SEGMENTATION OF NUCLEI IN CONFOCAL IMAGE STACKS USING PERFORMANCE BASED THRESHOLDING

William Beaver¹, David Kosman², Gary Tedeschi², Ethan Bier², William McGinnis², and Yoav Freund¹

¹Department of Computer Science and Engineering, UCSD, La Jolla, CA 92093
²Cell and Developmental Biology Department, UCSD, La Jolla, CA 92093

ABSTRACT

Combinatorial Transcriptional Fluorescent In Situ Hybridization (CT–FISH) is a confocal fluorescence imaging technique enabling the detection of multiple active transcription units in individual interphase diploid nuclei. As improved combinatorial labeling methods allow simultaneous measurement of gene activities to expand from five genes in a single embryo to tissue section to upward of twenty genes, transforming image stacks into usable data becomes an increasingly labor intensive task. In this paper we describe our progress towards a method for the computational analysis of confocal images from Drosophila melanogaster that involves the segmentation of the cell nuclei and of nascent transcription sites of specific genes. Using image processing and machine learning algorithms, we allow experimentalists to iteratively tune and improve the analysis system to reflect biological reality.

Index Terms— Image processing, Image segmentation, Machine vision, Pattern recognition

1. INTRODUCTION

Combinatorial Transcriptional Fluorescent In Situ Hybridization (CT–FISH) is a confocal fluorescence imaging technique that enables the detection of gene activity at the resolution of single transcription units on chromosomes in individual interphase diploid nuclei. The McGinnis and Bier labs are developing and employing CT–FISH methods to study the spatial and temporal activity of a variety of developmental regulatory genes in embryos of Drosophila melanogaster. Currently available CT–FISH methods are able to measure the activity level of five genes in a single specimen. Elaborations of combinatorial labeling methods under development promise the ability to measure twenty gene activities simultaneously.

To construct models of embryonic development from these measurements, there are a number of steps involved in the analysis. First, segment the cell nuclei using DAPI and other nuclear markers. Concurrently, detect transcription sites on chromosomes and classify nascent transcription sites for specific genes by their fluorescent combinatorial codes. Finally, combine and register image stacks from different embryos to an embryonic development model and assess the consistency of the patterns across different embryos [1].

In this paper we report our progress in developing accurate segmentation algorithms—a critical first step to the accuracy of later processing stages and ultimately to the final models built. There has been significant work on the problem of segmenting DAPI marked nuclei in confocal microscopy images. Most notable is the work of Lin et al. [2, 3, 4] and the work in Reinitz Lab [5, 6]. Our main contribution is the development of an adaptive segmentation method tuned for particular types of images using machine learning methods. Tuning is a critical step whenever a computer vision method is to be applied to new experimental data. Any change in the model organism, hybridization method, or the microscope results in changes to the generated images, and these changes, in turn, require changes to the computer vision algorithms. The advantage of machine learning is that it is rather than have the experimentalist tune the computer vision parameters directly, which requires a good understanding of computer vision, the experimentalist tunes the system by providing direct feedback of the form “this segment is a part of a nucleus” or “this segment is a combination of two nuclei”.

![Diagram](image.png)

Fig. 1. A high-level description the analysis system. The main low-level operation performed on the images is simple thresholding. An inner feedback loop controls selection of the threshold based on a quality scoring function. An outer loop controls the quality scoring function using human feedback and machine learning.

2. PERFORMANCE BASED THRESHOLDING

Our segmentation algorithm is based on the observation that, at least for our data, we can find a threshold on the intensity—for each nucleus in each frame of the stack—that correctly segments each nucleus from its surrounding objects. The problem is that the appropriate threshold changes from nucleus to nucleus and from frame to frame; it is not a global threshold. Our goal is thus to identify good local threshold values for each nucleus. To do this, we need a way to evaluate the quality of the segmentation resulting from different thresholds. To this end we design a performance scoring function \( F \) which assigns each segment \( x \) a quality score \( F(x) \) based on its shape. We then choose each segment based on the threshold that generates a segment with a high score. We call this method “Performance Based Thresholding”, or PBT for short. The PBT algorithm is outlined in Figure 2.

Our PBT algorithm accepts the lowest threshold that results in a sufficient quality segment. This is done to save computation time, to maximize the size of the segments, and to ensure that they maximally cover the nucleus. This last feature is important for later stages of the analysis where nascent transcription is spatially located.

