Automated analysis of time-lapse microscopy sequences

Devang Mundhra, Klas Erik Gustaf Magnusson, Nikhil Gupta
Stanford University

Abstract: In this study, we show that it is possible to (semi) automate cell tracking and lineage construction to give useful information about cell characteristics in almost "real-time" speed. Images are first preprocessed to extract the microwells of interest, stabilized, denoised. This is followed by segmenting the cells using blob properties and k-means clustering and tracking them using linear programming to minimize the sum of squared distances between centroids in consecutive frames. After correcting the trajectories manually, images of lineage trees and cell properties over time can be obtained.

1. Introduction

Robotic time-lapse microscopy allows researchers to capture activity in thousands of microwells almost in real time. This process generates enormous amount of data in the range of 100s of GBs, yet it has to be analyzed manually, which is a severe research bottleneck. We worked on this real world problem with Blau lab at Stanford to try to automate the analysis that is done manually at present. We divided project into a number of achievable goals with the ultimate goal being the automatic reconstruction of lineage histories of proliferating cells from image sequences, and tracking their properties like shape and size.

1.1. Problem definition

The cells are constrained in circular microwells that are 100 μm in diameter. At the beginning of each experiment there are zero or more cells in some of the microwells (some are empty). Over time the cells divide and the microwells become populated by more than one cell. The goal is to gather as much information as possible about the cells using computer vision techniques. A first goal is to count the cells in a reliable way. The second goal is to create a family tree of how the cells divide. This should include the exact times of division and information about which cells give rise to which. When this is done in a reliable way it is desired to gather as much information as possible about each cell automatically. Among other things we would like to follow how the shape of the cells evolve over time, the distance each cell travels, the trajectory followed by each cell and when the cells touch.

The main difficulty is that the image sequences lack persistent features like texture and gradients as the cells change shape and interact with the microwell walls. They also change in number over time and sometimes stick to or overlap with each other. Usual vision techniques of clustering/edge detection for segmentation and classical tracking algorithms are not robust to such conditions. So, such automated analysis of images would require work on robust image segmentation and tracking.

1.2. Previous work

There has been rising interest in this field since the last few years. In [10] the authors talk about automated lineage construction for C. elegans. They also take a similar approach to what we have taken, however the experiments with C. Elegans last for much longer (around 70 hours), and the motion of the cells is much slower as compared to stem cells. This makes cell associatin and hence tracking easier. So their results are not directly available to the problem that we had at hand. Also, they take a probabilistic approach to the problem of tracking and splitting, while we take a much simpler linear programming approach that requires no training of the system. In [5] the authors do tracking by using a Kalman motion filter, which would require some ground truth data obtained by manually correcting training data. In [8] a template matching approach is used for segmenting the cells, while we use a much simpler segmentation scheme based on morphological operations and clustering. This was needed as the cells we worked on change shape and size quite often, and there can be no fixed template. In [7] & [9], again an active contours based approach for segmentation is used which would not work in our case, as when the stem cells dilate, they

almost become indistinguishable from the background.

1.3. Main results

We have mainly implemented – A functional GUI, Preprocessing, tracking, manual corrections, lineage trees construction and a cell property tracking.

Using computer vision techniques, we have been able to achieve –

- Preprocessing:
  Robust detection and cutting out of microwells with activity from the bigger image sequences.
  Automatic stabilization of microwells.
- Segmentation :
  Background subtraction and get blobs (using K-means clustering for tricky cases) by morphological operations
- Tracking and manual correction:
  Perform tracking on these centroids using intuitive and robust Linear Programming.

This is easily extendible to more complicated cases.

1.4 Available Data

The data available consists of sequences of TIFF images. It was constructed by photographing separate parts of a sample with some microwells, so that all images tiled together form an image of the whole sample. The entire sample is photographed in this way in about every 2.5 minutes to produce a step motion movie of the cells moving around and dividing. Blau lab provided us with a data set with 24.7 Gb of high resolution images from an experiment where there are muscle stem cells growing in about 50 separate microwells. The data set has 46 image sequences with 418 images each. The resolution of these images is 1344x1024 pixels. We also have one image sequence with 434 672x512 pixel images taken with a lower magnification.

Both the magnification and the image resolution of the camera can be selected. Obtaining images with a lower resolution than 1344x1024 is easily done by re-sampling the high resolution images we have. Simulating the effect of a lower magnification can also be done to some extent by re-sampling the images to have lower resolution. The lighting becomes a little different when a lower magnification is used though.

The lighting conditions vary a lot between different image sequences, and they also vary between images in the same sequence. Another problem is that the images are rather noisy.

