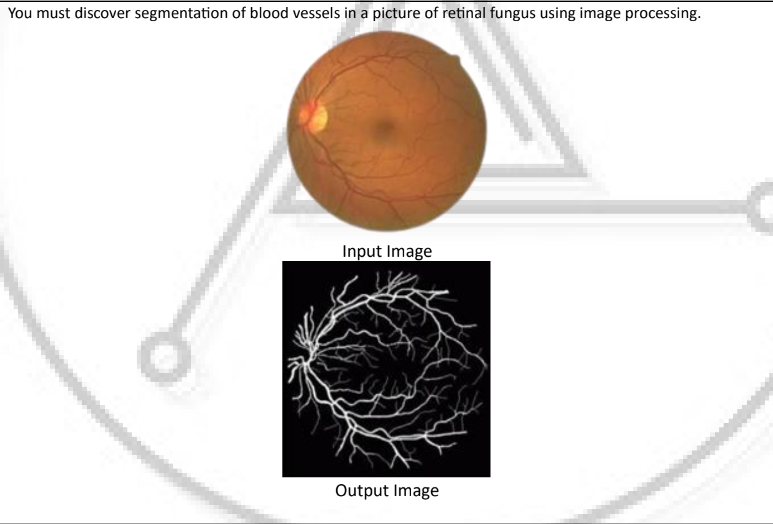
**Team Members: -**

Amait Vikram Bhardwaj

Krish Sheth

Jayant Lahoti

**Solution to the problem statement: -**



**Steps undertaken to solve the question: -**

*Step-1: -*

Reading the required image - We do this by using the imread function and specify the location of the image, in this case it is Test Images. We then store this image by assigning a variable to it. In this case, it is Img1

*Step-2: -*

Since the image that we use is relatively large in size, Therefore we resize the image using imresize function by specifying the desired dimensions of the resultant image and store it by assigning a variable to it, which here is Img2.

*Step-3: -*

Since, in the previous step the specified size cannot be in decimals, we lose a significant amount of precision. Hence, we convert the image into double datatype. We then store it in the variable named Img3

*Step-4: -*

We now convert the gray level of the previous image which was rgb to cielab gray level using the rgb2lab function. We do this because the RGB color space is device-dependent, meaning that the same RGB values can appear differently on different devices (e.g. a computer screen versus a printer). The CIELAB color space, on the other hand, is device-independent, meaning that colors will appear the same regardless of the device they are viewed on. Where l: - lightness of the colour, a: - green to red starting from negative to positive, b: - blue to yellow, negative to positive.

*Step-5: -*

We now use concatenate function on the cielab image using cat function where 3 means the three channels in the previous image and we concatenate those channels into 1 array and store it into fill variable

*Step-6: -*

Bsxfun function is used to perform element wise binary operations between fill and img4 and specify the function @times which is element wise binary multiplication where only the red channel will be the prominent channel and store it in variable img5

*Step-7: -*

We now reshape img5 and use 3 which is the existing dimension of the img5.

*Step-8: -*

We now apply principal component analysis using the pca function. It will return the coefficient and score of the principle component and we store it in [c,s] and now we reshape the score based on the size of the lab image and store this new value in the S variable

*Step-9: -*

We need all columns and all rows of the first channel hence (:,:,1) and now we convert this s into grayscale image for which we use the formula: -

S-min(S)/max(S)-min(S) where the division will be element wise hence the dot and store the resultant image in the variable grayimg.

*Step-10: -*

We now enhance the contrast of the gray image using adaptive histogram equalization. Since we are performing the enhancement block by block, we write numtiles and specify the size of the tile to be 8x8. Similarly we specify the number of bins to be 128. This will return the enhanced image.

*Step-11: -*

Now we need to apply average filter. For this we use the fspecial function where we specify the name of the filter and the size of the filter. We store it in variable filteravg. We apply this filter using imfilter specifying the filter used and the image it is applied on and show the result using the imshow function and assign it a title. We now use imsubtract to subtract enhanced image from filtered image.

*Step-12: -*

We calculate the threshold value by creating its function in a separate file which will be called by its specified name. Then we apply it by calling it on subtracted image.

*Step-13: -*

After the threshold has been calculated on the subtracted image, we use this value to convert subtracted image into binary image using the im2bw function and show the resultant image using imshow function.

Step-14: -

To reduce the noise from the binary image, we perform opening operation using the bwareaopen on values smaller than 100 pixels and give this resultant image a title, say clean image.

