TrackLife: A Pipeline for Detecting and Tracking Cells from Time-Lapse Microscopy

Darin Boyes, Jonathan Zhu, Michael Zheng, Shivank Sadasivan
Ray and Stephanie Lane Department of Computational Biology
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We developed a computational pipeline to model and simulate cellular dynamics from time-lapse microscopy images. The pipeline integrates watershed segmentation to delineate individual cells and the Hungarian method to solve the cell tracking problem as a graph-based optimization. This approach effectively simulates cell movement and behavior, capturing dynamic interactions and transitions across frames. Tested on the MI subset of the ALFI dataset, our model successfully tracks U2OS cells over 69 frames, simulating over long period of cellular activity. By leveraging segmentation to generate markers and optimization for trajectory linking, the simulation accounts for challenges like overlapping cells and low contrast.

Keywords: Cell tracking, modeling,watershed segmentation, Hungarian method

1 Introduction

Cell tracking from time-lapse microscopy plays a pivotal role in understanding the complex dynamics of cellular behavior, movement, and division within a controlled laboratory environment. This process is crucial for a variety of fields, including developmental biology, cancer research, and microbiology. By tracking cells over time, researchers can derive quantitative data such as cell velocity, movement direction, and division rates, which are valuable metrics for studying cellular processes under varying experimental conditions. These insights not only validate, but also refine computational models and simulations of biological phenomena, helping to uncover patterns in cell dynamics.

However, accurately detecting and tracking cells is a challenging task because of several inherent complexities. Cells exhibit diverse shapes and sizes, making it difficult to standardize detection techniques. Overlaps between cells during division or movement further complicate segmentation. Microscopy's inherent limitations, such as low focus accuracy, noise, and contrast issues, add another layer of difficulty. In addition, the dynamic and often unpredictable movements of cells, coupled with the large volume of data generated by time-lapse imaging, require computationally efficient and robust solutions.

Many current methods to track cells take place in two different segments: the determination of cells in a microscope image, and then tracking cells between these images in a video. Current methods to determine cell location rely on implanting the cells with a fluorescent protein or marker gene [1]. Algorithmic methodologies to analyze images are in continuous development and rely primarily on results from computer vision to determine the placement of cells; the fluorescent labeling of cells helps to distinguish the colored cells from the black background. While many algorithms achieve good performance, none get perfect results [2]. Furthermore, benchmarks for the development and testing of tracker algorithms rely primarily on fluorescently-labeled cells [3, 4], even though fluorescent labeling is not always possible. Tracking cells between images relies on a variety of methods, ranging from graphical approaches of shortest-path problems to propagating labels through a series of images based on maximum overlap between the images [2]. Again, while these can perform well, none of them achieve perfect accuracy. As such, there is a need for a more robust and generally applicable cell tracking algorithm that can be used on a variety of cell images with high accuracy.

In our project, we aim to follow the same framework of determining the location of cells in a microscope image and then track cells between such images, but we plan to do so with robust, sensitive algorithms that can effectively track cells for a variety of images.

2 Methods

To address these challenges, various methods have been developed, with image segmentation and cell tracking at the forefront. Image segmentation involves dividing an image into meaningful regions to isolate individual cells from their background. Techniques such as thresholding, edge detection, and more advanced approaches such as the watershed algorithm have proven to be effective in handling low-contrast and noisy images.

Once cells are segmented, cell tracking algorithms link the positions of cells across consecutive frames to establish movement trajectories. Approaches range from simple nearest-neighbor tracking to more complex probabilistic and optimization-based methods like the Hungarian algorithm.

These methods aim to accurately track cells, even under conditions of occlusion, division, or movement out of frame. To explore these aspects, we analyzed a dataset that provides a diverse and challenging benchmark for testing cell tracking methodologies.

2.1 Data Sourcing

Our dataset is sourced from Antonelli et al. [5] in their ALFI dataset. This dataset is comprised of thousands of time-lapse microscopy images of motile, dividing, and dying cells. We specifically focus on their MI dataset, which contains cells from U2OS, HeLa, and hTERT RPE-1 cell lines under varying concentrations of DMSO or MLN8237, a known anticancer agent [6]. As per the specifications of Antonelli et al. [5], images are taken 7 minutes apart; with 69 images for each cell culture, this means cells are tracked for just over 8 hours.

