Fourier Descriptors Of Planarian Worms

Dominik Chodounský

November 2019

1 Introduction

The goal of this project was to use computer vision concepts and tools in order to detect, describe and compare images of planarian worms. The dataset provides close up photos of planaria in a petri dish in different stages of their growth and in different positions.

2 Detecting the Planaria

After reading an image of the planaria, I convert it to grayscale values and apply thresholding to create a binary image. The threshold values I decided to go with were 50 and 255, which created the best blob-like features for the planaria without leaving out parts of their tails or being otherwise fragmented. Instead of thresholding, I also tried out running the image through a Canny edge detector, which didn't produce very satisfying results in comparison. Another approach in trying to optimize the recognition tool was performing histogram equalization on the image before thresholding, but this once again achieved worse results, as the differences in color between background noise and the planaria were less obvious.

In order to locate the worm and its contour within the image, I have used the cv2 function *cv2.findContours*. I have decided not to use chain approximation, as I needed the straight lines of contours at a later stage.

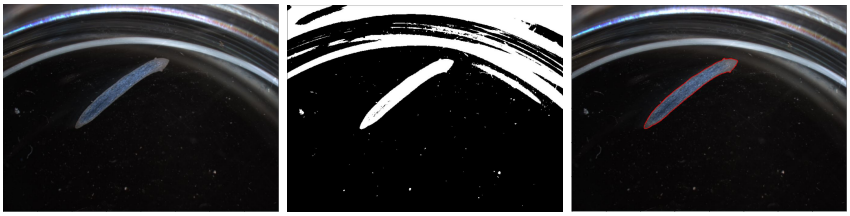
Having found all contours, I used *cv2.contourArea* to calculate the area of each contour and order them in decreasing order by their size. From looking at the dataset, the planaria is clearly going to be one of the few biggest blobs in the image.

At this point, I needed to distinguish between planaria blobs and blobs created by petri dish edges seen in the image. I attempted this in several different ways. The first was to find a pair of low and high thresholds for the planaria area. However, their sizes seem to have enough variation to make distinguishing them from also varying petri dish blobs quite impossible. The second method I tried was comparing the first and the last pixel of a contour and seeing if they are located within 1 pixel distance of each other. This way, I could find out if the shape of the contour is enclosed, which would have possibly

helped me distinguish the worms from some petri dishes. The final method I ended up using is based on finding perfectly straight lines longer than a given threshold (I chose 40 pixels) in the contour itself. The reason for this was, that the images never contain a full petri dish, its edge is always cut off by an edge of the image. This creates a straight line in its contour. Knowing this, I choose to simply highlight the contours which are the biggest (with at least 10 000 pixel area) and without straight lines and the accuracy seems to be fairly high based on my conducted testing.

(a) Regular photo of Planaria. (b) Thresholded binary image of photo. (c) Highlighted contour of planaria.

Figure 1: Results from detecting planaria in image *W54-PT-D11.TIF*.



(a) Regular photo of Planaria. (b) Highlighted contour of planaria. (c) Highlighted contour of planaria.

Figure 2: Results from detecting planaria in image *W1-PT-D11.TIF*.

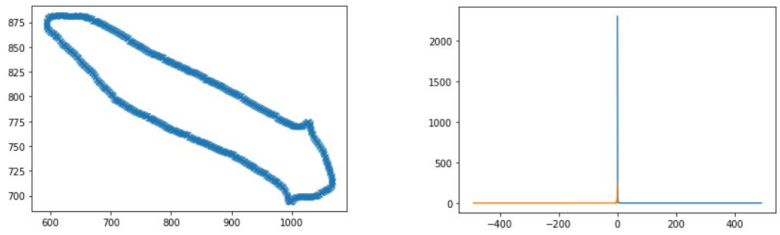
(a) Regular photo of Planaria. (b) Highlighted contour of planaria. (c) Highlighted contour of planaria.

Figure 3: Results from detecting planaria in image *W15-PT-D11.TIF*.

As we can see in Figure 2, the detection algorithm doesn't struggle with finding the contour of a planaria even if the petri dish is in the frame. Finally, Figure 3 shows that detecting multiple planaria shapes is also a successful.

3 Fourier descriptors

In order to be able to describe the general shape of Planaria, I used Fourier descriptors. Since the contour is just a set of coordinates tracing the outline of the shape, I needed to somehow describe the change of these coordinates in time, but this is difficult to do when they consist of two separate numeric values. However, I achieved this by separating the x and y coordinates and creating a set of complex numbers, where the real part is formed by the x coordinates and imaginary part by y coordinates. That gave me a combination of sine and cosine components describing the contour and I subsequently used the Fast Fourier transform (NumPy's *np.fft.fft function*) to convert them to the frequency domain. Figure 4 shows us the frequencies of a contour after applying Fast Fourier transform.



(a) Plot showing the planaria shape in complex domain.

(b) Plot showing the Fourier frequencies of a planaria contour.

Figure 4: Results from using the Fast Fourier transform on the contour made of complex coordinates.

In order to be able to compare the contours of different planaria, I centered the frequencies by setting the first coefficient to 0 and ran the inverse Fourier transform (*np.fft.ifft*), which allows for contours to have a common center point and enables me to compare them more easily.

[This is unfortunately the point where my work on this project ended due to time constraints, the following text describes how I would roughly go about implementing the rest of this task.]

Next step would be to normalize the range of values for all samples, so we can mathematically compare any two contours. At this point, we would have our descriptors of the general shape of planaria. Having one of these (or a combination of these) available, we could look at any image which contains a planaria, find its contour and transform it into a Fourier descriptor, which we then compare to that of a known planaria contour (for example by writing a distance function for two of these descriptors) without having to confirm the algorithm's decision ourselves.

Another addition to my tool could be finding the eigen vectors of the contours. Eigen vectors are useful when trying to avoid having to deal with linear transformations of the shape (rotation, different positions,...). Since these vectors are only affected by multiplication by a scalar value (eigen value), we can match the similar shapes much faster then using other methods.

4 Conclusion

Even though I did not have enough time to finish all parts of this project, I think it was fairly successful. Most parts needed to perform planaria description and comparison are finished and I learned how to go about implementing the tools needed to finish the job, which may be an interesting thing to do in the future. I was especially interested in isolating the planaria shape within images and I think I came up with a very reliable method of doing this for our dataset. Having tested out several different approaches and methods, I believe the one I ended up using will work the best in most cases.