Monitoring worms (Report)

Coursework - 1

Name: Sonia Mubasher

Student ID: 20129528

Word count: 1094 words

Table of Contents

1. Introduction

The coursework is about monitoring worm movement by processing a pipeline that converts a color image to binary image labelling different materials such as size, surface area of the worm, extracting the worm body and detecting edge etc. So, we were required to write a MATLAB code to perform this task of which the description of the main features which I implemented are as follows.

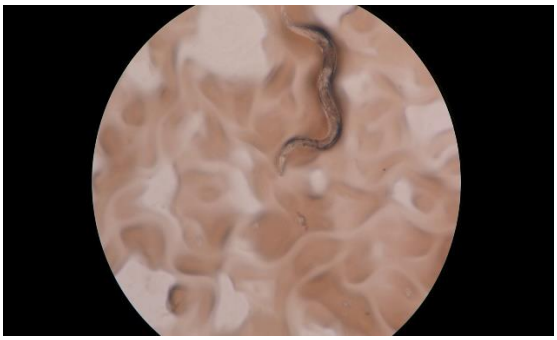
2. Description of the key features of implementation

2.1 Task

Our task was to detect and extract the worm's body, extract its boundaries, and measure its size or surface area.

2.2 Given

We were given a small set of color images as shown below



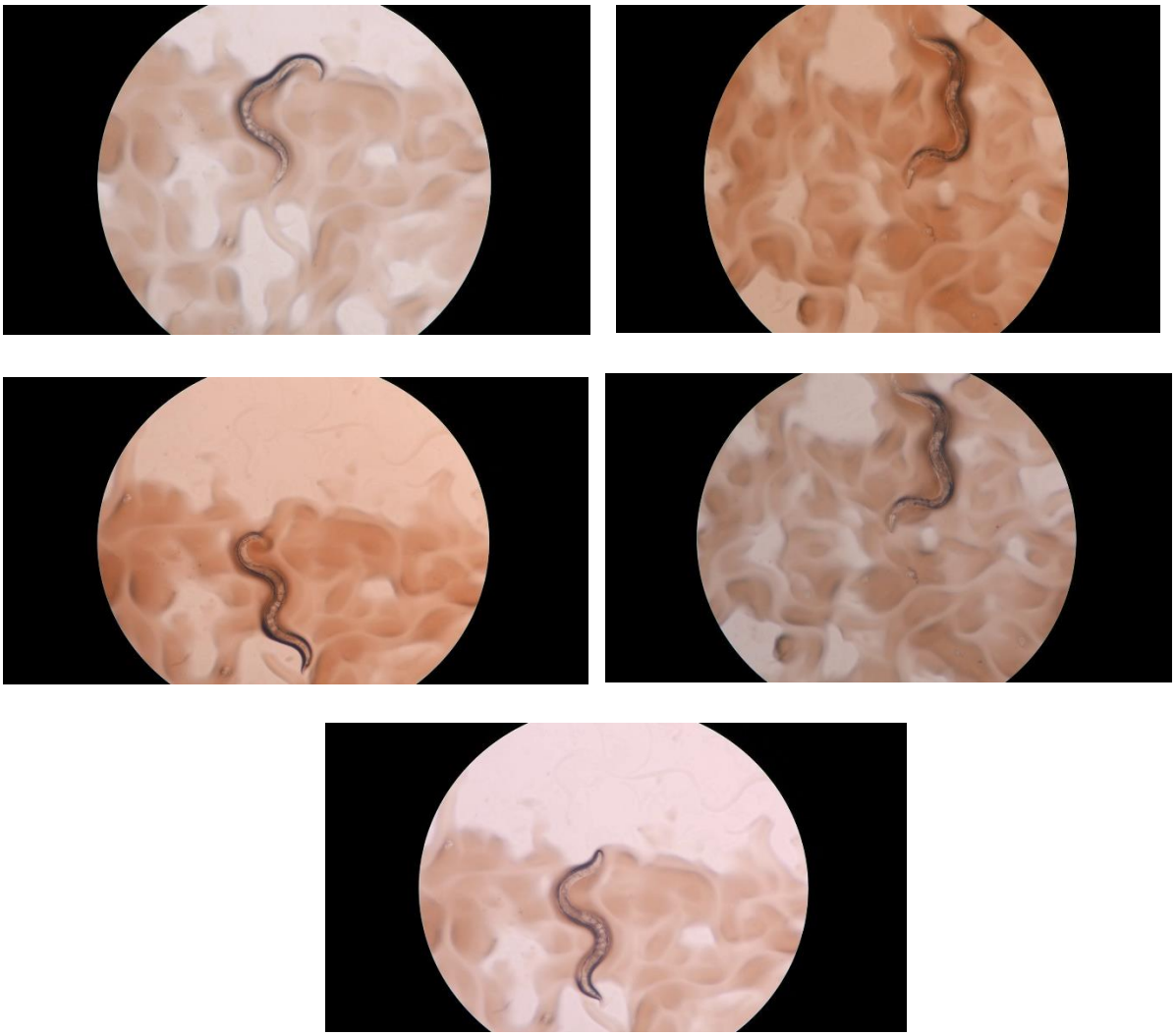


Figure 1: Original color images

As you can see from the above given figures, they differ greatly in noise from each other and we were required to propose a solution to extract the worm from these images with the same code.

2.3 Steps involved/Techniques used

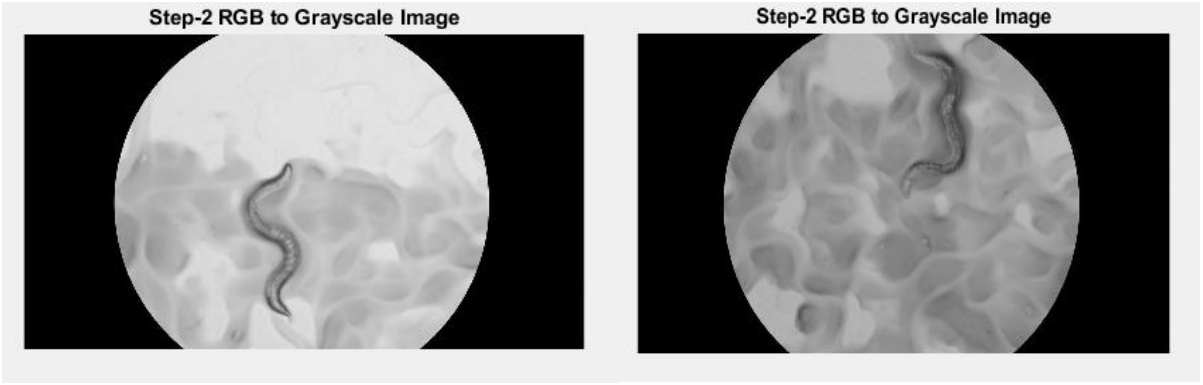
Following steps were involved to reach the required result.

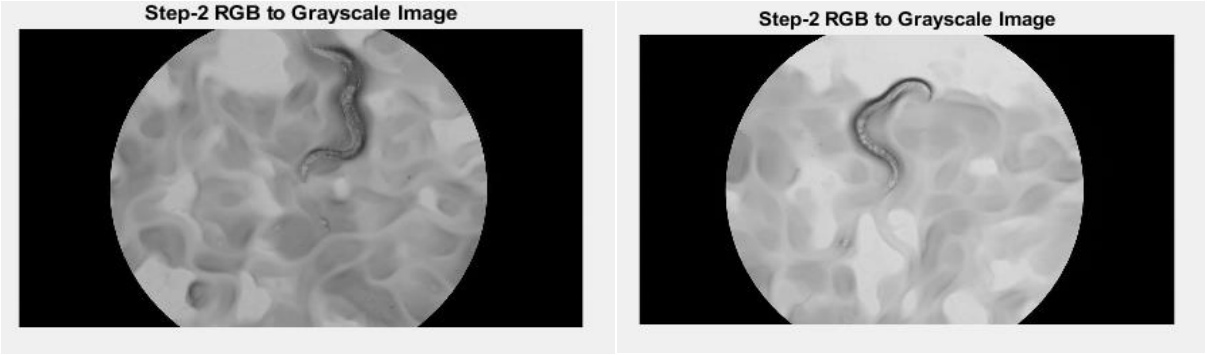
2.3.1 Read the images

I used For loop to read and then process all the images as the same code was supposed to work with all the images. I used imread function to read all the images. After reading all the images I decided to convert them to grayscale as it's much easier to work with grayscale images.

