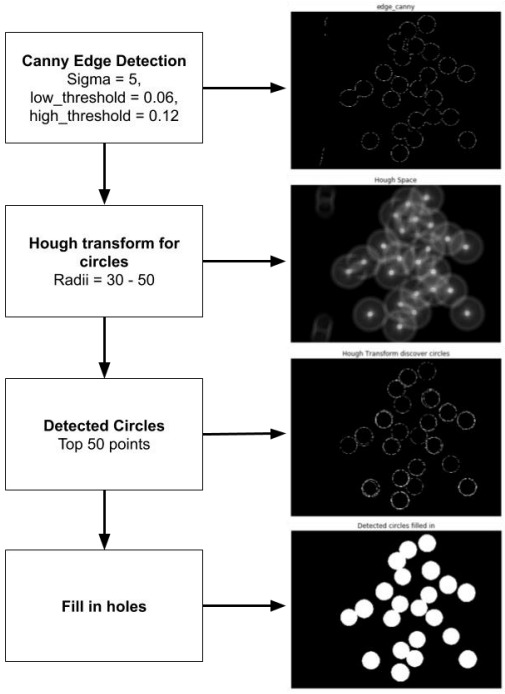
The adaptive thresholding applied to the blurred image results in an image rather similar to mean thresholding. The method used in the adaptive thresholding was coincidentally also mean. A major difference is that the coins appear to have a periphery in which there is little to no noise. This is however only the case in the middle of the image as there was a direct light pointed in that region. The coins to the left are still connected to noise. The next step after the thresholding is to close the image. This once again creates solid disks representing the coins. As the reflective coin had enough white pixels in it, we can use a simple disk with radius 5. As the background noise has sufficient holes in it, we can apply an opening with a disk of radius 38 to the image and get a rather good result. Some of the circles are not perfectly round as the noise distorted the shape. However, a larger disk used for opening would remove some of the coins.



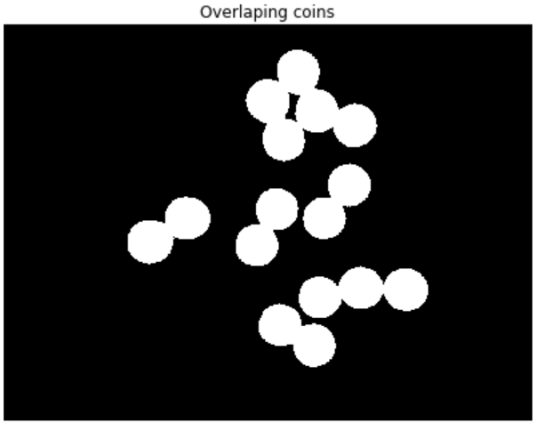
Honorable mentions: Canny Edge Detection with Hough Transform

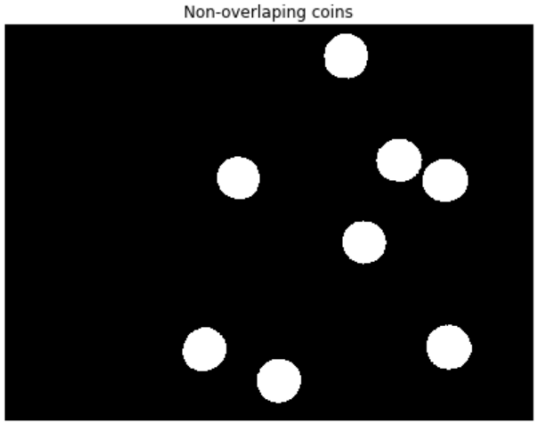
I also tried to perform the segmentation using Canny edge detection and hough transform circle detection. The final image only contains 22 out of the 24 coins thus it is an honorable mention.

