Automatic Construction of Cell Genealogical Histories

Matt Jachowski
Rajesh Ranganath

Introduction

Many research projects in the biological sciences have benefited from software and hardware that allows the automatic collection of large amounts of data. However, the full benefits of having this wealth of new data remain unrealized as much tedious analysis is still done by hand. Stem cell research is one sub-field suffering from this problem. Stem cell researchers have collected time-lapsed images of microwells populated with dividing stem cells. This data has allowed new analysis of stem cell division patterns. The data promises many new discoveries, but these discoveries are bottlenecked by the lack of automatic methods for generating cell genealogical histories.

Initially, it may seem that an off-the-shelf method to identify cells, such as a binary SVM, would be applicable to this problem, but the lack of persistent features and the difficulty of solving the overlap problem using joint classification leaves this method infesable to solve the cell genealogical history problem. Therefore we propose a more complex approach that hypothesizes cell count sequences, merges the hypotheses using a conditional random field, and finally establishes correspondence throughout time using a Kalman Filter.

Methodology

We achieve automatic construction of these cell genealogical histories from movies of dividing cells. Our technique involves five steps. In the first step, we apply the Hough circle transform to identify regions highly likely to be cells. Next, we apply two algorithms – local maximum finding and weighted k-medoids clustering algorithms – to estimate cell counts. Then, we input these two sequences of cell counts as observed variables for a conditional random field to generate the most likely sequence of cell counts. Given these determined cell counts, we match the positions determined by reapplying the local maximum finding and weighted k-medoids algorithms. Finally, we construct cell tracks by establishing interframe cell correspondence with a Kalman filter.

Identifying Cell Regions

Each movie frame that we consider contains a microwell and some number of cells contained within the well. To identify cell regions, we first apply a bottom-hat filter to the image to isolate the microwell and cell boundaries as in [1].

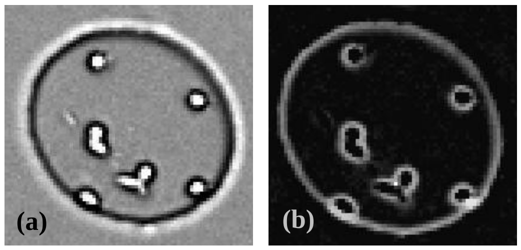


Figure 1. (a) Original Image. (b) Bottom-Hat Filtered Image.

In early stages of the project, we then proceeded by isolating the microwell boundary, easily achieved through segmentation or hough ellipse detection, and removing it from the image frame. However, we found this approach to be problematic as, in many images, cells hug the wall of the microwell, causing these cells to be either partially or fully removed by the microwell-removal process.

By applying a Hough circle tranform to detect cells directly from the filtered image, we found that we could achieve reliable and robust cell region identification without explicitly removing the microwell. Given an approximate range of cell radii, which can be learned from a few expert-labeled cell instances, we apply a series of Hough circle transforms for several radii in this range. We then sum the accumulators resulting from each of these transforms. The resulting accumulator exhibits strong peaks at cell centers. After smoothing the accumulator, we threshhold to leave only those regions most likely to be cell centers.

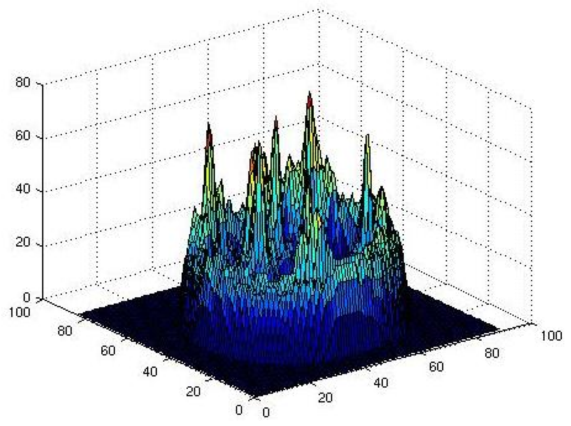


Figure 2. Accumulator resulting from hough circle transform being applied to above image.