Primary Matlab Code: -

Img1 = imread('Test Images/123.jpg');

Img2 = imresize(Img1, [584 565]);

Img3 = im2double(Img2);

Img4 = rgb2lab(Img3);

fill = cat(3, 1,0,0);

Img5 = bsxfun(@times, fill, Img4);

Reshaped\_Img4 = reshape(Img5, [], 3);

[C, S] = pca(Reshaped\_Img4);

S = reshape(S, size(Img4));

S = S(:, :, 1);

Grayimg = (S-min(S(:)))./(max(S(:))-min(S(:)));

Imgenhance = adapthisteq(Grayimg, 'numTiles', [8 8], 'nBins', 128);

Filteravg = fspecial('average', [9 9]);

Filtercoffee = imfilter(Imgenhance, Filteravg);

figure, imshow(Filtercoffee)

title('Filtered Image')

Substracted\_Image = imsubtract(Filtercoffee,Imgenhance);

level = Threshold\_Level(Substracted\_Image);

Binary\_Image = im2bw(Substracted\_Image, level-0.008);

figure, imshow(Binary\_Image)

title('Binary Image')

Clean\_Image = bwareaopen(Binary\_Image, 100);

figure, imshow(Clean\_Image)

title('Clean Image')

Function to calculate threshold value: -

function level = Threshold\_Level(Image)

Image = im2uint8(Image(:));

[Histogram\_Count,Bin\_Number]=imhist(Image);

i = 1;

Cumulative\_Sum = cumsum(Histogram\_Count);

T(i) = (sum(Bin\_Number.\*Histogram\_Count))/Cumulative\_Sum(end);

T(i)=round(T(i));

Cumulative\_Sum\_2 = cumsum(Histogram\_Count(1:T(i)));

MBT=sum(Bin\_Number(1:T(i)).\*Histogram\_Count(1:T(i)))/Cumulative\_Sum\_2(end);

Cumulative\_Sum\_3=cumsum(Histogram\_Count(T(i):end));

MAT=sum(Bin\_Number(T(i):end).\*Histogram\_Count(T(i):end))/Cumulative\_Sum\_3(end);

i = i+1;

T(i) = round((MAT+MBT)/2);

while abs(T(i)-T(i-1))>=1

Cumulative\_Sum\_2 = cumsum(Histogram\_Count(1:T(i)));

MBT=sum(Bin\_Number(1:T(i)).\*Histogram\_Count(1:T(i)))/Cumulative\_Sum\_2(end);

Cumulative\_Sum\_3=cumsum(Histogram\_Count(T(i):end));

MAT=sum(Bin\_Number(T(i):end).\*Histogram\_Count(T(i):end))/Cumulative\_Sum\_3(end);

i = i+1;

T(i) = round((MAT+MBT)/2);

