Automatic Cell Detection in Fluorescent Embryonic Stem Cell Images

Zahid Hossain, Diana Wan, Cezanne Camacho
Stanford University
{zhossain,jdwan,cezannec}@stanford.edu

Abstract

We studied two seperate algorithms to automatically detect and count embryonic stell cells from fluoresecent miscrocopy images. The first algorithm is based on a graph mining approach that makes strong assumptions about the topology of the cells. The second algorithm is a machine learning approach that exploits supervised learning technique and uses only 10 positive and 10 negative cell examples for training. For the given dataset the graph mining approach performs (85% accurate) slightly better then the machine learning approach (72% accurate). Nevertheless, we propose a platform that combines visual analytics with machine learning approach to tackle cell counting problem in general for all cell types.

1. Introduction

Even to this day and age cell detection and counting in the context of biology research is largely a manual and a labor intensive task. A reliable count is of paramount importance as it is often a fundamental step in various growth studies. The problem becomes particularly challenging as cell undergo morphological changes over time and an observation of it also suffers from limitations arising from different imaging modalities. In this project we studied the efficacy of two separate algorithms to automatically detect and count a very specific, but clinically important cell type, embryonic stem cells, from fluorescent microscopy images. We also extended each of the algorithms to mitigate spurious detections. Finally, we suggest a platform that combines visual analytics with computer vision to approach this problem generally for other cell types.

2. Related Work

A few methods of automatic counting have already been proposed and employed by researchers, and in the following sections we will examine the accuracy and efficacy of these techniques. These techniques include: size-based nuclei separation and recognition [7], support vector machine (SVM) learning models, which attempt to count cells based on learned nuclei recognition [6, 5], thresholding, which relies on high contrast between bright cells and a dark background to get an accurate cell count [1, 4], cell separation via thresholding or watershed algorithm [5, 4], and graph mining [3].

2.1. Thresholding

In fluorescence microscopy images, the cells we want to count are illuminated when compared to the background of the image. So, in theory it would be easy to convert an image into a black and white binary pixel representation from which a cell count could be extracted. One such method of cell counting relies on an algorithm that is defined by an initially chosen threshold level which determines the resulting binary image and its 2D matrix representation. Then, by calculating the average cell nucleus area with the regionprops command in Matlab, this algorithm extracts a cell count by dividing the area matrix with this average nucleus area and summing the resultant integers [1]. This method is effective for well-separated, high contrast images, and as long as the average nuclei area remains the same, this method is robust in the face of variation in the shape of nuclei. However, this technique falls short when cell nuclei are of varying areas, or when there are clusters of cells that this method cannot differentiate. Figure 1 shows the binarization of an image based on a fluorescent intensity threshold; in this figure you can see that smaller cells are not detected at all, and brighter cells become more difficult to spatially differentiate. Also, given a set of z-plane images, such as our set of confocal microscopy images, this method will likely overcount nuclei by recognizing and counting their perceived area in multiple depth planes.

2.2. Cell Separation

Cells that are clustered together are reliably recognized by a visual count, but often computational methods have trouble separating these cells and recognizing each cell individually. Two main separation methods are typically used: a threshold that tries to recognize lower image intensities in the area between cells, and the watershed algorithm which

Figure 1. **Left:** Input image. **Right:** Binarization after thresholding.

effectively creates edges where these low intensities are located [1, 4]. These methods are similar and the watershed algorithm has been shown to work well with a 2D image as long as image resolution is good enough to show a difference in fluorescent intensity when cells overlap or are right next to one another versus the fluorescent intensity when cells are well separated.

2.3. Machine Learning

One of the approaches in the current literature adopts SVM to train and detect cell nuclei [5]. SVM is able to find the hyperplane that can separate two classes with maximum distance. The training data is separated into training set and testing set. Among the training images, each image is manually divided into sub-images. Sub-images containing a cell nucleus are positive training samples while the rest are negative training samples. As for the testing images, a variable sized sliding window is adopted to search cell nuclei from top left to bottom right and to cater for the variable sizes of cells.

2.4. Graph Mining

This is an unsupervised learning approach proposed by Faustino, Geisa M., et al. that works well for counting embryonic stem (ES) cells in uorescence microscopy images. The challenge here is that the cells could appear fused due to very close spatial proximity or a single cell could potentially be counted twice for having multiple bright spots. In either case, a graph based approach, where an image is represented by a graph of connected components, outperforms the existing watershed algorithm. In this method, cells are modeled as domes and the image is segmented by thresholding at an increasing order of scalar value. At every threshold level, components are detected and tracked to construct a graph that captures how components may splits at a higher threshold. Intuitively, this is similar to walking in a mountain range: all the close-by mountains appear fused along the valley as one walks at the ground level, while the same mountains start separating as one climbs upward. This graph is then mined for simple paths, i.e. chain of nodes that dont split, to detect individual cells. An additional step of clustering is performed in the graph space to improve accuracy.

