# Algorithm for Detecting and Counting Tightly Bundled Axons in Normal Optic Nerve

**Christopher Cook**

Department of Electrical Engineering and Computer Science,

Case Western Reserve University, Cleveland, OH, Email: christopher.cook@case.edu

**Abstract –** This paper demonstrates a way to effectively identify and count cross-sectioned Axons bundled in a tightly packed Optic nerve. The main challenge in this application is the extremely high cellular density and low contrast. The method described in this paper uses local adaptive thresholding followed by various morphological operations to segment individual cells. The segmented cell candidates are then evaluated for specific physical characteristics and the original image is augmented with an overlay showing counted cells and rejected cell candidates. The average accuracy among all 6 test scenarios was 87% correct identifications and having an overall count 6% below counts performed by a trained researcher.

KEYWORDS

Adaptive Filtering, Image Processing, Edge Detection, Biological, Axon, Segmentation, Morphological Operations.

## I. INTRODUCTION

Many past and recent biological studies focusing on individual single or multi-celled organisms have required study of the actual cells, but knowing the size of an entire population. Until recently this had to be completed by hand. A researcher would load slide after slide under a microscope and count each individual cell with a hand counter. This process takes a huge amount of time and was most likely completed on only a small subset, large enough to estimate the entire population.

Recent advents in technology have made it possible to accurately discriminate for and count cells of interest. The problem is that many of the optimized scenarios benefit from a relatively low cell density, high contrast [5], or the use of multicolored stains. These low cell densities are often achieved by placing a sample in suspension allowing he cells to float apart for easier identification. The benefit of low cell density is that the cells will only touch on rare occasion as viewed under the slide. In addition to the low density, everything is generally viewed from a top down orientation, which yields the most recognizable and repeatable patterns. When multicolored stains are used, simple color channel manipulation will often provide an adequate means to identify and count cells.

The problem in this case is the cells originate from a cross section of an axon nerve bundle. The nature of a nerve bundle yields an extremely high cell density. Large portions of the evaluated image have many of the cells in direct contact. Even with a stain to enhance the cell wall, accurately discriminating two touching cells as two separate cells is difficult. Also as a result of using a cell cross section is that the cells have no visible nuclei or internal features to aid in identification. Everything must be evaluated on size and color of the cell wall.

Common with many microscope images, our samples lacked good contrast, consistent focus, and uniform intensity. These limitations indicated the need for some type of local adaptive filter. Many methods include background filtering, histogram equalization [3], and watershed type flooding [6]. The industrial manufacturing environment also implements local image filtering through an interesting moving evaluation frame for stain and pattern detection.

Initial attempts tried implementing some sort of moving inspection frame within the target image similar to [2]. The

**Figure 1 – Greatly reduced image of nerve bundle**

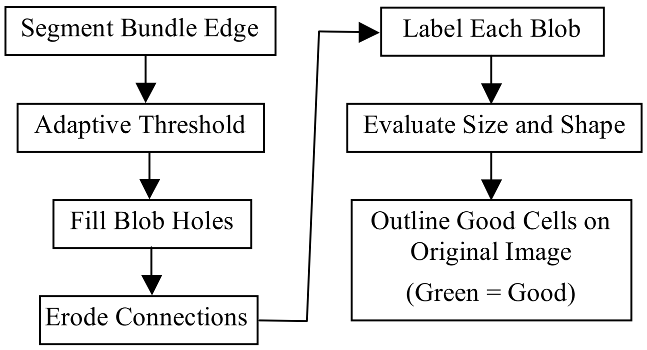


**Figure 2 – General outline of the algorithm process**

results were somewhat positive, but under the conditions of extreme gradients or constant intensity it behaved in an undesirable manor. Dark lines would appear in the middle of light backgrounds and vise-versa. Adaptive filtering [7] by evaluating each pixel compared to the surrounding NxN cells was the next logical choice as a modification to this method. This produced more consistent output across steep gradients and solid regions. It also turns out to be computationally simpler than a moving inspection window.

## II. METHOD

The basic process for positive cell identification is outlined in Figure 1.

