University of Nottingham
UK | CHINA | MALAYSIA

Monitoring Worms

Coursework – Item 2

Name: Sonia Mubasher

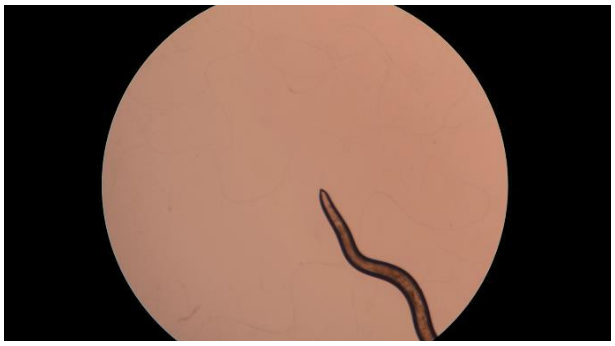
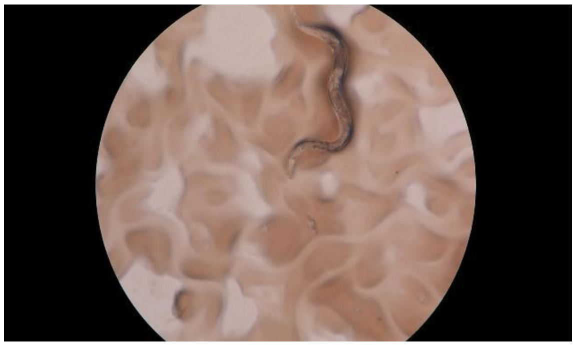
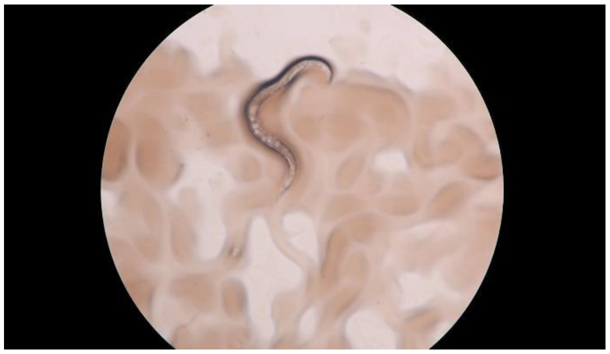
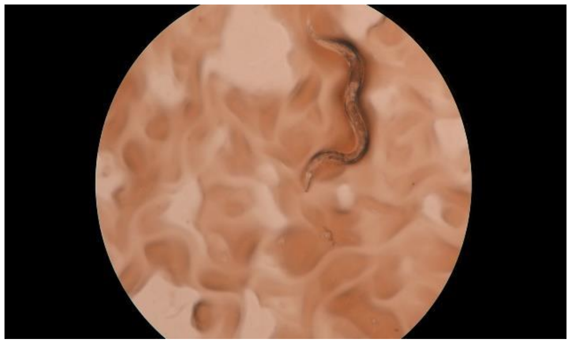
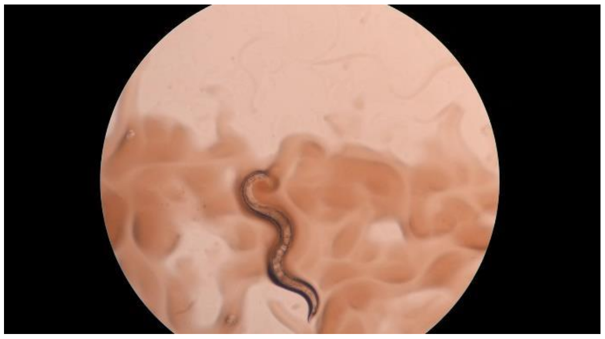
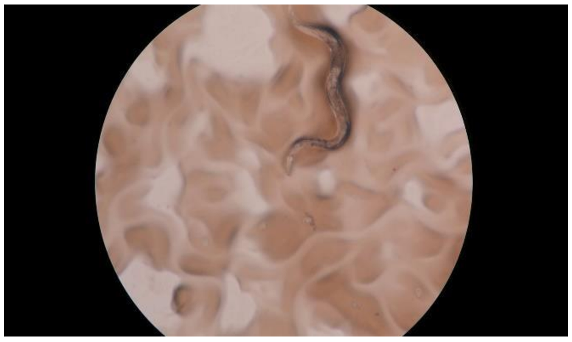
Student ID: 20129528

Word Count: 2538

Table of Contents

1. Introduction

This report includes explanation and justification on chosen techniques for extracting the worm body and detecting edge from the given set of images and critically evaluating on the methods and the obtained results by discussing the strengths and weaknesses of the chosen methods or stages.

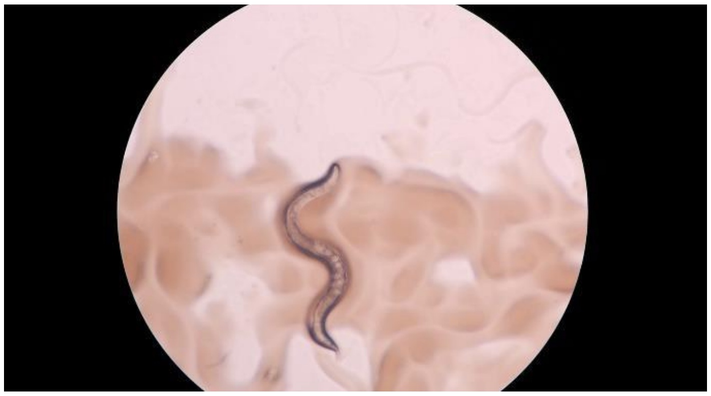


Figure 1: Original color images

2. General flow of stages

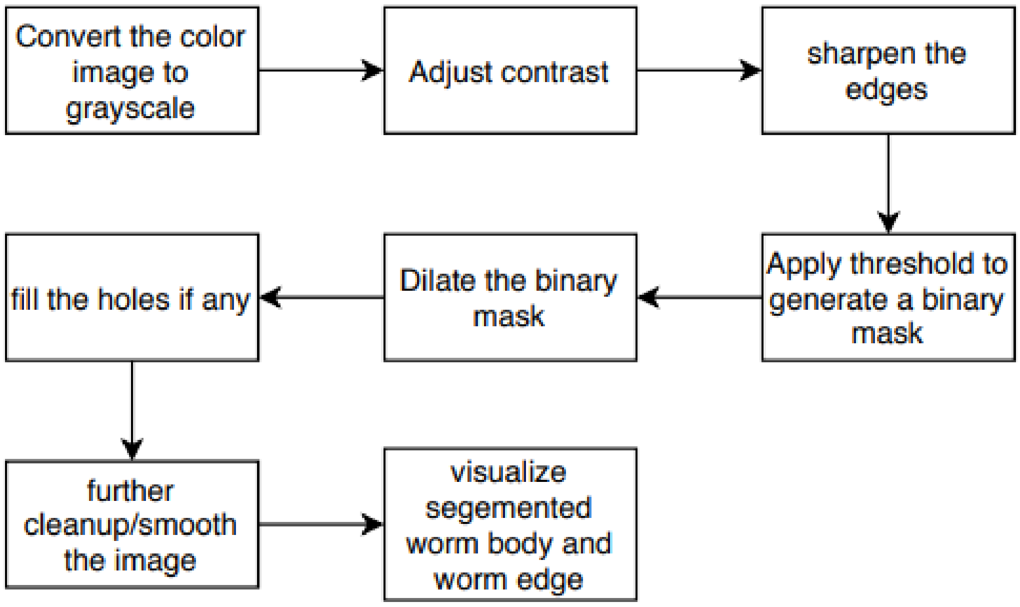

Convert the color image to grayscale
Adjust contrast
sharpen the edges
Apply threshold to generate a binary mask
Dilate the binary mask
fill the holes if any
further cleanup/smooth the image
visualize segemented worm body and worm edge

Figure 2: Flow of stages

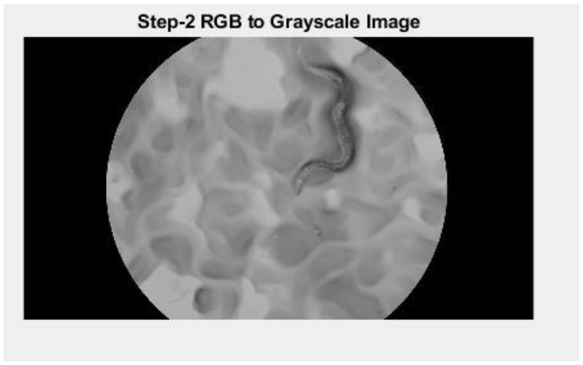
Figure 2 explains the order of stages which I followed to extract the worm body and detect the edge from the given set of images.

3. Image Pre-Processing

As you can see in figure 1 the given set of images highly differs in background noise from each other, so pre-processing of the images was required for the same implementation to work with all images.