2.2 Image Segmentation

Cell images cannot be automatically input into any tracking algorithm in their raw form; we require a segmentation method that can generate masks from cell images. Such masks must be binarized to distinguish between true cells and background. We attempted several different methods for image segmentation, discussed below.

Edge Detection With Simple 3x3 Kernel One of the classic methods for edge detection is to element-wise multiply and then sum a 3x3 kernel with each 3x3 region of pixels in the image. A typical kernel that works well in a variety of edge detection cases features -8 as the central element and fills all other elements with one. This kernel is then applied as follows:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} = -8x_{22} + \sum_{\text{all other } i,j} x_{ij}.$$

It follows naturally that when all pixel values are highly similar, the resulting value will be close to 0. However, image edges typically consist of boundaries between colors, resulting in different values across the matrix, and the resulting sum will be much higher than 0. When the kernel is applied to all possible 3x3 matrices in the image, the result is a matrix slightly smaller than the original image whose values distinguish edges (high) versus fill (low). Despite this, classical kernel detection fails at our microscope image dataset; this is most likely due to the images having monotone colors overall.

Sobel Filtering An improvement on the single 3x3 filter is the Sobel Operator or Sobel Filter. This involves transforming each 3x3 region of the matrix into two sums rather than one; like the single filter, these capture changes in pixel values but do so in both horizontal and vertical directions [7].

Formally, the Sobel Operator applies a horizontal transform given as

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}$$

and a vertical transform definied similarly as

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}.$$

This allows us to define a "gradient" for each point on the image, representing the amount and direction of change in the image's pixels. We define the amount of change as

$$G = \sqrt{G_x^2 + G_y^2};$$

much like the classic single filter, higher values of this gradient correspond to edges. The direction is calculated as

$$\Theta = \arctan \frac{G_y}{G_x}.$$

Despite improvements in many images, the Sobel filter is unable to achieve meaningful detection results on microscope images, again due to relatively similar pixel intensities across the image.

Canny Edge Detection A method building on the use of image gradients is the Canny edge detection method [8]. This method follows five parts; firstly, it applies a 5x5 kernel (much like classical edge detection 3x3 kernels) whose values are determined by a Gaussian probability density function whose mean is based on kernel size and current position and whose standard deviation is constant across all cells. In short, this helps to "smoothen" or blur the image and aims to reduce noise. Secondly, it calculates the gradients across the image. Thirdly, each pixel is analyzed in the direction of its gradient; if it is not a local maximum in that direction, the value is set to 0. Fourthly, two thresholds are applied to the remaining pixels to distinguish between edges and "weak" edges, which are edges in which we have less confidence. Finally, the weak edges are examined one more time to determine their connections to edges; weak edges without such a connection are determined to be noise. Once again, this method does not show promise in detecting cells in microscope images as it builds on methods that do not have such capabilities.

Heuristic Methods Since traditional methods did not work, we turned to experimenting with heuristic methods. One method that showcased some promise was to apply a 5x5 Gaussian blur kernel (as done in Canny edge detection) and simply threshold the image directly; this makes most portions of the image black but produces a few white dots in the image. While this does a poor job at converting to a mask in and of itself (due to high similarity in pixel intensity), this does identify certain points where cells exist as the white dots mainly occur where cells exist. As such, placing a boundary on these points (i.e., clustering them) would identify cells in our image and allow for binarization. If we think about each of these dots as a node in a fully connected graph, this then becomes a max k-cut problem. However, this type of thinking poses several problems:

1. The max k-cut problem is intractable in polynomial time;
2. There is no way to automatically determine an appropriate k; and
3. False positive dots exist due to noise in the image.

Fortunately, we can apply another heuristic method that takes advantage of these nodes existing in space with specific positions. By randomly selecting some number of dots and expanding their pixel range (i.e, placing larger white circles on top of the randomly selected dots), we describe a Monte Carlo method that covers a majority (if not all) of the cells, reduces false positives, and is trivial to show polynomial runtime. Our experimentation produced decent results when picking 25 dots at random and applying circles of radius 50 pixels over them.

Watershed While the above heuristic is acceptable for fast running time and approximate results, more accurate results can be obtained using lesser-known, yet higher-performing, algorithms. One example that produced the best results is the Watershed algorithm. Watershed takes its name from real-world watersheds, which are geographic areas that define a drainage basin's capture area. In simpler terms, any drop of water that falls in a specific watershed will flow into that watershed's drainage basin (Figure 7).