2.3.2 Convert RGB to Grayscale

I used rgb2gray function to convert the original color images to grayscale so, later I can work on them. Below are the samples of how the images looked like after converting them to grayscale.





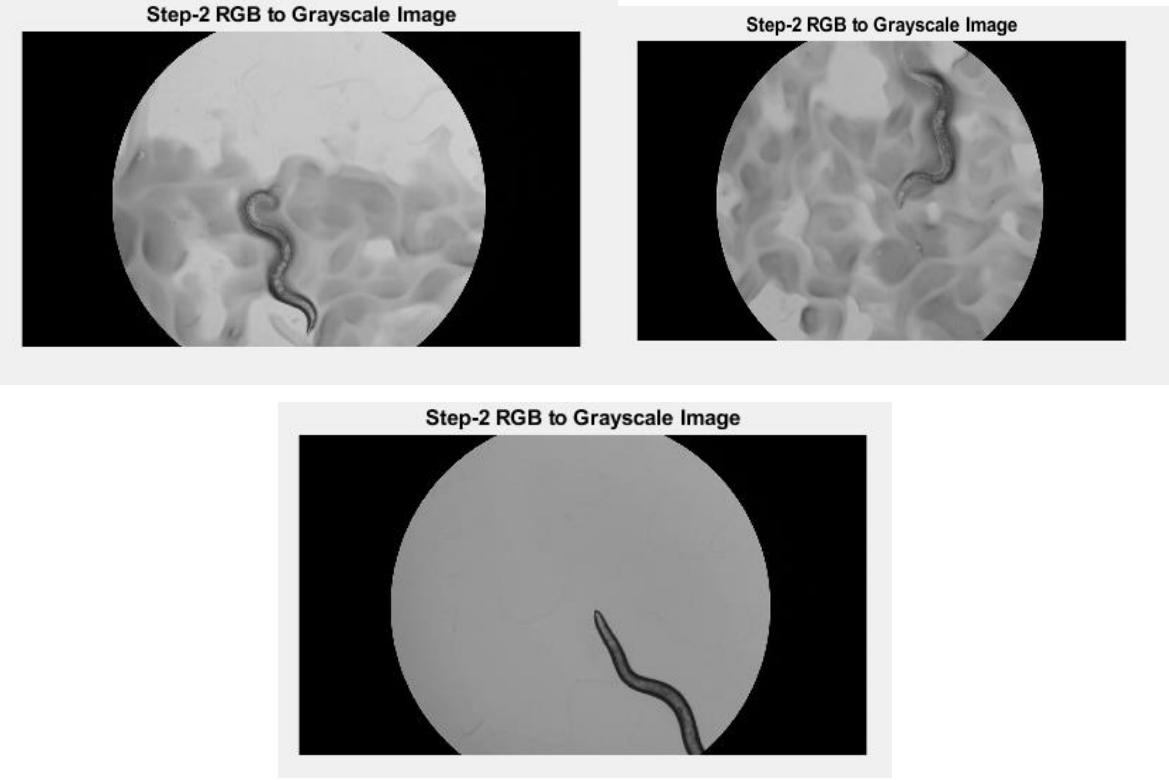


Figure 2: RGB to Grayscale

2.3.3 Adjust Contrast

After converting the images to grayscale, I realized that the worm was not so clear, and it will be hard to extract the worm from such kind of images, so I adjusted the contrast level of the above images to make the worm more visible. I used imadjust function to set the contrast level.

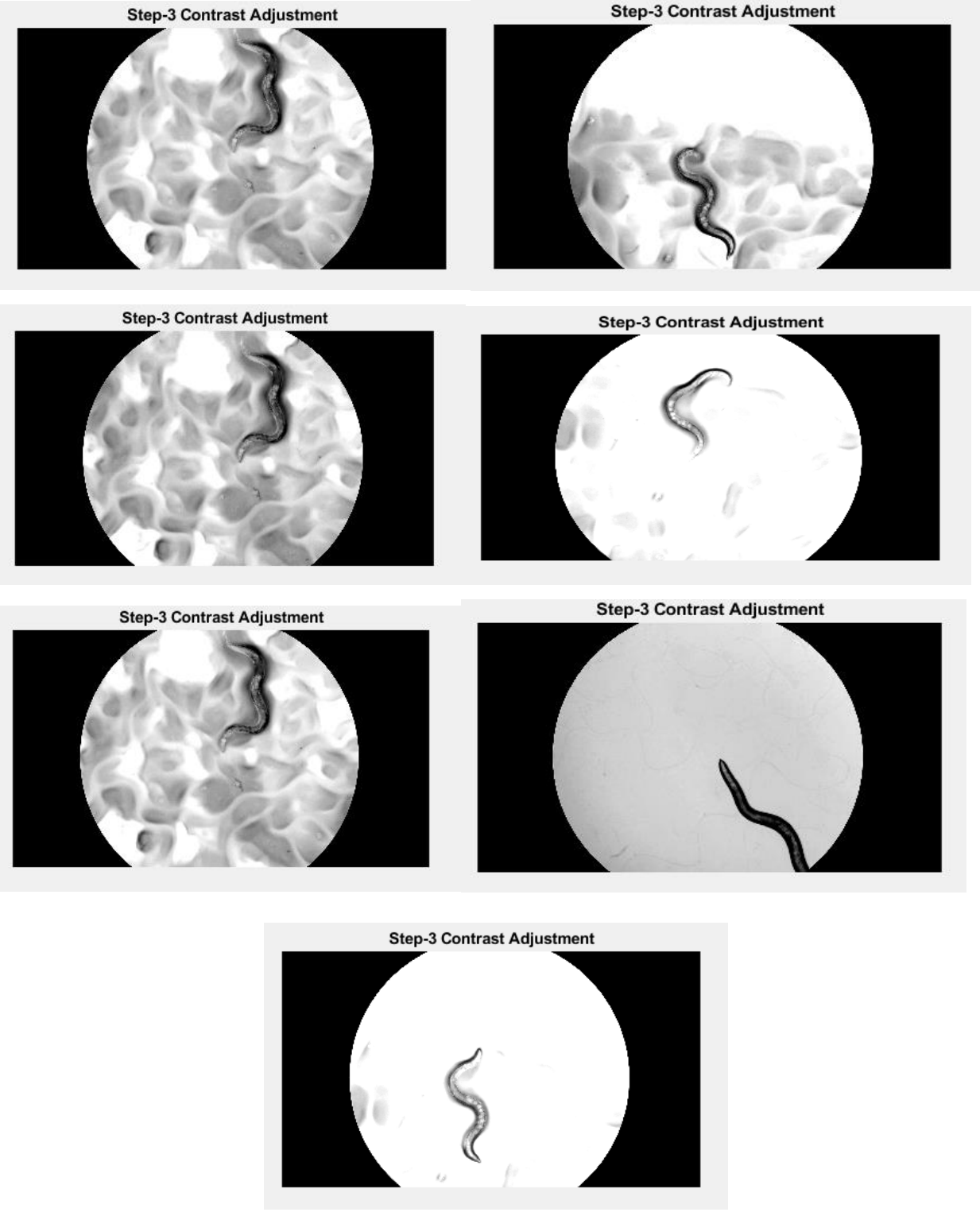
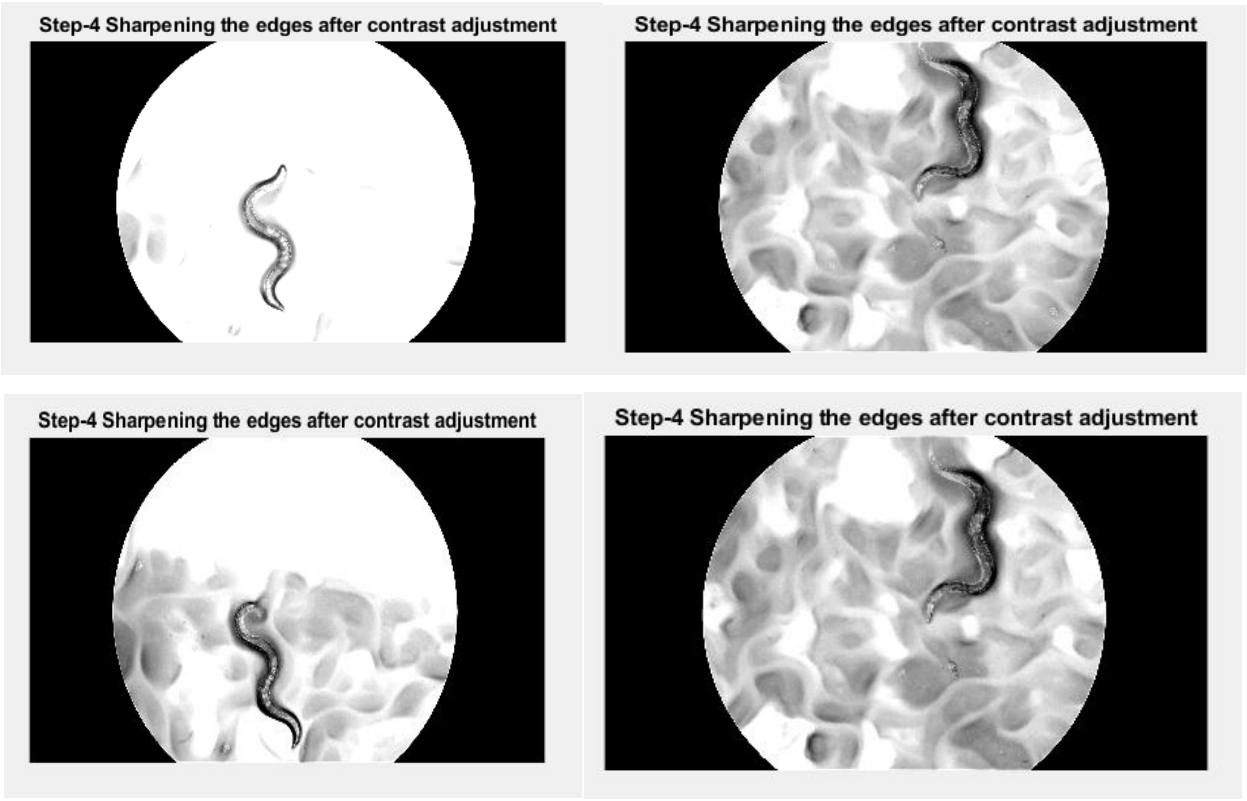