3.1 Converting color image to grayscale image

As the given set of images were stored in RGB color format. So, I first converted all the images to grayscale as an RGB image consists of 3-dimensional matrix where grayscale is of only 2 dimensions x, y and depth of 1 and the values range between 0-255 (8-bit unsigned integers) where a black pixel is represented by 0, and a white pixel is represented by 255. Therefore, some algorithms can only be applied on 2-D rather than 3-D , So I converted the given RGB images to grayscale as color information is not that helpful in identifying edges and extracting other important information from the images. I used rgb2gray function which converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance. The execution time is also fast for such images and is easy to perform as you have to treat less bits. Also, it takes less storage/memory and is simple but still provides a lot of information. It’s always a better choice to convert the image to grayscale before doing any further implementation instead of working on the color image itself as it would be really hard. also, it takes more time to process color images.



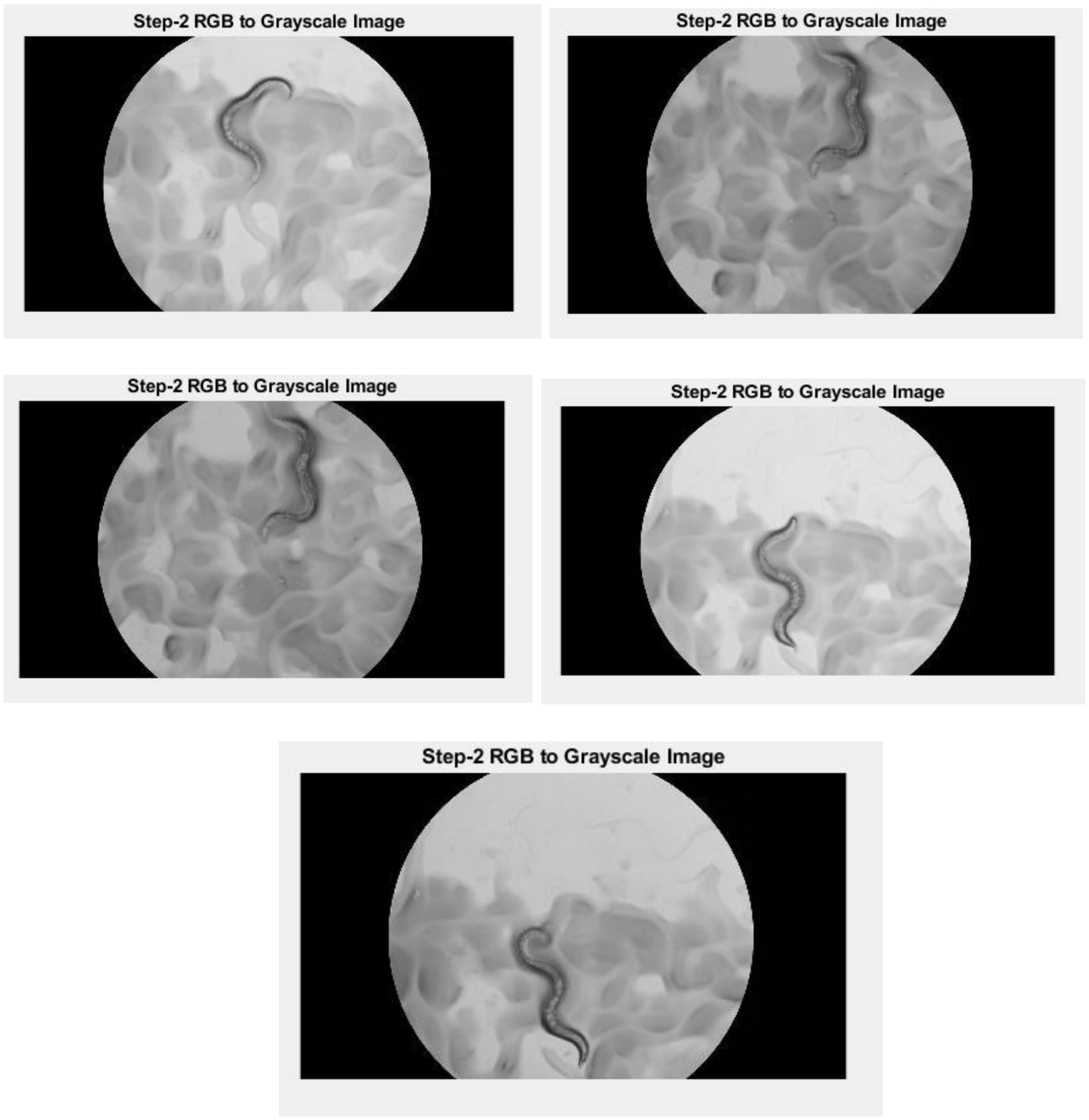
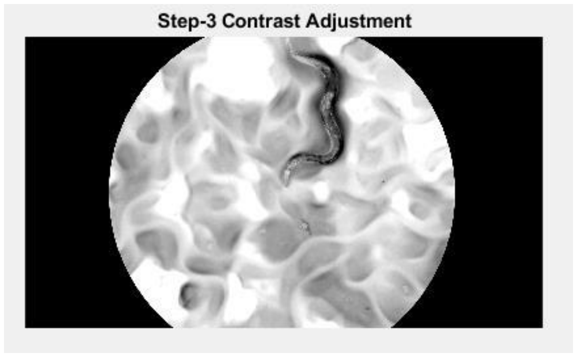

Step-2 RGB to Grayscale Image
Step-2 RGB to Grayscale Image
Step-2 RGB to Grayscale Image
Step-2 RGB to Grayscale Image
Step-2 RGB to Grayscale Image

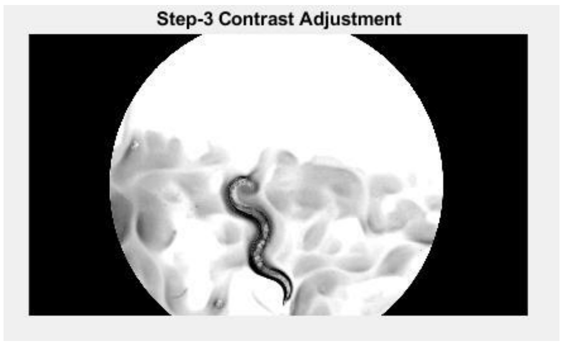
Figure 3: Results after converting color images to grayscale

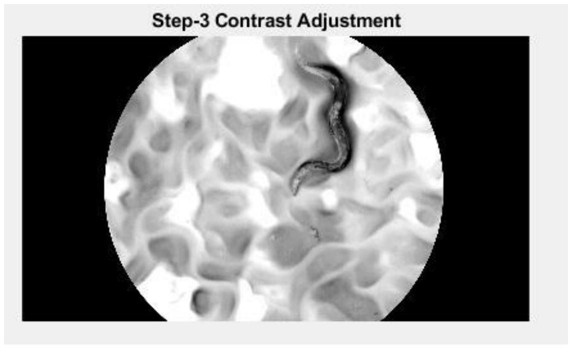
3.2 Image contrast

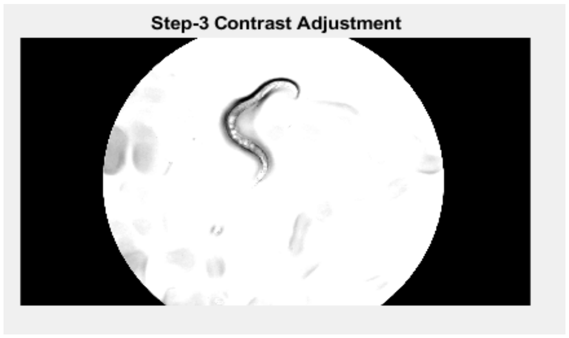
As you can see from Figure 3 all the images lack contrast as there are no sharp differences between black and white. So, by adjusting the contrast level of the images the black pixels become darker and white pixels become brighter and the worm body as you can see in Figure 4 is now more prominent and easier to extract as compared to Figure 3. I used imadjust function which maps intensity values in the grayscale image to new values. After checking with a lot of different values I chose the range from 0.25 to 0.67.

In this process, pixels values below 0.25 are mapped to black and pixel values above 0.67 are mapped to white and If you see the histogram you will see the base of the bell curve expand outward as we increase the contrast and the pixels are spread from the proximity to the middle towards white and black. I chose this technique to proceed further in my task because there was no clear difference between the worm body and the background noise in Figure 3 and it would be really hard to extract the worm body from such images so to make my task easier I chose to adjust the contrast level of the image so, if not 100% but 40%-50% of noise would be disappear and the body of the worm would be easier to extract. If you are given such images where the object to extract is not that prominent it’s always a good choice to adjust the contrast before any further implementation.