The performance function \( F(x) \) consists of two parts. The first is a mapping from the shape of the segment to a set of numeric fea-
1: for all framea, in stack do
2: \( \text{segments}_a = {} \)
3: for all thresholds \( t \) from low to high do
4: \( B = \text{frame}_a > t \)
5: \( CE = 2D \) connected elements in \( B \).
6: for all \( ce_j \) in \( CE \) do
7: \( p = F(ce_j) \)
8: \text{if} \( p \) ids “good” then
9: \( \text{mask} \) (remove) \( ce_j \) from \( \text{frame}_a \).
10: \( \text{add} \) \( ce_j \) to \( \text{segments}_a \).
11: \text{end if}
12: \text{end for}
13: \text{end for}
14: \text{Connect segments between segments}_a \) and \( \text{segments}_{a-1} \)
15: \text{end for}

Fig. 2. Pseudocode for Performance Based Thresholding

Fig. 3. Segment geometry. \( p \) represents object centroid. (a) Convex Hull for object. \( R \) is radius measure from \( p \) to a pixel on the object’s perimeter \( P \). (b) The ellipsoid with the same second order moments as the object. This ellipsoid has major and minor axes \( (\lambda_1 \lambda_2) \) and \( f1 \) and \( f2 \) represent the foci of the ellipse.

3. LEARNING PERFORMANCE SCORING FUNCTIONS

Given a feature vector that captures the salient shape information about a segment, we wish to compute a score that will quantify the likelihood that the segment correctly captures the shape of the nucleus. This is the most important—and the least well-defined—step in the PBT algorithm. We construct \( F \), the function mapping feature vectors to performance scores, using a machine learning algorithm. A machine learning algorithm receives as input a “training set”—here a set of randomly selected segments, each with an associated label, stating whether the segment is “good”, “too small” or “too large”. The learning algorithm selects a function to approximate the relationship between segments and labels in the training set and is likely to perform well on similar, but yet unseen segments. There are many learning algorithms; decision trees, nearest neighbors, neural networks, and support vector machines are a few of the more popular ones. We use a learning algorithm called Adaboost, invented by Freund and Schapire [7]. Adaboost has several advantages in this context:

- The performance of the algorithm is not sensitive to the number of irrelevant features. This frees the designer to add any feature they think might be useful, rather than performing a priori feature selection to reduce the dimensionality of the feature vector.
- The features do not have to be strongly correlated with the desired label. For a feature to be useful, it is sufficient if it has a weak, but statistically significant, correlation with the label.
- The performance of boosting does not change if the features are scaled, shifted, or transformed by any monotone function.

We use a learning algorithm based on Adaboost that creates structures similar to decision trees: “alternating decision trees”. The
algorithm for learning alternating decision trees (or ADTs) is explained in detail in [8]. A small fraction of an ADT we constructed for the quality scoring function of DNA stained with DAPI is given in Figure 4. In our case, the outcome of the function is a vector of three scores, one for each of the three possible classes we simply refer to as “good”, “too small” and “too big”. The evaluation of the ADT involves testing the condition in each square node and then adding the contributions in the corresponding circular node to the corresponding score. After all of the square nodes have been processed in this way, the final three scores quantify the degree of confidence in each of the three labels. In order to accept a segment, we require that the “good” score be large, and significantly larger than the scores for the two “bad” labels.

The result of the learning process is a performance function \( F(x) \) with known empirical error rate. If we wish to improve performance beyond this rate, we gather more example data representing those objects where the function is incorrect. Take the segmentation from PBT (Section 2), use it as input to our performance function, and, via the same interactive mechanism used to correct labelings from the pixel count heuristic (Section 4), correct the mislabelings made by the performance function. This new, labeled data is added to the training set, and we train a new predictor exactly as before.

4. TRAINING METHODOLOGY

Typically, the process of labeling training data is laborious as a very large number of examples must be labeled individually, by a human. To overcome this problem we use the “active learning” methods of Abramson and Freund [9]. These methods greatly decrease the number of examples labeled manually.

Given a CT–FISH image stack, we generate a first iteration performance function by thresholding each frame of the stack using Otsu’s method [10] and labeling segmentations according to a simple pixel count heuristic. Objects whose pixel count falls within a variable range are labeled “good”. This range is determined by visual inspection of the effects that modifying the range has on the labeling. This method allows the experimentalist to achieve a large number of labelings in a short amount of time. For cases where the pixel count heuristic does not accurately classify an object, the object is simply reclassified by selecting it in an interactive interface. Once a number of frames are labeled (resulting in, typically, hundreds of positive and negative labeled examples per frame) we extract the morphological features described in Section 2. These features, along with the appropriate class label, are now used as inputs to the Adaboost learning algorithm.

