Biological Cell Tracking

David Hernandez and Nivedha Sivakumar

May 2, 2018

1 Introduction

Why Cell Tracking? Living organisms, including cells, are constantly moving and evolving, requiring temporal study of their motion and features to fully understand them. For example, unicellular organisms move to find nutrients or avoid toxic substances. Tissue cells within multicellular organisms move during development of embryos, wound healing, tissue maintenance, generation of blood vessels, cancer metastasis and immune response [4]. Understanding the underlying mechanisms of cell motility is crucial for the development of treatments, for example to treat diseases caused by abnormalities in cell locomotion, and for a better general understanding of the natural world. Time-lapse microscopy of live cells produces massive amounts of data which are difficult to digest and analyze manually as a human observer [3]. Computer vision techniques address this problem as they can analyze microscopy images in an efficient and reproducible manner, producing useful quantitative data, for example about cell response to stimuli, cell growth and division rates, and much more.

Challenges Automated cell tracking is a complex vision problem to solve due to several reasons. Common challenges include variation in microscopy techniques, video quality, magnification, illumination, occlusion, variability in cell appearance, and cell proliferation and death. For this project we use a typical, challenging E. Coli dataset. It includes phase microscopy images of mobile E. Coli cells with air bubble debris in the image and large shifts in illumination. Furthermore, the E. Coli have complex motion as they can occlude each other and exhibit run and tumble behavior. In run and tumble behavior, flagella of E. coli cause the bacterium to either spin counterclockwise in place or move forward in a straight line, creating a "random walk" behavior [1]. Cells can stop suddenly and change direction, making them difficult to track with a motion model. Additionally, this random walk can be biased, for example, towards nutrients in the search for food or away from toxins. An example of an application of tracking for this dataset is the analysis of this biased random walk in response to different stimuli.

2 Approach

2.1 Overview

There are two major steps to the cell tracking problem: cell segmentation and multi-cell tracking. The overall workflow of our approach to cell tracking is as illustrated in figure 1.

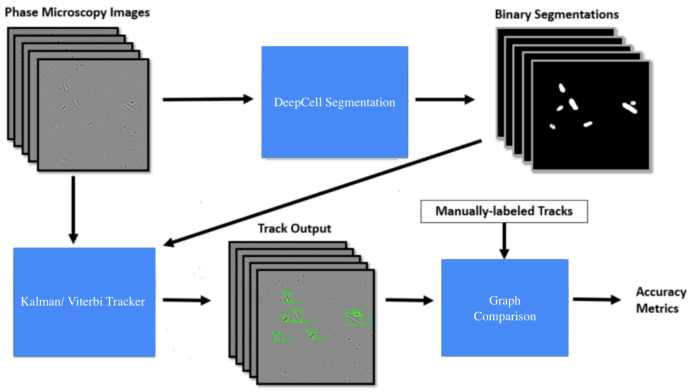

Figure 1: Workflow diagram: Diagram showing the full tracking process, starting with cell segmentation by the DeepCell network producing binary outputs, followed by the generation of tracks by the Tracker, and finally ending with calculation of accuracy metrics by comparing the track outputs to manually-labeled ground-truth tracks

Cell Segmentation Accurate segmentation is a challenging problem and usually requires problem-specific tailoring of algorithms. Some segmentation approaches that have been used include pixel-based techniques, the watershed algorithm and active contours. While these approaches have enabled the analysis of numerous experiments, they are limited in their robustness to experimental conditions and in generalizability. A major recent advancement is the use of deep convolutional neural networks for cell segmentation, which has proven to be very robust to experimental conditions such as lighting, debris in the image and variable appearances of cells. For our project we used DeepCell [5], a software package for a fully convolutional neural network designed specifically for cell segmentation. We trained the network with 10 manually-labeled images from a training video dataset and then ran it on the other 4 videos. This provided the segmentations used by the tracking algorithms described below.