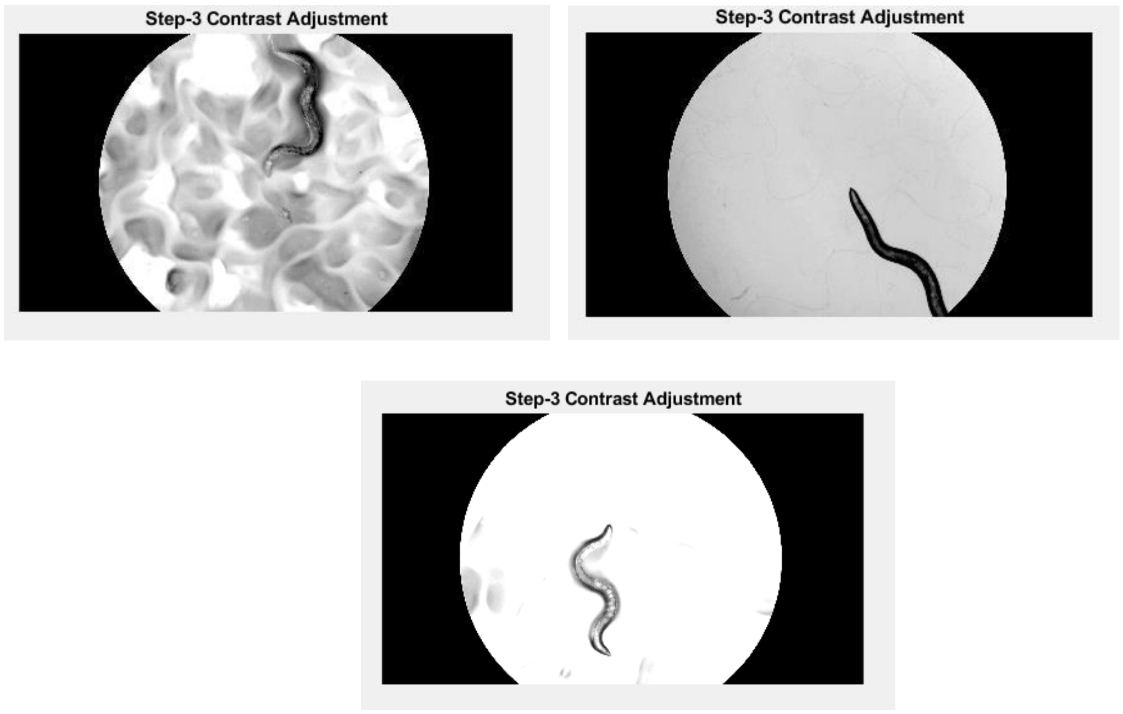
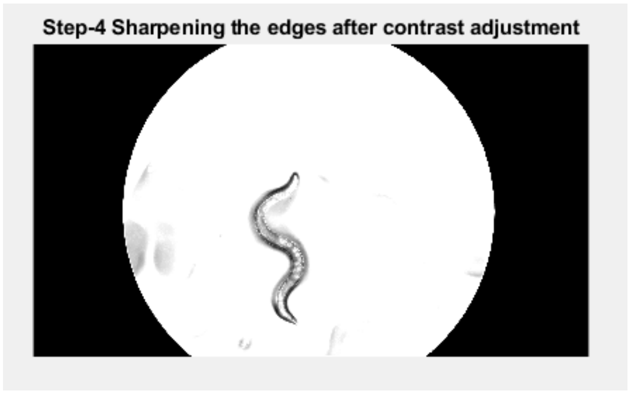

Step-3 Contrast Adjustment
Step-3 Contrast Adjustment
Step-3 Contrast Adjustment

Figure 4: Results after adjusting contrast level in Figure 3

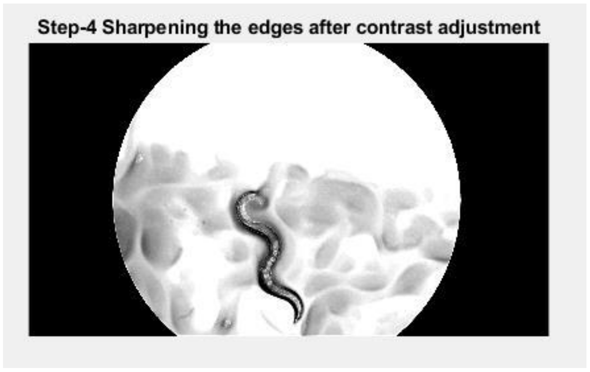
3.3 Image sharpening

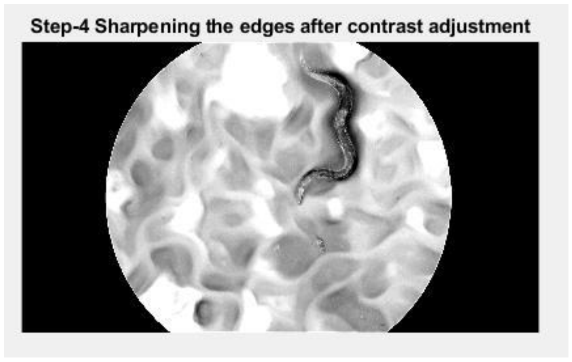
From Figure 4 as you can see that the edge of the worm is not that sharp and if the image is sharpened it would be much easier to detect the edge with less background noise. I used unsharp mask filter to proceed with my task which is used to sharpen an image. The sharpening process works by first creating a slightly blurred version of the original image. For this I used gaussian low pass filter that allows low frequency image information to pass through and blocks high frequency image information which effectively blurs or smoothes the image. The primary advantage of the unsharp filter over other sharpening filters is the option to control the sharpening process as many sharpening filters do not provide any user-adjustable parameters.

I used fspecial function to produce blurred version of Figure 4 which returns a rotationally symmetric gaussian lowpass filter of size 10 with standard deviation 7. After that I subtracted the blurred image from Figure 4 in order to produce an edge image(i.e. roughly extracted edges) and added the edge image to Figure 4 to produce a sharpened image i.e. Figure 5.

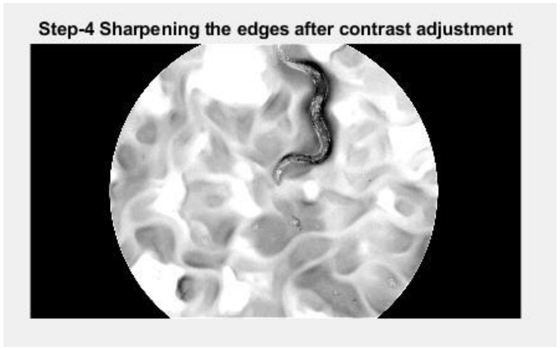

Step-4 Sharpening the edges after contrast adjustment

Step-4 Sharpening the edges after contrast adjustment


Step-4 Sharpening the edges after contrast adjustment


Step-4 Sharpening the edges after contrast adjustment

Step-4 Sharpening the edges after contrast adjustment


Step-4 Sharpening the edges after contrast adjustment

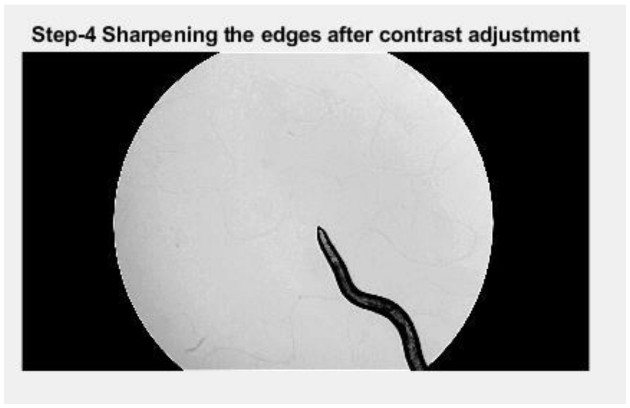


Figure 5: After sharpening Figure 4

However, this was not a suitable choice for my task as if you see above Figure 5 there is not much difference between the sharpened image and the previous Figure 4. Sharpening Filters are very sensitive to noise. So, it's always better to apply noise reduction first as sometimes sharpening filters may create unwanted edge effects or increase image noise. Using unsharp mask filter was not suitable in this case instead Gaussian high pass filter would be a good choice as it allows high frequency image information to pass through and blocks low frequency image information which effectively sharpens the image as compared to unsharp mask filter.

