Hybrid Method of Biomedical Image Segmentation

Ming-Hung Hung
Department of Electrical Engineering and Computer Science,
Case Western Reserve University, Cleveland, OH, Email: mxh218@case.edu

Abstract – In this paper we present a general frame work for axon segmentation in retinal image and count the number of axons. The algorithm contains three phases. First phase is preprocessing. The goal of this phase is to get a binary image with cell objects in it. In this section, we use two different thresholding methods one is FCM clustering and the other is Otsu's Method and I also compare the results of both methods. The next phase is separate cell objects from the binary image we get after phase 1. Here we separate the objects based on its size and shape. To separate objects by shape, we use V/H scan line shape detection to check the smoothness of objects. The last phase is to count the number of object in a binary image.

I. INTRODUCTION

Optical fundus assessment is widely used in medical community for many reasons, such as the optical sensor is more sensitive than human eyes especial gray-scale. For human vision, on an average, we only can tell the difference between gray levels is more than 10-20 (on 0-255). But for highly sensitive sensors, it can tell up to 22bit (about 4 million) gray levels. Thus a machine vision can see more details. Another reason we use optical fundus assessment is to reduce processing time. Like we are trying to do in this paper (to count the number of axons or cells), there may be several ten thousands of axons in optical nerve image. If we count all of them by ourselves, it may take several days to do it.

In this paper, our goal is to locate and count closed bundles in an optical nerve image. To achieve this, first we need to separate the objects we want from the image. Thresholding is a fundamental approach to segmentation and when the intensity of objects and background are differ enough it can achieve pretty well results. Unfortunately, that is not the case in our input image. Thus we need to use different kinds of image segmentation techniques to separate axons from the image.

II. ALGORITHM

PHASE 1: Preprocessing

Step 1: Image Enhancement
Figure 1 is the input image for the algorithm to be developed. From the image, we can observe that this image is contaminated by some ripples and some boundaries of cells are very clear but some are very ambiguous. Thus, the first step is to perform histogram equalization and enhance the contrast. The anticipated result is to make the intensity distribution is bimodal.

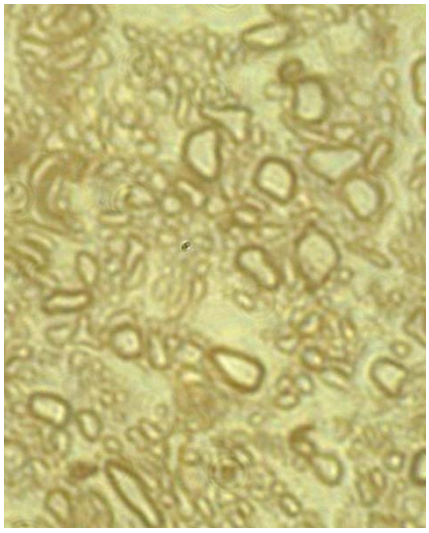

Figure 1.

Step 2: Thresholding
In this step, we will use thresholding techniques to separate background and cell's boundary from the image. Two thresholding methods will be introduced.

Fuzzy C-Mean (FCM) Clustering Method - The main process of FCM [1] is performed under feature space. To get the feature space, we use the intensity of different color layer from input image (Figure 1) as the 3-D feature space. Since the intensity of the same pixels in different color layer are differ. Also, in this paper we use a supervised FCM. The idea is try to make the result more reliable with the help of defining pixels characteristic of each class type (boundary and back ground) by a human

expert. Figure 2 is training set in feature space, and circle markers are the center of each class.

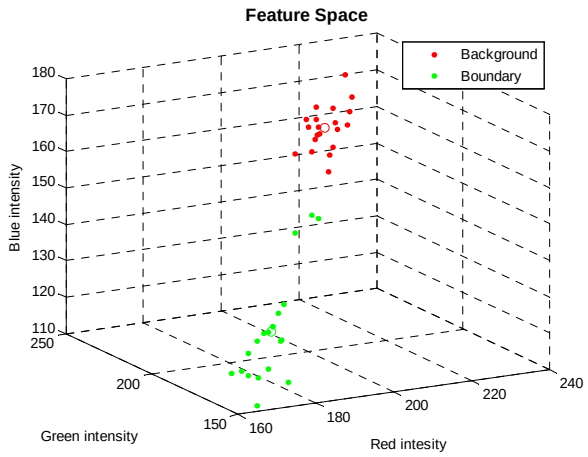

Figure 2. Training set in feature space

To begin the FCM algorithm, we set membership to each class as u_{ij} and class center as v_j (class centers come from the training set). Where *j* is the classes and *i* the pixel number, x_i is a vector of the gray levels in the images. Second, set a cost function *J* as

$$J = \sum_{j=1}^{Nc} \sum_{i=1}^{N} u_{ij}^2 \left\| x_i - v_j \right\|^2$$

Where *N* is the sum on the pixels and *Nc* is the sum on the classes. First we need to calculate v_j .The formula to compute the cluster centers v_j is:

$$\hat{v}_j = \frac{\sum_{i=1}^{N} u_{ij}^2 x_i}{\sum_{i=1}^{N} u_{ij}^2} \qquad (1)$$

