A Hole-filling Algorithm for Automated Axon Counting

Jonathan Wallace

Department of Biomedical Engineering,

Case Western Reserve University, Cleveland, OH, Email: Jonathan.Wallace@cwru.edu

Abstract

This paper presents an algorithm for the automated counting of axons in microscope images of cross-sections of the optic nerve. The approach uses a hole-filling algorithm to identify closed bundles in the image. Six images in which axons were labeled by a trained observer (TO) were used as test data for the algorithm. Results show that the algorithm counted the axons with an average relative error (n=5) of (34±11) %. This error is far too large for the algorithm to be useful at present. It is believed however that this error is due primarily to the image segmentation method used, and that the algorithm could be improved with some modifications to increase performance.

KEYWORDS

Automated counting, optic nerve, imaging/image processing/image analysis

INTRODUCTION

The ability to accurately count the number of axons in a cross-sectional microscope image of the optic nerve may serve as a method by which to quantify the number of rods and cones present on the retina. Manually counting the axons in such an image, and especially in large sets of such images, would prove to be an overwhelming task. A fully automated method which could reliably count the axons in such an image would allow for large data sets to be quickly analyzed.

Commercial software for histological evaluation is available (e.g., BioQuant). However, the purpose here was to develop an algorithm customized to the specific application presented above. The goal was to develop an approach which is specific to the types of data available for such research so as to optimize the ability to quantify such images.

The problem of axon counting is really one of segmentation. The ideal algorithm would count every axon in the image while correctly rejecting extracellular space. The difficulty in accomplishing such a task lies in the low contrast between the interiors of axons and the space between axons (see Figure 1 below). There is, however, a good deal of contrast in such images between the borders of axons and the rest of the image (i.e. both the regions inside of axons and between axons). It is for this reason that the scope of the problem was narrowed to correctly labeling only axons which are completely contained within the image. Thus, the axon borders could be segmented and under this construct, an axon could be viewed as a closed object completely contained within the image.

This definition of an axon is not sufficient for segmentation, as it is quite probable that extracellular space exists within the image as a closed object. Methods for distinguishing such regions are proposed in the discussion that will follow. However, it was believed that a good deal of extracellular space could be correctly rejected by the application of a hole-filling morphological algorithm. Such an algorithm is normally used to fill holes in an image, as the name would suggest. However, in order to accomplish this, the holes themselves must be identified. By applying a hole-filling algorithm and considering not the filled image which is normally desired, but rather the holes as calculated by such an algorithm, one can isolate all of the closed objects contained within an image.

The approach of this work was to study whether or not such a design would produce good results, as compared to the axon counts supplied by a trained observer. Additionally, the question of whether or not such an approach is plausible, despite the results obtained, was posed and discussed.

IMPLEMENTATION

The basic idea of the algorithm was to use a hole-filling morphological process [1] to identify and count closed bundles in the image. Some pre-processing of the image was required to provide a suitable format for this approach (a segmented binary image.) Additionally, some post-processing was used to remove small bridges between objects, merge objects in close proximity to one another, and shrink objects to single points to identify the location and number of objects. The images used were supplied in the tagged image file format (.tiff). The algorithm was implemented using MATLAB.

The basic outline of the algorithm is as follows: (1) import the .tiff input file into the workspace; (2) translate the image to grayscale; (3) adjust image contrast to span full 0-255 range; (4) use Otsu's Method [2] to optimally threshold the image; (5) apply morphological hole-filling algorithm to identify closed objects in the image; (6) perform a morphological erosion to remove small bridges which may connect adjacent closed objects; (7) morphologically shrink the image to reduce closed objects to single points (or rings in the case of closed objects with interior black pixels); (8) perform morphological closing of the image to merge objects that are located within close proximity of one another; (9) Reduce rings to single points by performing a morphological dilation to widen the single-pixel thick ring borders, applying the hole-filling algorithm to fill in the rings, and then morphologically shrinking the image to reduce rings to single points; (10) Count the points in the image to yield a cell count; (11) Label the original input image with the locations of the closed objects

Many of these steps were performed using functions provided in the MATLAB Image Processing Toolbox, such as imerode, imclose, imdilate, and bwmorph. Otsu's Method and the hole-filling algorithm were implemented by the author. The full source code for the algorithm is available in Appendix A.