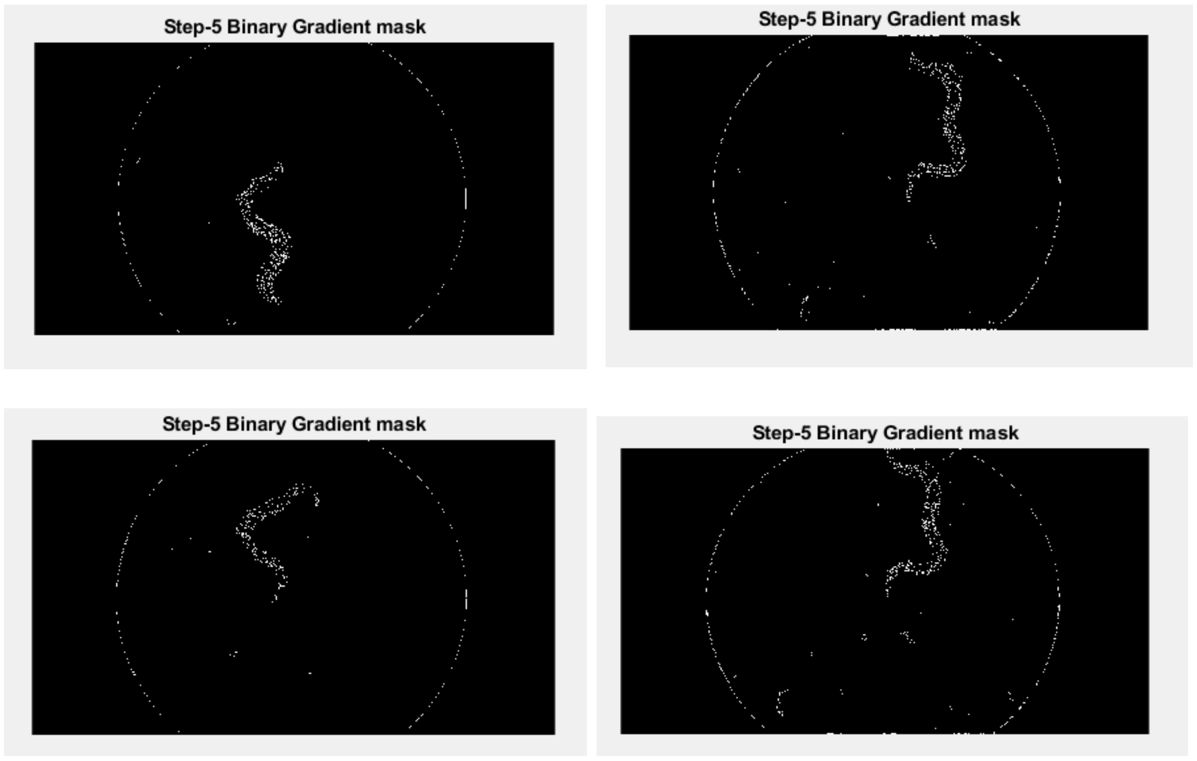
4. Image Segmentation

After Pre-processing of the given images, the worm-body is now more prominent as compared to the original images as you can see in Figure 5. Image segmentation is the process of subdividing a digital image into multiple meaningful regions, I started by first generating binary gradient mask from Figure 5 to accomplish my task.

4.1 Generating Binary Gradient Mask

Now as you can see the worm body to be segmented differs greatly in contrast from the background noise in Figure 5. I used image thresholding to proceed as it's a simple yet effective, way of partitioning an image with high levels of contrast into a foreground and background. This technique is a type of image segmentation that isolates objects by converting grayscale images into binary images. Changes in contrast can be detected by operators that calculates gradient of an image. I used edge and the sobel operator to calculate the threshold value that calculates the gradient of image intensity at each pixel within the image and it shows how abruptly or smoothly the image changes at each pixel. I used a fudge factor of 1.0 to tune my threshold value as I wanted less background noise but still wanted the worm body to be visible. As you decrease the fudge factor you will see the result with more white pixels (with more background noise) and if you keep on increasing the fudge factor the background noise will eliminate completely but the worm body will also not be completely visible. So, after so many trials and errors I convinced myself to use fudge factor = 1.0 as it replaces each pixel in an image with a black pixel if the image intensity is less than 1.0 , or a white pixel if the image intensity is greater than 1.0.

I used edge and sobel operator again with the threshold value I calculated above to produce the Binary gradient mask as you can see below in Figure 6.


Step-5 Binary Gradient mask
Step-5 Binary Gradient mask
Step-5 Binary Gradient mask
Step-5 Binary Gradient mask

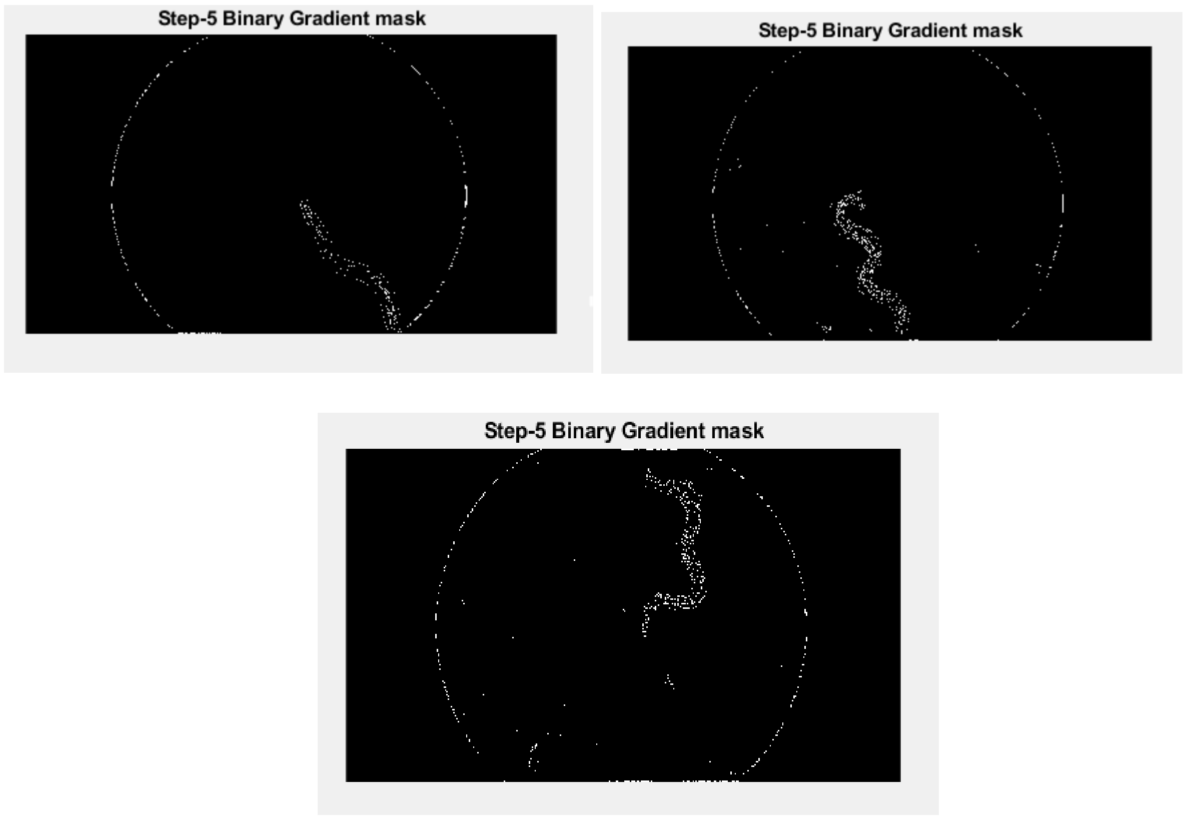

Step-5 Binary Gradient mask
Step-5 Binary Gradient mask
Step-5 Binary Gradient mask

