C. Elegans Recognition from Images

Bo Xue
Electrical Engineering and Computer Science
University of Michigan, Ann Arbor
`xuebo@umich.edu`
Project Codes: GitHub

Abstract

Caenorhabditis elegans nematodes are powerful genetic model organisms which are instrumental for research on a wide range of biological questions. Sometimes it is very useful to track each individual worm, thus it is necessary to run an algorithm to recognize worms from different shoots. In this paper, we try different ways to extract features and analyze individual's distinctness from others. After many ways tried, we give a conclusion and a suggestion on the possible improvements.

1 Introduction

Caenorhabditis elegans nematodes are broadly used in biological research because of their simple shapes and easy to grow under a short generation time (about 3 days). In some research projects, it is necessary to track each individual in a short or even longer time. Thus a large population of nematodes could come with a great cost, because of the need to manually track and analyze the features in a large data pool. This calls the need of an algorithm to automatically track and analyze each individual's feature.

In this article, we proposed a serial of ways to identify each worms including three parts of work: extract worms, extract features, and high-level analysis. For the last part, we implement different ways and show the corresponding results. Section 2 illustrates the methods we use. Section 3 shows how we put these methods into experiments. Section 4 gives the corresponding results. Section 5 concludes the results and draws possible improvements.

2 Method

For this part, we show the methods we choose to identify each worms based on different images. This work can be divided into three steps:

- Identify worm areas: For each images, we need to extract the worms' areas in order to get each individual's features and track them. All the following steps will be based on the areas generated by this step.
- Extract features: For each worm, we need to get its features, like area, length, and color mean.
- High-level analysis: Based on the features we get from previous step, analyze all the features' distribution and one worm's feature distribution in different images. From the analysis, we propose ways to identify worms and implement them.

2.1 Identify Worm Areas

First we need to make images grey-scale. After this, the main work for extracting worm areas is to increase the contrast of each images. Although there are a lot of methods to do this, we implement another way here.

Like the model shown in Figure 1, we project the original red line onto the blue line. The axes represent image intensity. The red line shows the original image, while blue line shows the enhanced image. In the low intensity part(in Figure 1 is less than 60), we lower the output intensity(in Figure 1 the range is 20, while the original intensity is 60). For the high intensity part, the range is shortened, which expands the range for medium intensity. This can be expressed in the following equation:

$$I' = \begin{cases} k_1 \times I, & \text{if } I \text{ is less than } f_a \\ k_2 \times (I - f_a + g_a), & \text{if } I \text{ is between } f_a \text{ and } f_b \\ k_3 \times (I - f_b + g_b), & \text{if } I \text{ is higher than } f_b \end{cases} \quad (1)$$

While I' is the enhanced intensity, I is the original intensity. k_1, k_2, k_3 are the slope of the three blue lines.

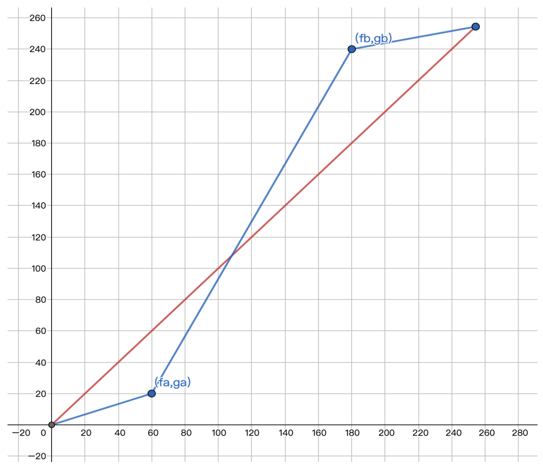

Figure 1: **Non-linear Enhancement** The model of non-linear enhancement. The axes represent image intensity. The red line shows the original image, while blue line shows the enhanced image

After enhancing images' intensity, we make the image a binarized image, use the Matlab function `bwconncomp` to get the connected area in the image, and remove the small connected areas.

2.2 Extract Features

To extract features(area, length, color mean) for each worm, we need to get each worm's middle line(lies in the middle of a worm) to illustrate the worm's length(area feature is already given by the Matlab function `bwconncomp`, and we can easily calculate the mean color value for each worm area). Here, we propose a new way to do this, shown in Figure 2.