When applied to images, we transform an image into a topographical map by assigning pixel value as height (Figure 7). Much like their real-world counterparts, this topographical map defines one or more watersheds that can be used to segment the image. A common method to do this also takes inspiration from real life, Meyer's Flooding Algorithm. This method simulates "rain" falling on the topographical map of this image and then flowing into the drainage basins of the various watersheds. In more technical terms, points are scattered across the image's pixels and the direction of steepest decrease (i.e., the gradient) is calculated for each one. The points then traverse in their direction of steepest decrease, and this process is repeated until no more traversal occurs. At this point, there will be many points in several distinct basins. The collections of points in the basins are traced back to their starting points; all of the starting points for the points ending up in a single basin defines that basin's watershed and therefore becomes a distinct area of the image.

While the Watershed method has high time complexity and is not guaranteed to converge, our experimentation showed that Watershed had incredible sensitivity to segmenting cells from the background in the microscope images. As such, we used Watershed to produce masks for all microscope images, which enabled us to apply tracking algorithms.

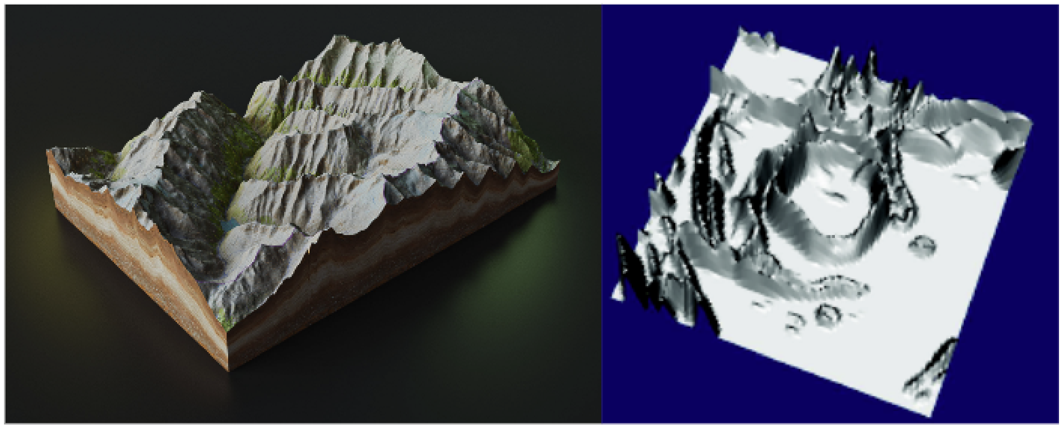


Figure 1: Watersheds. (Left) An example of a real-life topographical map defining a watershed, with the outer mountains being its boundaries; any drop of water that falls in this region will flow to the watershed's lake, which acts as its drainage basin. (Right) An example of an image-constructed watershed, where pixel intensities correspond to height. Images sources from 3d-mapper.com (left) and wikipedia.com (right).

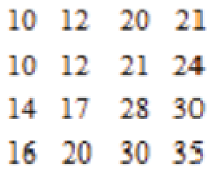
2.3 Cell Tracking

Linear Assignment Problem Given that we have a set of markers belonging to each segmented object in the image, we can then find an optimal assignment between frames that links each of the cells in one frame to a unique cell in the next frame. A classical way of achieving this is by framing this problem as the Linear Assignment Problem, or LAP. One way of intuitively thinking about how to set up the LAP is by assigning workers to tasks, but the cost is the amount of time that it takes for a specific worker to do a specific task. The input to the problem is thus a weighted bipartite graph and the output is a minimum-weighted bipartite matching that assigns elements in the first bipartite set to elements in the second bipartite set. Cell tracking can also be set up as a linear assignment problem, where we might have m cells in frame i, n cells in frame $i + 1$, and we want to find the minimum weight assignment of frame i cells to frame $i + 1$. The weight for the cell tracking problem can be represented as the euclidean distance between cells between the two frames.