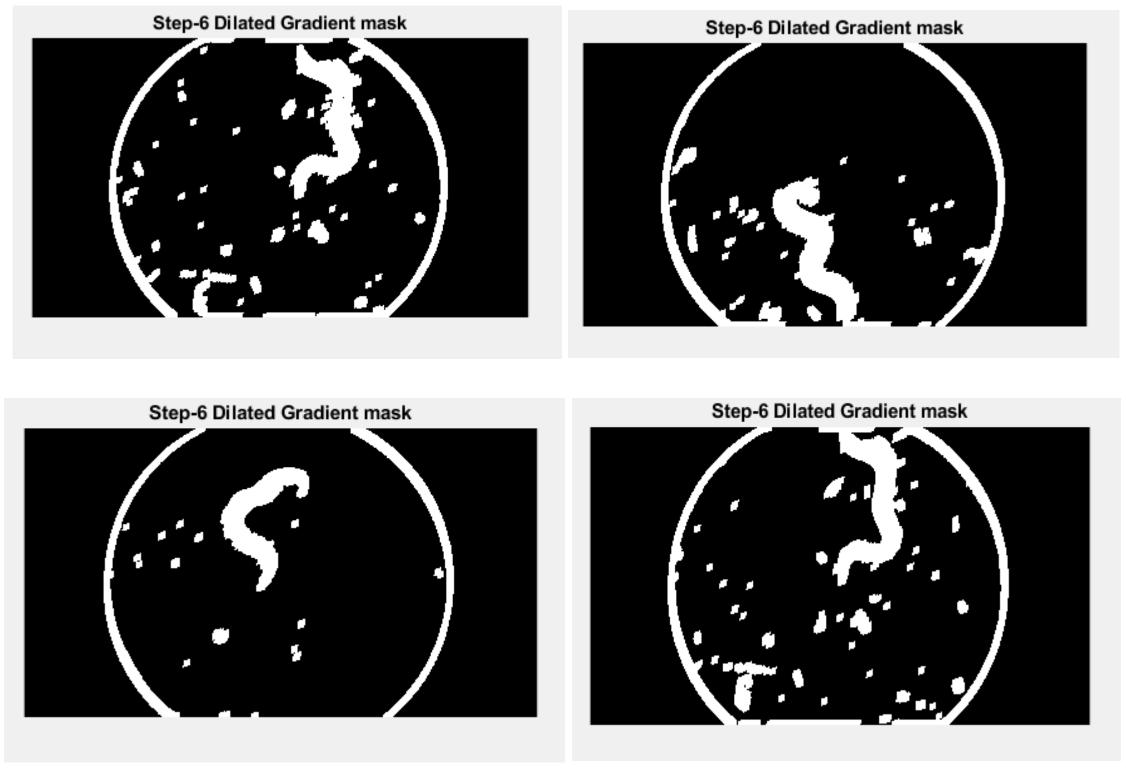
Figure 6: Binary Gradient Mask

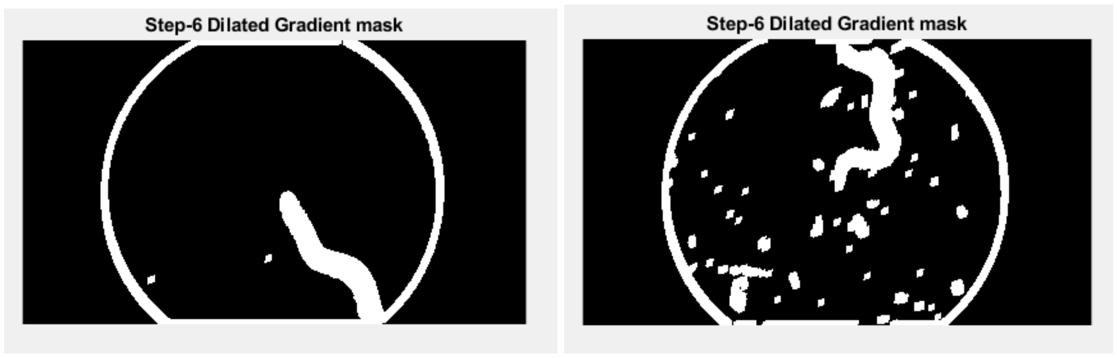
To extract an object from an image it's always a better choice to create a binary gradient mask of that image first with a specific threshold value. I chose to calculate my threshold value based on edge detection by using edge and sobel operators as It works by detecting discontinuities in brightness due to which the dotted lines of high contrast (the worm body) were created but if you don't tune this threshold value you will see the background noise as well as its contrast level is also high but if we chose a specific fudge factor to tune this threshold value then only the worm body will be visible and 60% - 70% approx. noise will be disappeared as you can see in Figure 6.

4.2 Dilate Binary Gradient Mask

Now as you can see in the above Figure 6 the binary gradient mask shows dotted lines of high contrast in the image, but these lines do not quite delineate the outline of the worm body as compared to the original image. I used morphological processing of images to proceed in my task. It treats images as sets of pixels. In my case I have a Binary image where White pixels represents Foreground objects and Black pixels represents the background objects. It operates by changing the assignment of a pixel from one set to another i.e. foreground to background or background to foreground sets depending on the structuring element where the structuring element can vary in size and can have various shapes.

From the above figure 6 as you can see there are these gaps which will be disappear if the sobel image is dilated using linear structuring elements. Dilation expands the foreground or background object using the structuring element. So, I created two perpendicular linear structuring elements by using strel function which is a binary valued neighborhood, in which the true pixels are counted in the morphological computation and the false pixels are not. The center pixel of the structuring element , called the origin, identifies the pixel in the image being processed and dilated the binary gradient mask using the vertical structuring element of size 26 followed by the horizontal structuring element of size 26 as well using imdilate function as you can see in Figure 7.


Step-6 Dilated Gradient mask
Step-6 Dilated Gradient mask
Step-6 Dilated Gradient mask
Step-6 Dilated Gradient mask



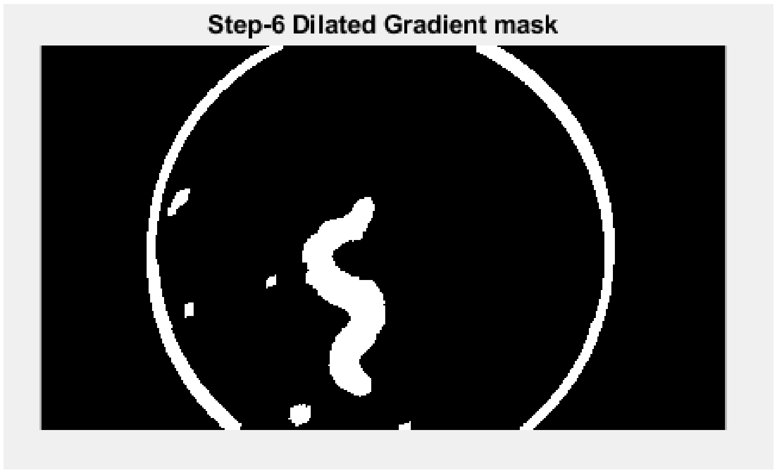


Figure 7: Dilated gradient mask

Dilation was a good choice to proceed in my task as the dotted lines were not really connected due to which it was impossible to extract the worm body so by dilation the foreground pixels were expanded due to which they got connected to each other and a clear boundary of the worm was then created. But as you can see the edge of the worm is still not that smooth but good enough because of the structuring element I chose. I tried with other structuring elements such as disk and square as well, but the result was even worse than the current result. So, I proceeded with two lines perpendicular to each other.

4.3 Fill in the holes if any

The dilated gradient mask as you can see in the above Figure 7 shows the outline of the worm quite nicely , but there are still holes in the interior of the worm body which can be filled using imfill function which fills holes in the input binary image i.e. Figure 7 where a hole is a set of background pixels that cannot be reached by filling in the background from the edge of the image as you can see in figure 8.

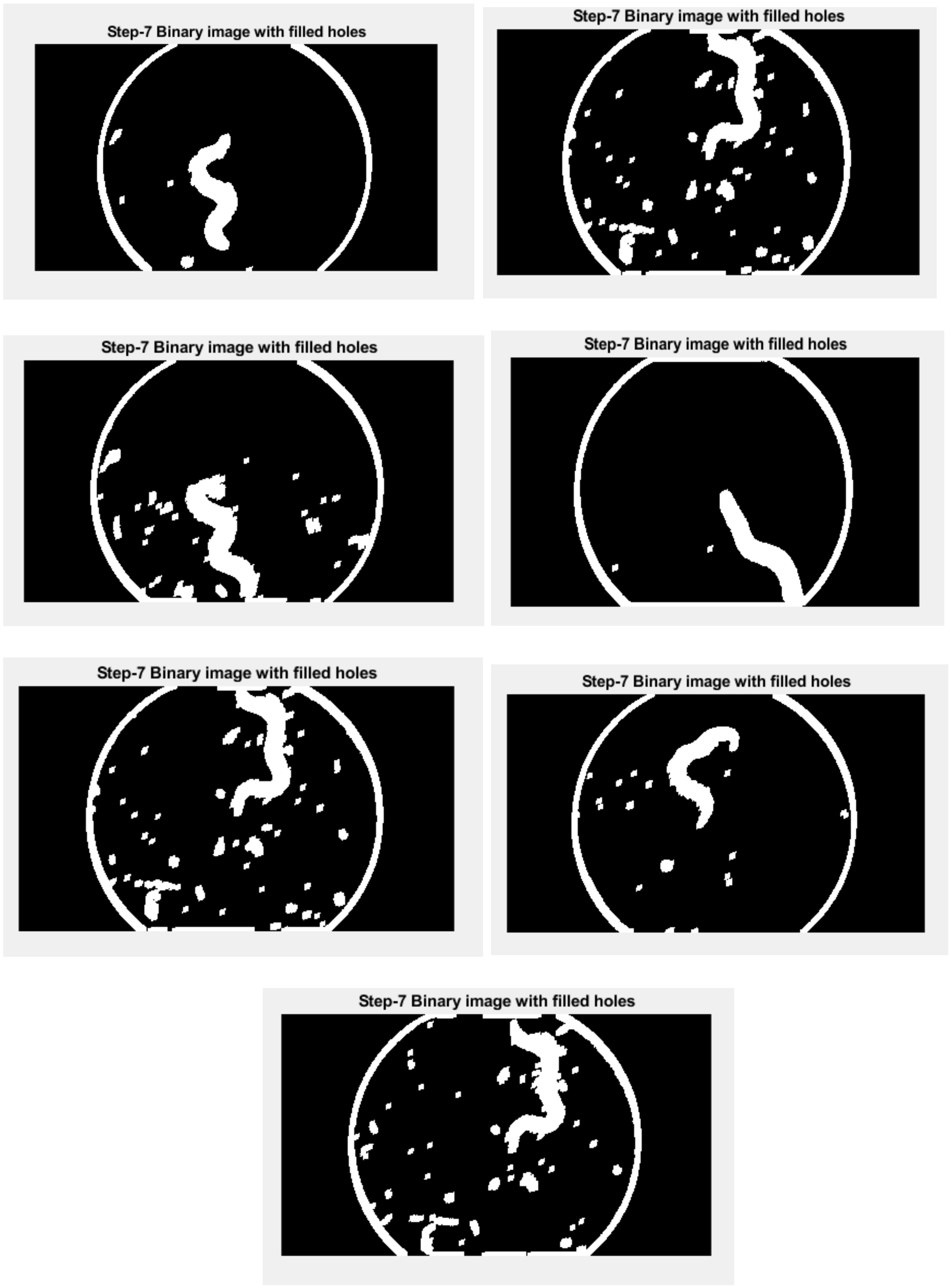
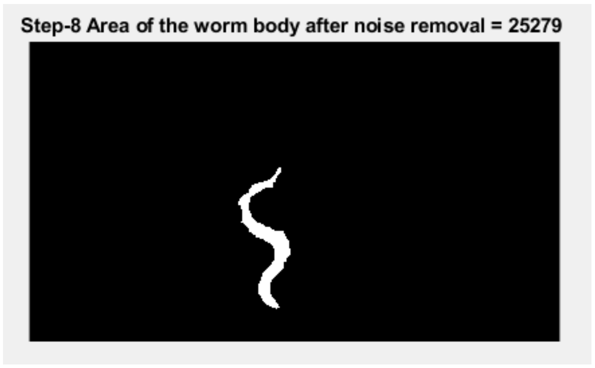
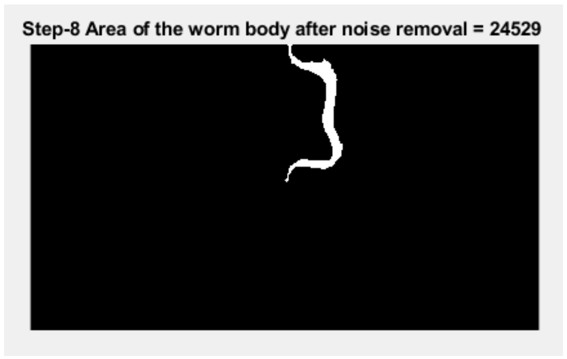