3. Method

We decided that the most accurate and precise algorithms were robust in the face of cell size and brightness variation, which are strengths of human visual intuition. Also, algorithms that were generalizable and scalable were more ideal. Since we have seen that thresholding is the most simplistic and least effective approach, we decided to implement both machine learning and graph-mining cell-counting methods.

3.1. Graph Mining Approach

We adopted this technique from Faustino et al. [3]. In this approach each cell in an image is modeled as a smooth 2D function that has a single local maxima in its neighborhood. Figure 2 demonstrates this cell model. Note that in this model the shape of the cell does not necessarily have to be circular or blob like.

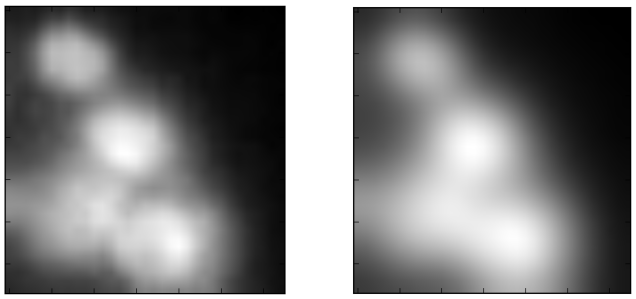


Figure 2. Left: grayscale image representation of the cells, Right: the corresponding heightfield representation of the cells. Each local maxima in the heightfield corresponds to one cell on the left image.

The overall graph mining process can be divided up into the following 5 steps.

1. Preprocessing
2. Histogram Partitioning and Connected Component Detection
3. Graph Construction
4. Graph Mining
5. Graph Clustering

The exact values of all the parameter that we may introduce in this section will be reported in Section 4.

Preprocessing: First we convert the image into grayscale and only use this grayscale representation for the rest of the pipeline. Unlike the assumed model as shown in Figure 2, a real image is often contaminated with noise that give rise to spurious peaks. Hence we first smoothen the image using a Gaussian filter with a standard deviation of σ to get

rid of some of these noise as well remove any discontinuities. Next we remove the background by applying a simple threshold $\tau = \mu_I + x \cdot \sigma_I$, where μ_I is the average intensity value, σ_I is the standard deviation of the intensities and x is a parameter. We set all the pixels to 0 which has intensities less than τ.

Histogram Partitioning and Connected Component Detection: In this step we partition the histogram of the image (intensities values ranging from $[0, 255]$) in n equal sized segments. For example, in Figure 3, a histogram is partitioned into $n = 4$ segments and one of the segments (marked in red) is used to create a binary image containing only the pixels that belong to the selected histogram segment. This binary image is then used to find all the connected components using a 4-connected filter. Each component is given a unique label and this process is repeated for every segment on the histogram. This yields a single image, we term as *label image* M, that has the same dimension as original, where each pixel is labeled (see Figure 4). We traversed the histogram from the largest range to the smallest, i.e. from right to left, and assigned labels such that pixels with higher intensities get smaller label values. This makes the graph mining step, discussed later, computationally easier.

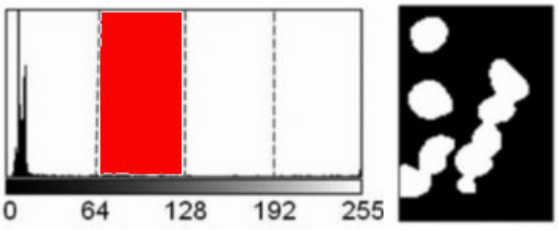


Figure 3. Left: Histogram which is partitioned into $n = 4$ segments. Right: Binarized pixels within the red segment on the histogram.

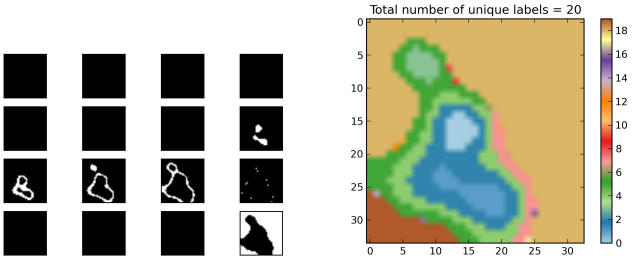


Figure 4. The input image is the same as Figure 2 **Left:** binarized images from histogram segmentation. The histogram was segmented into $n = 16$ parts. **Right:** All detected labels (color coded) in one single image, we term as *label image*, M.