Cell Tracking Once segmentations have been attained, the next challenge is individual cell tracking over a sequence of images. This is a challenging problem because cell segmentations, hereafter refered to as "detections", can contain multiple cells and because the results of segmentations are rarely perfect. Some examples of algorithms which have been applied to cell tracking in the past include multiple hypothesis tracking and joint probabilistic data association. For this project we implemented two trackers. The first combined a Kalman filter for tracking cell motion with the Hungarian algorithm for assignment of cell tracks to detections. The second tracker is an extension of a previous paper that used the Viterbi algorithm for tracking [2]. Our tracker introduces the use of individual cell tracks, rather than working only with detections, and also integrates a Kalman Filter motion model. These modifications provide the advantage of allowing tracking of individual cells, rather than just the movements of cells or groups of cells between detections, and make the integration of a motion model feasible.

2.2 Kalman-Hungarian Tracker

Kalman Filter Our first tracker works in a three-step process. First is the prediction step of the Kalman filter in which it produces estimates of the current state variables (in our case, state = [position, velocity]), along with updated state covariance. The second step is the use of the Hungarian Algorithm to assign tracks to detections. This can be thought of as assigning measurements (used in the measurement update of the Kalman Filter) to tracks. This assignment is done in a way that minimizes a cost metric, in our case the Mahalonobis Distance between tracks and detections. Finally, once these assignments have been made, the measurement update step of the Kalman Filter is performed, updating track states with the cell detection positions.

The process update of the Kalman Filter, as in equations 1 and 2, updates the predicted state of each track using the current estimates of position and velocity, and then updates the covariance of the new distribution of that predicted state. By including velocity in our state, the tracker is able to predict where the cells are going to move, making it robust to missed detections and multi-cell segmentations where multiple cells overlap for a short period of time.

$$\hat{x}_k^- = A\hat{x}_{k-1} \tag{1}$$

$$P_k{}^- = AP_{k-1}A^T + Q \tag{2}$$

Next, the Hungarian algorithm is used to assign detections to some tracks. This allows the measurement update to then be performed using the equations 3, 4 and 5. This step uses detection centroids as measurements to improve the accuracy of state estimates for each track. A total of only 4 tunable parameters, including initial covariance, process and measurement noise covariance and maximum number of missed detections, make the algorithm easy to modify for new datasets.

$$K_k = P_k^- H^T (HP_k{}^- H^T + R)^{-1} \tag{3}$$

$$\hat{x_k} = \hat{x_k}{}^- + K_k(z_k - H\hat{x_k}{}^{-1}) \tag{4}$$

$$P_k = (1 - K_k H)P_k{}^- \tag{5}$$

Hungarian Assignment The Hungarian method is a combinatorial optimization algorithm that solves the assignment problem in polynomial time. For the purpose of tracking, it assigns cell tracks to detections in a way that minimizes an overall cost metric. For each pair (track, cell) a cost must be computed for the potential assignment of the track to the cell. Our goal is to minimize total cost while assigning each track to exactly one cell segmentation and vice versa.

It is to be noted that we may have different numbers of tracks and cells in image frames. Since the Hungarian algorithm can only be used for one-to-one correspondence of tracks to detections, we had to experiment with different heuristics to handle differences between the numbers of tracks and cells. For frames with more detections than tracks, we start new tracks at the detections that are not assigned a track by the Hungarian algorithm. For frames with more tracks than cells, if tracks are within a pre-determined maximum search distance, we simply assign them to the closest detection. If the track is beyond this search distance from all detections, it is not assigned to any cell and is considered a missed detection. In this case, it is assumed that the segmentation algorithm failed to segment the cell in that frame and, thus, only the prediction step is carried out for that track until an assignment

is eventually found. To keep tracks from continuing forever without ever being assigned to detections, we set a maximum number of consecutive missed detections that each track is allowed before the track is ended.

We experimented and analyzed tracking performance with several cost functions for the Hungarian Assignment, and with other parameters such as maximum search distance. The cost function that worked best was the Mahalanobis distance. This is a distance metric that accounts for the covariance of a distribution, in this case the state covariance from the Kalman filter, when determining how far away a point is from the mean of a distribution, in this case the state estimate of position. Other distance functions we experimented with were Euclidean, Manhattan, and Minkowski norm-3 distance.