2. Approach

Our main approach is the "tracking by detection" approach, to segment the cells in each image individually, and then try to build associations between cells frame to frame. The basic flow of the system is as follows –

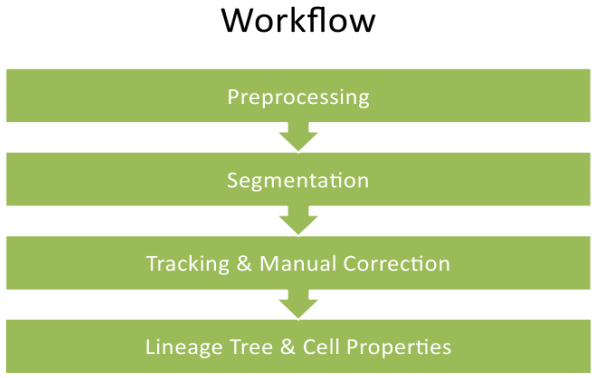

Figure 1 Approach

2.1 Cutting out microwells

To find the microwells containing moving cells in the images we start by resizing the images to have ¼ of the initial resolution in both x- and y-direction. This gets rid of a lot of image noise and also makes the microwell edges easier to find using an edge finding algorithm.

Then we apply a Sobel edge finder to the image. After that we apply a sequence of morphological operations to the resulting binary image. First we fill in possible gaps in the well borders using dilation with horizontal and vertical lines. Then we close the image using a disk shaped structuring element, also to fill in gaps in the borders. Then we fill all holes in the binary image, including holes that are cut by one side of the image or by two adjacent sides of the image. After this we open the image using a large disk shaped

structuring element, to get rid of noise between the wells. After this we take all the connected regions in the image to be possible microwells.

Using only the above algorithm, together with a condition on possible sizes of microwells gives good results in most cases, but using it together with an algorithm for fitting circles to the well boundaries turned out to give slightly better results in cases where unwanted objects close to the well boundaries.

To fit circles to the well boundaries we apply the same Sobel edge finder as above to the possible well regions found above. Then we fit a circle to all edge points found, using least squares. We use both the circle centre and radius as parameters. We could of course force the circle to have a known radius since all wells in the data set had the same radius. But we did not want the algorithm to depend on the size of the wells as future data sets might have a different resolution or well size.

After we have fit a circle to the points we see if the distance from any point to the circle boundary is less than 5 % of the circle radius. If this is not the case we iteratively remove the 5 % of the points that are furthest away form the circle, and fit new circles and until all points lie within 5 % of the radius from the circle. This turned out to remove unwanted objects that had had merged to the cell boundary, effectively.

To determine which micorwells have cells in them we looked at the sum of the absolute differences between consecutive images in a sequence with every fifth image out of the first fourth of the original image sequence. We look at only the first fourth of the original image sequence to be able to find cells that die in the beginning of the experiment. For such cells we would run the risk of setting the threshold on accumulated absolute difference to high we used the whole sequence. We look at every fifth image to allow all cells to move far enough for the absolute image difference to be large. Looking at a small subset of the images also makes the algorithm a lot faster. Using the cell detection algorithms described under segmentation below would probably be a better way of determining which wells have cells in them, but the well cutting algorithms were developed before we did segmentation, and they work satisfactory the way they are.

In the final step we reject all circles with a radius that deviates by more than 20 % form the median radius of all circles. We save images of all the wells that do not go outside the boundaries of the original image by more than 33 % of the circle radius. This seems to be a reasonable tradeoff between processing as many wells as possible while also avoiding cases where cells disappear outside the image.

2.2 Stabilization

The image sequences provided have a constant jitter motion, mostly by the motion of camera while acquiring images. It was necessary for us to get rid of this jitter to segment the cells more robustly and to determine the background for background subtraction.

Image stabilization is achieved using ImageJ plugin provided in [3]. It uses the currently shown slice in an image stack as the "template" and then estimates the geometrical transformation needed to best align each of the other slices with the "template". The estimation and alignment are performed using the Lucas-Kanade algorithm. Once a slice is aligned, the "template" will be updated on the fly using the formula: new_template = a * old_template + (1 - a) * newly_aligned_slice, where "a" is the "template update coefficient".

2.3 Segmentation algorithm

Segmenting out the cells reliably is an important task to be solved before a lot of the other goals could be achieved. It is difficult because the cells change shape over time. There are a lot of different ways to find cells in an image. Most of them either try to find cell boundaries; try to find areas corresponding to cells, or simply threshold some feature of the image pixels. Finding cell boundaries is likely to be hard when the cells are close together, and in some cases where the boundaries are diffuse.