For the pixel *i* is of class *j*, $u_{ij} = 1$. Otherwise, $u_{ij} = 0$. Thus to calculate the center of each class, equation (1) can reduce to just calculating the mean of points of each class. The next step is to minimize the cost function *J*. To minimize *J* with respect to u_{ij} we have

$$\hat{u}_{ij} = \frac{1/\left\| x_i - v_j \right\|^2}{\sum_{k=1}^{N_c} 1/\left\| x_i - v_k \right\|^2}$$

Finally, we will get a set of similarity map (probability map) u_{ij} for each class [1]. Figure 3 shows the result.

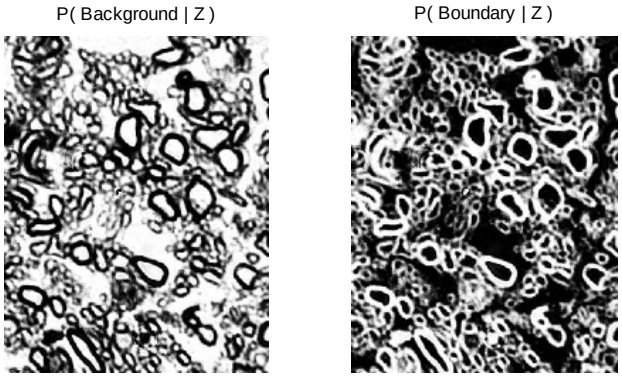

Figure 3. Probability maps

The outputs of the FCM algorithm are probability maps for each tissue. We then assign each pixel to the most probable class; this would get a binary image of each class (Figure 4.).

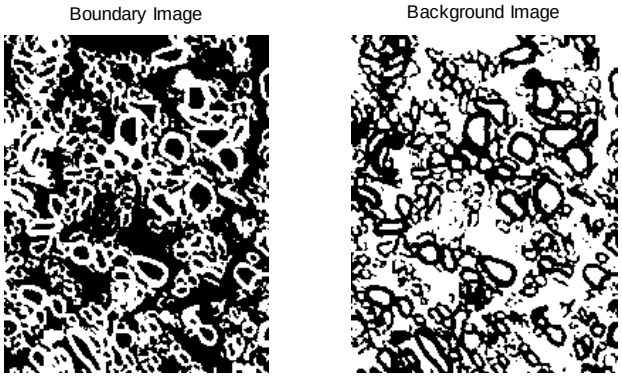

Figure 4.

Localized Otsu's Thresholding Method - Otsu's method chooses the threshold to minimize the intra-class variance of the black and white pixels. It treats normalized histogram as a discrete probability density function, as in

$$p_r(r_q) = \frac{n_q}{n}, \quad q = 0, 1, \ldots, L\text{-}1$$

Where n is the total number of pixels in the image, n_q is the number of pixels that have intensity level r_q, and L is the total number of possible intensity levels in the image. Otsu's method chooses the threshold value k that maximizes the between-class variance [2].

In our case, if we apply the Otsu's method on the entire image the result will not well. Since the intensity of cell's boundary is differ in different area of the image. And I also apply a 4x4 Median Filter to get rid of some noisy due to the reflection of the fluid. Figure 5 is a magnified image of Figure 1 after applying Otsu's method with/without image division. Clearly from this image, Otsu's method with division and median has better result (cell won't mix up with boundary).

Otsu Method with division & Median Filter

Otsu Method without division

Figure 5.

PHASE 2: Image Segmentation

In this section, we will perform image segmentation based on the priori knowledge of the characteristic of objects we are interest in.

Step 1: Segment Image by Size

From the result after above steps, we will have binary image contain cell (or cell's boundary) and background objects in it (see Figure 4). Usually the size of background is bigger than cell. Thus, we can delete those objects whose size greater than the predefined threshold. We can easily determine the threshold by knowing magnification rate, resolution of the sensor and the size of cell we are looking for. Figure 6 shows the results before/after segmentation by size.

Step 2: Segment Image by Shape (H/V Scan Line Shape Detection)

The result from previous step (right image of Figure 6) still contains many background objects. Thus, we will further remove it based on its shape characteristic. It is reasonable to assume the shape of the cell is round and smooth. Hence, we can detect the smoothness of each object in the image.