2.3 Tracking with the Viterbi Algorithm

Previous Work Cell tracking with the Viterbi Algorithm has been proven to be a robust cell tracking method. Generally, it uses a probabilistic Markovian Framework to only add cell tracks which are probabilistically-likely. One of our group members (David) submitted a conference paper recently to the MARSS conference which evaluated the combination of the DeepCell segmentation algorithm in [5] with the Viterbi tracker in [2]. This tracker uses "event" probabilities to form tracks between detections, which could contain one or more cells, and uses heuristics such as "swap" operations to further improve tracking. This resulted in tracking that did a good job of following cell movements between detections, but did not naturally result in individual cell tracks without post-processing. Also, it used a Brownian motion model which is not ideal for many datasets in which cells can move with a clear direction for periods of time, such as the E. Coli dataset used for this project. Adding a motion model to this implemention is infeasible as it would lead to an exponential increase in the number of possible states that would need to be considered to find the most likely cell tracks.

Contributions of our Project We implemented a novel tracker which uses a similar idea of adding probabilistically-likely tracks using the Viterbi algorithm with key differences. First, our tracker is based on individual cell tracks, rather than just the movement of cells or groups of cells between detections. This allows us to easily reconstruct individual cell tracks in an output image sequence and, importantly, to add to a motion model without having an exponential increase in run-time. Furthermore, our implementation still assigns tracks to detections if it is probabilistically-likely and can therefore still be represented as a forest graph model for analysis of track accuracy, as is explained in the results section below.

Algorithm Summary Our algorithm involves repetition of a few key steps until completion. It begins by initializing tracks at each detection in the first frame. Then, in a loop, it iteratively performs the Kalman filter prediction update for each track, assigns tracks to detections in a probabilistically-motivated fashion based on the Kalman Filter state covariance, creates possible tracks representing missed detections, and finally performs the measurement update of the Kalman Filter based on track assignments to detections. In each iteration, track scores are updated based on how likely their assignment to a detection or missed detection is. Tracks are added to the set of possible tracks until no likely track can be found that has not passed the maximum number of allowed consecutive missed detections, a parameter which can be tuned for different datasets, or until the last frame has been reached. Then, the single most-likely track is added to the output graph and final set of tracks. The whole process is repeated until all detections have at least one track passing through them and tracks beginning at all frames have been considered.

3 Results

Figure 2: Graphical representation example: From left to right: Normalized cell image, binary output from deep cell, representation as a column in forest graph F

Evaluation with a Graphical Framework The cell tracks within a video can be represented as a directed acyclic graph, or forest graph, where nodes represent individual detections that can contain one or more cells, and edges represent migration of cells or groups of cells between those detections. Each column of nodes represents the detections in an individual frame of the video. An example of such a forest graph is shown in Fig. 2. For each video we manually labeled the first 20 frames, making ground-truth graphs that could be used to evaluate our trackers' outputs. For each tracker we implemented, we started with a graph with the same set of nodes and no edges, and used the tracks generated to form edges between nodes by looking at which detections the tracks were assigned to.

Then, graph difference operations were used to get information about the number of shared edges between the two graphs, as well as the number of edges that exist in one graph but not the other.

Quantitative Results The results of the segmentation network are presented in Table 1. Despite occlusion and variable lighting, the segmentation network was able to pick out 94.2% of the cells in the image. Furthermore, there was not a single false positive detection which shows that the segmentations made by the network can be very reliably considered to result from cells.

DeepCell Segmentation Accuracy			
Video Number	Missed Detections	Total Detections	Percent Accuracy
Video 1	0	207	100.0 %
Video 2	52	579	91.0 %
Video 3	30	682	95.6 %
Video 4	20	292	93.2 %
Overall	102	1760	94.2 %

Table 1: *DeepCell detection accuracy on E. coli dataset*: Results from manually checking frames 1, 15, 30, ..., 300. False positive rate was 0% in these frames

Table 2 contains the accuracy of results for different cost metrics for Hungarian assignment. The distance metrics tested were Mahalanobis, Euclidean, Manhattan and Minkowski (norm-3) distances. From the below results, it is evident that the best results are obtained for Mahalanobis distance as the total number of incorrect edges (sum of edges in truth but not test, and in test but not truth) is lowest for this distance metric. This is an intuitive result as Mahalonobis distance takes into account the covariance of data to compute correspondences. The number of edges reported in Tbl. 2 is summed over all videos.