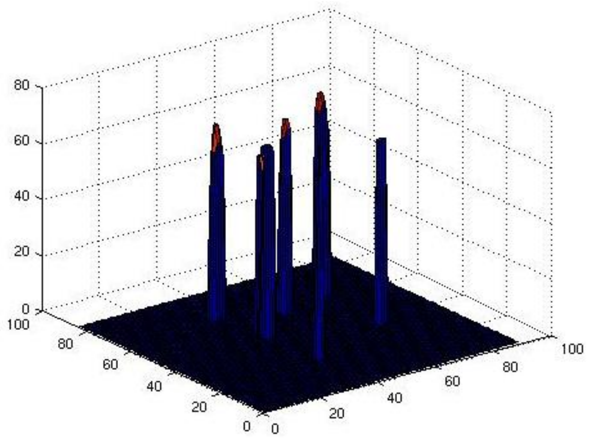


Figure 3. Smoothed and threshholded accumulator. Spikes represent likely cell regions.

Estimating Cell Counts

Given these likely cell regions, we then estimate cell counts with two different clustering techniqes. The first technique finds local maxima in the smoothed and threshholded accumulator. The second technique repeatedly applies the weighted k-medoids clustering algorithm to the smoothed and threshholded accumulator until it finds a reasonable k.

The k-medoids algorithm is related to the k-means algorithm, differing in an added requirement that cluster centers be cluster points. Given k, the algorithm finds k clusters and the point within each cluster with the minimal distance to all points within the cluster. Since each cluster point has a weight – its accumulator count – and higher weights correspond to higher likelihood of points being cell centers, we employ a weighted version of k-medoids to encourage k-medoids cluster centers to converge on cell centers. To optimize the weighted k-means objective function subject to the constraint that the centroids belong to the set of input points, we use the Partitioning Around Medoids algorithm[2]. Finally, since k is unknown, we implement an iterated process, where we first hypothesize one cell, then two cells, and so on, until some value of k results in the algorithm assigning cluster centers that are closer than the approximate diameter of a cell. Since two cell centers must be at least one cell diameter apart, we can reject this clustering and settle on k-1 as the probable cell count.

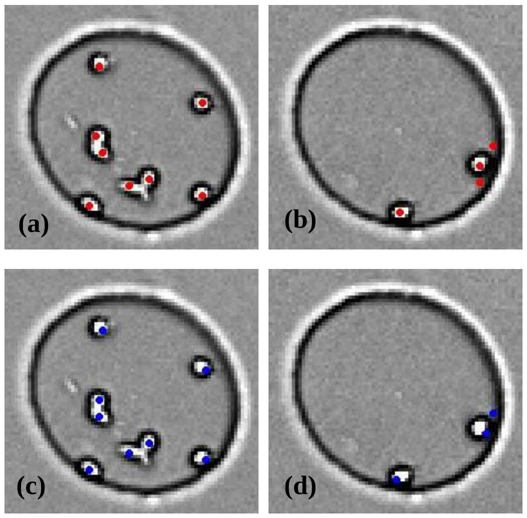


Figure 4. Estimates of cell counts/positions from the local maxima finding algorithm (a,b), and the weighted k-medoids algorithm (c,d). (a) and (c) are examples of when the algorithms work well. (b) and (d) are examples of when the algorithms are lacking.

Conditional Random Field

Given the cell counts estimated by k-mediods and the local maxima finding algorithms, it may seem plausible to simply use the cell count estimates of one of the previous frames to localize cells, but this leads to avoidable errors. Consider figures 4b and 4d where local maxima finding and k-mediods fail to identify correct cell counts due to Hough responses created by the microwell boundary. Another possible avoidable error can occur when one technique outputs the correct cell count and the other fails to, due to assumptions made in each algorithm. The combination of these errors can lead to different inaccurate sequences of cell counts from each algorithm.