Template matching is one way of finding areas that correspond to cells. In this method a template of the cells expected appearance is created. Then this template is correlated with the image to produce a new image where the pixel intensity gives a measure of how likely it is that there is a cell in that position. This approach is taken in [8], where the template used is simply a dark circle with a brighter ring around it. This also seems to be the approach taken by the team that worked on this project last year. They used machine-learning techniques to find a good template. Template matching is likely to handle cells that are close together better than algorithms finding boundaries or using thresholding. The main problem with this approach is that the muscle stem cells that we study can stretch out, so that they no longer match the template. Another problem is that some cells are dark with a bright ring around them, and some cells are bright with a dark ring around them. Using the local variance of the pixel intensities might be better than using the intensities themselves. Local variance was used successfully in [6] to find cells undergoing mitosis. Here a machine-learning algorithm was trained to recognize 24x24 pixel sub-images of dividing cells in 5 consecutive image frames.

Since the images are noisy, and have extreme light variances at times, we decided to work on gradient of the images for getting more robust segmentation. As the background does not change from image to image, we are be able to use background subtraction to preprocess the images to increase the performance of other segmenting algorithm. Background subtraction is used as a part of the segmenting algorithm described in [5]. Background subtraction is done by averaging the stabilized image over all the frames of the sequence. After this, the image is converted to binary image by thresholding, the value used for this operation has been determined empirically by us to be 0.0255 for getting good results. Then basic morphological operations like erosion and dilation are performed. Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries. Together, they provide much higher robustness to noise. Segmented objects with size less than 50 pixels were considered as noise and rejected.

Thresholding does not work in the cases where cells are clumped together; the segmented areas must be split into individual cells in some way. When the cells overlap, or when a cell is splitting into two, we determine the centroids based on local maxima in intensity image, and on positions of previous centroids. Splitting is done using simple k-means clustering, and on the basis of previous centroids (we split areas that contain more than one centroid form the last image).

We observed that most of the false positives occur near cell boundaries mostly due to jerk in the camera movement, so segments that are outside or close to the cell boundary are ignored to provide further robustness to noise.

2.4. Tracking

Let $q_k(\tau)$ denote cell centroid number k in image number τ. Further assume that there are m centroids in frame t and n centroids in frame t+1. Then we do tracking only by connecting the centroids $q_i(t)$ i = 1,2,…,m in frame t to centroids in frame t+1 $q_j(t+1)$ j = 1,2,…,n. We always match the maximum number of centorids possible, which is max(m,n). We do this without looking at any information from previous or future time frames.

We found the motion of the cells to be very unpredictable. For this reason, we decided to model the probability distribution of a cell centroid position in frame t+1 as a two dimensional Gaussian distribution with the covariance matrix σI, centered on the centroids position in the previous frame.

$$f(q^{(t+1)}) = \frac{1}{2\pi\left|\sum\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(q^{(t+1)}-\mu\right)^{T}\sum^{-1}\left(q^{(t+1)}-\mu\right)\right) =$$

$$= \frac{1}{2\pi\left|\sigma I_2\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(q^{(t+1)}-q^{(t)}\right)^{T}\left(\sigma I_2\right)^{-1}\left(q^{(t+1)}-q^{(t)}\right)\right) =$$

$$= \frac{1}{2\pi\sigma}\exp\left(-\frac{\left\|q^{(t+1)}-q^{(t)}\right\|^{2}}{2\sigma}\right)$$

To get the most likely matching of centroids we take the matching that maximizes the product of the transition probability densities of all the cells in frame t. If m > n we simply leave the unmatched centroids out of the product. We maximize the product by maximizing its logarithm, so our maximizing objective becomes

$$\log\left(\prod_{i=1}^{m} f(q^{(t+1)})\right) = \sum_{i=1}^{m} \log(f(q^{(t+1)})) =$$

$$= -m\log(2\pi\sigma) - \frac{1}{2\sigma}\left\|q^{(t+1)} - q^{(t)}\right\|^2$$

We see that the most likely matching of centroids is given by minimizing the sum of the squared distances between matched centroids in the two frames. We do this by solving a linear programming problem.