Cost Matrix The first step to solving the LAP is to construct a cost matrix. To do so, imagine we have 2 frames, f_1 and f_2, containing m and n cells, respectively. The cost matrix, C, is an $m \times n$ matrix of weights, and C_{ij} represents the Euclidean distance between cell i in f_1 and cell j in f_2.

$$C = \begin{bmatrix} C_{11} & . & . & . & C_{1n} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ C_{m1} & . & . & . & C_{mn} \end{bmatrix}$$

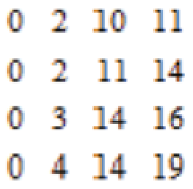
Hungarian Method - Step 0 Having constructed a cost matrix, we can then solve the linear assignment problem in polynomial time using the Hungarian method. The steps of the Hungarian method will be illustrated using an example from Medina-Acosta and Delgado-Penín. In the example C is a 4×4 matrix, so we can imagine that there are 4 cells in f_1 that we want to assign to the 4 cells in f_2.



10	12	20	21
10	12	21	24
14	17	28	30
16	20	30	35

Figure 2: Hungarian Method. Input Cost Matrix.

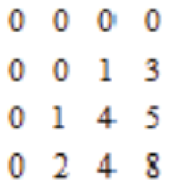
Hungarian Method - Step 1 From the cost matrix C, we first subtract the minimum of each row from each value in that row. In the example, the minimum of row 1 is 10, row 2 is 10, row 3 is 14, and row 4 is 16. So, we need to subtract the minimum of each row from each value in the respective row. In other words, subtract 10 from all values in row 1, 10 from all values in row 2, 14 from all values in row 3, and 16 from all values in row 4.



0	2	10	11
0	2	11	14
0	3	14	16
0	4	14	19

Figure 3: Hungarian Method. Step 1.

Hungarian Method - Step 2 We then repeat the same procedure as in step 1, but for each column in the resulting matrix from step 1. In the example, we subtract 0 from each value in column 1, 2 from each value in column 1, 10 from each value in column 3, and 11 from each value in column 4.



0	0	0	0
0	0	1	3
0	1	4	5
0	2	4	8

Figure 4: Hungarian Method. Step 2.

Hungarian Method - Step 3 The next step is to draw the minimum number of horizontal and vertical lines through each 0 in the resulting cost matrix. In the example, we draw the lines 1, 2, and 3 through each 0. Now, if the number of lines drawn is equal to the number of rows, then we go to step 5, otherwise, we continue to step 4. Step 3 and 4 repeat until the number of lines drawn is equal to the number of rows in the matrix.

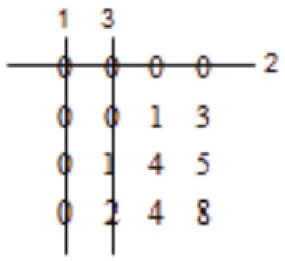


Figure 5: Hungarian Method. Step 3.

Hungarian Method - Step 4 Now, we have an uncovered section and an intersection section. The uncovered section consists of 1, 3, 5, 5, 4, and 8, while the intersection points are the first 2 0s in row 1. From here, we subtract the minimum of the uncovered section from all values in the uncovered section, and add that same minimum value to the intersection points. Then, we go back to step 3.

$$\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 1 & 3 & 4 \\ 0 & 2 & 3 & 7 \end{array}$$

Figure 6: Hungarian Method. Step 4.

Hungarian Method - Step 5 Below is the aggregation of one additional iteration of step 3 and step 4, then the assignment of rows to columns in step 5. The assignment by finding the row with the minimum number of 0s, then assigning that row to a column that shares a 0 value with that row. Then, that row and column are removed from the possible assignments and the process continues with selecting rows with a minimum number of 0s.

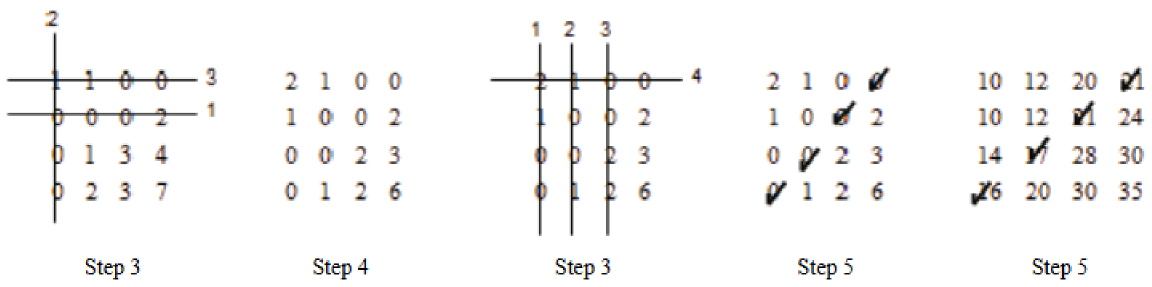


Figure 7: Hungarian Method. Continuation through to step 5.