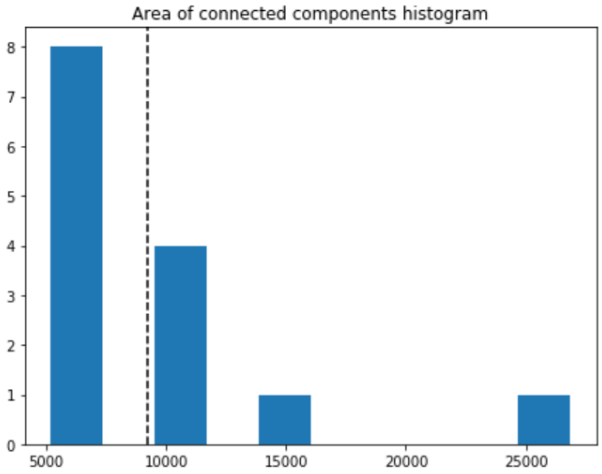
The first step was to perform canny edge detection on a rescaled (¼) green channel image. After some experimentation I settled on a sigma of 5 to sufficiently blur the image, an upper threshold of 0.12 to catch most of the circles, and a lower threshold of 0.06 to add the rest of the circles to the image. Following this I used the hough circle detection functionality to search for circles with radii in the range of 30 to 50 pixels. The next step was to display the top 24 circles. These only included 16 of the 24 coins so I increased the total number of peaks in the accumulator matrix to 50. This problem was the case because quite a few of the coins were overlapping. The two coins that ultimately did not get displayed in the final image were surrounded by 3 other coins or more. This method could have been viable for an image of coins that did not have as much overlap.



1.4 Overlapping and non-overlapping coins







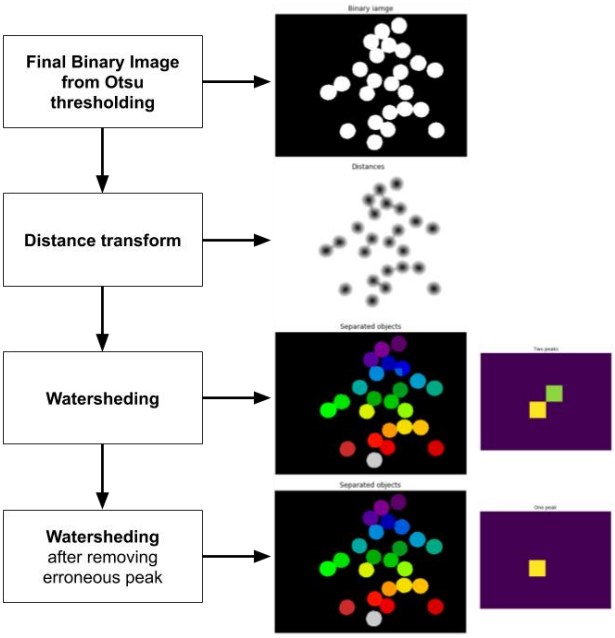
The two types above were categories using an automated approach. Using the fact that overlapping coins form one connected component, we can tell them apart from the single coins by their increased size. We extract the area of all connected components using the region props function and plot the areas on a histogram. Knowing that two overlapping coins will be almost twice as big as a single coin and looking at the distribution we can see that using the mean is sufficient to differentiate between the two types.

There are 8 non overlapping coins, leaving 16 overlapping coins, which together form 6 connected components. If we had many more coins and especially more large groups of overlapping coins, then this method could misclassify some coins. The pairs of overlapping coins would be at risk. A more sophisticated approach should be implemented in

that case. A possibility could be to use the same methodology used in the Otsu thresholding as this tries to distinguish between two classes of pixels/items.