To produce a more accurate sequence of cell counts we implemented a first order conditional random field where each state represents a fixed cell count. We use the two hypothesized count sequences returned by k-medoids and the local maxima algorithm as the observed variables of our model. The mathematical model of our conditional random field is given below.

$$Score(\boldsymbol{\pi}, \mathbf{x}) = \exp\left(\sum_{i=1}^{L}\sum_{j=1}^{K} w_j f_j(\pi_i, \pi_{i-1}, i, \mathbf{x})\right) = \exp\left(\sum_{j=1}^{K} w_j F_j(\boldsymbol{\pi}, \mathbf{x})\right)$$

$$P(\boldsymbol{\pi}|\mathbf{x}) = \frac{Score(\boldsymbol{\pi}, \mathbf{x})}{Z(\mathbf{x})} = \frac{\exp\left(\sum_{j=1}^{K} w_j F_j(\boldsymbol{\pi}, \mathbf{x})\right)}{\sum_{\boldsymbol{\pi}'} \exp\left(\sum_{j=1}^{K} w_j F_j(\boldsymbol{\pi}', \mathbf{x})\right)}$$

Where $\mathbf{x}$ is the observed input, $\boldsymbol{\pi}$ is the parse (a sequence of states), and f_j is the jth feature

Our condtional random field had two types of features: type 1 features and type 2 features. Type 1 features are functions of multiple sequences of observed variables, while type 2 features are functions of only one. The score at time *i* is boosted by type 1 features if all the observed cell counts are equal to π_i. Type 2 features are defined by the following formula:

$$f(\pi_i, \pi_{i-1}, i, \mathbf{x}) = p(\pi_i; \pi_{i-1}, \sigma) * p(\pi_i; \mu, \sigma)$$

where μ is the expected value of the observed sequence with respect to a normal distribution centered at *i*, and p is the normal PDF.

Using our CRF model with the two hypothesized cell count sequences, we implemented the Viterbi algorithm to find the most likely state parse given the observed variables. Since our states represent cell counts, the Viterbi algorithm output for the each position represents the most likely cell count.

Finding Cell Centers Given Fixed Cell Counts

Given the cell counts for each frame as determined by the CRF, we reapply the local maxima finding and weighted k-medoid algorithms to pinpoint cell centers. In the case of the weighted k-medoid algorithm, the benefit of predetermined k is clear: we no longer need to iterate through possible values of k. However, it is harder to force the local maxima finder to return exactly k local maxima for any given frame. We resolve this by applying the following method: if the result returns more than k local maxima, pick the k points with the highest accumulator weights, otherwise return all of the local maxima found.

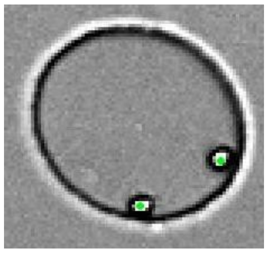


Figure 5. The incorrect cell counts displayed in (4b) and (4d) have been corrected by the CRF.

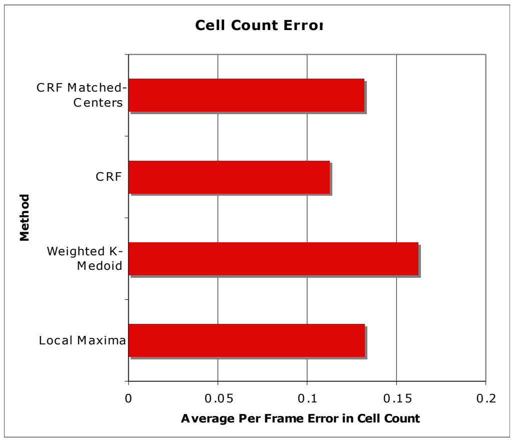
We then apply the Hungarian algorithm [3] to match the corresponding cell centers from each algorithm, and output the single cell center in each pair with the highest hough circle accumulator weight.

Error Analysis: Cell Count and Position Error

We evaluate our cell count and position error on a 235 frame movie independent from the data used during the project development. We define the overall cell count error to be the average of the per-frame cell count errors. The overall cell position error is calcuated as the average, over all frames, of the average cell position error in each frame (in pixels).

$$\text{Cell Count Error} = \frac{1}{N_{frames}} \sum_{(\text{frame } i)} \frac{|count_{true}(i) - count_{observed}(i)|}{count_{true}(i)}$$

$$\text{Cell Position Error} = \frac{1}{N_{frames}} \sum_{(\text{frame } i)} \frac{1}{N_{cells}} \sum_{\substack{(\text{cell } c \text{ in} \\ \text{frame } i)}} \|pos_{true}(c) - pos_{observed}(c)\|_2$$