### *SEGMENT BUNDLE EDGE*

The point of this step is to find the absolute outside boundary for the nerve bundle. This prevents any evaluation outside the desired nerve bundle area. The entire bundle image is evaluated at once by subtracting two different black and white images created at different threshold values to emphasize the bundle edge. All holes are filled in the resulting mask and it is eroded and dilated to remove noise. Any portion of the image lying outside of the bundle perimeter is changed to the average intensity of the entire bundle image.

### *ADAPTIVE THRESHOLD*

Prior to filtering the image is symmetrically padded on all sides to prevent any error from cells occurring on the edge. The adaptive threshold filter then converts the grayscale image into a black and white image by evaluating the center pixel against a threshold value set by the mean of all pixels contained within specified distance from the center.

### *FILL BLOB HOLES*

After adaptive thresholding the image needs additional processing to remove noise and separate any joined blobs. First, the black and white image is then evaluated to throw out any out any extremely large areas (>3000 pixels). Any white space this large is due to a void within the axon nerve bundle and should be ignored. Next, the every remaining blob has its holes filled to make it a solid object.

### *ERODE CONNECTIONS*

The white area representing possible cells is now eroded back. This gives two results: First, it helps separate any cells that may be touching each other and second, it removes some of the noise around the cell boundaries. The size of the structuring element in this step is critical so that it will provide proper cell separation, but avoid removing small cells.

### *LABEL EACH BLOB*

At this point the user has the option to remove any cells touching the border or let them remain for further analysis. Each blob is now given an indexed value using the method described in [1] to help individually identify each cell for later calculations.

### *EVALUATE SIZE AND SHAPE*

Each cell candidate is evaluated for area and perimeter to determine if it a real cell or just debris. Many more factors can be analyzed, but my general lack of knowledge in this area prevents advanced identification techniques.

### *OUTLINE GOOD CELLS*

Once the good cells have been identified, the original image is overlaid with a green outline of all the identified cells. Possible candidates thrown out during size and shape evaluation show up as a red overlay. This step serves primarily as verification for correct identification. It is also helpful to further tune system performance to achieve optimum results.

## III. RESULTS

This algorithm has two main user controllable parameters: adaptive filter size and boundary cell exclusion. The size of the adaptive filter is greatly influenced by the average cell size. Too large of a filter window won't pick up the slight variations of small cells and too small of a filter effectively enhances noise. For all of the scenarios presented the adaptive filter size stays constant at 11 x 11 pixels. The boundary cell exclusion flag will discard any cells lying on the edge of the image. For all scenarios presented this flag will remain at 1 to discard all edge cells.

### *SCENARIO 1*

The image given for scenario 1 as seen in Figure 3 is fairly dark and has extremely low contrast. Both the original and