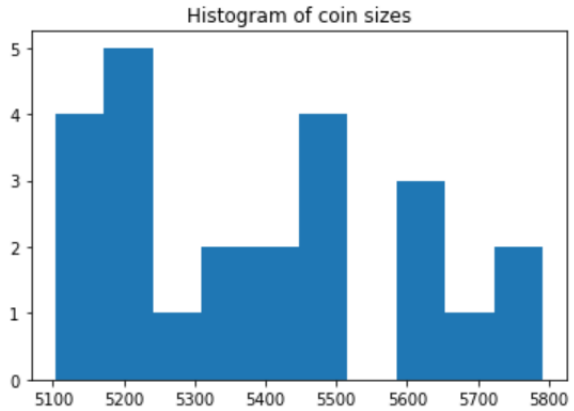
1.5 Segmenting the coins

To properly separate and segment the coins I used watershed segmentation. First we transform the binary image provided by the thresholding into a gradient map. Next we apply watershed segmentation using the gradient map and markers created by the local peak function. In the process one of the coins erroneously was assigned two peaks. With manual editing the second peak was removed and we can see that the segmentation now works smoothly (the blue coin is fully blue).



1.6 Bounding boxes and number of coins

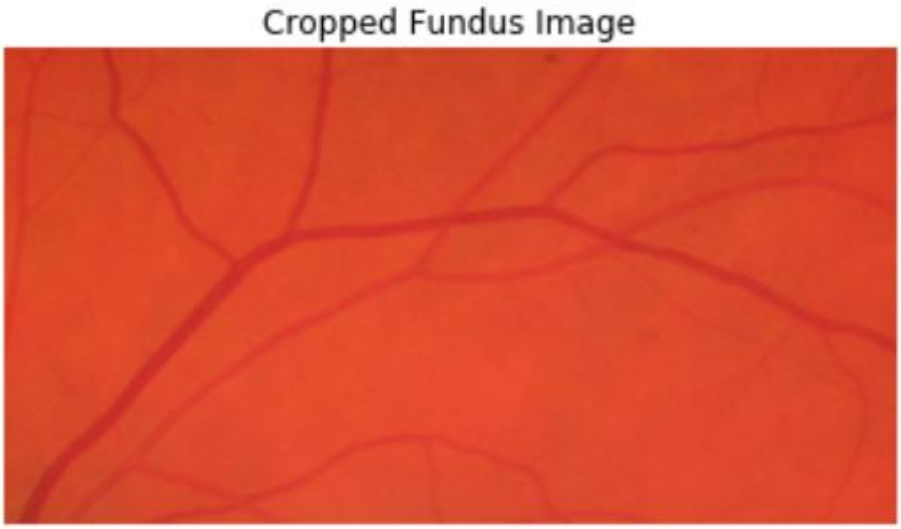
Region props was applied to the watershed segmentation and 24 connected components were found.



The sizes of the coins are all fairly similar. Some are larger than expected and this can be attributed to the shadows of the coins being misidentified as part of the coin. The size of coins that touch other coins also depends on the quality of the watershed segmentation.

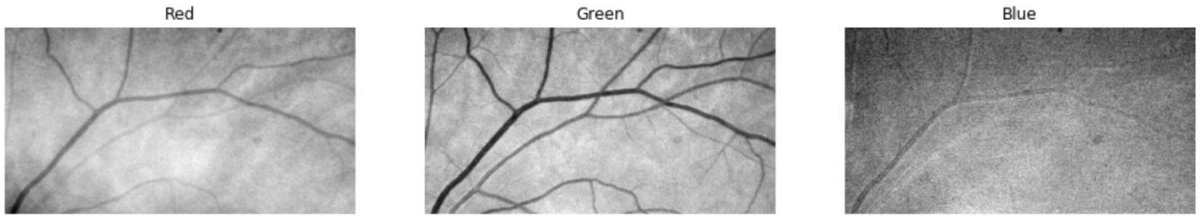
2.1 Extracting vessels from raw fundus images