Kalman-Hungarian Tracker Accuracy with Different Cost Metrics					
Distance metric	Number of edges in ground-truth	Number of edges in Kalman Filter output	Number of Edges in Intersection	Number of edges in truth but not in test	Number of edges in test but not in truth
Mahalanobis	1329	1291	1206	123	85
Euclidean	1329	1374	1194	135	180
Manhattan	1329	1354	1179	150	175
Minkowski Norm-3	1329	1382	1174	155	208

Table 2: *Kalman-Hungarian tracker output accuracy*: Comparison of output directed acyclic graph from the Kalman Filter-based tracker with manually-annotated tracks over 20 frames over all video frames

The breakdown of results for the Kalman-Hungarian tracker using Mahalanobis distance cost over the 4 test videos is shown in Tbl. 3. The results for the Viterbi Tracker for the same videos are shown in Tbl. 4.

Kalman-Hungarian Tracker Accuracy Using Mahalonobis Distance					
Video Number	Number of edges in ground-truth	Number of edges in Kalman Filter output	Number of Edges in Intersection	Number of edges in truth but not in test	Number of edges in test but not in truth
Video 1	134	129	126	8	3
Video 2	477	475	449	28	26
Video 3	564	547	495	69	52
Video 4	154	140	136	18	4

Table 3: *Kalman-Hungarian tracker output accuracy with Mahalanobis distance for Hungarian assignment*: Comparison of output directed acyclic graph from the Kalman-Hungarian based tracker with manually-annotated tracks over 20 frames in each video

Qualitative Results of Kalman-Hungarian Tracker We picked some interesting cases to qualitatively discuss results of the Kalman-Hungarian cell tracker. Refer to the attached videos and README files for examples of good and bad qualitative results of cell tracking. From figure 3, we can observe that the tracker works well for cells that exhibit long straight motion. The black-and-white figure on the left shows tracking on binary masks, and the greyscale image on right shows tracking on the original microscopy images. Figure 4 illustrates that the tracker also works well on low velocity tracks; most tracks adhere well to the actual cell movement. Some cells in this figure are in the tumble phase of 'run and tumble behavior', however, it is easier seen as a video sequence. The Kalman-Hungarian tracker shows good results for cells in both the run and tumble phases of their random walk.

Viterbi Tracker Accuracy					
Video Number	Number of edges in ground-truth	Number of edges in Kalman Filter output	Number of Edges in Intersection	Number of edges in truth but not in test	Number of edges in test but not in truth
Video 1	134	131	122	12	9
Video 2	477	483	409	68	74
Video 3	564	571	414	150	157
Video 4	154	144	124	30	20

Table 4: *Viterbi tracker output accuracy*: Comparison of output directed acyclic graph from the Viterbi tracker with manually-annotated tracks over 20 frames in each video

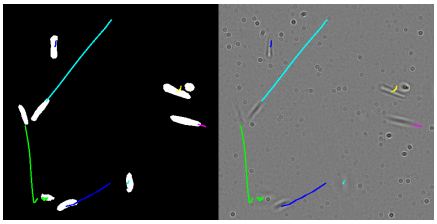

Figure 3: Good tracking for straight cell movement

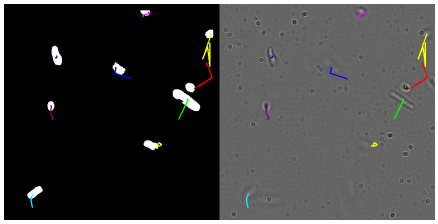

Figure 4: Good tracking for low velocity tracks

The image sequences below illustrates other cases where the tracker works well. In a few cases, DeepCell failed to detect cells in some frames. However, the blue track in figure 5 tracks the cell even when it is not segmented correctly by DeepCell. The image in middle frame has a missed detection but, despite this, the blue line picks up the cell track once again when the detection reappears in the bottom image. Yet another example of good tracking is illustrated in figure 6, where the blue and pink tracks stay with their corresponding cells through a multi-cell detection by using the velocity information from their state.