Figure 3: Contrast Adjustment

2.3.4 Sharpen the Image

After adjusting the contrast level, I sharpened the edges of the worm so it would be much easier for me to detect the worm without any noise. For this I blurred the above image by using gaussian filter and subtracted the blurred image from the image above. This gave me an Unsharp mask which I later added to the above image which resulted in sharpening the edges of the worm body.



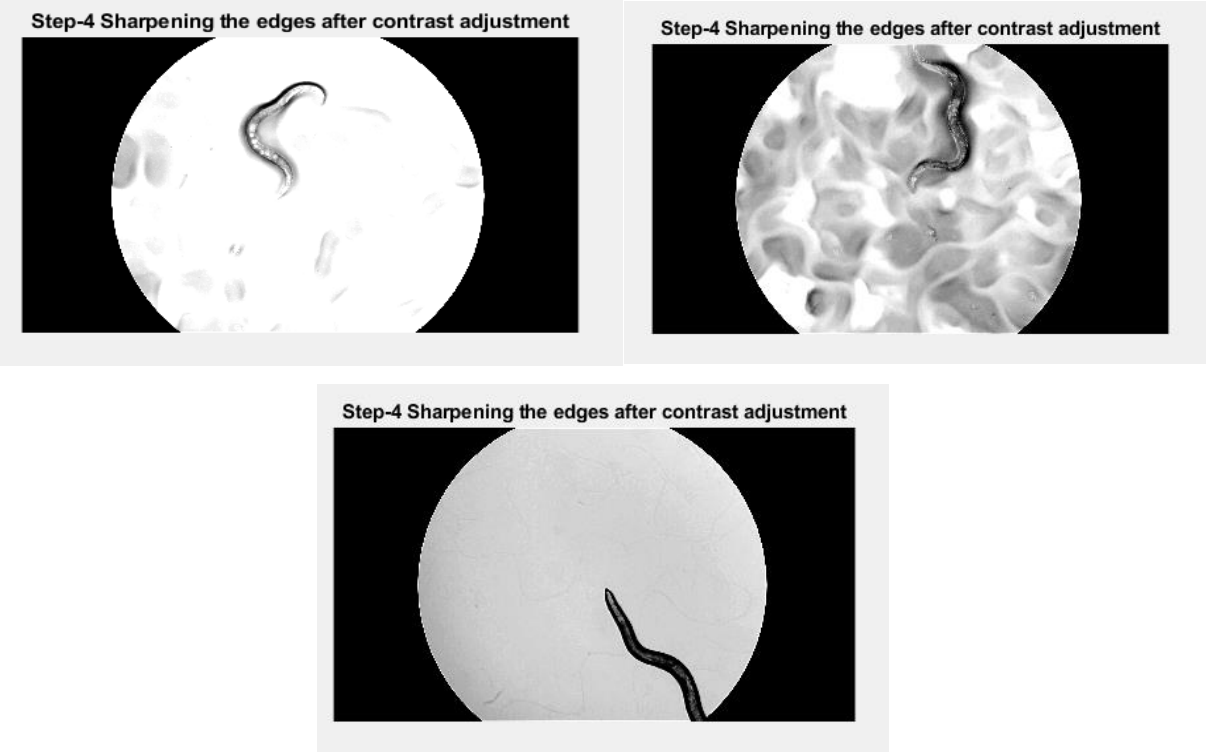


Figure 4: Sharpened images

2.3.5 Apply threshold and Generate Binary Gradient mask

Now the worm body to be segmented differs greatly in contrast from the background noise. Changes in contrast can be detected by operators that calculates the gradient of an image. To create a binary mask containing the segmented worm, I calculated the gradient image and applied threshold. I used edge and the Sobel operator to calculate the threshold value. Tuned the threshold value and used edge again to obtain a binary mask that contains the segmented worm body.