2.1.1 Load and display image



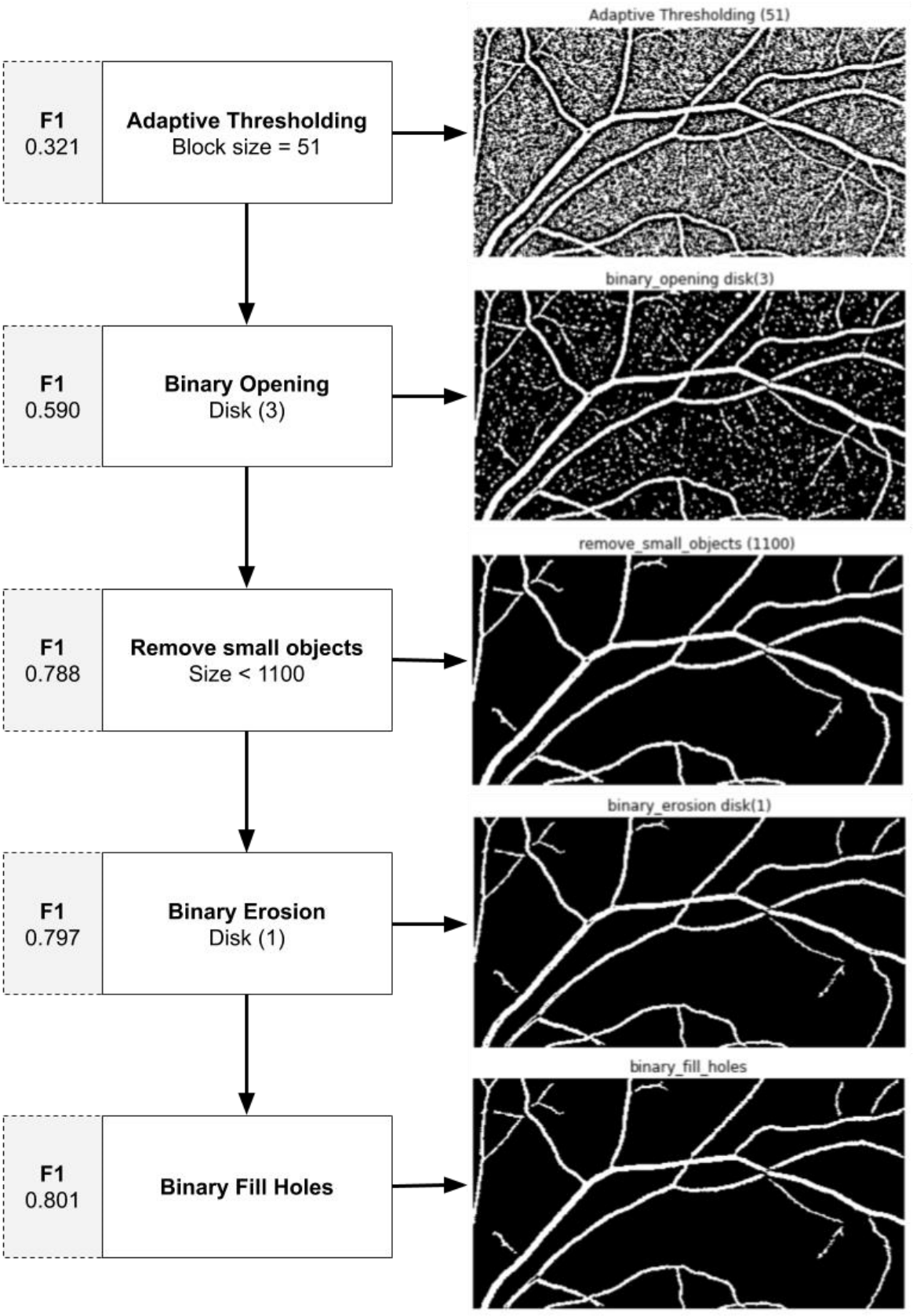
2.1.2 Segmenting the Fundus Image

My first step was to identify which color channel provided the most information. Below the channels are displayed. The green channel was chosen as it has the highest contrast between blue vessels and background pixels.