TESTING

Given that the algorithm was designed to label only closed objects contained completely within the borders of the image, it was understood that axons which crossed the image border would not be recognized. Each result of the algorithm was therefore compared only to the number of axons counted by the TO which did not intersect the borders of the image (hereafter referred to as a standard count (SC)). The absolute and relative errors between the algorithm axon count (AC) and the SC were calculated for each of the six images (except one image in which the SC was zero and the relative error could not be computed.) The mean and standard deviation of the relative errors was then computed.

RESULTS AND DISCUSSION

Each of six images of the cross-section of an optic nerve was processed using the algorithm described above. The results were examined visually to ensure that the algorithm did indeed label closed objects contained completely within the images. Each of the labeled images along with its corresponding TO labeled image is shown in Figure 1. Figure 2 illustrates the performance of the algorithm on a larger image for which no TO labeled image was provided (and thus for which no analysis was performed.)

Upon examination of Figure 1, it is clear that the algorithm does in fact label some of the closed objects completely contained within the images. Some of the closed objects, however, are not labeled. It is believed that this is due to the low contrast in the images. In the initial development of the algorithm, a simple thresholding scheme was used in which the user adjusted the threshold level until it could be asserted visually that a good segmentation was achieved. In order to automate the algorithm, this approach was replaced with Otsu's Method for optimal thresholding which maximizes the between-class variance of the result. It is believed that this method is not sufficient to segment all of the closed objects. For example, in images A, C, and F it is clear that some of the closed objects are not labeled and were not segmented properly. If a better segmentation method were used, the results obtained may improve. Other approaches, such as fuzzy c-means classification, may perform better than the thresholding method that was implemented.

Another issue that has not been addressed is that of the possibility of closed extracellular space in the image. The hole-filling approach was implemented because of its ability to remove extracellular space connected to the edge of the image and isolate only closed objects. However, if an area of closed extracellular space, for example, within a ring of axons were present in the image, this area too would be improperly labeled as an axon. An advanced approach might use other characteristics besides a closed border to label axons. For example, shape and size along with other criteria might be used for segmentation after an initial classification has been performed.

The algorithm was never intended to count neurons which crossed the image border. Therefore if this approach is to be further developed it is suggested that after labeling the data a region of interest be defined which extends only to the labeled axons which are closest to the border. The number of axons per unit area could then be calculated, and an adjustment made to the final axon count to correct for the lost area in the image. Alternatively, a separate algorithm could be implemented to isolate only the axons which cross the image border, and the two counts could then simply be summed.

Values for the AC and SC of each image along with the errors associated with these values are shown in Table 1. The average relative error (n=5) was (34±11) %. The algorithm is obviously not sufficiently developed for use in practical application. By visually examining the results, however, it is clear that the algorithm does have some potential as it is successful in labeling closed objects. The idea of using a hole-filling algorithm for this particular application is novel, but the question must be raised as to whether or not such an approach would offer an improvement over a good segmentation algorithm. The reasoning behind using this approach was that it would eliminate extracellular space connected to the border of the image. However, if is eventually going to be necessary to implement other criteria for the labeling of axons, such as size and shape, there may be no need to worry about eliminating these regions as these characteristics may provide enough information to rule them out. If so, then using both steps in the same algorithm would be unnecessary and redundant.

SUMMARY

A hole-filling algorithm for the labeling and counting of axons in cross-sectional microscope images is presented. The algorithm was tested on six reference images which had been previously labeled by a trained observer (TO). The number of axons labeled by the TO is compared to the number labeled by the algorithm. The algorithm does not demonstrate adequate performance. Possible reasons for this are discussed, along with suggestions for future development.