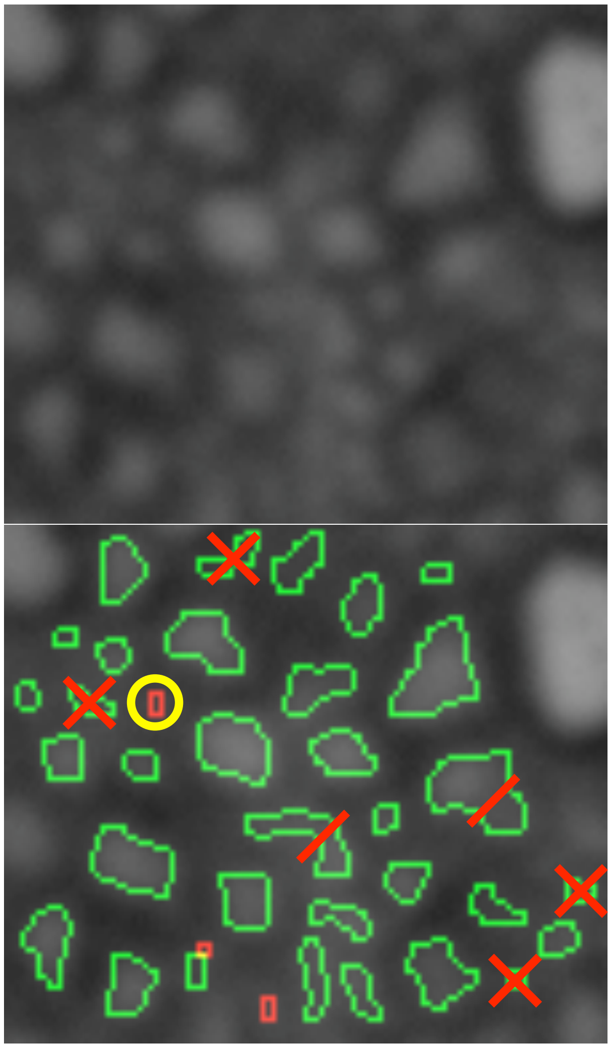


**Figure 3 – Scenario 1 (original – top / counted – below)**
**Green Outlines – Identified Cells**
**Red Outlines – Rejected Cells**
**Red Xs – False Positives**
**Yellow Circles – False Negatives**
**Red Slashes – 2 Cells identified as 1 cell**

count overlaid images are shown for the first scenario, but the original image will be omitted for all remaining scenarios. Most of the cells are fairly uniform in size with the exception of one vastly larger cell on the image edge. The algorithm in this case yielded 4 alpha errors and 3 beta errors. The overall poor quality of this image gives rise to the high alpha and beta error rates.

## *SCENARIO II*

In figure 4 the overall intensity of the image is much brighter. The algorithm had little trouble picking out the inside edge of the large cells. One will notice that more than 2 cells exist in this image but the two cells in the bottom right corner both come into contact with the edge and are therefore removed from the initial cell segmentation and

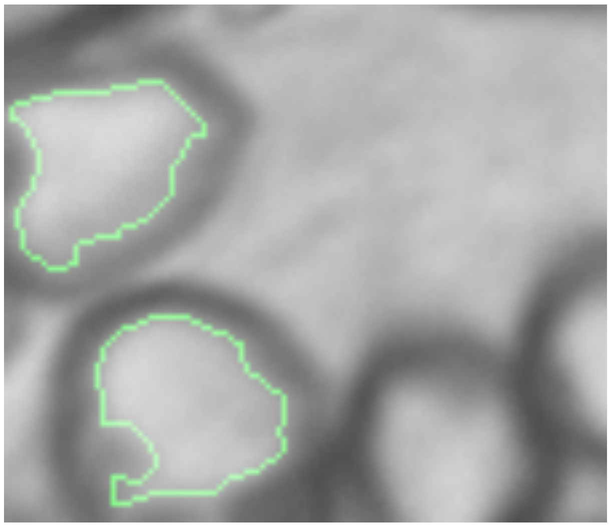


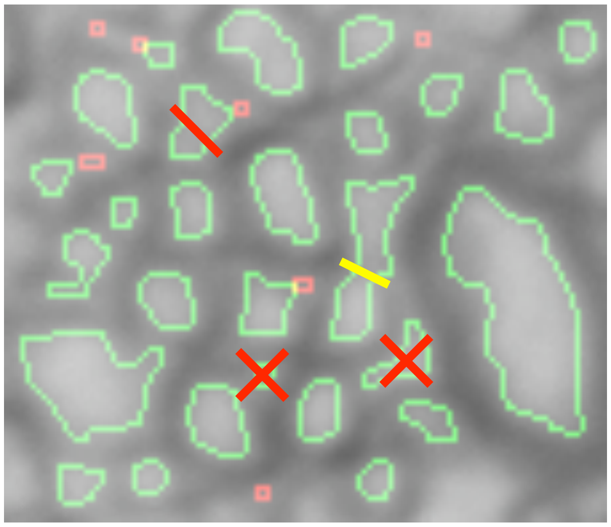
**Figure 4 – Scenario 2**

identification process. If the user chose to include boundary cells then those two cells would also be included. It was particularly difficult to optimize the algorithm to handle both low and high contrast images without having to adjust parameters in between simulations. The need to not adjust parameters between runs is essential for providing a completely automated counting process. The accuracy for this image was 100% with no alpha or beta errors.