We introduce the binary variables xij that are 1 if centroid i in frame t is matched to centroid j in frame t+1. Let dij be the squared distance between centroid i in frame t and centroid j in frame t+1. We then put xij and dij into the vectors x = [x11 x12 ... x1n x21 ... xmn]T and d = [d11 d12 ... d1n d21 ... dmn] T. Our optimization problem can now be written as

$$\text{minimize } \mathbf{d}^T\mathbf{x}$$

$$st.$$

$$\sum_{i=1}^{m} x_{ij} <= 1 \quad j = 1,\ldots,n$$

$$\sum_{j=1}^{n} x_{ij} <= 1 \quad i = 1,\ldots,m$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} = \max(m,n)$$

$$x_{ij} \in \{0,1\} \quad i = 1,\ldots,m \quad j = 1,\ldots,n$$

The first constraint means that one cell in frame t can give rise to at most one cell in the next frame. The next constraint says that one cell in frame t+1 can only be generated from one cell in the previous image, or be created in some other way. The latter can occur for example by mitosis or when two cells that have been sticking together separate. The third condition says that if there are a smaller number of cells in frame t than in frame t+1, or if the number of cells is equal, all cells in frame t must be matched to cells in frame t+1. If on the other hand there are more cells in frame t, all cells in frame t+1 must be generated from cells in frame t. The last constraint says that the variables xij have to be binary. This constraint can be relaxed to

$$x_{ij} >= 0 \quad i = 1,\ldots,m \quad j = 1,\ldots,n$$

$$x_{ij} <= 1 \quad i = 1,\ldots,m \quad j = 1,\ldots,n$$

This is because solutions to the optimization problem must be corner points of the allowed region, and this will force all xij to be either 0 or 1.

The number of variables for each time step is small, and the problem is solved efficiently with MATLAB's built in linear optimization function "linprog".

3. Results

3.1 General Results

Each of these images contained about ten microwells but only a few of them (about 1 or 2) had any cellular activity in them. These microwells were differentiated automatically and only those microwells with cell movements were cut individually and saved. This resulted in *reducing the disk storage required to save the data by about 92%.*

3.1.1 Cutting Out Microwells

Following is an image of a typical time-lapse microscopy sequence and a corresponding microwell with cell activity inside identified and cut automatically from image.

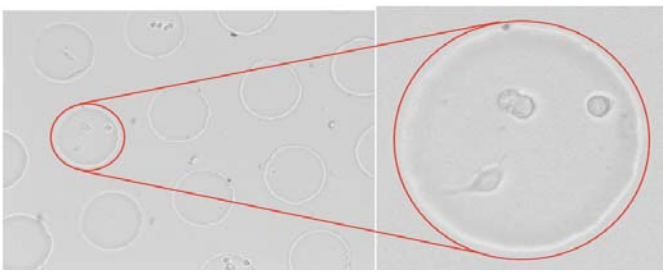

Figure 2 Active microwell identified and cut

3.1.2 Segmentation

After preprocessing the image (image stabilization, lighting invariance, denoising) the images cells were segmented on each frame. Cells with pixel area less than 200 pixels are discarded as noise. Below is a snapshot of the segmented cells for the frame shown above.

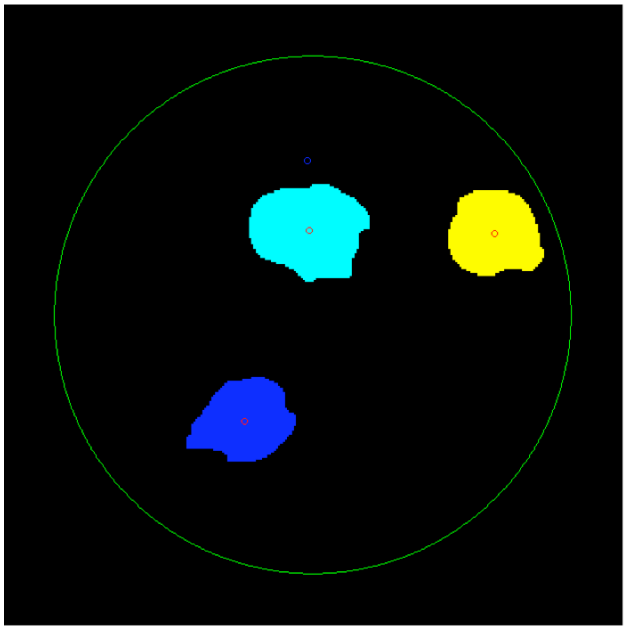

Figure 3 Segmented cells inside the microwell

A few remarks can be made on the segmented image and the correspondence between the original image and the segmented image.

- the segmented blobs corresponding to each cell in the image is only an approximate representation of the actual shape of the cell.
- when the cells come close to each other and touch or overlap, then the multiple cells are represented as a single blob (there are two cells corresponding to the light blue blob in the images above).
- the microwell boundary is represented as a circle and only changes inside this boundary is recorded to avoid getting false positives.