5. EXPERIMENT: COMPARISON TO OTHER THRESHOLDING ALGORITHMS ON DAPI STAINING

Using the features described in Section 2 and the method in Section 3, we learned a scoring function \( F(x) \) for DAPI stained nuclei. The first (most significant) nodes of the ADT representing \( F(x) \) are shown in Figure 4. We interpret this ADT to indicate that the three most discriminating features of nuclei are based on two features: \( \mu_3 \) (mean of the eigenvalues) and Convexity. Values of \( \mu_3 \) where 3593 < \( \mu_3 \) < 392956 are likely “good” objects while objects with \( \mu_3 \) below the lower threshold are likely “too small” and \( \mu_3 \) above the upper threshold are “too large”. The third node of the ADT can be be interpreted as objects with convexity > 0.84 are also likely to be labeled as “good”.

To measure the performance of PBT, we compare it against hand labeling of segmentations of other thresholding techniques; namely mean thresholding (where per frame threshold is simply the mean intensity value) and Otsu’s method [10]. For each method, we report the number of correct and incorrect labelings as compared to the hand-labeled ground truth dataset.

![Fig. 4. An alternating decision tree showing the first three decision rules of our \( F(x) \) for DAPI. The tree consists of alternating levels of ovals (prediction nodes) and rectangles (splitter nodes). The numbers within the ovals define contributions to the prediction score for each label. The labels used are G=good, S=small, and B=big.](image)

Table 1. Performance of various thresholding algorithms in detecting objects vs. ground truth segmentation from CT–FISH confocal image stack of DAPI. The actual number (ground truth) of 2D nuclei segments in this 18 frame stack containing approximately 600 nuclei, each spanning about 9 frames, is 3661. \( \text{diff} \) = difference from ground truth, \( \text{US} \) = under-segmentation (segment on nuclei but including too little of it); and \( \text{Conn} \) = Connected (nucleus not detected as a separate segment but combined with another nucleus.)

<table>
<thead>
<tr>
<th>method</th>
<th>count</th>
<th>diff</th>
<th>% found</th>
<th>US</th>
<th>Conn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>2535</td>
<td>-1126</td>
<td>69%</td>
<td>7572</td>
<td>347</td>
</tr>
<tr>
<td>Otsu</td>
<td>2741</td>
<td>-920</td>
<td>75%</td>
<td>4134</td>
<td>186</td>
</tr>
<tr>
<td>PBT</td>
<td>3505</td>
<td>-156</td>
<td>96%</td>
<td>20</td>
<td>173</td>
</tr>
</tbody>
</table>

As Table 1 indicates, performance based thresholding outperforms the other methods in terms of accuracy. Figure 5(b) visually shows the performance of Otsu’s thresholding on a sample image. Here we see both over- and under–segmentations. Using PBT on the same image gives marked improvement (Figure 5(c)). As can be seen, PBT is not immune to incorrect segmentation, but, as the accuracy of \( F(x) \) increases, so does the accuracy of PBT.

6. APPLICATION TO DETECTION OF CHROMOSOMAL TRANSCRIPTION SITES

While the above technique is shown successful in locating nuclei from CT–FISH confocal microscopy, we qualitatively show that the technique is successful at also locating other gene expression from CT–FISH. Using the same techniques outlined for DAPI, we learn another \( F(x) \)—one that identifies transcription of Distal-less (dll)—and we collocate dll with the DAPI segmentations; this effectively gives us a map of nascent transcription. Figure 6 shows a single nucleus with its collocated dll transcription as extracted from clusters of dll expression in the limb field of Drosophila at stage eleven of embryonic development.

![image](image)
Fig. 5. Comparison of segmentation results. (a) Original image. (b) Example of Otsu thresholding with over-segmentations (red, downward sloping diagonal striping), under-segmentations (yellow, upward sloping diagonal striping), and good segmentations (green, vertical striping). (c) Example of PBT thresholding under-segmentations (yellow, upward diagonal striping) and good segmentations (green, vertical striping). Note: no over-segmentation with PBT.

Fig. 6. 3D close-up image of a single limb field nucleus and transcription site

7. ACKNOWLEDGMENTS

Molecular graphics images were produced using the UCSF Chimera package from the Resource for Biocomputing, Visualization, and Informatics at the University of California, San Francisco (supported by NIH P41 RR-01081).

8. REFERENCES