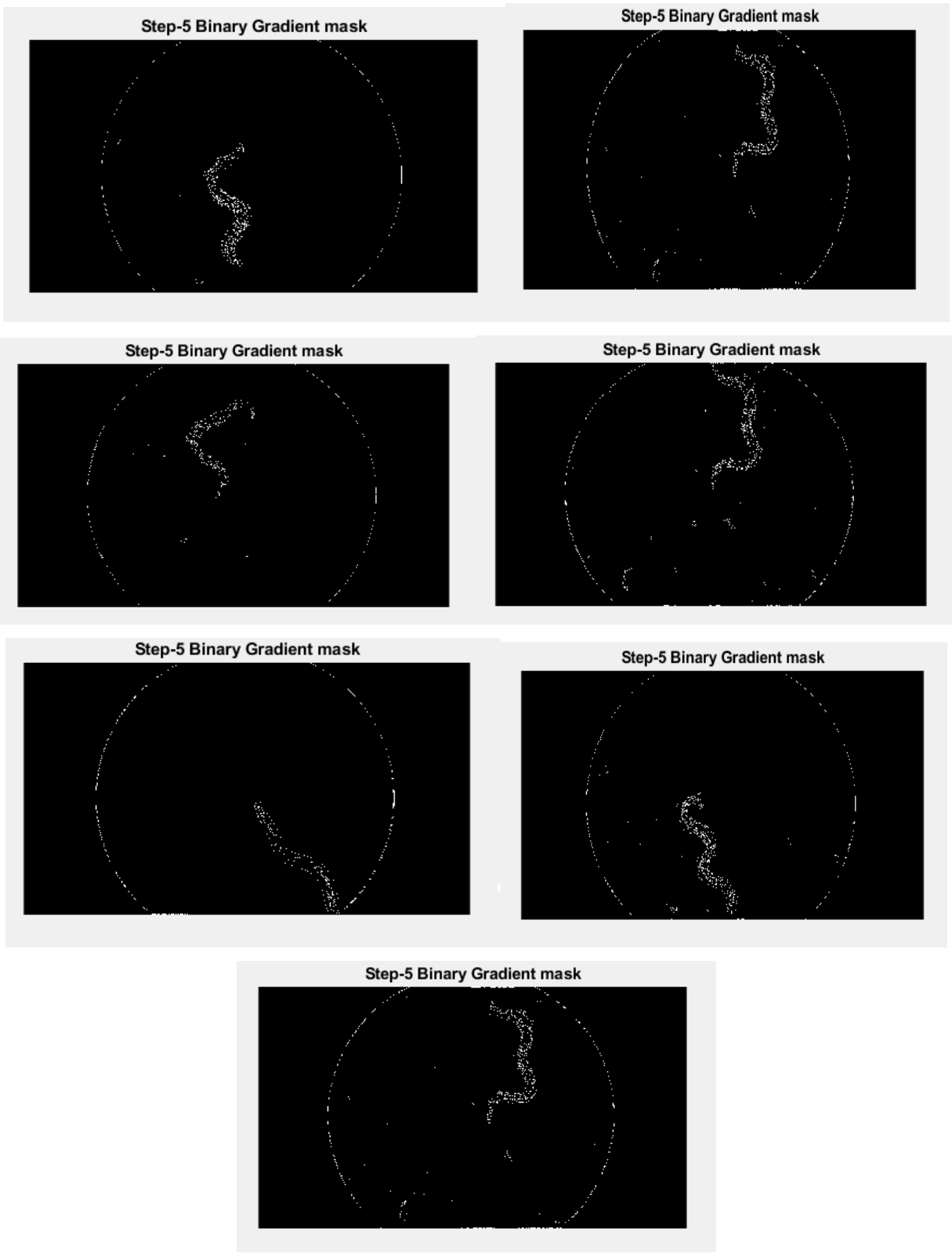
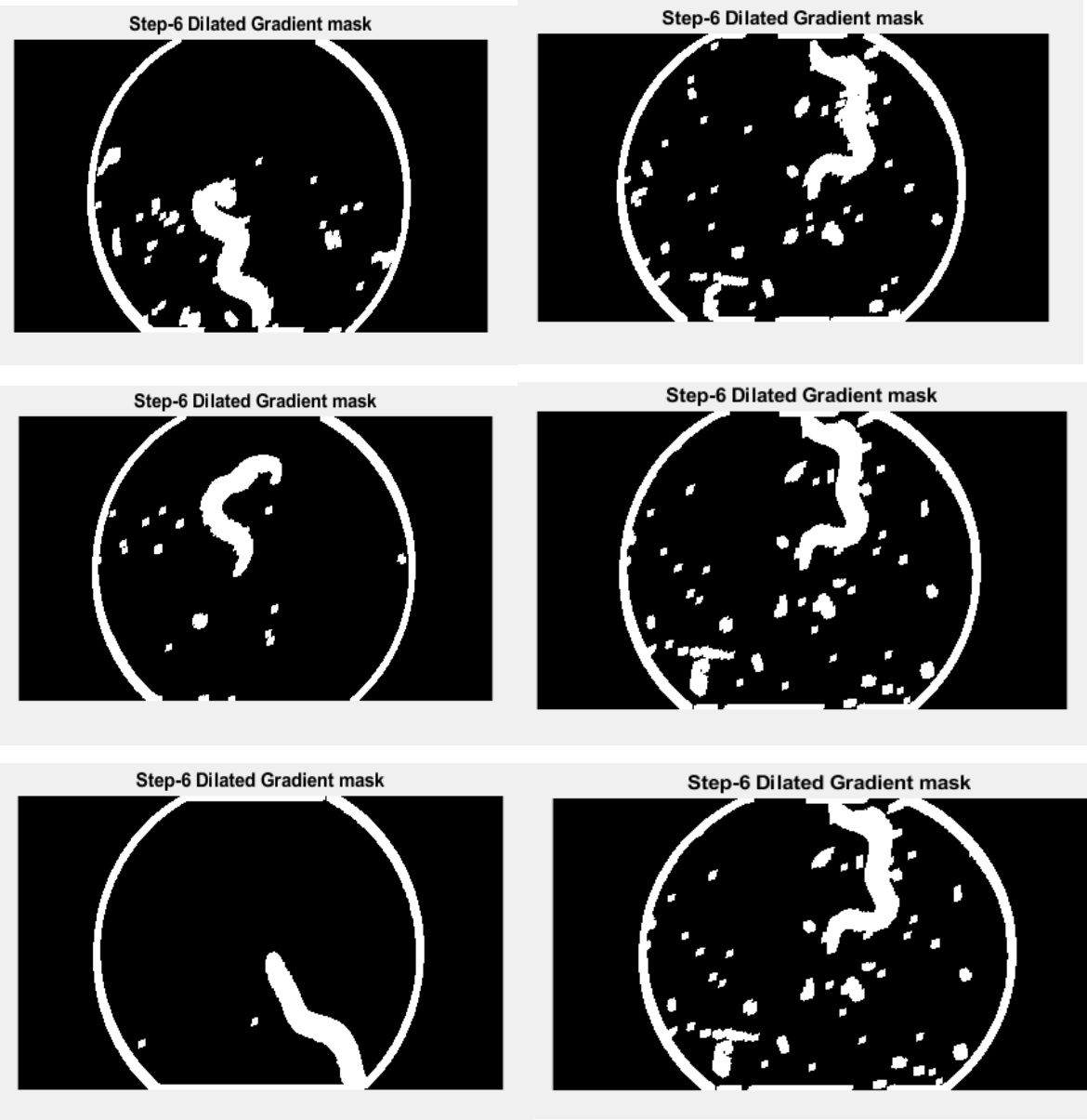


Figure 5: Binary gradient mask

2.3.6 Dilate the binary mask

The binary gradient mask shows dotted lines of high contrast in the image. These lines do not quite delineate the outline of the worm body as compared to the original image, these gaps will disappear if the Sobel image is dilated using linear structuring elements. So, I created two perpendicular linear structuring elements by using strel function and dilated the binary gradient mask using the vertical structuring element followed by the horizontal structuring element by using imdilate function.