## SCENARIO III

Figure 5 shows some interesting output evaluations from this algorithm. The two incorrectly identified cells are shown with red Xs. There are also two tightly positioned cells incorrectly identified as one cell. The yellow slash in figure 5 indicates the algorithm correctly counted the cell, but incorrectly identified the shape of the cell. It is interesting to point out that one of the other cells on the far left also has an incorrectly identified boundary.

**Figure 5 – Scenario 3**



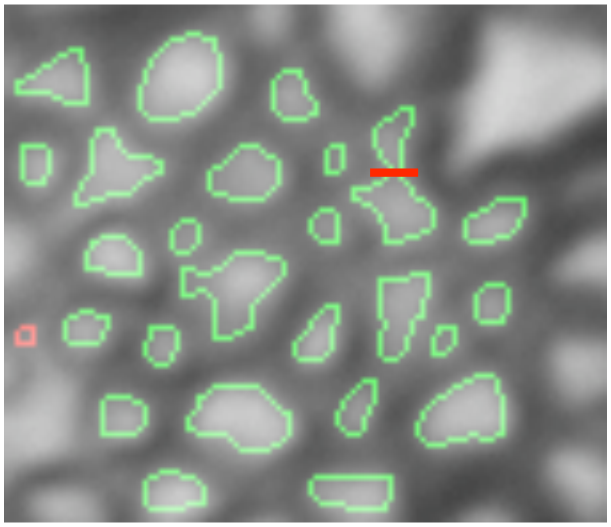


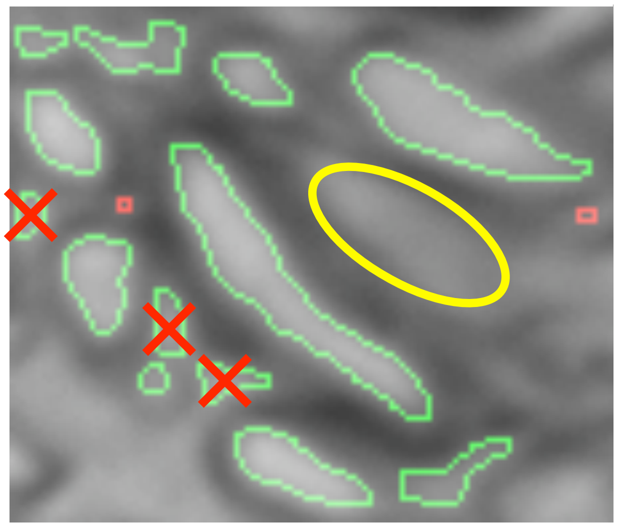
**Figure 6 – Scenario 4**

An interesting aspect of scenario 3 is how the algorithm correctly rejected the 7 smaller candidate cells. This image produced an overall accuracy of 89% with 2 alpha errors and 1 beta error.

*SCENARIO IV*

The algorithm processed the image shown in figure 6 for scenario 4 with a great deal of accuracy. The results were almost flawless. The one problem as shown in figure 6 is that again two adjacent cells were identified as a single cell. Attempts were made to fix this problem by eroding more of the cell away during processing, but that had the adverse effect of preventing the smallest cells from being detected. The accuracy on this image is 96% with no alpha errors and one beta error.