Graph Construction: An undirected graph of connected components is then constructed from the label image M. We used a matrix representation for this graph and looked at 4 neighbors of each pixel in M in one single scan to populate the matrix.

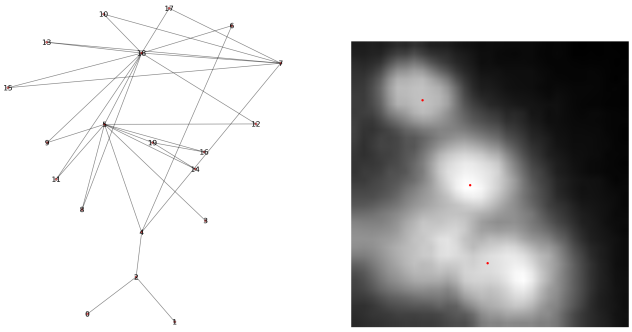


Figure 5. **Left:** An undirected graph of connected components generated from the label image M as shown in Figure 4. **Right:** Detected cells, each of which corresponds to a simple path, are marked as red dots.

Graph Mining Every cell in the input image corresponds to a simple path in the undirected graph. A simple path in the graph is defined as a path where every node appears only once and have at most 2 edges. For example, in Figure 5, the paths $\{0\}, \{1\}$and$\{3\}$ and all simple paths. However, note that a simple path may contain more than 1 nodes. The peak of each cell is marked as the cell location and since we constructed M such that the peaks have smaller labels we traverse the graph rooted at smaller node values and mark off any node that we have already visited. This not only prunes the search space significantly but also guarantees that we always traverse from some peak.

Graph Clustering A naïve graph mining will detect many spurious cells due to multiple local maximas within a single cell. This may occur due to noise or some special structure of the cell itself, e.g. multiple cell nucleus or other organelles. In most cases we already know the approximate scale of the cells in a given image. This information could be provided by a biologist once before running the system in batch or could be extracted from the imaging metadata. We used that information to compute pair-wise distances between every possible pair of detected cells. Next, for every detected cell, we prune all the other detected cells that are within that scale threshold. This however makes an implicit assumption that cells do not overlap each other in the image. A simple pair-wise distance computation was reasonably fast in our case but for very large dataset with potentially many cells one could employ space partitioning algorithms like quad-tree to speed up this process.

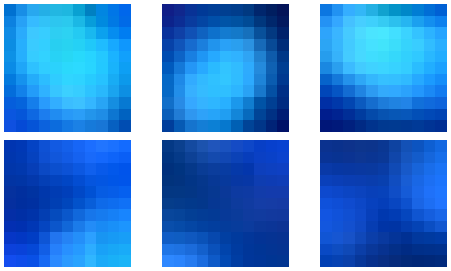


Figure 6. **Top Row:** 3 positive training examples. **Bottom Row:** 3 negative training examples.

Figure 5 shows only 3 cells were detected instead of 4. We described our algorithm in the light of a specific input image as shown in Figure 2 on purpose to demonstrate exactly where our technique succeeds and where it fails. The underlying reason for missing one extra cell in the detection can be understood by referring to Figure 3. In this figure, the two cells peaks in the bottom (light blue) were mistakenly merged during the histogram partitioning step because the number of partitions $n = 16$ was not high enough. However, if n is too high it will lead to too many disjoint components in the labeling process due to the discrete nature of the data and this may again yield poor performance.

3.2. Supervised Learning Approach

3.2.1 Support Vector Machine

Support Vector Machine is a commonly used approach of pattern classification in machine learning. SVM determines the separating hyperplane by maximizing the distance of it to the closest points in the training set [2]. Popular SVM kernels include linear, polynomial, radial basis function and sigmoid. It generally involves two steps when applying SVM to a classification problem, which are training and predicting. In training step, positive samples and negative samples are presented. In the predicting step, the model generated from the training step is used to predict each input sample to be either positive or negative judging from its position to the separating hyperplane.

3.2.2 Training

Our goal is to detect cells so we should feed SVM with positive training patches that contain cells and negative training patches that does not contain a complete cell (no cell, or a partial cell). We extracted patches of 11 by 11 from the fluorescent cell image dataset. Examples are shown in Figure 6:

The size 11-by-11 is not arbitrary. It is observable that in this dataset, most cells have similar sizes and a single cell can fit into a 11-by-11 square window just about right. In selecting positive patches, we tended to include cells with more variable shapes as possible. As mentioned in the previous sections, cells vary in shapes. Therefore if we only selected cells with one shape, it is possible that our trained model would not be able to detect cells with other kinds of shapes. As for negative patches, we included a variety of background patches and also patches with partial cells.