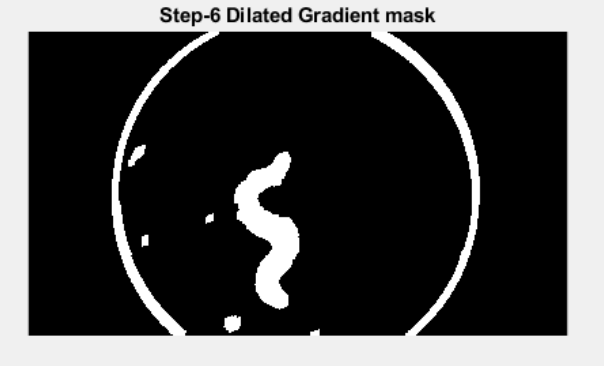
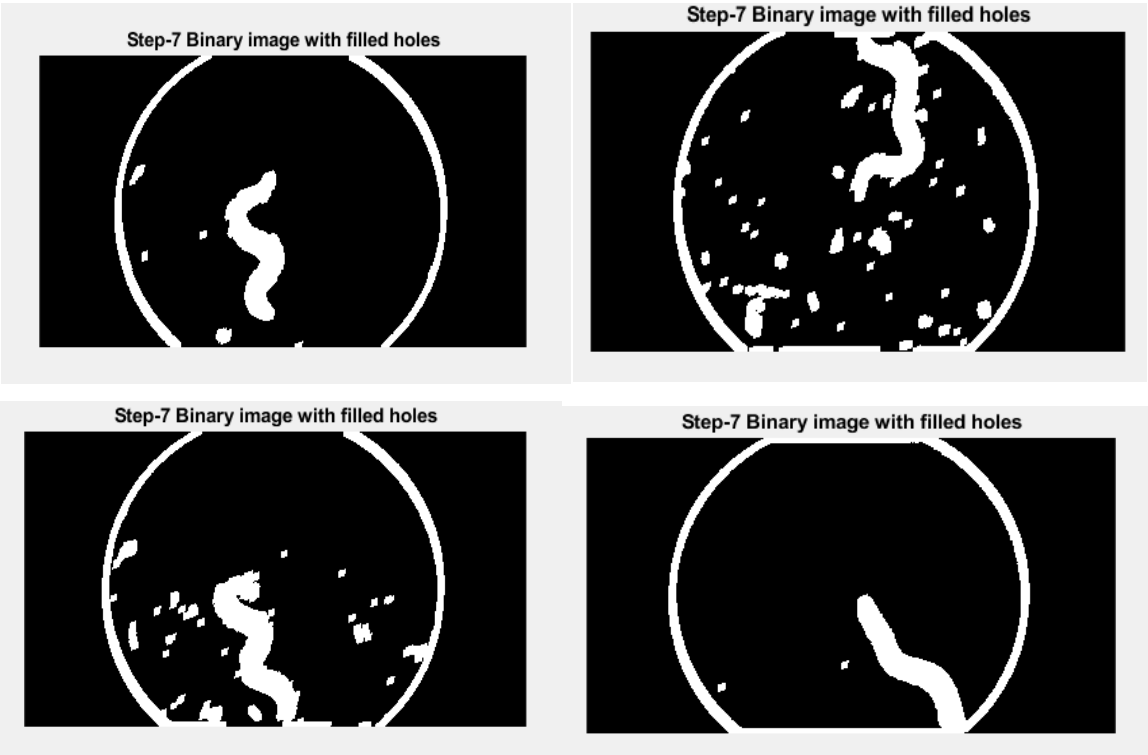


Figure 6: Dilated gradient mask

2.3.7 Fill in the holes if any

The dilated gradient mask shows the outline of the worm quite nicely, but there are still holes in the interior of the worm body. To fill these holes, I used the imfill function.



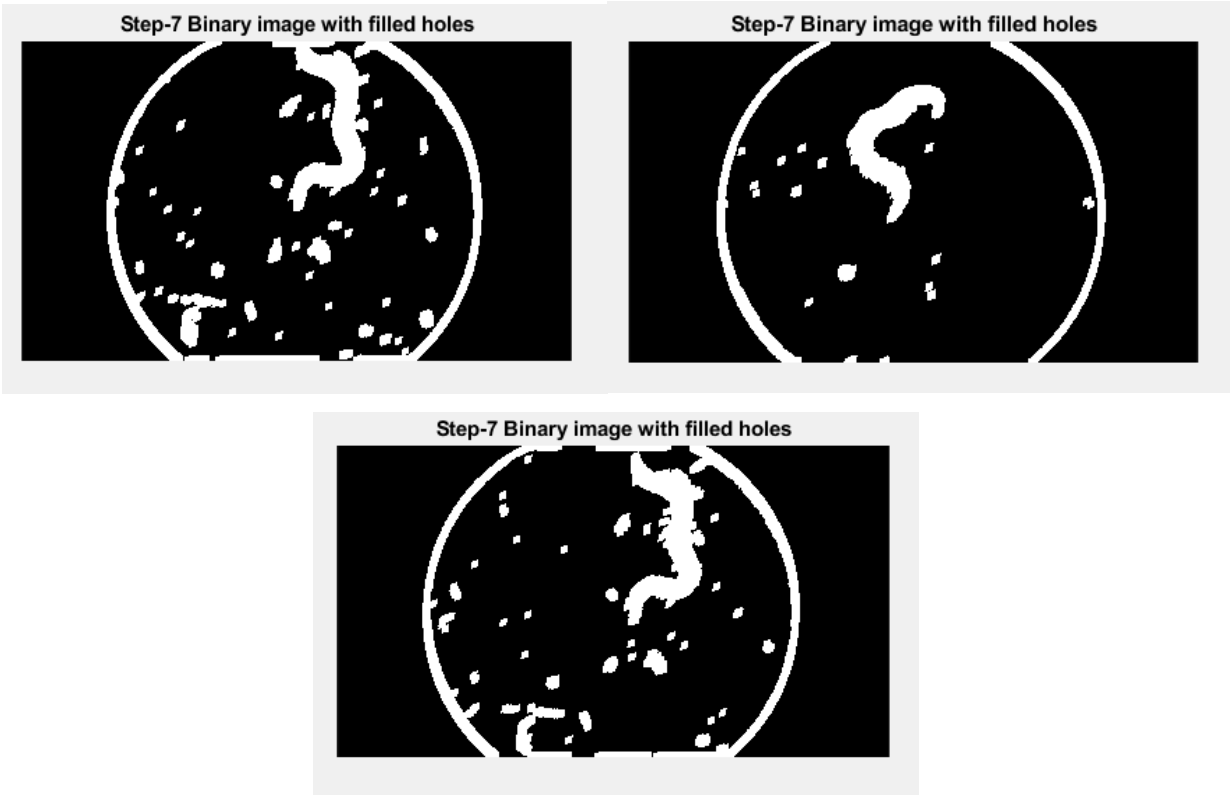
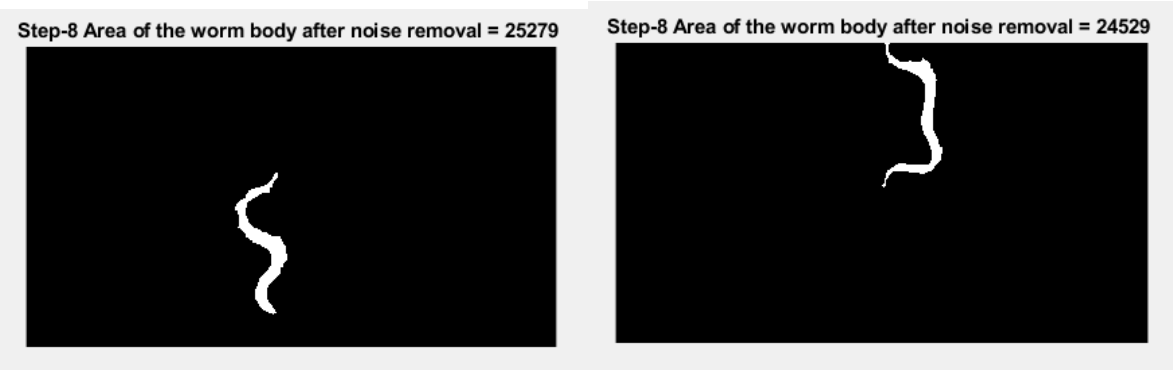


Figure 7: Binary image with filled holes

2.3.8 Further cleanup/smooth the image

Finally, in order to make the segmented worm body look natural, I cleared the image (removed noise from the image) by eroding the image with square and disk structuring element. I created the two structuring elements using the strel function and erode the image using imerode function. Still there was noise in the image so to further cleanup I used bwareaopen function to remove all the other objects from the image and leaving just the worm body.