**Figure 7 – Scenario 5**



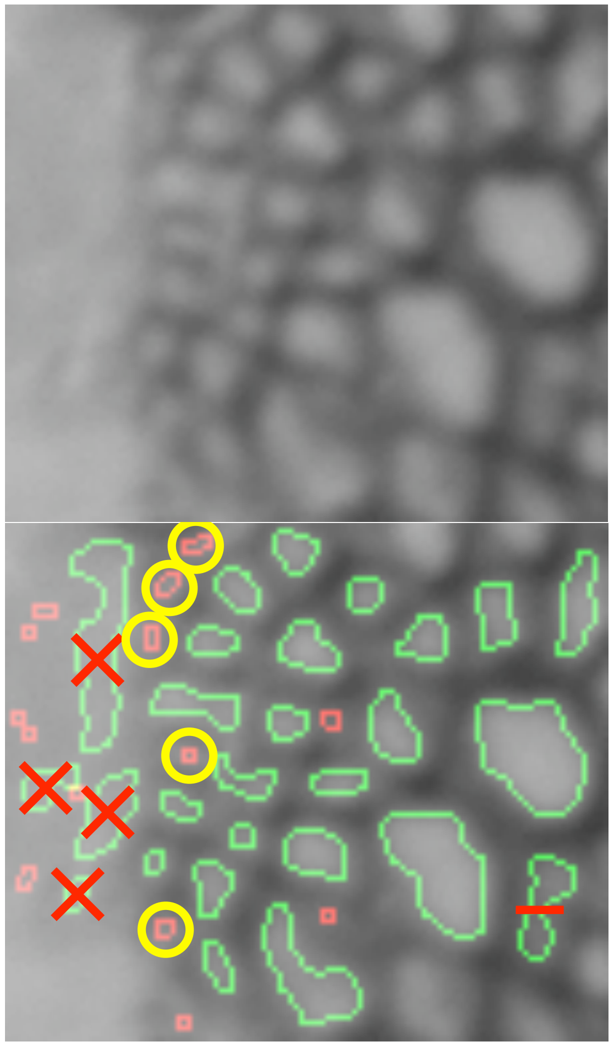


**Figure 8 – Scenario 6 (original – top / counted – below)**

*SCENARIO V*

The image for scenario 5 was quite different than any of the other images. As seen in figure 7, many of the cells were long oval shaped sections surrounded by “white space” that is easily confused as a cell. The 3 alpha errors occur where the non-cell area around a cell is a similar size and shape to a normal cell. The completely missed area identified by a yellow ellipse does not show the typical cell wall gradient on the right side. This is what caused the algorithm to completely ignore that area. It just appears as a large blank area. The detection accuracy in this scenario was only 78%.

*SCENARIO VI*

This scenario proved to be the most difficult. The image provided and shown in figure 8 shows what looks like the edge of the nerve bundle where the axons become smaller and smaller to seemingly vanish at the edge. It was very difficult for the algorithm to distinguish between cell and space along the edge region. A total of 6 cells were missed including the half counted cell to the right of the image.

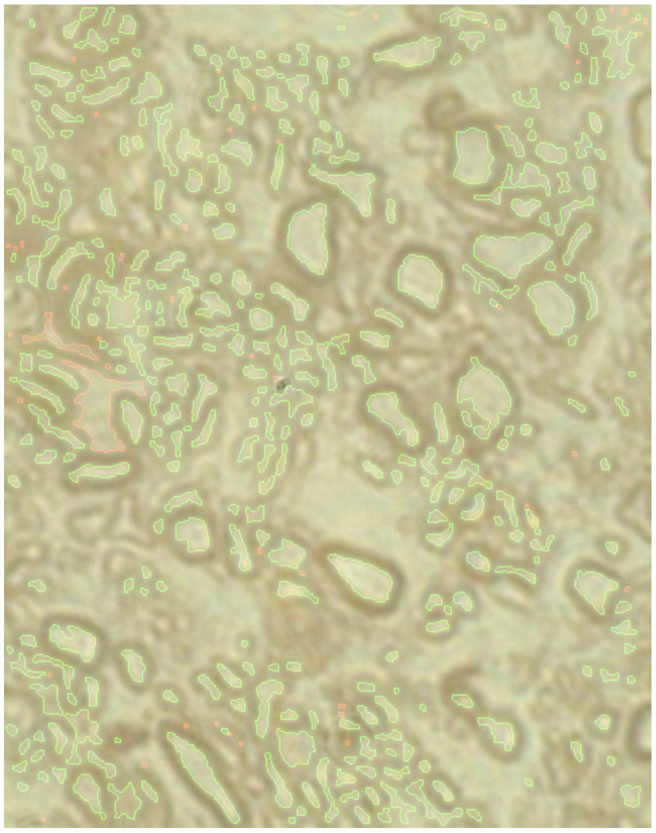


**Figure 9 – Original assignment image (480x610 pixels)**

The algorithm also determined that 4 cells existed in the void space to the left when, in fact, nothing does. The accuracy in this image is only at about 80%. That number is deceivingly high since the number of total cells in the image is larger than the other images.