Another remark that is not apparent from this image is that if the cell dilates a lot, its central portion starts merging with the background resulting in over-segmentation. Another reason for over-segmentation is that we split segments that have multiple local maxima in smoothed and background subtracted intensity image. For cells where the center is darker than the background this often results in over segmentation, especially when the cells dilate.

- a dilated cell can be over-segmented to multiple small cells.
- if the cell dies, it is accounted for till it disintegrates and its size reduces to less than the threshold (200 pixels here). This might give an error in noting the life time of a cell that dies.
- Sometimes false positives are observed due to noise or some random structure not accounted for.

Directions to resolve a few of these problems are mentioned in the section *'Discussion and Directions for Future Works '.*

3.1.3. Tracking

The cells are tracked after that and the segmenting and tracking can be corrected manually when the automatic algorithm makes an error. The figure below shows the tracking of cells in a microwell over 6 frames.

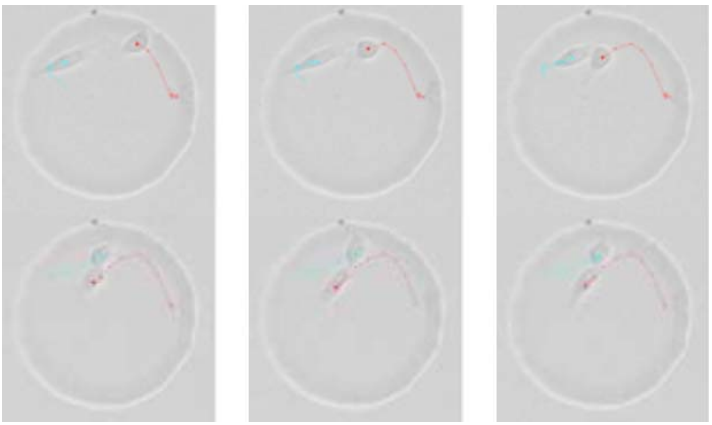

Figure 4 Cells tracked over 6 frames

3.1.4 GUI

To run all the different functions on the many images in an easy and efficient manner, a GUI was developed with the functions listed in separate categories as below.

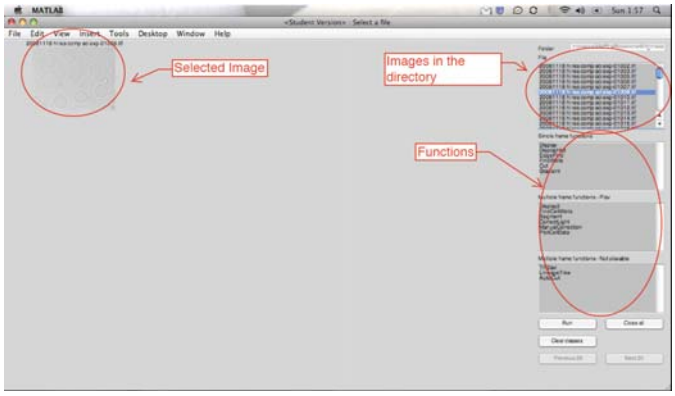

Figure 5 GUI developed and used

To process complete directories with a single command, a GUI was developed to fasten the process of automated cutting and tracking.

3.1.5. Obtaining relevant information

After the tracking is complete and manually corrected information about the cells is saved. This information can be used later to create lineage trees and study the characteristics of the cell like the distance traveled by each cell or the shape of the cells in terms of the ratio between its major axis length and its minor axis length, on a frame-by-frame basis.

Below are plots of lineage trees (Figure 6) and some cell properties (Figure 7 and 8) of a microwell for the complete sequence.

In the lineage tree plot below, the numbers at every node represents the level it is in, where each split in the cell represents a new level. The color of each branch in this and the following figures is the same color as was used for the trajectory in tracking and manual correction to ease relating the plot with the cells. The x-axis shows the frame number, the y-axis is immaterial here.

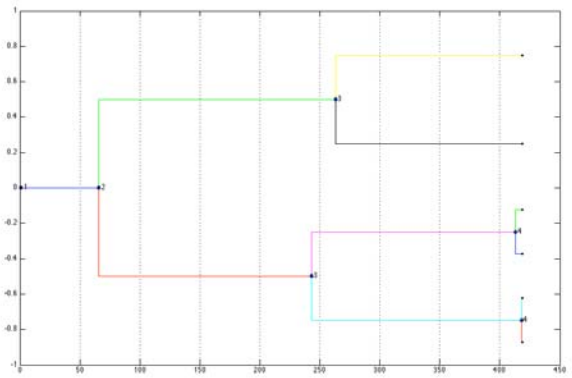

Figure 6 Lineage trees

The total distance traveled plot shows the cumulative distance traveled by a cell (in pixel units) at the frame on the x-axis. The slope indicates the speed of cell, the larger the slope, the more movement in that frame.