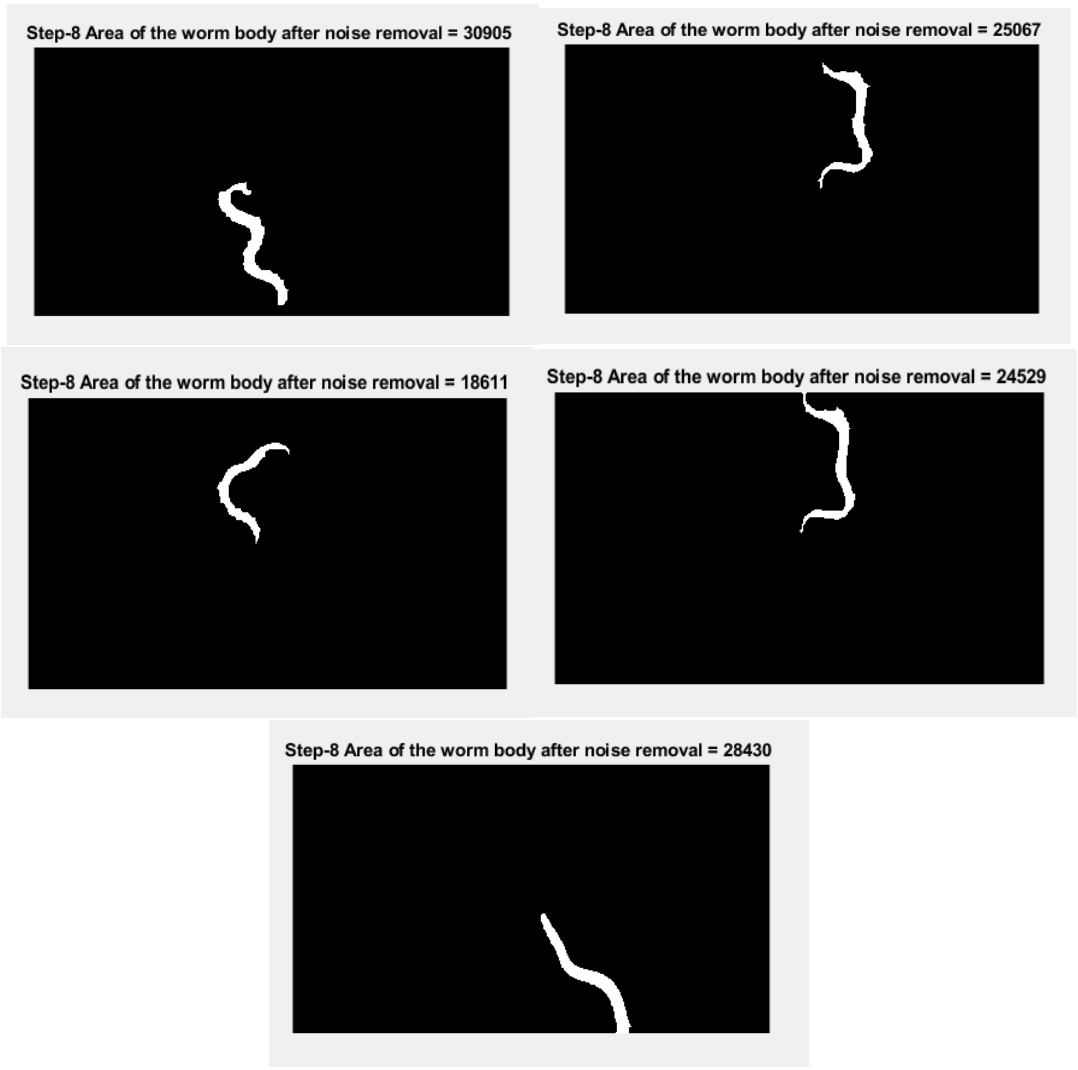


Figure 8: Binary mask of the worm

2.3.9 Visualize the segmentation

After that I used the labeloverlay function to display the mask over the original image.