Strong indications about the versatility of this algorithm come from the results in scenarios 5 and 6. The ability and accuracy of this algorithm to detect axons of greatly varying eccentricity produces a problem if the general population of cells contains a high percentage of both oval and circle silhouettes. The problems dealing with large void space in an image is apparent in figure 8. Processing of the entire nerve bundle requires some sort of filtering to effectively remove all the space around the nerve bundle. This step would have normally filtered out the left part of the image in scenario 6, but since only a portion of the axon bundle was processed that initial bundle-filtering step did not occur. Overall the total cell count from the algorithm came to 94% of the total.

*ADDITIONAL TRIALS*

As a form of comparison between different methods of axon counting this algorithm was run on two other larger and more complicated images. The first image, seen in figure 8, is roughly 40x larger than the previous scenarios. It is very positive to see that both the 10 or so very large axons are

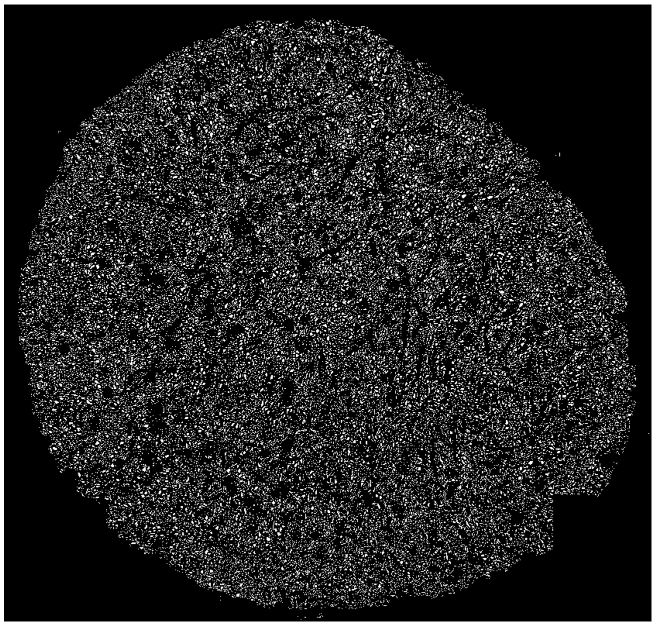


**Figure 10 – Low Resolution image representing all of the "good" counted cells in the entire nerve bundle**

detected in addition to many of the smaller axon segments. However it is also troublesome that many cells in the image are easily identifiable and are not marked as cells. Looking through the multiple states of image processing yielded the result that many of the cells have a small portion the "bleeds into the surrounding void areas. If this happens then when the large void areas are discarded some of the cells go along with it. Attempts were made to prevent this by eroding images further. This often would fix the problem of cell bridging, but at the same time would discard many of the smaller cells that make up a much larger portion of the image. While the algorithm does do a good job at identifying the cells with a stronger cell wall gradient many of the smaller cells with fuzzy or washed out boundaries are omitted. No attempt was made to validate these results to date but the evaluated total number of nerve cells produced by the algorithm is 451 cells. The algorithm was also run on a high-resolution image of the entire nerve bundle, figure 1. Since the image was so large it was of little benefit to show cell-tracing accuracy in this paper. Instead figure 10 shows a black and white image of the axon nerve bundle reduced b 82x where each white blob represents a counted axon. The total number of counted axons by the algorithm was 45,167 cells and if we assume that only 94% were counted as demonstrated in the first 6 scenarios then the total axon count in the bundle would be 48,050 cells.

## IV. DISCUSSION

The presented counting algorithm utilizing adaptive thresholding works fairly well for well-behaved cell populations and where voids in the inspected image do not resemble cells themselves. For a more robust inspection and counting process higher contrast images with greater focus help tremendously. The greatest challenge was balancing different algorithm parameters to provide accurate detection for small, large, sharp, fuzzy, low contrast, and high contrast cells.