3.2.3 Prediction

For each image in the testing set, we moved a sliding window of 11 by 11 from top left to bottom right. By using this exhaustive search method, we got a testing patch at each pixel location and we used our training model to determine whether this patch contained a cell or not. If a patch contained a cell, we marked the central position of this patch with a yellow marker. One thing worth mentioning is that, during our experiment, we initially could only detect cells that had a good contrast to their backgrounds and this method failed on cells that were less conspicuous. After analyzing the problem, we decided to normalize all the training patches before feeding them into SVM in our training step, and normalize all the testing patches before predicting. This improved method worked perfectly and was able to detect all of the cells present in an image. However, there were duplicate detections, and this will be addressed in the next sub-section of this paper.

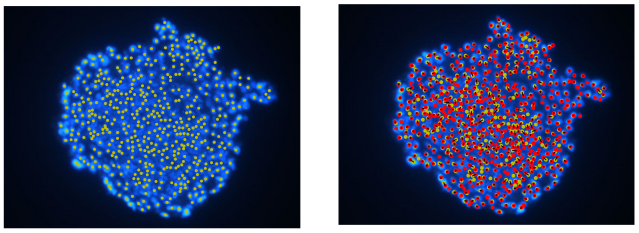
3.2.4 Post-Processing

There are several methods to eliminate duplicate detections, such as non-max suppression, merging points that are too close together, etc. We opted to use the merging method we moved a larger sliding window across the whole image and at each location we merged points that had Euclidean distances smaller than a threshold. This method worked very well in eliminating duplicate detections. The value of the threshold should be empirically selected. If the value was too large, we would have some false negatives; if the value was too small, we would have some false positives. The optimal threshold value should be a tradeoff between false positives and false negatives.

4. Results

For detection problems, precision and recall are often used to measure the performance of an algorithm. Precision measures how relevant the retrieved items are and recall measures the ability to retrieve relevant items. F-measure is just the harmonic mean between precision and recall. To calculate precision and recall, we need to know the ground truth data first. We manually labeled cells in 10 cell images from the dataset. Due to limitation of time and manual labor, we only included 10 images in our testing set. For each algorithm we compared our results with this ground truth data.

4.1. Graph Mining Results



(a) (b)

Figure 7. (a) Detected cells (yellow) by the graph mining technique. (b) Matching between the detected cells (yellow) and the hand labeled cells (red).

We ran the Graph Mining algorithm with a parameter setting of ($\sigma = 2$ pixels, $x = 0.3$ and $n = 16$). Figure 7 shows the results of the graph mining technique which we compared with the hand labeled ground truth. For comparison we perform a bipartite matching between the detected cells and the ground truth subject to a threshold that is approximately the scale of a single cell. We ran this algorithm on our entire dataset and observed an performance (see Figure 8, average precision = 85%, average recall = 83%) that is within an acceptable range for the biologists.

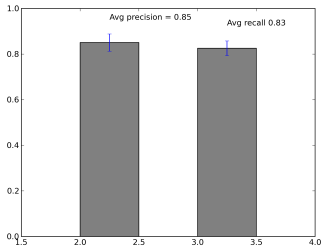


Figure 8. Precision/Recall for the Graph Mining technique.

Figure 9 shows where the Graph Mining algorithm failed. The red dots corresponds to the false negatives while the yellow dots to false positives. We noticed a pattern among the false positives, i.e. in most cases they live in the valley of surrounding cells, which tells us there is still some room for improving the precision measure.

4.2. Supervised-Learning Results

The precisions and recalls for the Supervised-Learning approach is given in Table 4.2. Learning curve is usually used in measuring the performance of a machine learning algorithm. Test error measures the relationship between error rate and the size of the training set, and it usually decreases as the training set size increases. Training error is the error rate of testing on the rest of the original training set while the training set size here increases. Usually the training error increases as the training set size increases. However, in our approach, we had a small training set of 10 positive patches and 10 negative patches. We decided that due to the small size of the training set, training error might not be a good measurement of performance and should not be adopted. The learning curve of test error is shown as below. It can be seen that the error rate decreases with the increase of training set size and it is believed that the error rate will remain relatively constant when the training set size is large enough.