Before Removing Objects By Size

After Removing Objects By Size

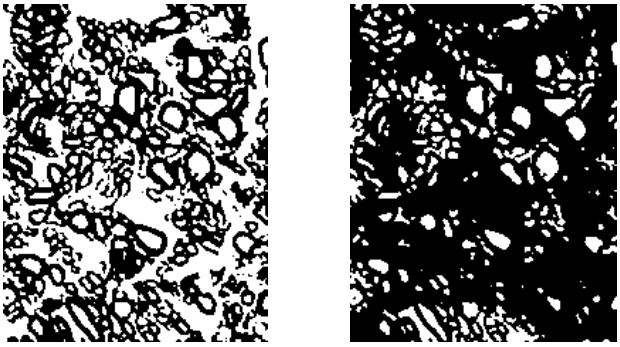

Figure 6.

To detect the smoothness, we use a horizontal and a vertical scan line passing through all objects. If the boundary of the object has more than four "disjoint" intersection points with the H/V scan line (Figure 8), we catalog the object as background. Because we are dealing with the image in discrete domain, we need to do some extra detection to avoid following situation.

Figure 7.

Figure 8.

Figure 7 and 8 are magnified images of cell's boundary. When horizontal scan line passing through the area circled in red, we will get more than four "continues" intersection points. Without any posterior process, the system will regard this object as background (actually it is a cell). To avoid this kind of misjudgment, we need to

further detect the continuity of those intersection points by checking its coordinates.

Figure 9 shows the result before/after segmentation by shape.

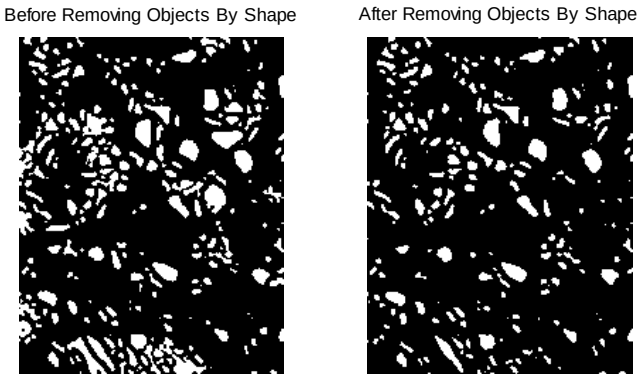

Figure 9.

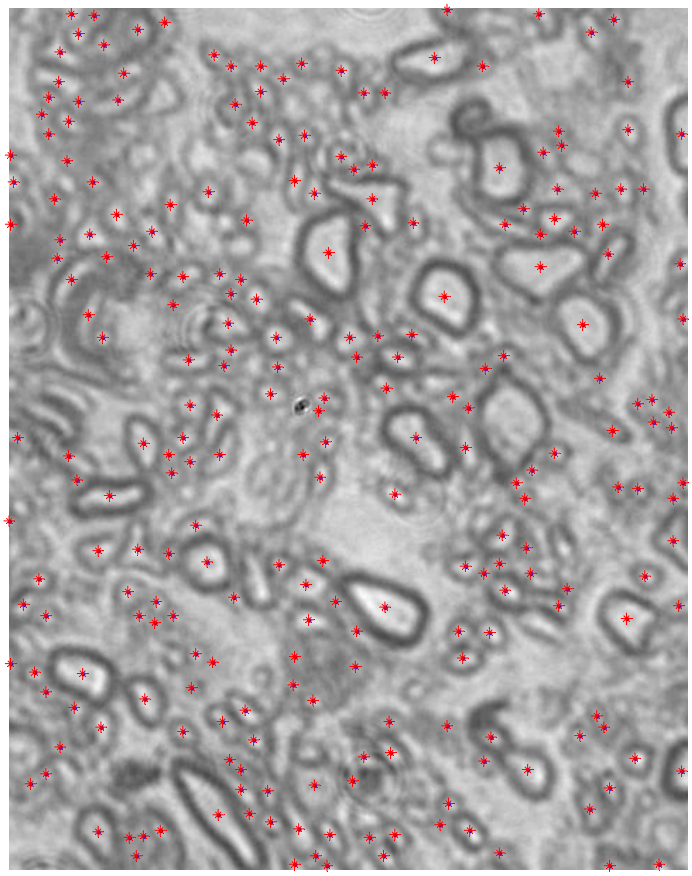

Figure 10.

PHASE 3: Count the Number of Cells

To count the number of objects in a binary image, we scan all pixels and check its connectivity (In the binary image, value 1 represents object and value 0 represents background). When we meet first non-zero value, we give that pixels a label value start from 1. Next, when we meet another non-zero value, we check its surrounding's label value. If the surrounding label values are all zero, we give that pixel a label value 2, otherwise, we give the pixels t a label value the same as surrounding label value. Repeat above steps until all pixels are examined [3]. The objects number is then the maximum label value. Figure 10 shows the result. In Figure 10, cells are marked by red asterisks and the number of the axons is 246.