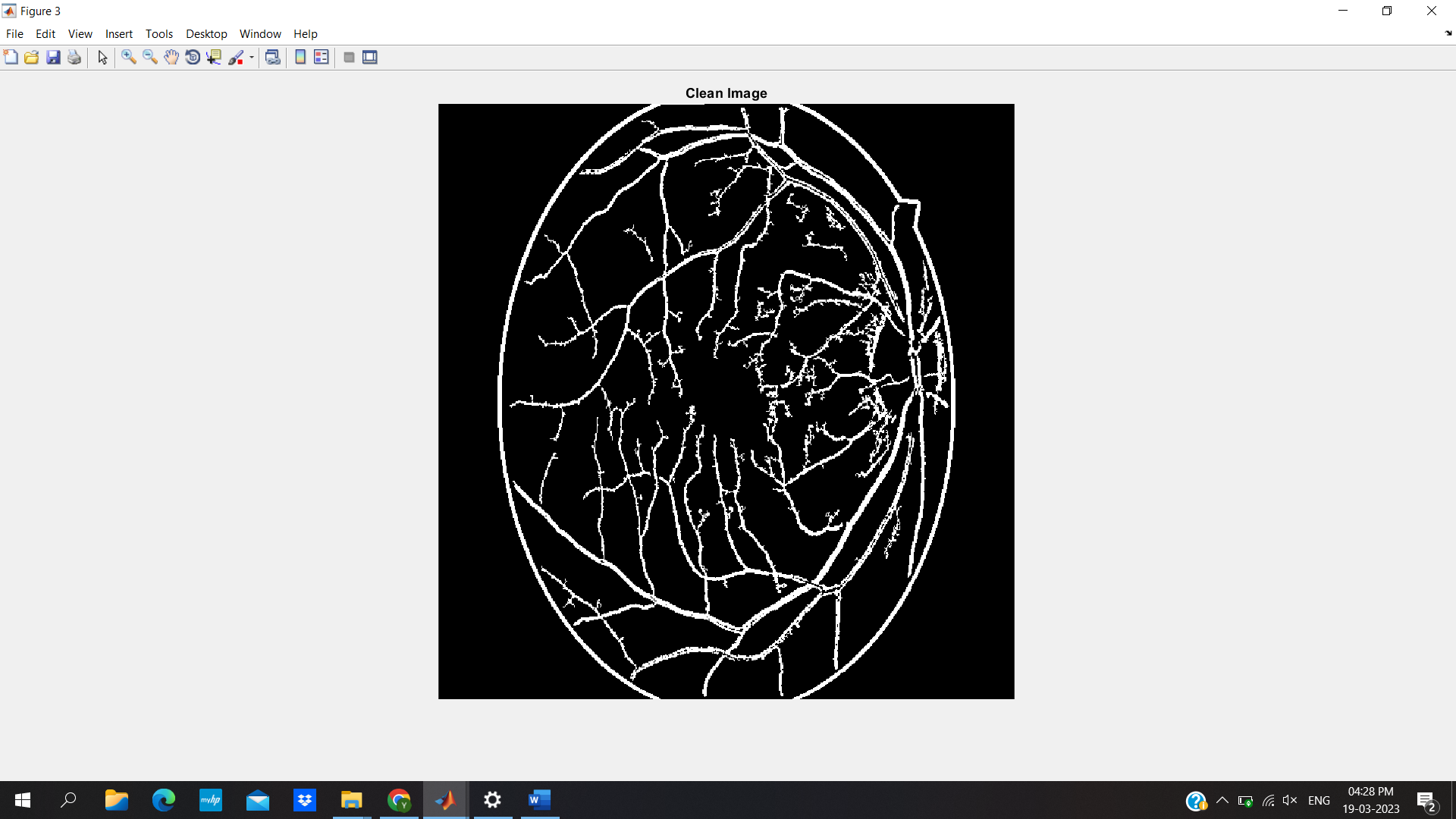
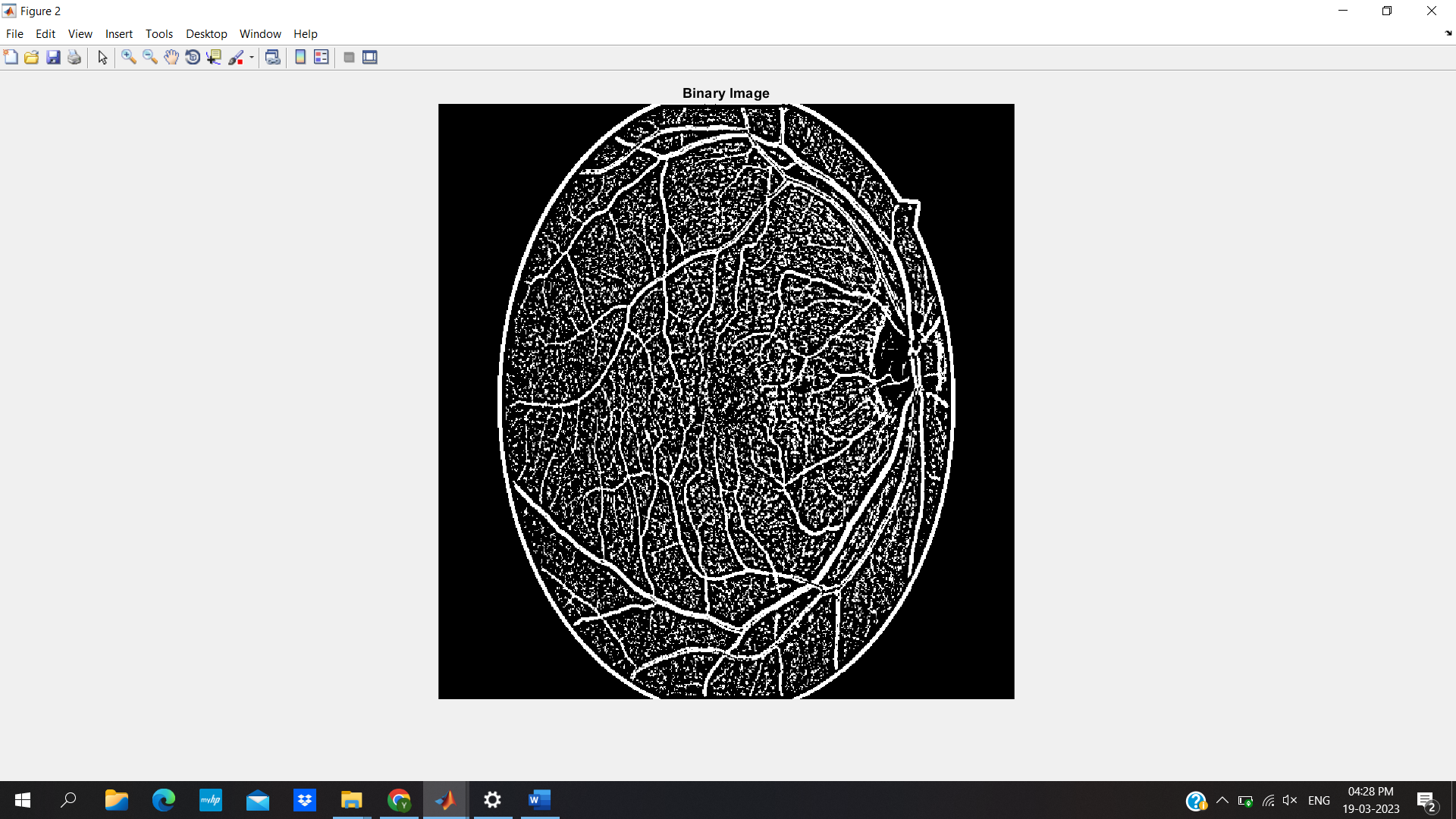
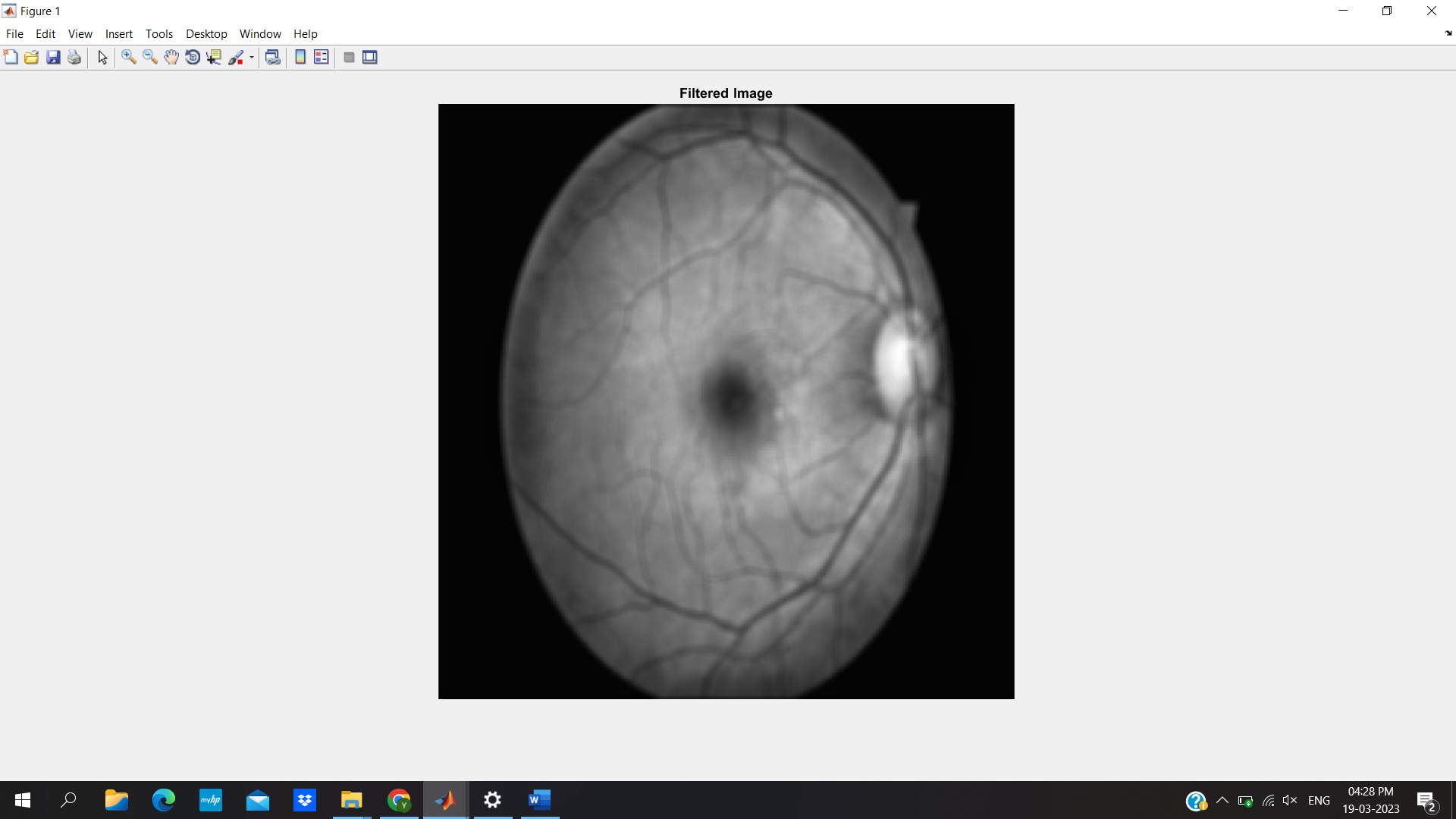
Threshold = T(i);

end

level = (Threshold - 1) / (Bin\_Number(end) - 1);

end

Screenshots: -



For Secondary Image: -