The figure shows the binarized image for a single worm. The pink crosses show the sample of the boundary, which is given by the Matlab function. We can consider the middle line generated by linking all the middle points together, and the middle points are generated by find the opposite point for each boundary sample point. In order to get the opposite point for each boundary sample, we examine every consecutive three points from the boundary sample, and find the examined points that satisfies the following three requirements:

- The point in the examined group that is of the minimum distance from the operated point is the middle one instead of the side ones. This point is called the "goal point".
- The final goal point for each boundary sample should be of the minimum distance from the examined point among all the goal points.

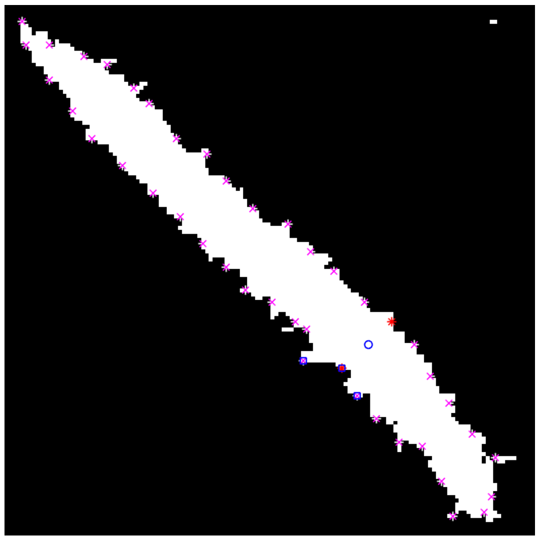

Figure 2: **Get Middle Points** Pink crosses show the sample of the boundary, red stars shows the operated point and the goal point, blue squares show the examined points, blue circle shows the middle point.

- Most of the points on the line that links the operated point and the goal point should be in the worm area. This ensures the algorithm stable for some irregular shaped worms(like severely winding shapes that head and tail are very closed to each other).

In Figure 2, the red stars show the operated point and the final goal point, the blue squares show the current examined group, which is also the "goal group" here, the blue circle shows the middle point for the operated point.

2.3 High-level Analysis

In order to identify worms from extracted features above, we consider in two ways: fixed binarizing threshold, considering different binarizing thresholds.

2.3.1 Fixed Binarizing Threshold

This method only uses fixed binarizing threshold to extract all the worm areas and features. The main idea of this method is to find the most similar worm in different images for each worm, which is the corresponding worm in that image. In order to define the similarity between two worms, we calculate the weighted distance of the two worms' features(each feature is a 5 dimensional vector). The following steps illustrate how to define the five weights for each feature channel.

The first three features' weights are assumed to be the same because they all represent color features and each color channel(RGB) are assumed to be of the same importance. For the $4th$ and the $5th$ weights, we try different values for them using use the normed weights(the vector of the five weights is normed to be 1) to find the weights that gives a maximum distance between all the worm and the worms' closest worm, which would make every worm more distinctive.

However, this may cause a problem that because the variance of each feature channel are totally different, the weights can be significantly affected. For example, the color features' range are $[0, 255]$, but the area feature may be hundreds or even thousands, thus the distance may depend greatly on the area and give a very low importance for the color feature. In order to fix this, we try to norm each feature by dividing the mean value of each feature channel for all the worms, which makes each feature's variance the same.

2.3.2 Considering Different Binarizing Thresholds

Because different image has different mean intensity, one worm would be different with the same binarizing threshold in different images. Thus different images require different thresholds, which a big noise generated by thresholds. We should consider the impact brought by different thresholds. Besides, with the increase of threshold, some worm may be divided into several parts because the middle part is much darker than the head and tail, which makes breakpoints on the curve showing feature value with regarding to different thresholds. In order to fix the above problems, we propose the following ways:

- Suppose we can have a fully correct answer of who is who in all the training images, and all the worm are divided into three groups by three colors(this is easily finish because the original images we have here only have worms with red, green and blue fluorescent). For the following process, do the same for the three colors.
- Like the above method, we need to calculate the distance of two worms, i.e., to find weights for color, area, length. In order to do this, we need the final weights fit the correct result well. Basically, the main goal is to try different weights and find the best one. However, different from the above method, the distance of two worms in one feature is the distance of the two lines representing the two worms(each line represents feature value with regarding to different thresholds), instead of the direct weighted difference for each feature channel. Compared with the latter method, the former method can take the error of binary-image-threshold into account, which is worse. In this step, the objective function is the sum of the distance between all the worms that are actually the same worm given by the truth, i.e. $\sum_{h=1}^{3}\sum_{i,j} distance_{i,j} \times weight_h$, where $h = 1, 2, 3$ is the number of feature channel, i and j represent different lines of the same worm from ground truth.
- To define what is the distance of two lines, we need to know that, besides the fact that the distance should only count on the points of the two lines that have the same threshold value, we have to admit that, because worms with higher threshold can represent more on the actual worm area instead of some dark background, the points with higher threshold should be counted with higher weight, e.g. like $e^{\omega t}$, where t is the threshold, ω is another parameter that we need to trained/tried.
- In order to remove the breakpoints, we do this: when detected there is breakpoints(if there is a huge difference with little change of threshold), increase the brightness of the area generated by the former threshold, then extract all the worm areas again until there is no breakpoints.

So totally there are 4 parameters we need to train/try, weights of (color, area, length), ω.

3 Experiment

We implement all the above methods with the images uploaded on GitHub. For the Fixed Binarizing Threshold method, our test images are like the following one shown in Figure 3. The image is merged from different colored images captured by different filters, so there are 4 colors here. For the Considering Different Binarizing Thresholds method, our test images are like the following one shown in Figure 4. The image is the original image, which is not merged, so we have single color in one image(here totally 3 colors: red, green, blue).

4 Results

4.1 Fixed Binarizing Threshold

After trying different weights for area and length feature(colors' weights are correspondingly changes because the weight is normed one), the change of $M-th$ minimum distance between all different worms is shown in Figure 5 and Figure 6. From these figures, we can see that the optimal weight locates at the border, which means colors' weights are almost zero here. This is a bad thing because this weight does not take the color feature into account. This happens because the variances of area feature and length feature are much higher than the colors'. Another problem we can find here is that

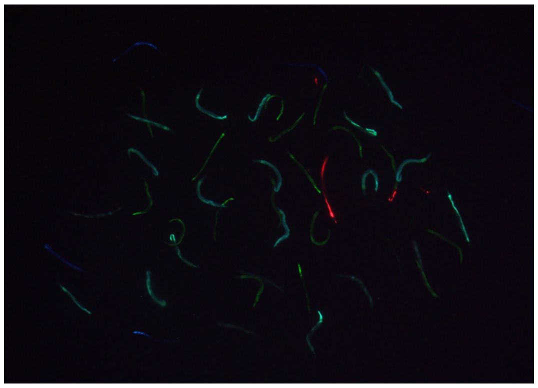

Figure 3: **Original Image for Fixed Binarizing Threshold** The image is merged from different colored images captured by different filters. The worms are in totally 4 colors.

Figure 4: **Original Image for Considering Different Binarizing Thresholds** The image is not merged and this is captured by a red filter so there are only red worms here.

for different minimum distance(i.e. different M), the optimal weights can change greatly, and from other results by different M values, we find the change of optimal weights is randomly and severe. This is also bad because our expected weight is stable no matter what the parameters change.

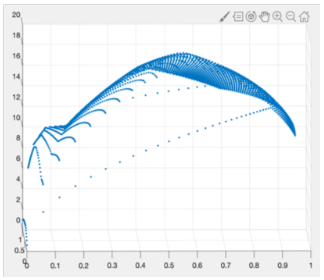

Figure 5: $3-th$ minimum distance. x, y axes are weights, z axis is the distance.

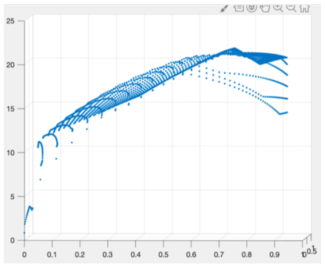

Figure 6: $6-th$ minimum distance. x, y axes are weights, z axis is the distance.