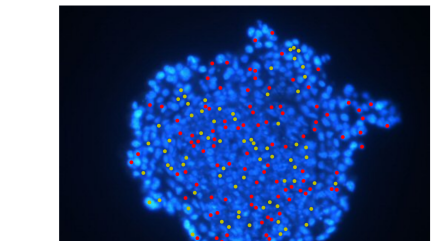
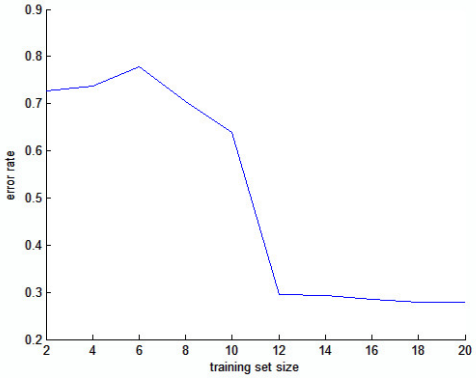


Figure 9. Failure cases of Graph Mining: Red: False Negatives, Yellow: False Positives.

File #	Precision	Recall	F-measure
1	0.7667	0.8164	0.7908
2	0.7774	0.7172	0.7461
3	0.8303	0.7455	0.7856
4	0.7476	0.7979	0.7719
5	0.6778	0.7125	0.6947
6	0.5522	0.6915	0.6140
7	0.7667	0.8058	0.7858
8	0.7558	0.7708	0.7632
9	0.7410	0.7540	0.7474
10	0.5902	0.7099	0.6445
Average	0.7206	0.7521	0.7344



5. Conclusion

It is hard to design a generic cell detection technique that will encompass all cell types and imaging modalities.

Both the graph mining and the supervised machine learning approaches worked remarkably well for the embryonic stem cells. After interviewing a few biologists we learned that cell counting is mostly done manually because none of the existing system seem to be adequate. We tried some of the automation tools that are available to better understand where they fall short. What seemed to be lacking is perhaps not the quality of each algorithm but a platform that integrates visual analytics with automation through a good interface. For example, there is really no easy way for a biologist to interact with the automation results and correct wherever it made mistakes. We realize that any automation system is unlikely to be perfect and hence there is a serious need for a platform that a) learns from the data and mistakes over time b) allows biologists to correct mistakes through very simple interfaces. Though our machine learning based approach fell slightly short in terms of performance compared to the graph mining approach, we envision that an ensemble of machine learning approaches combined with a visual analytics platform will tackle this problem better in the long run. Typically a biologist work on a particular cell type and imaging modality for a number of years which also gives the aforementioned platform a much greater opportunity to learn over time. In addition, biologists can share these learned parameters with each other to cover diverse cell types and imaging modalities.

Acknowledgments

We would like to thank Dr. Silvio Savarese along with all the TAs for teaching this incredibly valuable course of CS231a. We would also like to thank Dr. Ian Driver and David Glass from the Stanford Bioengineering department for providing us with the dataset and various supports all throughout.

References

[1] B. K. Al-Khazraji, P. J. Medeiros, N. M. Novielli, and D. N. Jackson. An Automated Cell-Counting Algorithm for Fluorescently-Stained Cells in Migration Assays. *Biological procedures online*, 13(1):1–6, 2011.

[2] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine learning*, 20(3):273–297, 1995.

[3] G. M. Faustino, M. Gattass, C. J. de Lucena, P. B. Campos, and S. K. Rehen. A Graph-Mining Algorithm for Automatic Detection and Counting of Embryonic Stem Cells in Fluorescence Microscopy Images. *Integrated Computer-Aided Engineering*, 18(1):91–106, 2011.

[4] M. G. Forero and A. Hidalgo. Image Processing Methods for Automatic Cell Counting In Vivo or In Situ Using 3D Confocal Microscopy. 2011.

[5] J. Han, T. Breckon, D. Randell, and G. Landini. Radicular Cysts and Odontogenic Keratocysts Epithelia Classification Using Cascaded Haar Classifiers. In *Proceedings of the Twelfth Annual Conference of Medical Image Understanding and Analysis*, pages 54–58, 2008.

[6] J. W. Han, T. P. Breckon, D. A. Randell, and G. Landini. The Application of Support Vector Machine Classification to Detect Cell Nuclei for Automated Microscopy. *Machine Vision and Applications*, 23(1):15–24, 2012.

[7] T. Shimada, K. Kato, A. Kamikouchi, and K. Ito. Analysis of The Distribution of the Brain Cells of the Fruit Fly by an Automatic Cell Counting Algorithm. *Physica A: Statistical Mechanics and its Applications*, 350(1):144–149, 2005.