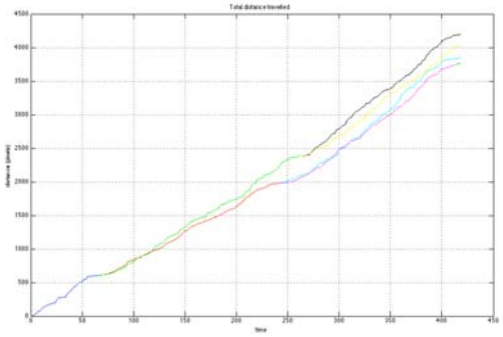

Figure 7 Plot of distance travelled by each cell

The plot below indicates the size of each cell in pixel units.

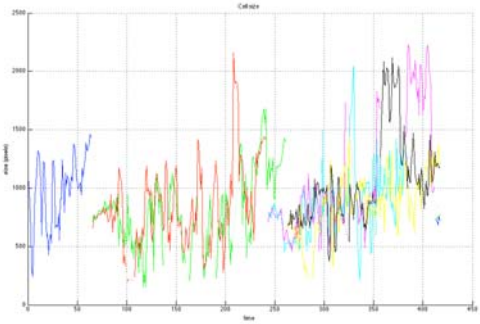

Figure 8 Size of each cell

As was noted above and can also be seen from the image above, the cell size does not seem to be very useful for analysis as the size of the cell has

some errors due to the segmentation method used. Also, for the purposes of this study, the cell morphology is a more interesting attribute that needs to be visualized. This can be conveniently plotted by plotting the ratio of the major axis length to the minor axis length of the segmented cell regions as shown below. Thus the closer the ratio is to 1, the more circular it is in shape.

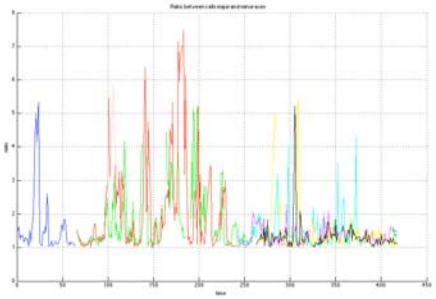

Figure 9 Cell eccentricity

3.2. Measures of Performance

3.2.1. Cutting out microwells

In the dataset consisting of 46 high-resolution images, we manually identified 70 microwells with cells in them. The algorithm for automatic cutting of microwells was able to find 67 of these wells, without finding any false positives. This corresponds to a precision of 0.96 and a recall of 1.0.

Two of the wells that the algorithm failed to find contained only one small cell that moved very little. The algorithm did therefore not find enough movement in the wells to detect them as occupied. In the third case there were a couple of regions with higher intensity than the background, close to the well boundary, that made the segmenting algorithm fail. All wells in the cut images were nicely centered in the middle of the image, except for the ones that were too close to the boundary of the large image to allow this.

3.2.2. Evaluating segmenation and tracking using manual correction

For evaluating the performance of the segmentation and tracking algorithms we ran the algorithms on 20 randomly selected microwells out of the 67 occupied microwells that the micowell cutting algorithm found in the data set. Then we corrected the created trajectories using the manual correction functions that we have developed. We chose to correct the tracks only when there were 4 or fewer cells in the images. This was because in a few sequences it becomes extremely hard, or impossible, even for a human to track the cells when there are more than 4 cells. Instead of just leaving these frames out we left all frames with more than 4 cells out. It is also the case that almost all image frames in our data set have 4 or fewer cells. Table 1 shows how many frames there were with different cell counts.

# of cells	# of frames	fraction of frames
1	2496	0.299
2	3253	0.389
3	215	0.026
4	1284	0.154
> 4	1112	0.133

Table 1. The number of frames in the test data that had 1, 2, 3, 4 and more than 4 cells.

3.2.2.1 Segmentation

For evaluating the cell detection we looked at the detections classified as cell centroids by the segmenting and tracking algorithms, and the centroids that were selected using manual correction. We also looked separately at the performance on frames with 1, 2, 3 and 4 cells present. We found the average precision and recall to be 0.928 and 0.903 respectively. The precisions and recalls depending on the number of cells in the frame are given in Table 1.