Step-7 Binary image with filled holes
Step-7 Binary image with filled holes
Step-7 Binary image with filled holes
Step-7 Binary image with filled holes
Step-7 Binary image with filled holes
Step-7 Binary image with filled holes
Step-7 Binary image with filled holes

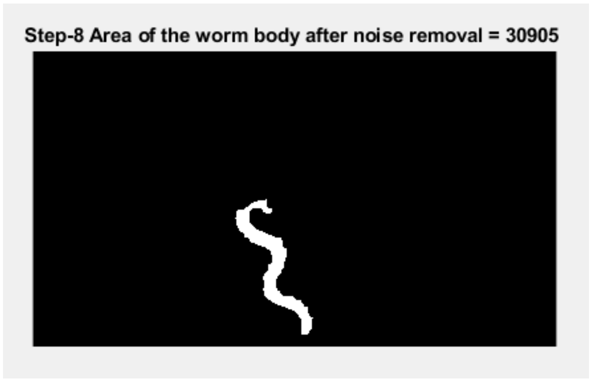
Figure 8: Binary image with filled holes

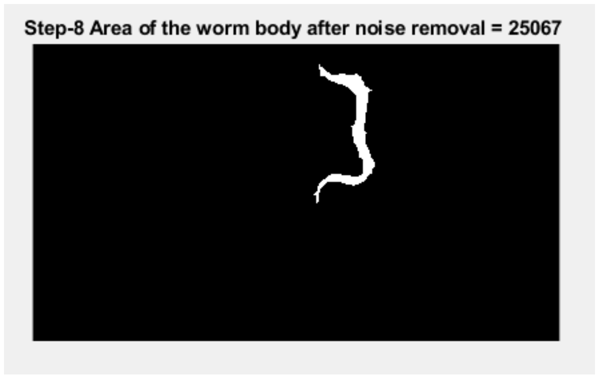
4.4 Extract Body of the worm

As you can see from the above Figure 8 now the worm body is completely visible but still there is so much noise in the image in order to just get the body of the worm and to make the segmented worm body look natural further processing of the image was required. I used erosion from the morphological operations to continue as erosion shrinks a foreground or background object, using the structuring element. So, the small foreground objects as you can see in the above figure 8 will be disappeared by using imerode function with two structuring elements square of size 14 and disk of size 14 as well. After eroding the image 90% of the noise was removed from the image but still there were still some parts in the image with noise, so I further used bwareaopen function to remove those pixels from the image. It removes all connected components that have fewer than 900 pixels from figure 8, producing another binary image Figure 9. This operation is also known as area opening.









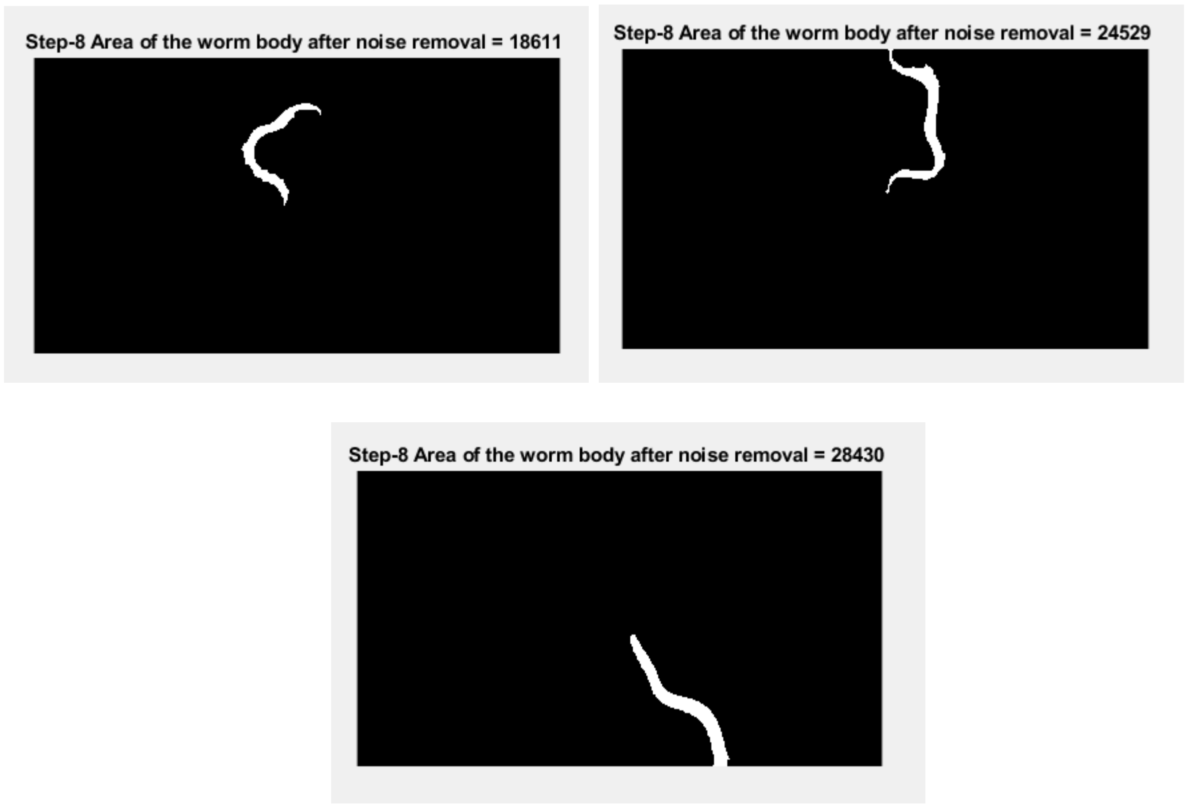

Step-8 Area of the worm body after noise removal = 18611
Step-8 Area of the worm body after noise removal = 24529
Step-8 Area of the worm body after noise removal = 28430

Figure 9: Segmented worm body

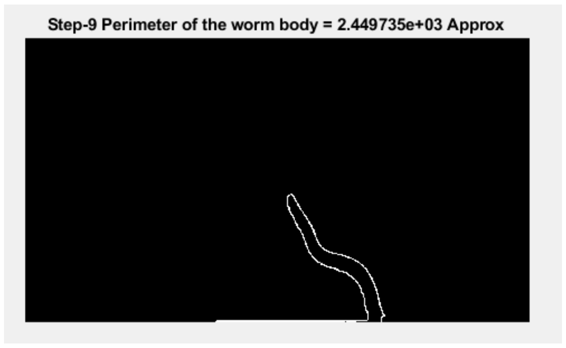
Erosion was a good choice for me to proceed in my task as you can see in Figure 8 the worm body is completely visible but there is still a lot of background noise due to which it becomes hard to detect the worm in the image. So, by eroding the above Figure 8 with suitable structuring elements all the noise was disappeared and just the worm body was left in the image as you can see in Figure 9.

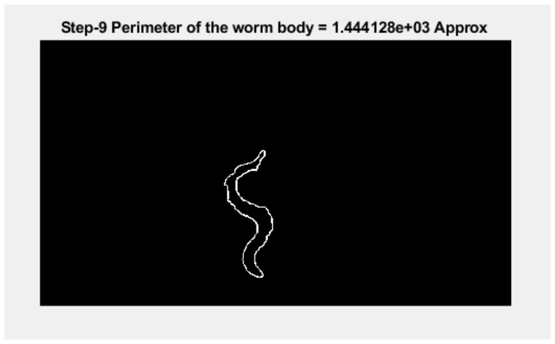
4.5 Calculate area of the worm body

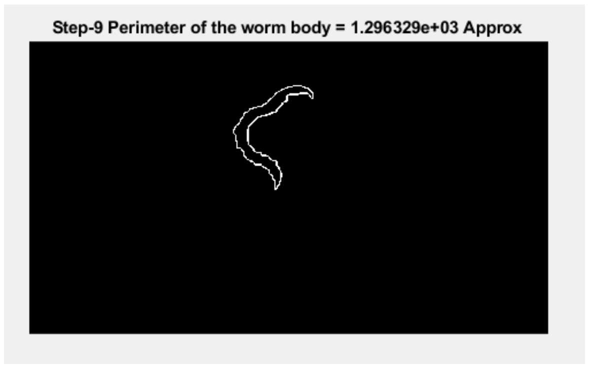
To measure the area of the segmented worm body I used regionprops function which calculated the area in pixels. The total area for each image is displayed in the above figure 9.