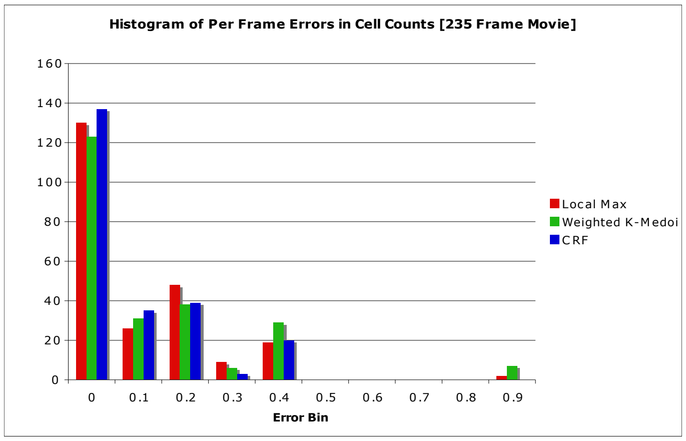


Figure 6. Cell count error is on the order of 10%. As expected, CRF cell count error is the lowest.

Figure 7. A histogram of per-frame cell count errors shows that over half of the frames are labeled with zero error in cell count.

The average cell count error is on the order of 10%, as shown in figure 6. Note that each frame in the 235-frame movie analyzed had 2-8 cells, so any under- or over-labeling, even if by one cell, had a signficant impact on the error. As desired, the cell count error is smallest for the CRF-determined cell counts. Additionally, the cell count error for the final cell finding algorithm – matching the results of the local maxima finding and weighted k-medoids algorithms with pre-determined k – is lower than either of the two algorithms with unknown k. As shown in figure 7, over half of the frames were labeled with zero error in cell count. And, the CRF cell count eliminates the most egregious errors in the local maxima finder and weighted k-medoids counts.

As shown in figure 8, the average cell position error is on the order of 1 pixel. Given that the cells are approximately 6 pixels in diameter, and human labeling error is also on the order of 1 pixel, the cell position labelings generated by our method are excellent.

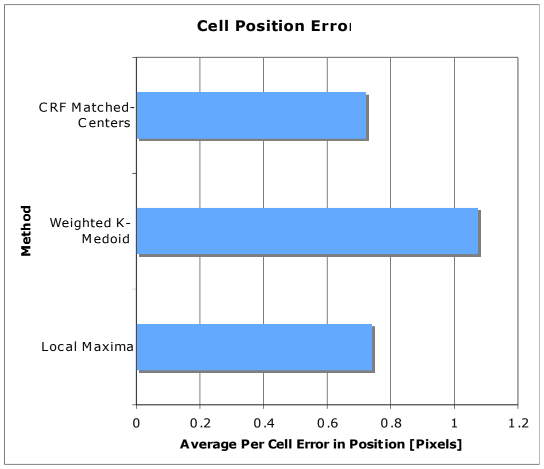


Figure 8. Cell position error is approximately 1 pixel for cells with diameters of approximately 6 pixels.

Cell Tracking and Track Merging

In the final step of the algorithm, we combine our cell position data to synthesize cell tracks through time. First level cell tracking is achieved through use of a Kalman filter [4]. The state of cell *i* at frame *k* is given by:

$$\mathbf{x}_k^i = \begin{pmatrix} x_k^i \\ y_k^i \\ \dot{x}_k^i \\ \dot{y}_k^i \end{pmatrix}, \qquad \mathbf{x}_k^i = \mathbf{A}\mathbf{x}_k^i + \mathbf{w}_{k-1}^i, \qquad \mathbf{A} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

We observe cell positions, given by:

$$\mathbf{z}_k^i = \begin{pmatrix} \hat{x}_k^i \\ \hat{y}_k^i \end{pmatrix}, \qquad \mathbf{z}_k^i = \mathbf{H}\mathbf{z}_k^i + \mathbf{v}_{k-1}^i, \qquad \mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

w and **v** represent process and measurement noise drawn from gaussian distributions.

Given a set of cell states for frame *k-1*, we calculate the cell states for frame *k* by applying the Hungarian algorithm to match observations in *k* with states from *k-1*, and then applying the appropriate Kalman filter. Any cells observed in *k-1* that are not observed in *k* are propagated according to the Kalman filter prediction to get a probable location in frame *k*. We allow such propagations to occur up to two frames in a row before we determine that a cell must have died. This allows us to continue tracking a cell even if we fail to identify it for one or two frames. Such a case might occur when two cells temporarily overlap, or our method simply fails to identify the cell.