# of cells	Centroid precision	Centroid recall
1	0.8582	1.0000
2	0.9353	0.9204
3	0.9946	0.8514
4	0.9552	0.8399

Table 2. Precision and recall for cell detections, depending on the number of cells in the processed image. The average precision and recall were 0.928 and 0.903 respectively.

The main problem with cell detection is that we tend to loose cells when they come close to each other. This is why the centroid recall goes down

significantly when the number of cells increases in Table 2.

3.2.2.2 Tracking

For tracking we looked at the line segments connecting cell centroids from consecutive images in the cell trajectories. We defined a line as being correct if it both starts and ends at the same points in the automatically generated trajectories and in the manually corrected trajectories. We only included lines that either start or end at cell centroid positions in the manually corrected data. We did not find edges between false detections to be relevant for the performance of the algorithm. In any case these edges should not be seen as mistakes, because they make it easier to label a whole trajectory as not belonging to a cell. Looking at this subset of lines we got an average precision and an average recall of 0.990 and 0.884 respectively. We see that among the centroids that we are able to detect we are able to match almost all correctly, as the edge recall is almost as high as the centroid recall. The line precisions and recalls depending on the number of cells in the frame are given in Table 3.

# of cells	Line precision	Line recall
1	0.996	0.996
2	0.992	0.904
3	0.978	0.845
4	0.985	0.810

Table 3. Precision and recall for lines between cell centroids, depending on the number of cells in the processed image. The average precision and recall were 0.990 and 0.884 respectively.

The average length of correct trajectory segments in the uncorrected data was 38 time steps. This corresponded to 23 % of the lifetime of the cell being tracked. Figure 9 and 10 show histograms of the lengths of the correct trajectory segments in frames and fractions of the cell lifetime respectively.

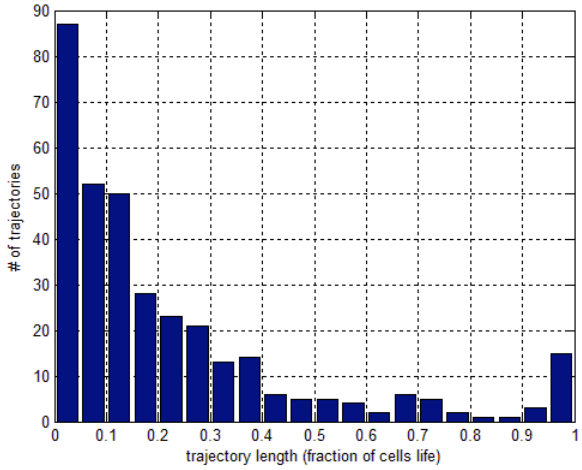

Figure 9. Histogram over the lengths, in frames, of correct trajectory segments.

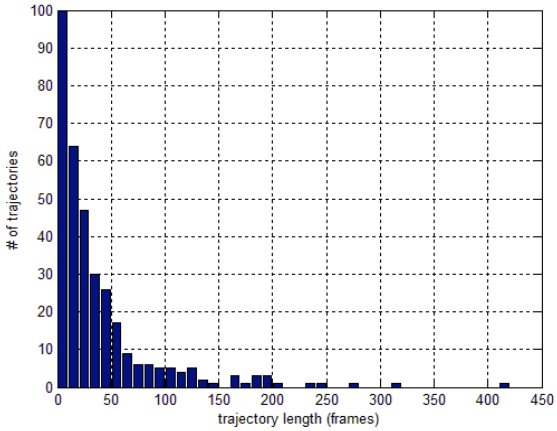

Figure 10. Histogram over the lengths, in fractions of the cell lifetime, of correct trajectory segments.

3.2.3 Algorithm Running Time

On a 2.4 GHz Intel Core 2 Duo Processor running MATLAB R2008b (Student Version), only segmenting the cells in 418 frames takes 45 seconds (about 0.1 seconds per frame) and automated tracking of the cell takes 2 minutes 15 seconds (about 0.3 seconds per frame) which gives almost real-time performances.

3.2.4 Manual Correction Time

We recorded the time needed for correcting each of the image sequences mentioned above, and found that the average processing time was 6 minutes and 35 seconds.

4. Discussion and Directions for Future Works

The study shows that it is possible to (semi) automate cell tracking and lineage construction to give useful information about cell characteristics in almost "real-time" speed.

Though the method mentioned above works quite well, as noted previously there is still scope for improvement, especially in making the segmentation more robust and reducing false positives. A few problems in particular are mentioned below and some directions are suggested to better the method.