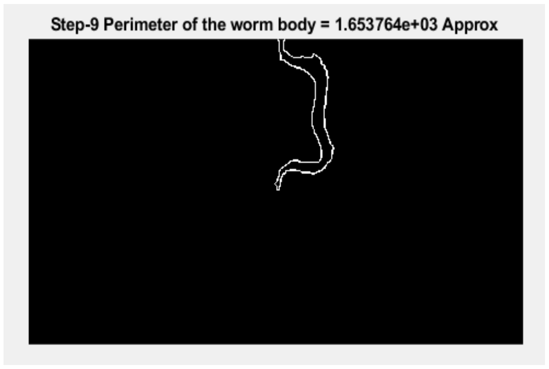
4.6 Extract Boundary of the worm

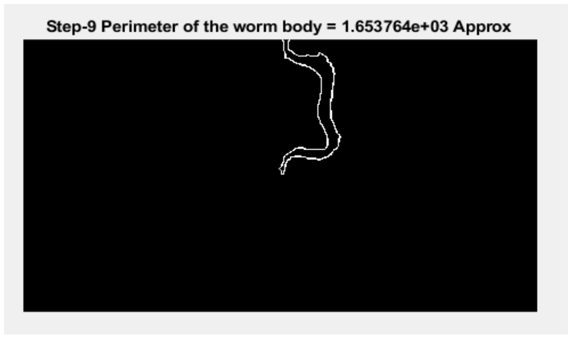
To detect the edge of the worm I first used edge and sobel operator but due to irregular region the boundary of the worm was not totally connected. So, again I proceeded with morphological operations I dilated the above Figure 9 and produced a new binary image and from that new image I subtracted the previous image which resulted in boundary of the worm body as you can see in Figure 10.

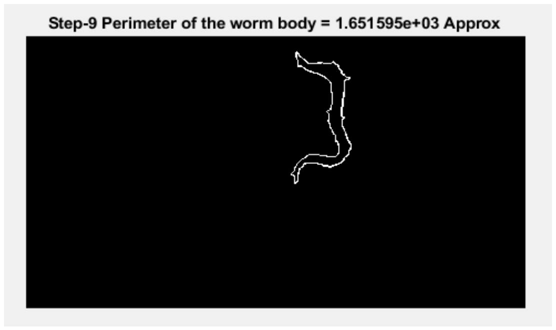












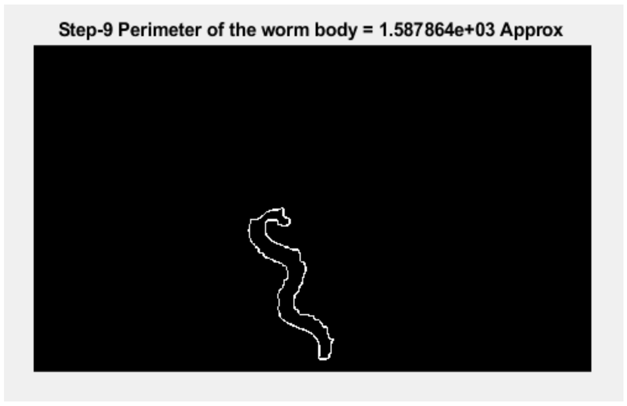


Figure 10: Boundary of the worm body

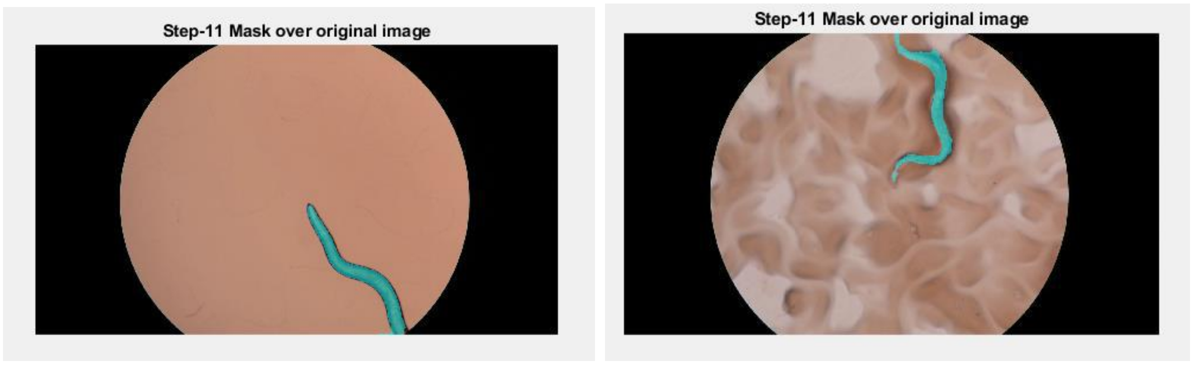
4.7 Calculate perimeter of the worm

To measure the perimeter of the worm body I used regionprops function which computes the perimeter by calculating the distance between each adjoining pair of pixels around the border of the region but as in my case the image contains discontinuous regions it returns unexpected results. The perimeter of the worm body can be seen in Figure 10.

5. Image Visualization

5.1 Visualize segmented worm body

To visualize segmented worm body, I used labeloverlay function which fuses the input image with a color where the binary mask of the image Figure 9 is true and displays the binary mask of the worm over the original image as you can see in Figure 11.


Step-11 Mask over original image
Step-11 Mask over original image

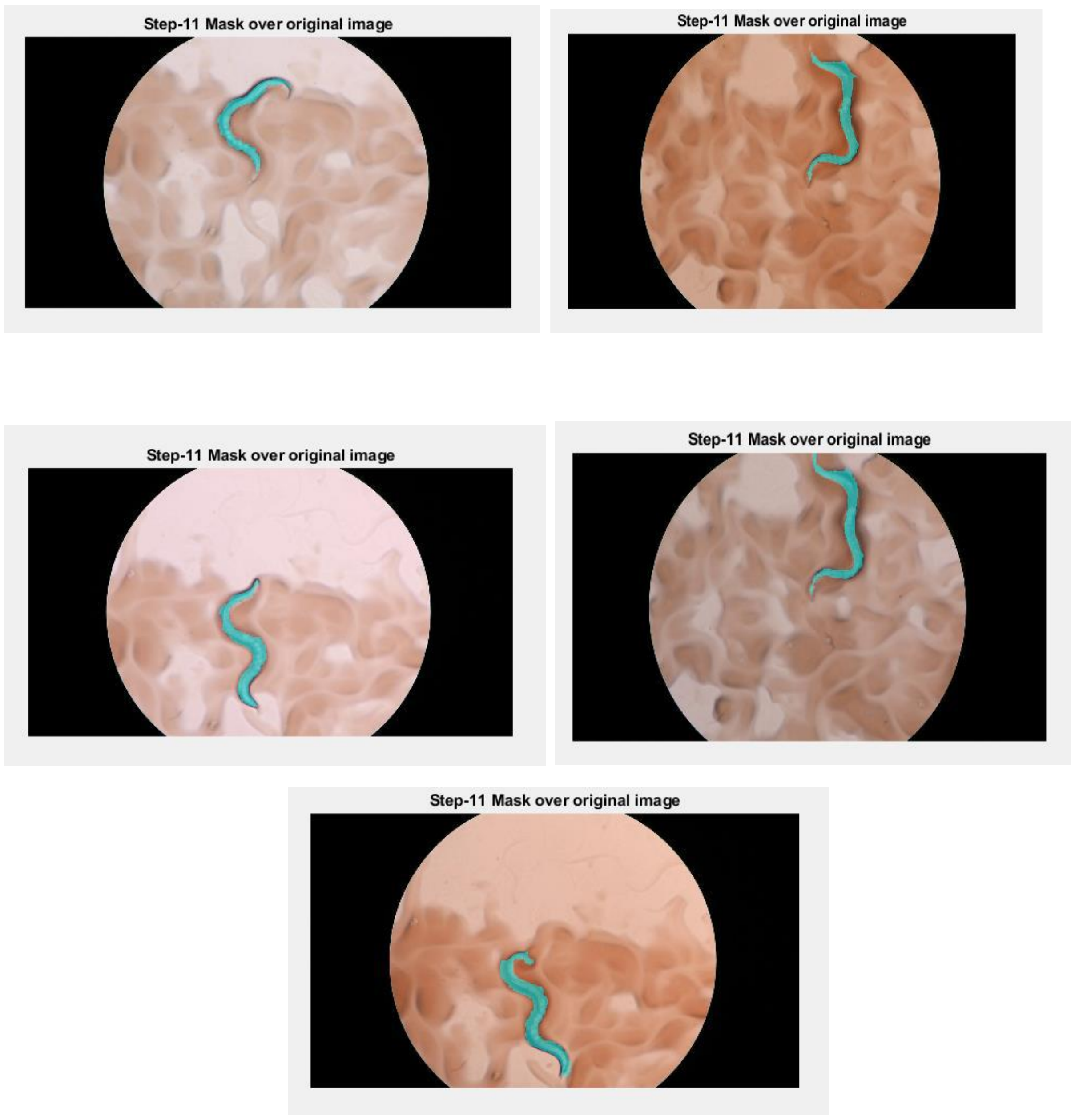

Step-11 Mask over original image
Step-11 Mask over original image
Step-11 Mask over original image
Step-11 Mask over original image
Step-11 Mask over original image

Figure 11: Mask over Original Image

5.2 Visualize edge of the worm

To visualize edge of the worm, I used labeloverlay function which fuses the input image with a color where the image displaying edge of the worm Figure 9 is true and displays the boundary of the worm over the original image as you can see in Figure 12.