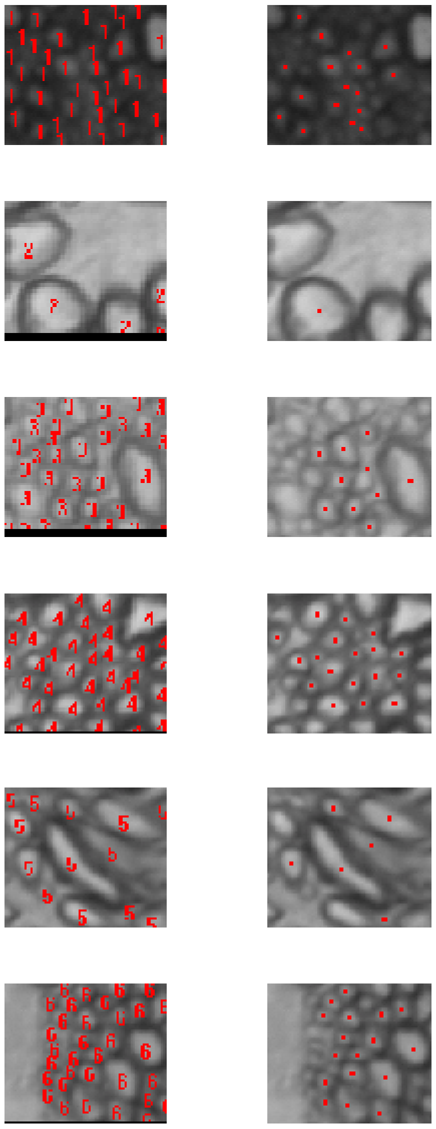

Figure 1 The left column shows the images labeled by a TO; the right column shows the corresponding images labeled by the algorithm; the test images will be referred to hereafter from top to bottom as A, B, C, D, E, and F

Image	SC	AC	Absolute Error	Relative Error
A	30	17	13	0.43
B	0	1	1	N/A
C	20	10	10	0.5
D	24	18	6	0.25
E	8	6	2	0.25
F	24	17	7	0.29

Table 1 Standard counts, algorithm counts, and the absolute and relative errors between these values for each of the six text images

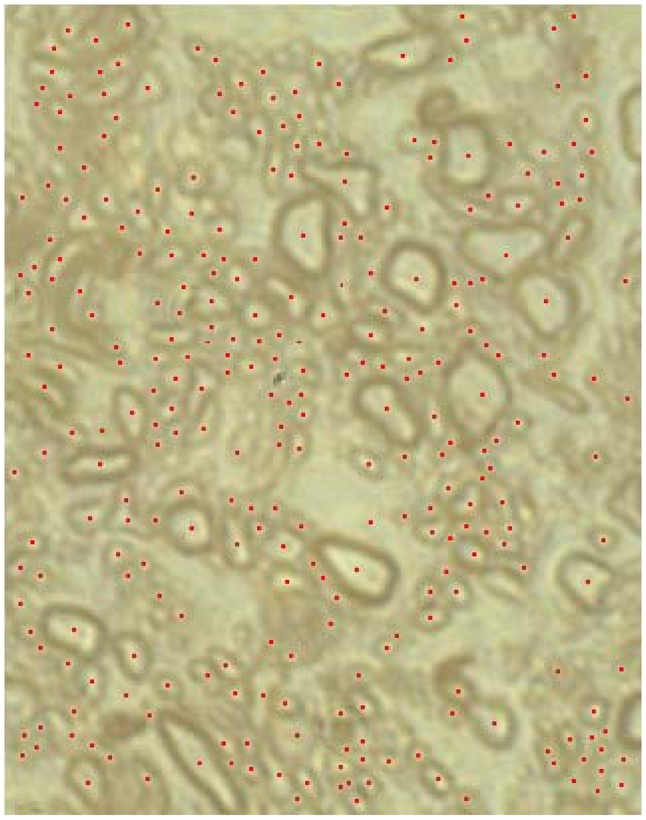

Figure 2 A large image algorithm labeled image

ACKNOWLEDGMENTS

TO labeled images and test data were provided by Dr. Howell at Case Western Reserve University (CWRU). Guidance and instruction were provided by Dr. Merat at CWRU. MATLAB was provided by the CWRU software center.