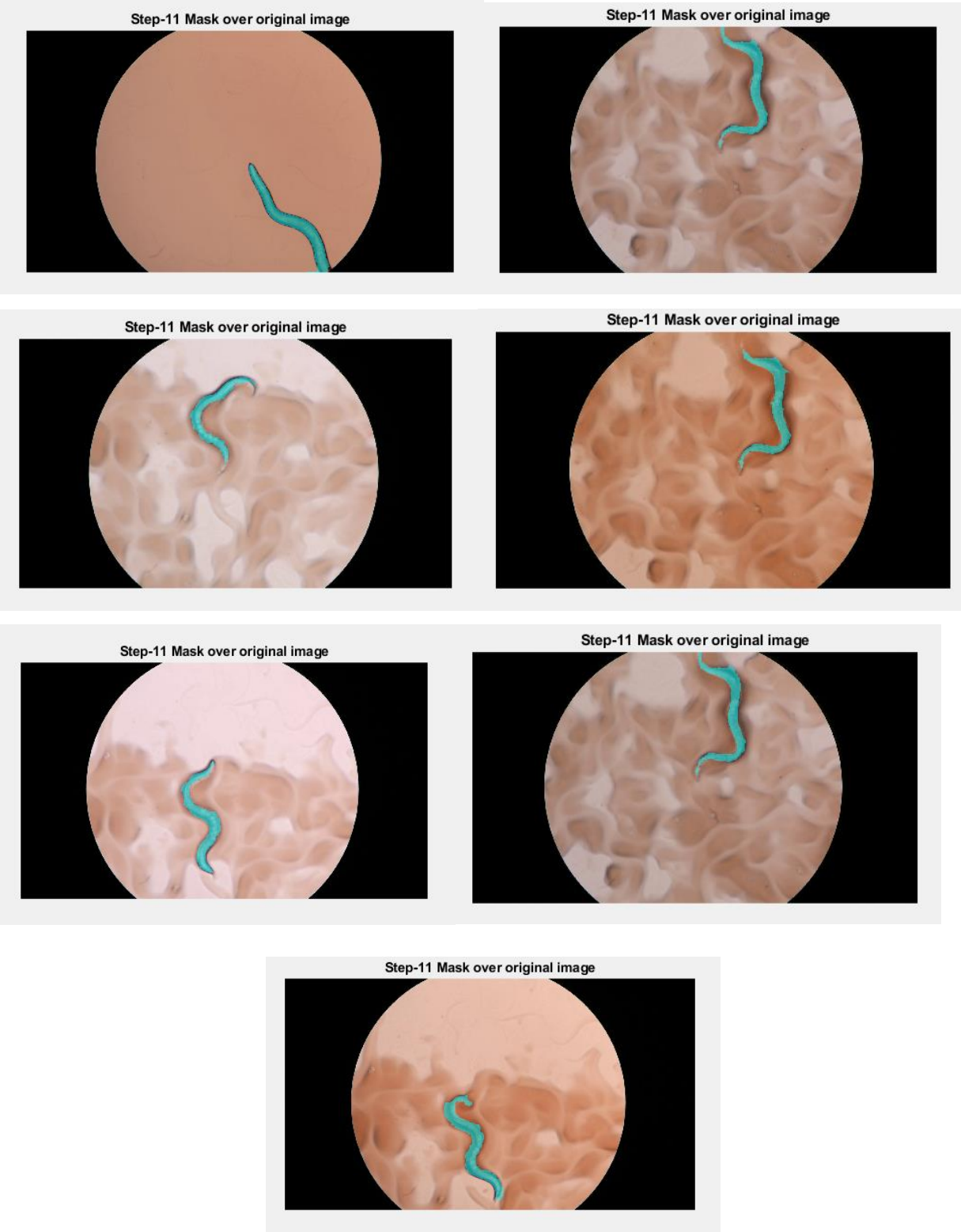
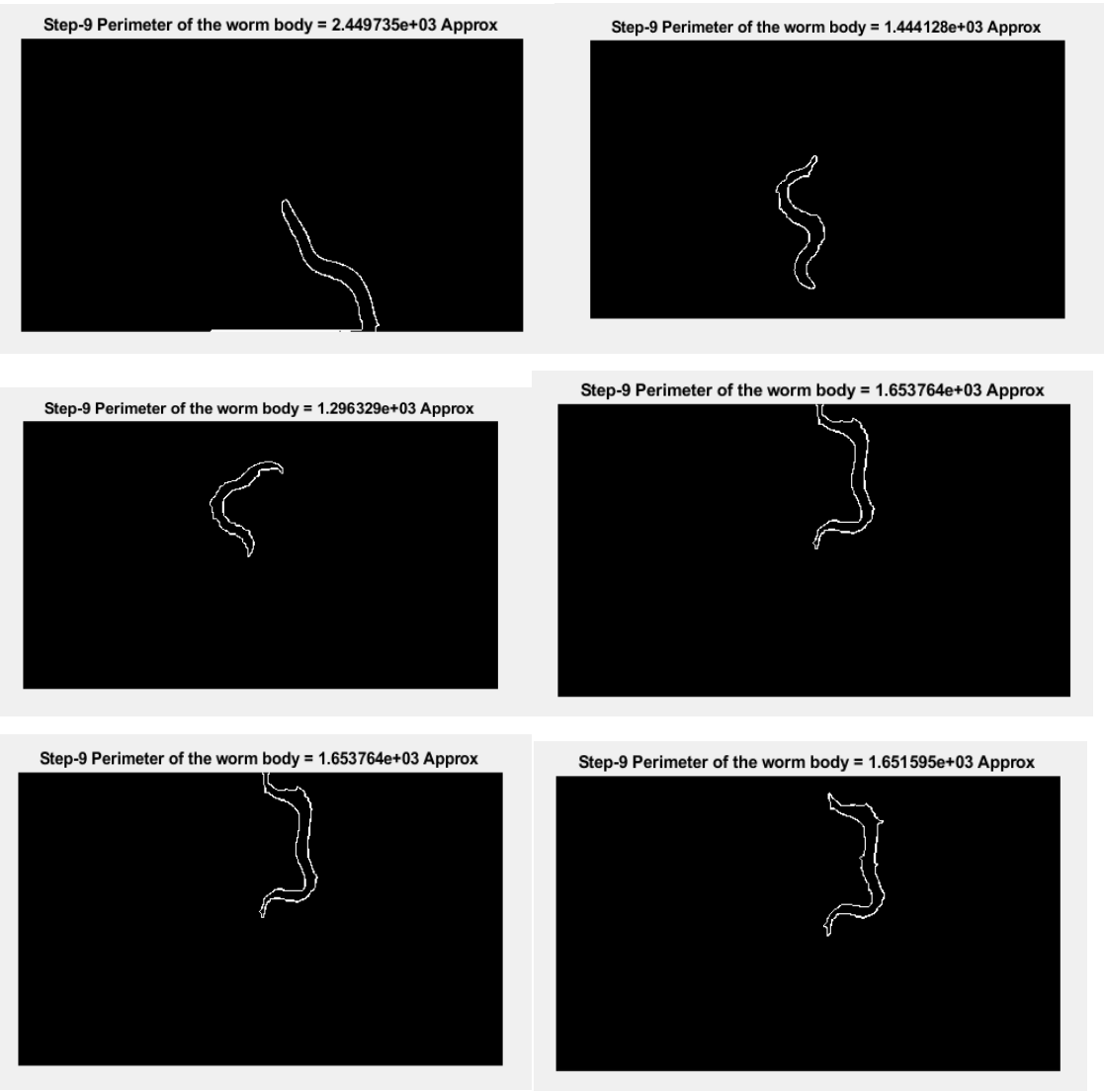


Figure 9: Mask over original image

2.3.10 Extract boundary of the worm

To extract the boundary of the worm I dilated the above developed binary mask of the worm with two structuring elements square and diamond. I created the two structuring elements using strel function and dilated the image using imdilate. After this I used imsubtract function to subtract the binary mask of the worm from the dilated binary mask which resulted in the boundary of the worm body.



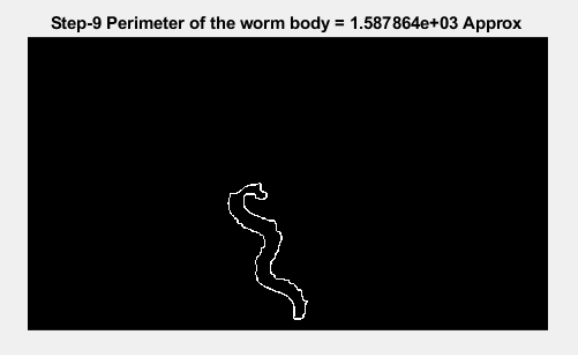
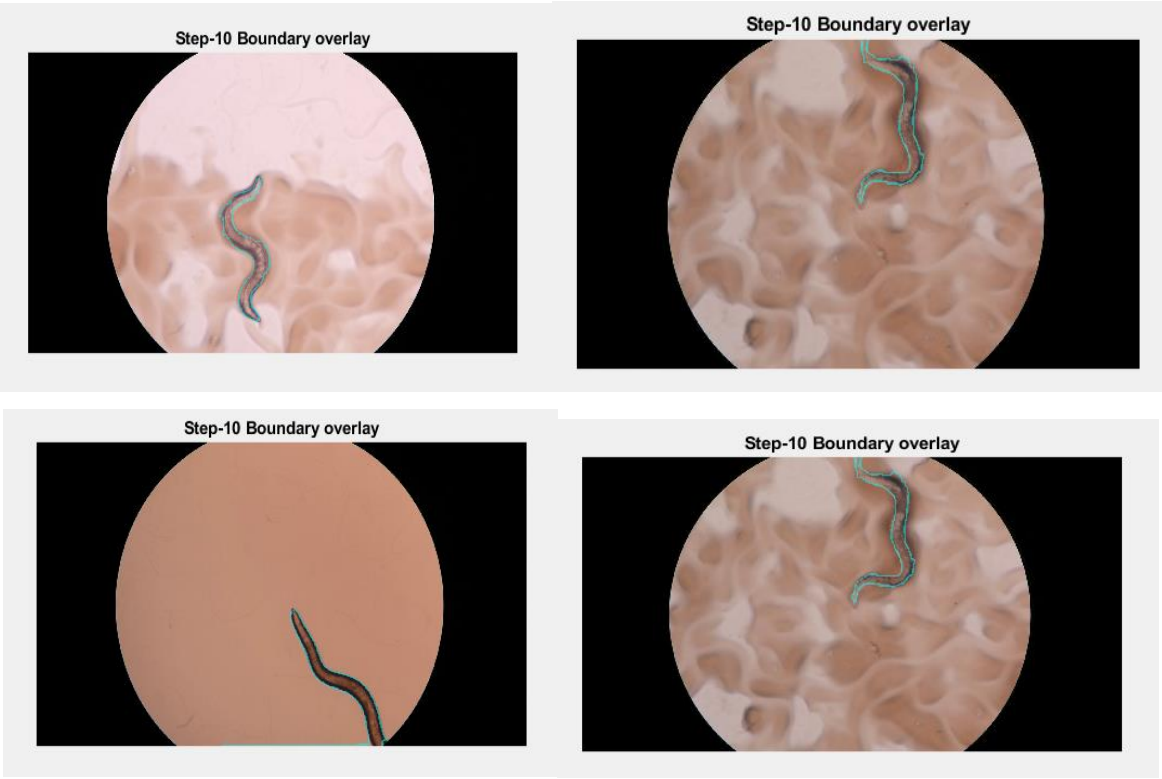


Figure 10: Boundary of the worm

2.3.11 Visualize the boundary

After that I used the labeloverlay function to display the boundary over the original image.