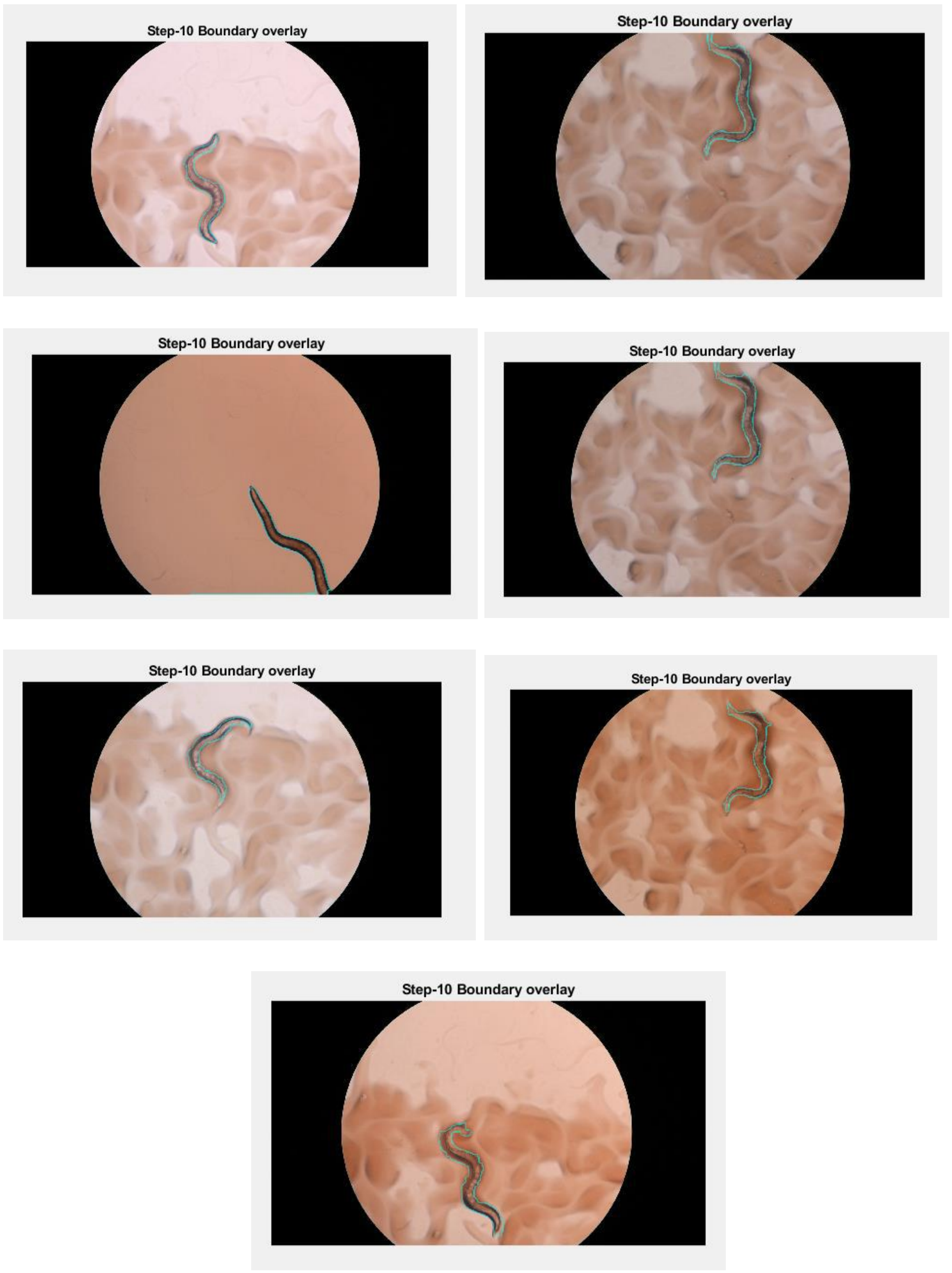
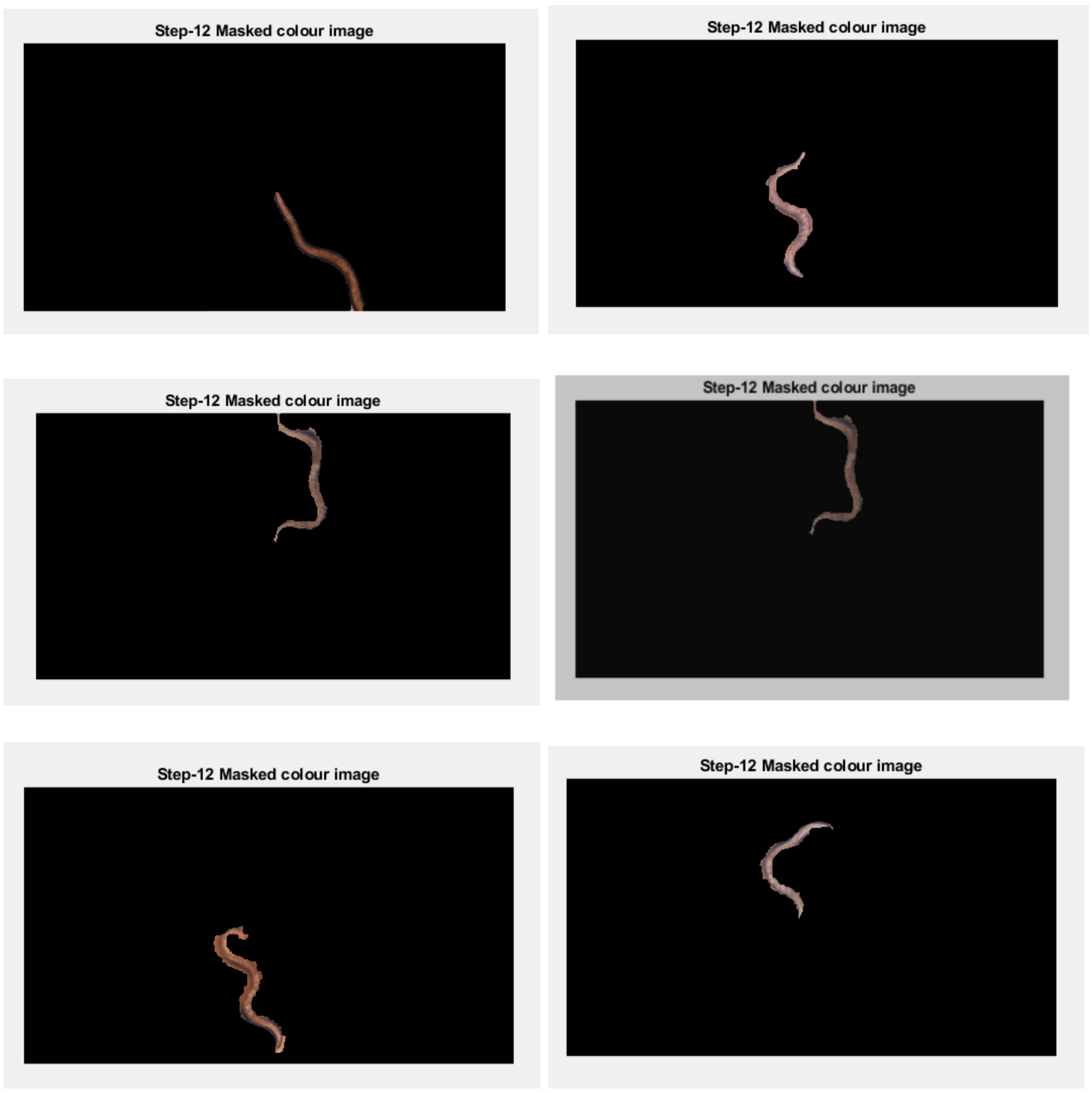


Figure 12: Edge over original image

5.3 Generate masked color image

To generate masked color image of the worm I extracted the worm (original pixels) based on the produced binary mask above and produced a new color image that only contained the worm (original pixels) from the original image as you can see in Figure 13.


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

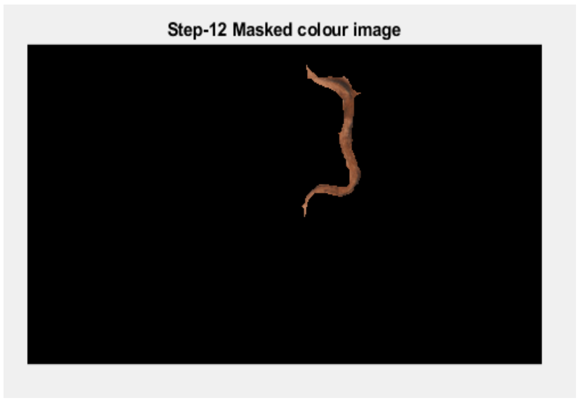


Figure 13: Masked color image

6. Conclusion

In conclusion, there are many other ways to extract an object from an image of which some of the techniques I used above. In addition, if an image goes through a number of pre-processing stages in right order before the actual processing starts then it becomes much easier to accomplish the task.