III. DISCUSSION

In phase 1, our aim is to get a binary image which contains as much cell objects as possible. I tried two methods, one is FCM clustering algorithm and the other is Otsu's method.

The result of FCM clustering (Figure 4.R) is not good (compared with Otsu's Method (Figure 6.L)). In Figure 4 many cells still connect with background, especially in upper area. The performance of FCM depends on the image you choose to build the feature space. In this paper I use Red, Green and Blue intensity of the original image to build the feature space. Since R,G and B image come from the same image, their intensity distribution shares the same characteristic. Thus, this decreases the benefit of FCM algorithm. To overcome this problem, we need more images come from different type of sensor, such as infrared rays, X-ray or MRI image …etc, as input images to build the feature space. Also, for a supervised FCM, the training set defined by our self will greatly influence the result. To have a better result, the more training set points we choose the better result we'll get. And the training set points we choose for each class must cover all situations as possible as we can.

The next thresholding method I used in this paper is Otsu's method. As mentioned in previous section, localized Otsu's method has better performance, but we still lose several cell objects after applying Otsu's method. That is because the input image is highly contaminated.

Another reason is the color of nucleus is the same as the background. This makes the recognition of small cells more difficult.

In phase 2, we are trying to remove background (noise) objects. The idea of removing object by size is very simple. Just calculate the number of pixels of each object. We have a useful function BWLABEL in Matlab. This function can count the number of objects and it also can label which pixel belong to which object by the method I mentioned in previous section, and the output of the function contains two data, one is the number of objects and the other is an image of input in label domain. To calculate the *i*th object size, we just count the number of *i* in label domain.

The difficulty of phase 2 is to remove object by shape. At first, I try to use signature of the image. But in the input image, cells have different kinds of shape. If we use signature method, the detection will become very complex. Thus I use H/V scan line to detect the shape.

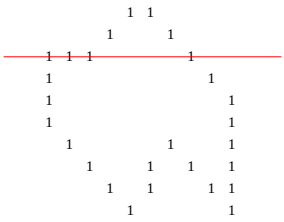

Figure 11.

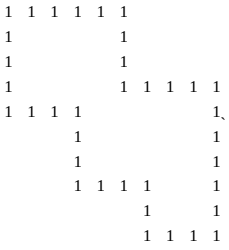

Figure 12.

Figure 11 is a binary boundary image. To detect the number of intersection points of boundary and V/H scan line, we add the values along columns and rows. The results represent the number of intersection points. To further determine “disjoint” intersection points, we compare the difference of its coordinates. For example, in Figure 11 the y-coordinates of intersection points of boundary and red line are Y = [1, 2, 3, 8]. Then we perform circular shift of Y one element to the left, we get Y’ = [2, 3, 8, 1]. Next, we calculate |Y’-Y| = [1, 1, 5, 7]. The number of non-one elements is the number of “disjoint” intersection points. In the example, the number of “disjoint” intersection points is 2.

H/V scan line shape detection is simple and fast, but it cannot deal with the condition like Figure 12. Figure 13 shows the result of over-counting due to the failure of H/V scan line shape detection.

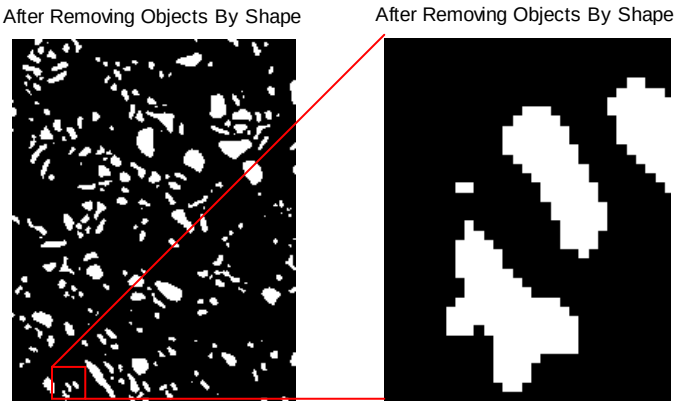

Figure 13.

APPENDIX

In Appendix, file **FinalProject (Main).m** is the main program. And **FinalProject (FCM).m** only performs supervised FCM algorithm. The result of **FinalProject (FCM).m** will store in the file **DataTmp.mat**. Last, **ExpertDef.m** is function for supervised Fuzzy C-Mean.

REFERENCES

1. **Bezdek, James C.** *Pattern Recognition with Fuzzy Objective Function Algorithms.* s.l. : Kluwer Academic Publishers, 1981.

2. **Rafael C. Gonzalez, Richard E. Woods.** *Digital Image Processing 2nd.* s.l. : Prentice-Hall, Inc., 2002.

3. **Milan Sonka, Vaclav Hlavac, Roger Boyle.** *Image Processing, Analysis, and Machine Vision.* s.l. : Thomson-Engineering, 1998(2 edition).