Given these cell tracks, we then apply a higher level merging algorithm to merge tracks that represent the same cell or descendents of the same cell. This algorithm determines a track label for each track based on its distance, in both space and time, to other existing cell tracks. A typical result is shown in figure 9. The dataset we worked with was entirely unlabeled, so figure 8 represents the result of our cell tracking and track merging algorithm applied to hand-labeled cell position data.

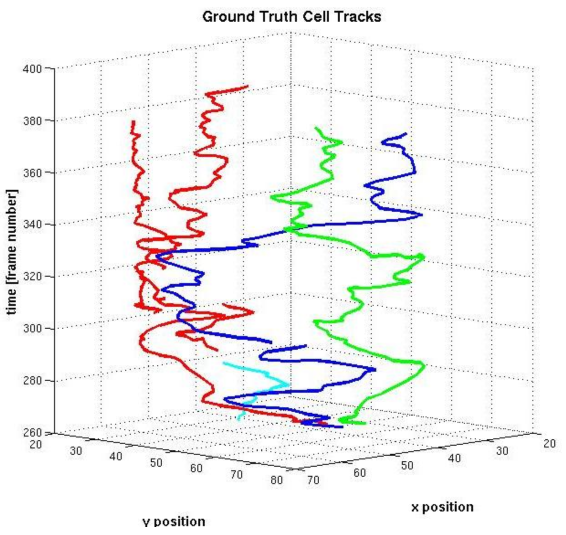


Figure 8. Cell tracks generated from hand-labeled cell position data. There are 4 parent cells. One of the cells (red) divides, yielding two children.

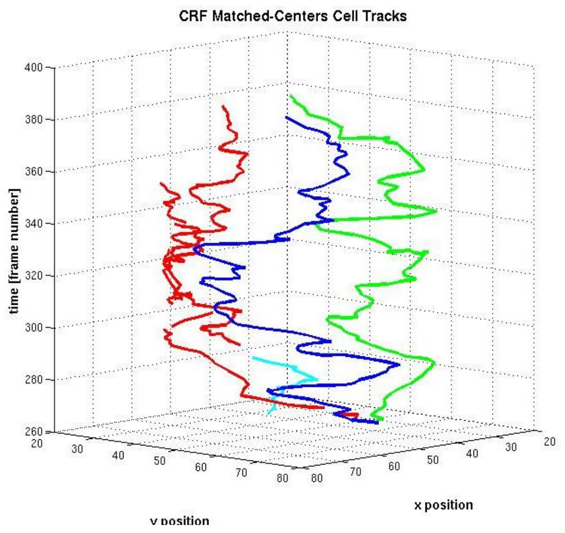


Figure 9. Cell tracks generated by our algorithm. The plot correctly shows 4 parent cells, and one of them (red) dividing. The plot incorrectly crosses the green and blue paths.

Conclusion

This paper details our technique for reconstructing cell genealogical histories. We accomplished this by first identifying and accurately counting the cells, then using the counts to confidently localize the cells. Given this data we use the Hungarian algorithm combined with a Kalman filter to establish correspondence through time. Our method localized the cells accurately and produced promising cell histories as a final result.

Moving forward we would ask the biologists to provide us with higher image and time resolution data to allow us to leverage more complex techniques, such as inference on cell contour shape, to achieve more robust cell detection. We would also like to incorporate the estimated cell positions into our CRF to further improve our results. Finally, we would like to quantify cell histories and compare the results to published techniques such as [5].

References

[1] "Tracking Stem Cell Genealogical Trees in Mircrowell" Ranganath and Kakaradov. CS223B Project. 2007
[2] Partitioning Around Medoids. 1987
[3] "Hungarian Algorithm" Matlab implementation by Alex Melin, 2006.
[4] "Kalman Filter Toolbox" Matlab implementation by Kevin Murphy, 1998.
[5] "Cell population tracking and lineage construction using Multiple-Model Dynamics Filters and Spatiotemporal Optimization"; Li, Kang; Kanade, Takeo; Proceedings of 10th Intl Conference on MICCAI; 2007, pp. 295-302.