REFERENCES

[1] R. Gonzalez and R. Woods, *Digital Image Processing*, 3rd ed., Upper Saddle River, NJ: Pearson, pp. 660-662, 2007.

[2] N. Otsu, “A threshold selection method from gray-level histograms,” IEEE Trans. Sys., Man., Cyber., vol. 9, pp. 62–66, 1979.

Appendix A – Source Code

Main function:

PROCESS IMAGE

```
function [input labeled_neurons cell_count] = count_neurons(name)

%Read in the input file
input = read_image(['C:\Jon\eecs 490\Midterm Project\data\' name]);

%Prepare the image (translate to grayscale from RGB, threshold image using
%optimal Otsu method)
image = prepare_image(input);

%Calculate the image dimensions for later use
dim = size(image);

%Isolate the regions of neurons inside the cell membrane and not connected
%to the border of the image using a hole-filling morphological algorithm.
%This process will also return extracellular regions which do not intersect
%the border of the image
[filled_image holes] = fill_holes(image);

%Remove small bridges which may connect adjacent cells and extracellular
%areas
se1 = strel('disk',2);
eroded_holes = imerode(holes, se1);

%Shrink the objects returned from the hole-filling algorithm to single
%points
hole_locations = bwmorph(eroded_holes, 'shrink', Inf);

%Combine objects found in close proximity to one another
se2 = strel('disk', 5);
closed_hole_locations = imclose(hole_locations, se2);

%In the case of rings formed by the shrinking algorithm, fill the rings and
%then reduce rings to single points
blurred_closed_locations = imdilate(closed_hole_locations, se1);
filled_image = fill_holes(~blurred_closed_locations);
hole_locations = bwmorph(filled_image, 'shrink', Inf);

%Count the number of points in the image to calculate the number of cells
%in the image
cell_count = sum(sum(hole_locations));

%Label the objects in the original image which were counted as cells by
%creating a new image with red dots superimposed over the original image
labeled_neurons = input;
for i = 1:dim(1)
    for j = 1:dim(2)
        if(hole_locations(i,j)==1)
            if(i==1)
                k=0;
                l=1;
                m=1;
                n=1;
            elseif(i==255)
                k=1;
                l=0;
                m=1;
                n=1;
            elseif(j==1)
                k=1;
                l=1;
                m=0;
                n=1;
            elseif(j==255)
                k=1;
                l=1;
                m=1;
                n=0;
            else
                k=1;
                l=1;
                m=1;
                n=1;
            end
            labeled_neurons(i-k:i+l,j-m:j+n,1) = 255;
            labeled_neurons(i-k:i+l,j-m:j+n,2) = 0;
            labeled_neurons(i-k:i+l,j-m:j+n,3) = 0;
        end
    end
end
```

Other functions:

READ IMAGE

```
function image = read_image(name)
%EECS 490 Fall 2007
%Jonathan Wallace
%
%A simple program to load image data into the workspace
%
%The user should not need to call this function.

clc
image = importdata(name);
```

PREPARE IMAGE

```
function image = prepare_image(input)
%EECS 490 Fall 2007
%Jonathan Wallace
%
%The user should not need to call this function.

gray_image= rgb2gray(input);
enhancontr = imadjust(gray_image, stretchlim(gray_image),[], 1);
```

```
image = otsu_optimal(enhancontr);
```