After normalizing the feature by dividing each feature's mean value, we get the results shown in Figure 7 and Figure 8. From these figures we can see that the optimal weight does take the color information into account but for different images, the optimal weights are still different. To choose the final weight, we do this way: if the peak of one image is flat, then this image's optimal weight should have a less heavier weight when calculating the final weight because its distance changes relatively less with the change of weights compared with the image with higher peak.

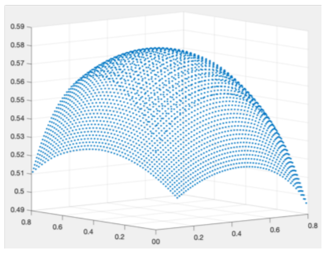

Figure 7: Overall distance for image1.

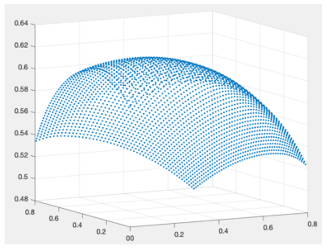

Figure 8: Overall distance for image2.

Table 1: Identification behavior

Identification Accuracy	72.7%
Identification Rate	8.5%

Using the weight given by the above way, we get a final result identifying worms shown in Table 1. From the table we find that the identification accuracy is high enough but the identification rate(the ability to identify) is very low. The data are calculated from 4 images, 120 worms.

4.2 Considering Different Binarizing Thresholds

Considering different binarizing thresholds, we plot each feature channel's value with regard to the change of threshold. And some of the results are shown in Figure 9 and Figure 10. From these figures, we can see the following problems:

- Different worms' feature lines intersect with each other, which means we can not recognize different worms.
- Distance between the same worm's lines is too small to distinguish worms.
- There are many breakpoints, which leads to incomplete information extraction for each worm.

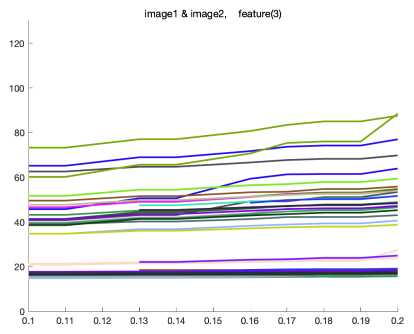

Figure 9: Blue color's value with threshold from 0.1 to 0.2.

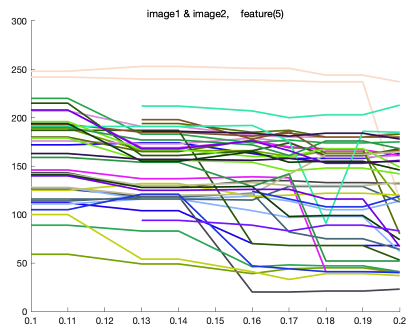

Figure 10: Length's value with threshold from 0.1 to 0.2.

After removing breakpoints and trying different weights for intensity, area, length and ω. And the overall optimal weights are shown in Table 2. We can see that the optimal weights do not take the area and length feature into consideration, which is not expected.

Table 2: Optimal Weights

intensity	1.0000
area	0.0012
length	0.0012
ω	0.0000

5 Conclusion

In this article, we propose different ways to identify worms from different images. The results we get are not good enough, which gives a very low ability of identification. This means there are still possible improvements for this topic, which are shown below:

- Get better input images: In our input image, the worms can be just partly seen because the fluorescent on the worm is just partly distributed. And the brightness of our input images' worms is not equal in every parts of the worm. This may cause that our worm' shape is not accurate and not the same under different thresholds.
- In order to recognize each worm, we need to extract some features that is totally stable among worms, like the color distribution. But this should be done after ensuring that our shape of worms is accurate, otherwise we color distribution is also not accurate.
- Before extracting features and analyzing features, we should focus more on how to extract worm's areas and middle line accurately, otherwise our data is not trustful.

6 Acknowledgment

Thanks very much for the help from Prof. Anastasopoulos and Dr. Gourgou. They really taught me a lot and made me know more. This direct study is very meaningful to me and gives me an experience of self-exploring research.