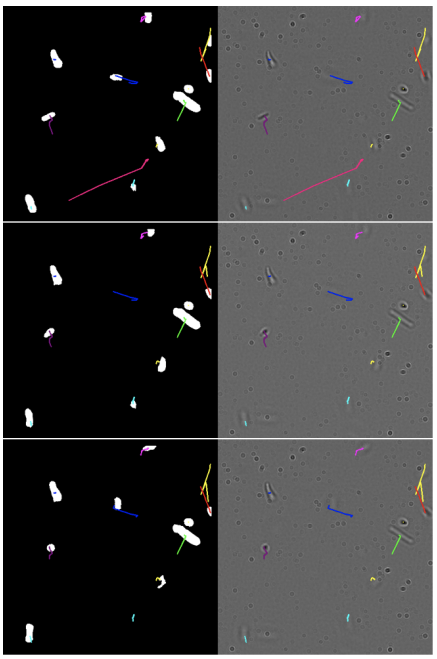

Figure 5: Good tracking for missing segmentations

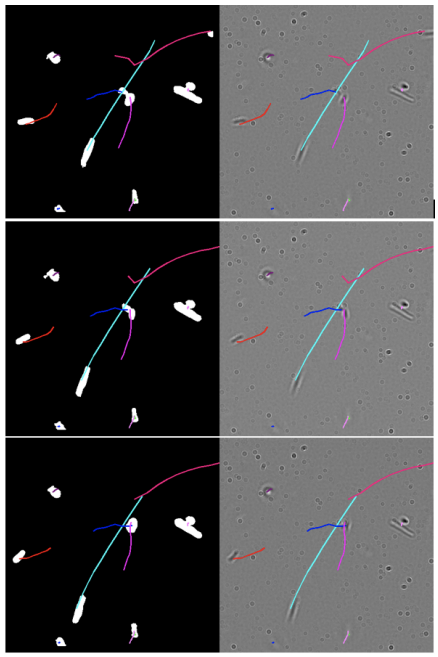

Figure 6: Good tracking for merging and separating cells

Although there are several positive results, we observe some cases when the tracker did not perform well. Figure 7 is an example of the tracker assigning multiple tracks to the same detection, even though it only contains a single cell. Figure 8 is an example of a track (in example, blue) which completely changes direction after being incorrectly assigned to a different cell. Both of these examples demonstrate shortcomings due to the limitations of the Hungarian assignment algorithm.

Qualitative Results of Viterbi Tracker The Viterbi tracker output had noticeably distinct behavior compared to the Kalman-Hungarian tracker. It still exhibited some of the advantages of adding a motion model and had improved assignment handling in a few cases, but it proved difficult to tune the different parameters in the algorithm to correct

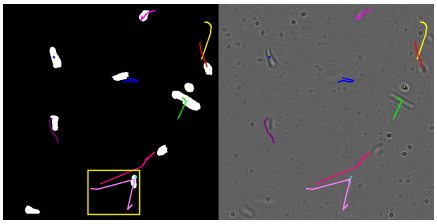

Figure 7: Multiple track assignments for a detection

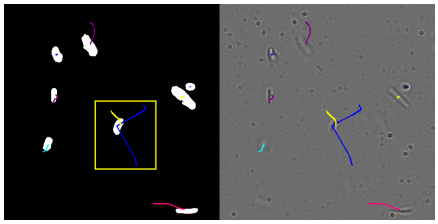

Figure 8: Random track switching

for some common errors. Fig. 9 shows an example in which the tracker mixed up assignments of tracks to nearby cells, demonstrating the limitations of a motion-only based probabilistic model, as multiple tracks are assigned to individual cell detections. Fig. 10 shows how the motion model did help track cells through multi-cell detections in some cases, but still had an erroneous assignment of the extra yellow track to the wrong cell.

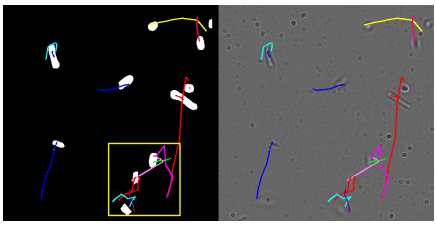

Figure 9: Confusion in track assignments with tracks switching between nearby detections

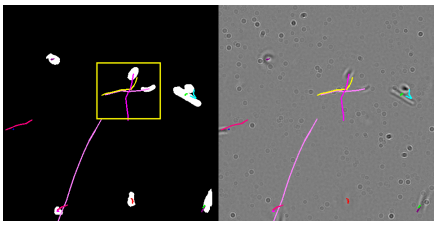

Figure 10: Working motion model but multiple tracks assigned to single-cell detection

4 Conclusion

For our final project we implemented 2 cell trackers: one that uses a combination of a Kalman filter and the Hungarian algorithm, and another which applies the Viterbi algorithm framework to tracking of individual cells with an integrated motion model. Experimental results show that both proposed algorithms are able to do robust multiple object tracking within dynamic scenes while addressing challenges such as tracking cells through multi-cell detections and tracking the run and tumble behavior of cells.