After experimenting with a couple of methods I discovered that a lot of methods do more harm than good. The laplacian filter added more noise to the image. Top hat filtering made the gradient of the noise rather small, which caused problems when thresholding as the noise now connected up easily and created one large mesh in which the little veins were lost. Blurring the image also resulted in a loss of information that was not recoverable. In the end what worked best was the simple adaptive thresholding with morphological operations. Each parameter in all of the steps were tested to find the optimum value which results in the following results. The processing pipeline is shown below.

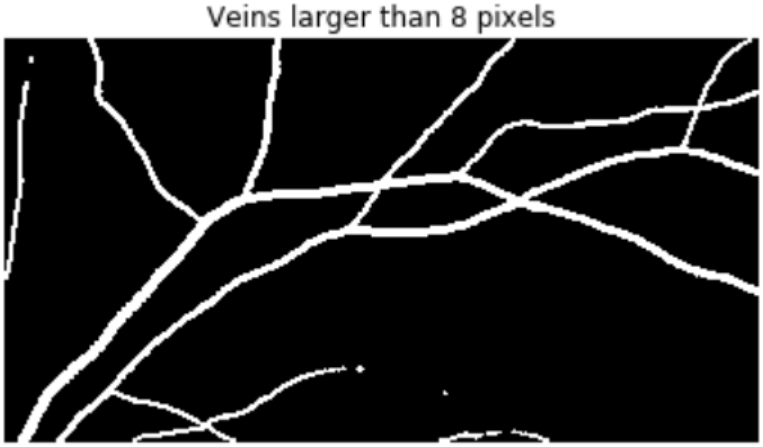
Jaccard Similarity Score	Precision	Recall	F1
0.961	0.802	0.800	0.801


F1
0.321
Adaptive Thresholding
Block size = 51
Adaptive Thresholding (51)
F1
0.590
Binary Opening
Disk (3)
binary_opening disk(3)
F1
0.788
Remove small objects
Size < 1100
remove_small_objects (1100)
F1
0.797
Binary Erosion
Disk (1)
binary_erosion disk(1)
F1
0.801
Binary Fill Holes
binary_fill_holes

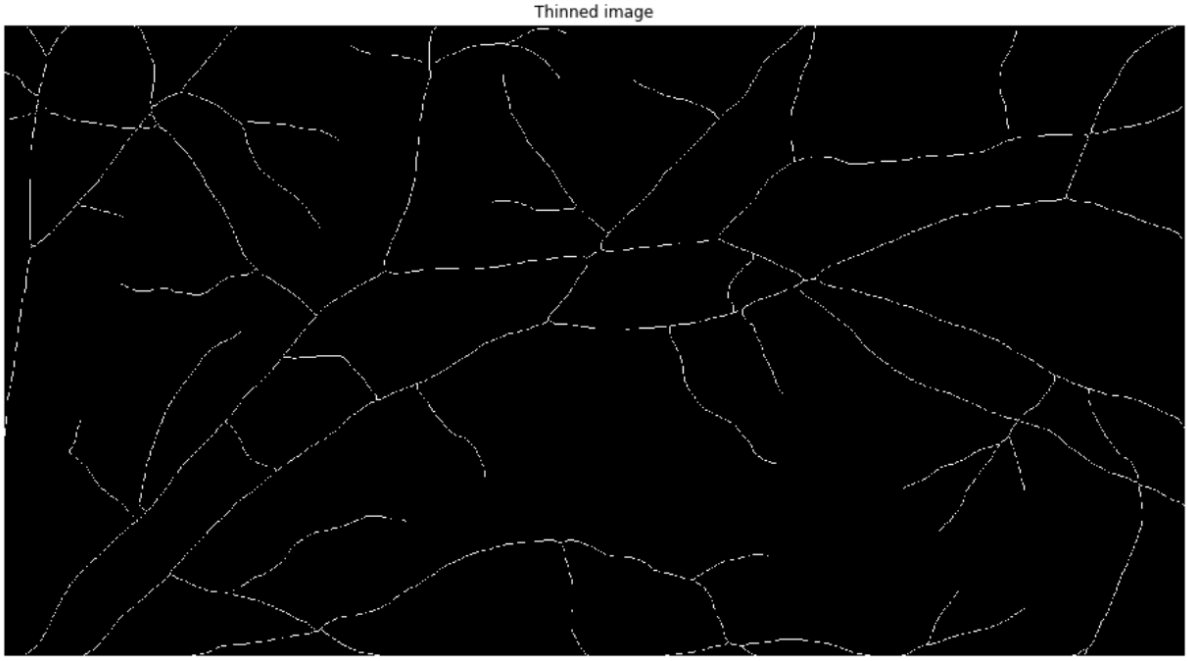
2.2 Processing a binary image of segmented vessels