The largest challenge was determining how to select the proper thresholding method and how to separate cells that joined together during the thresholding and segmentation stage. If these two aspects could be improved the accuracy of the automated cell counting algorithm would drastically improve. This method proved a starting point for automated cell counting in images with extremely high cell density. Inherent limitations in the algorithm and poor image quality prevent more accurate results at this time.

## V. FUTURE IMPROVEMENTS

### *ADAPTIVE THRESHOLD*

While the adaptive threshold model worked well, there were still areas within the entire axon that were not properly thresholded. Further study on this matter may be necessary. The addition of some type of flooding algorithm [6] in combination with the prior used methods may be beneficial.

Methods for determining cell boundaries through the use of snakes and other dark line tracing methods have also been previously used [4]. Adaptive thresholding could be used to improve the overall contrast within local segments of the image. At this point a snake algorithm could follow the darkest portion of a line around each cell. This would allow the counting algorithm to utilize the gradient pattern specific to a certain cell family or characteristic. Now something like a void of the same size and shape would not be registered as a cell. Instead the algorithm would determine that the cell border gradient does not match that of a predetermined cell type and ignore it.

### *EVALUATE SIZE AND SHAPE*

As mentioned somewhat previously, Cells are not just governed by a specific size and shape. Other characteristics such as wall thickness and intensity gradient also play a factor. Consultation with a field expert would be required to properly gage these characteristics and include them into my model. Once these types of judgments are assigned a numerical value, they can be easily implemented into a software package to automatically count cells.

One of the difficult aspects is that it is easy for a trained operator to apply years of judgment when identifying cells while a computer program has only a limited set of evaluative functions and data to judge from. Giving a group of untrained individuals sample pictures with every good cell indicated only provides half of the story.

## ACKNOWLEDGEMENTS

The provided images were acquired from Professor Scott Howell and Dr. Frank Merat from Case Western Reserve University in Cleveland, Ohio.

## REFERENCES

[1] Haralick, Robert M., and Linda G. Shapiro, “Computer and Robot Vision”, Volume I, Addison-Wesley, 1992, pp. 40-48

[2] Keyence Corporation “CV-2100 High-speed Digital Image Sensor: Users Manual”, Keyence Corporation, 2003, Stain Detection, Pages 4.73 – 4.77

[3] Refai, H.; Li, L.; Teague, T.K.; Naukam, R. (2003)” Automatic count of hepatocytes in microscopic images”, Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on Volume 2, 14-17 Sept. 2003 Page(s):II - 1101-4 vol.3

[4] Theerapattanakul, J.; Plodpai, J.; Pintavirooj, C. (2004) “An efficient method for segmentation step of automated white blood cell classifications”, TENCON 2004. 2004 IEEE Region 10 Conference Volume A, 21-24 Nov. 2004 Page(s):191 - 194 Vol. 1

[5] Qiang Feng; Shenglin Yu; Huaiyin Wang (2006) “An New Automatic Nucleated Cell Counting Method With Improved Cellular Neural Networks (ICNN)”, Cellular Neural Networks and Their Applications, 2006. CNNA '06. 10th International Workshop on 28-30 Aug. 2006 Page(s):1 – 4

[6] Yongming Chen; Biddell, K.; Aiying Sun; Relue, P.A.; Johnson, J.D (1999) “Automatic cell counting method for optical images, An”, Engineering in Medicine and Biology, 1999. 21st Annual Conf. and the 1999 Annual Fall Meeting of the Biomedical Engineering Soc.] BMES/EMBS Conference, 1999. Proceedings of the First Joint Volume 2, 13-16 Oct. 1999 Page(s):819 vol.2

[7] Xiong, Guanglei (xgl99@mails.tsinghua.edu.cn) at Tsinghua University, Beijing, China. [Adaptive Thresholding Algorithm] For more information, please see http://homepages.inf.ed.ac.uk/rbf/HIPR2/adpthrsh.htm