The use of a Kalman filter provides an improved performance by using quantities that are partially or completely neglected in other tracking techniques. It provides estimates of both position and unmeasured velocity and gives information about the quality of those predictions by providing variance estimates. For the Kalman-Hungarian tracker, its recursive structure, based on the Markov Assumption, allows its real-time execution without storing observations or past estimates. For the Viterbi tracker, it provides a way to account for cell motion in a way that allows for improved tracking of individual cells through missed detections and multi-cell clusters.

Each implementation comes with its own set of shortcomings. The Kalman-Hungarian tracker suffers the most from the inherent limitations of the Hungarian assignment algorithm. The algorithm is only able to perform one-to-one assignment and thus requires the management of extra tracks and detections with imperfect heuristics. These cause occasional mismatching of tracks to detections and the creation of new, unnecessary tracks. Future areas for improvement to this algorithm include investigation into better heuristics or alternatives to the Hungarian algorithm for assignment of tracks to detections.

The Viterbi tracker was able to provide comparable performance to the other Viterbi trackers with minimal tuning and testing of probability models for motions and cell counts. It resulted in decent tracking of individual cells, but requires improvements before having higher accuracy than the other existing Viterbi trackers in the Directed Acyclic Graph evaluation framework. One potential area for improvement is to include features of the segmentation contours to improve probabilistic matching of tracks to nearby cells, as was done in the other two Viterbi trackers. The current implementation is purely centroid-based and does not take advantage of all of the available information from each image. Potential features include contour features such as area, circularity measures, mean distance and standard deviation from the centroid. Features could even be extended to include visual features of the pixels in the raw microscopy images corresponding to the binary segmentations. Use of these features would allow the tracker to make more informed decisions when assigning tracks to detections.

Finally, to fully evaluate both algorithms, we would have to test their generalizability to new, diverse datasets and do more investigation into other state-of-the-art cell trackers which are available. This is an important part of cell tracking algorithms because each biologist who may be interested in using a tracker will have a unique experimental setup and a different set of applied microscopy techniques. Furthermore, to make either tracker useful to biologists who may not have programmatic experience, the trackers would have to be integrated into an easy-to-use software package.

References

[1] J. T. Locsei. "Persistence of direction increases the drift velocity of run and tumble chemotaxis". In: *Journal of Mathematical Biology* 55.1 (July 2007), pp. 41–60. ISSN: 1432-1416. DOI: 10.1007/s00285-007-0080-z. URL: https://doi.org/10.1007/s00285-007-0080-z.

[2] Klas EG Magnusson et al. "Global linking of cell tracks using the viterbi algorithm". In: *IEEE transactions on medical imaging* 34.4 (2015), pp. 911–929.

[3] Erik Meijering, Oleh Dzyubachyk, and Ihor Smal. "Chapter nine - Methods for Cell and Particle Tracking". In: *Imaging and Spectroscopic Analysis of Living Cells*. Ed. by P. Michael conn. Vol. 504. Methods in Enzymology. Academic Press, 2012, pp. 183–200. DOI: https://doi.org/10.1016/B978-0-12-391857-4.00009-4. URL: http://www.sciencedirect.com/science/article/pii/B9780123918574000094.

[4] Ahmet Sacan, Hakan Ferhatosmanoglu, and Huseyin Coskun. "CellTrack: an open-source software for cell tracking and motility analysis". In: *Bioinformatics* 24.14 (2008), pp. 1647–1649. DOI: 10.1093/bioinformatics/btn247. eprint: /oup/backfile/content_public/journal/bioinformatics/24/14/10.1093/bioinformatics/btn247/2/btn247.pdf. URL: http://dx.doi.org/10.1093/bioinformatics/btn247.

[5] David A Van Valen et al. "Deep Learning Automates the Quantitative Analysis of Individual Cells in Live-Cell Imaging Experiments". In: *PLoS Comput Biol* 12.11 (2016), e1005177.