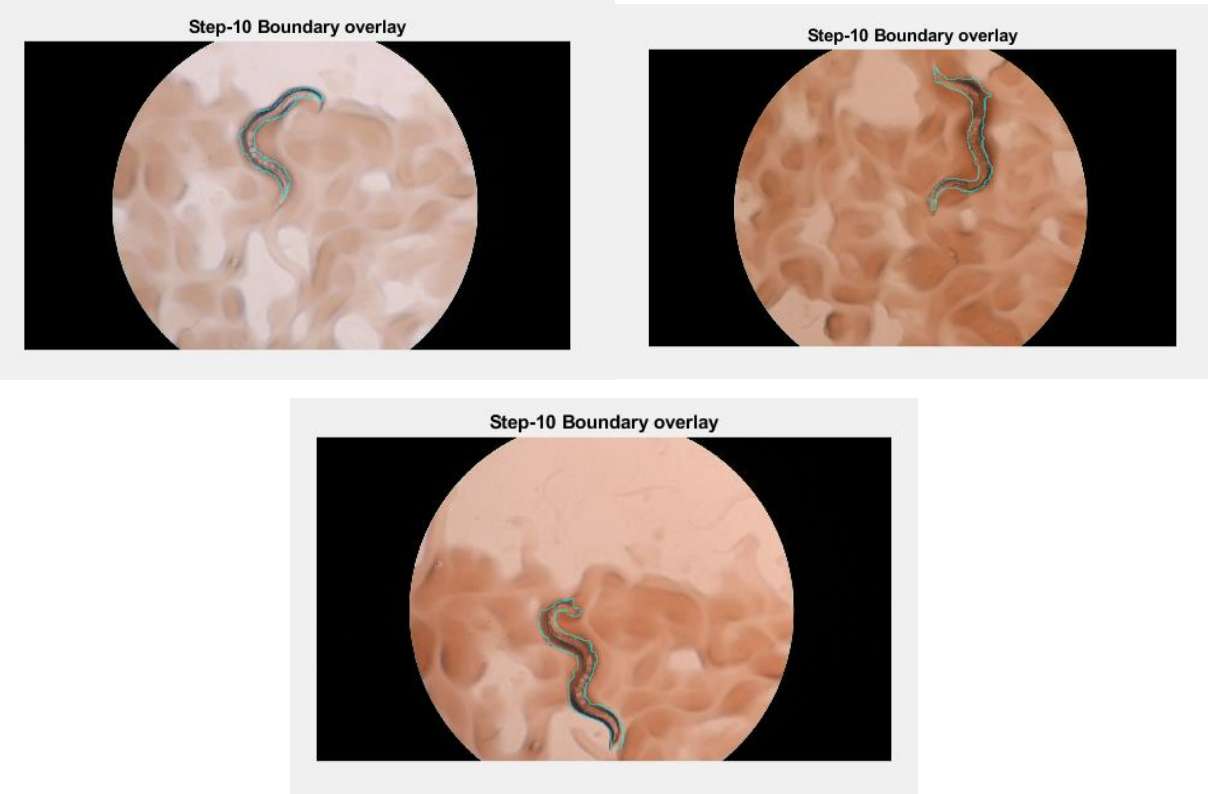
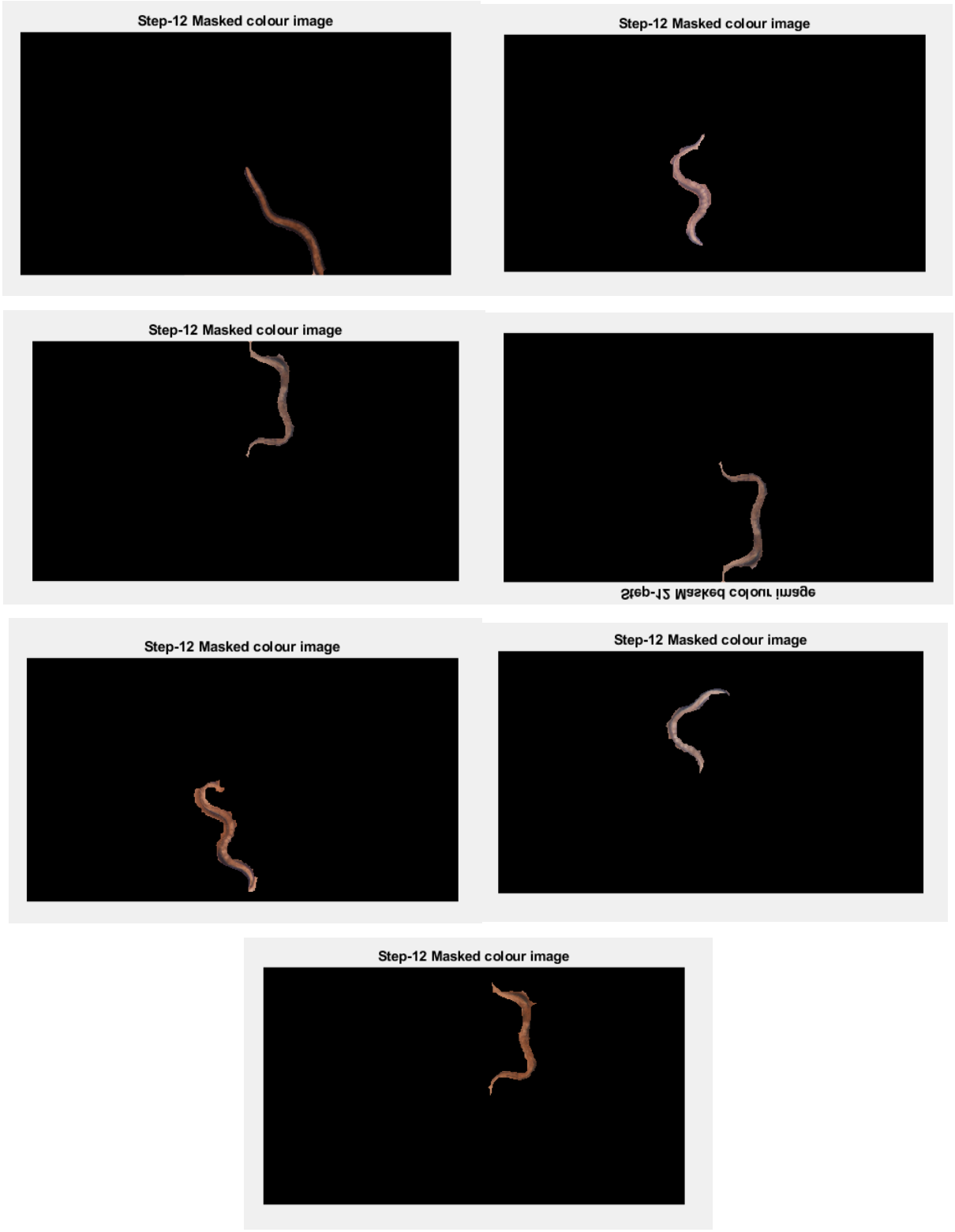


Figure 11: Boundary overlay

2.3.12 Generate masked color image

After that I extracted the worm (original pixels) based on the produced mask above and produced a new color image that only contained the worm (original pixels) from the original image. For this I declared another variable and named it “mask” and put the binary mask in that variable and in Binary_mask I put the Orginal_image and used bsxfun function to produce the masked color image of the worm.


Step-12 Masked colour image
Step-12 Masked colour image
Step-12 Masked colour image
Step-12 Masked colour image
Step-12 Masked colour image
Step-12 Masked colour image

2.3.13 Calculate Area

After generating the binary mask of the worm, I calculated the area of the worm by using regionprops function and stored the result in a cell array.

2.3.14 Calculate Perimeter

After generating the binary mask of the worm, I calculated the perimeter of the worm by using regionprops function and stored the result in a cell array.