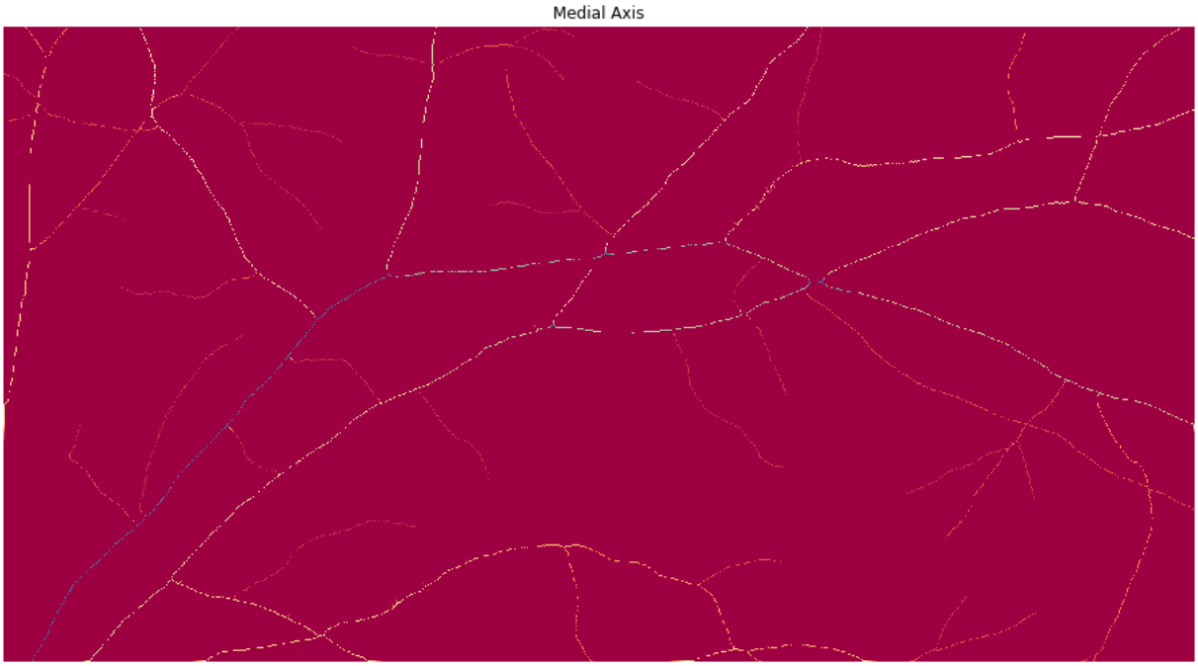
2.2.1 Veins larger than 8 pixels

The veins smaller than 8 pixels in diameter were removed using binary opening using a disk of radius 4. A disk was used as opposed to a square as the disk is orientation neutral.



2.2.2 Thinning and medial axis transforms





The maximum vein width is 12.37 pixels.

2.2.3 Length and Orientation

The length of the vascular network is 12,016 pixels. This was calculated by summing all of the values in the thinned image. The value slightly differs for the skeletonised version (12,360). This makes sense as the skeleton captures the true size of the object it is representing. The orientation of the vascular network can be seen below. The hough space of the thinned vascular network can be seen next to it. We can see the largest values (green) are in the ranges of 60-80 and -80 to -60, which matches up with the polar plot.