OPTIMALLY THRESHOLD IMAGE

```
function output = otsu_optimal(input)
%EECS 490 Fall 2007
%Jonathan Wallace
%
%This function uses Otsu's method for optimal thresholding, which maximizes
%the between-class variance of the output. The user should not have to call
%this function.

dim = size(input);
NM = dim(1)*dim(2);

%Compute the normalized histogram of the input
H = zeros(1,256);
for i=1:dim(1)
    for j=1:dim(2)
        g = input(i,j)+1;
        H(g) = H(g)+1;
    end
end
p = H./NM;

%Compute the cumulative normailzed histogram
P = zeros(1,256);
P(1) = p(1);
for k = 2:256
    P(k) = P(k-1) + p(k);
end

%Compute the cumulative means m
m = zeros(1,256);
m(1) = 0;
for k = 2:256
    m(k) = (k-1)*p(k)+m(k-1);
end

%Compute the global intensity mean mg
mg = 0;
for k = 2:256;
    mg = mg + (k-1)*p(k);
end

%Calculate the between-class variances
v = zeros(1,256);
for k = 1:256
    if(P(k) == 0)
        v(k) = 0;
    else
        v(k) = ((mg*P(k)-m(k))^2)/(P(k)*(1-P(k)));
    end
end
```

```
%Find the maximum (or maxima)
thresh = 0;
maximum = 0;
maxima = 1;
for k = 1:256
    if (v(k)>maximum)
        maximum = v(k);
        thresh = k - 1;
        maxima = 1;
    elseif (v(k) == maximum)
        thresh = thresh + k - 1;
        maxima = maxima+1;
    end
end
thresh = thresh/maxima;

output = (input>=thresh);
```

FILL HOLES IN IMAGE

```
function [filled_image holes] = fill_holes(input)
%EECS 490 Fall 2007
%Jonathan Wallace

dim = size(input);
I = ~input;
Ic = input;
F = zeros(dim(1), dim(2));

F(1, :) = ~I(1, :);
F(dim(1), :) = ~I(dim(1), :);
F(:, 1) = ~I(:, 1);
F(:, dim(2)) = ~I(:, dim(2));
F = logical(F);

H = ~imreconstruct(F, Ic);

filled_image = H;
holes = H & Ic;
```

GENERATE FIGURE

```
[input_1 counted_1 cell_count_1] = count_neurons('Sample_1.tif');
[input_2 counted_2 cell_count_2] = count_neurons('Sample_2.tif');
[input_3 counted_3 cell_count_3] = count_neurons('Sample_3.tif');
[input_4 counted_4 cell_count_4] = count_neurons('Sample_4.tif');
[input_5 counted_5 cell_count_5] = count_neurons('Sample_5.tif');
[input_6 counted_6 cell_count_6] = count_neurons('Sample_6.tif');

standard_1 = read_image('C:\Jon\eecs 490\Midterm
Project\data\Sample_1_counted.tif');
standard_2 = read_image('C:\Jon\eecs 490\Midterm
Project\data\Sample_2_counted.tif');
standard_3 = read_image('C:\Jon\eecs 490\Midterm
Project\data\Sample_3_counted.tif');
```

```
standard_4 = read_image('C:\Jon\eecs 490\Midterm
Project\data\Sample_4_counted.tif');
standard_5 = read_image('C:\Jon\eecs 490\Midterm
Project\data\Sample_5_counted.tif');
standard_6 = read_image('C:\Jon\eecs 490\Midterm
Project\data\Sample_6_counted.tif');


figure
subplot(6,2,1); imshow(standard_1);
subplot(6,2,2);imshow(counted_1);
title('Abs Error = 13, Rel Error = 43%')
subplot(6,2,3); imshow(standard_2);
subplot(6,2,4);imshow(counted_2);
title('Abs Error = 1, Rel Error = N/A')
subplot(6,2,5); imshow(standard_3);
subplot(6,2,6);imshow(counted_3);
title('Abs Error = 10, Rel Error = 50%')
subplot(6,2,7); imshow(standard_4);
subplot(6,2,8);imshow(counted_4);
title('Abs Error = 18, Rel Error = 25%')
subplot(6,2,9); imshow(standard_5);
subplot(6,2,10);imshow(counted_5);
title('Abs Error = 2, Rel Error = 25%')
subplot(6,2,11); imshow(standard_6);
subplot(6,2,12);imshow(counted_6);
title('Abs Error = 7, Rel Error = 29%')
```