- To reduce the errors in segmentation when multiple cells overlap or touch each other, it is suggested that a seeded watershed algorithm be used. This seems to be a good idea since the gradient image has local extrema at cell boundaries even when the cells overlap and so the watershed algorithm should be able to detect the boundaries robustly.
- Cell mitosis can be detected by keeping a count on the number of cells. An increase in the number of cells over a few frames can be attributed to one cell splitting into two cells whose centroids are closest to the parent cell centroid.
- There are still a number of false positives and missed cells. To reduce them, a manual ground-truth data set can be created by manually lineaging images and then using a training process to find the moments about some cell features like its centroids area and eccentricity. Thus a probalistic model of these cell features can be propagated through linear programming and decisions on splitting and false positives can be made based on maximum probabilities. One other problem with this approach is that the cells might move differently in other experiments. Therefore it might be a good idea to stick to the simple Gaussian probability distribution that we have assumed, to avoid overfitting the model to the particular data set that we were given.
- One could try to extend this approach by using a Kalman filter or a Particle filter to predict where the cells are going to be in frame t+1 based on the position, and possibly some other features of the cell in previous frames. If we call the predicted centorid locations xhat(t+1) we could minimize the squared distance between xhat(t+1) and x(t+1) instead of minimizing the distance between x(t) and x(t+1). One problem with the approach is that the cell motion is very random, and that a lot of the errors made are caused by large unpredictable motions that are unlikely to be modelled accurately by a Kalman filter or Particle filter. The Particle filter might have a greater chance of modelling this, because it allows more flexibility in the motion model.
- In addition to missed cells and false positives, a single cell in image at t + 1 can be detected as two due to segmentation errors. It must be ensured that we can reliably distinguish cell division from oversegmentation. This can be done by noticing that when the cells split, the usually have approximately the same areas. Thus a function taking parameter as the difference between the two cell size can be constucted and a threshold can be set to decide if the cells are children of a parent cell or if it is a single cell stretched.
- No heuristics has been developed to explicitly account for dead cells. This can be done by noting cells which do not move at all over some particular number of consecutive frames and mark the cell dead when it loses its motility. This lowers the chances of false negative if the cell is missed in frame t and marked dead in frame t+1.
- Changes can be done on the segmentation front to threshold the images adaptively and not modify the blobs much so that the recorded cell size is a more accurate function of the actual cell size.

5. References

[1] Handout by Professor Helen Blau handed out in class.

[2] "Cell Tracking Project" of The Robotics Institute, CMU http://www.ri.cmu.edu/ research_project_detail.html?type=publication&project_id=579&menu_id=261

[3] K. Li, "The image stabilizer plugin for ImageJ," http://www.cs.cmu.edu/~kangli/code/ Image_Stabilizer.html, February, 2008.

[4] Visual Dynamics Reseach Group, Oxford University http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/ISARD1/condensation.html

[5] Li, K., Miller, E.D., Weiss, L.E., Campbell, P.G., Kanade, T., "Online tracking of migrating and proliferating cells imaged with phase-contrast microscopy", In: Proc. IEEE Conf. Comp. Vision and Patt. Recog. Workshop, p. 65. IEEE Computer Society Press, Los Alamitos (2006)

[6] Kang Li, Eric Miller, Mei Chen, Takeo Kanade, Lee Weiss, and Phil Campbell, "Computer vision tracking of stemness", In: Proc. IEEE International Symposium on Biomedical Imaging (ISBI): Special Session on In Vivo Microscopic Image Analysis, May, 2008, pp. 847 - 850.

[7] Hailin Shen, Nelson Glyn, Kennedy Stephnie, Nelson David, Johnson James, Spiller David, White Michael R. H., Kell Douglas B., "Automatic tracking of biological cells and compartments using particle filters and active contours", In: Chemometrics and intelligent laboratory systems, 2006 vol. 82, no1-2, pp. 276-282

[8] Nezamoddin N. Kachouie, Paul Fieguth, John Ramunas, and Eric Jervis, "Probabilistic Model-Based Cell Tracking", In: International Journal of Biomedical Imaging, Volume 2006, Article ID 12186, Pages 1–10.

[9] Kang Li, Eric D. Miller, Mei Chen, Takeo Kanade, Lee E. Weiss, Phil G. Campbell, "Cell population tracking and lineage construction with spatiotemporal context", In: Medical Image Analysis 12 (2008), Pages 546 – 566

[10] Omar Al-Kofahi, Richard J. Radke, Susan K. Goderie, Qin Shen, Sally Temple, Badrinath Roysam, "Automated Cell Lineage Construction", February 2006, in Landes Bioscience.