3 Results

Watershed and Cell Tracking To evaluate the effectiveness of our approach, we tracked cells across frames from the 0th to the 50th frame with an interval of 10. The results of our segmentation and tracking pipeline are visualized in the provided images, which show the progression of cell dynamics over time. Each frame highlights segmented cells using the watershed algorithm, with unique colors representing individual cells. The trajectories generated through the Hungarian method accurately link cells across frames, even under challenging conditions like overlapping cells.

While every frame was processed to create a comprehensive GIF of cell movement, we present frames 0, 10, 20, 30, 40, and 50 to illustrate the results for the paper. These selected frames capture key transitions in cell behavior, movement, and division, showcasing the robustness of our pipeline in maintaining consistent cell identities across time points.

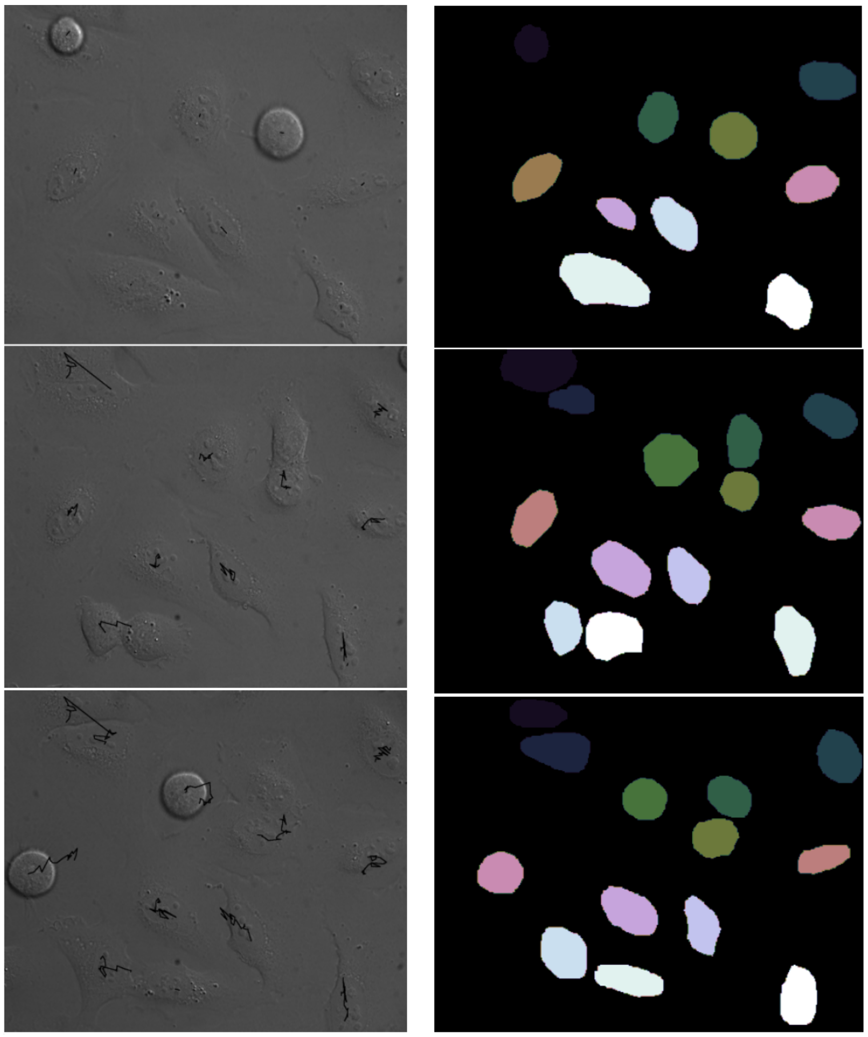


Figure 8: Cell tracks and respective watershed (0th, 10th and 20th frame)

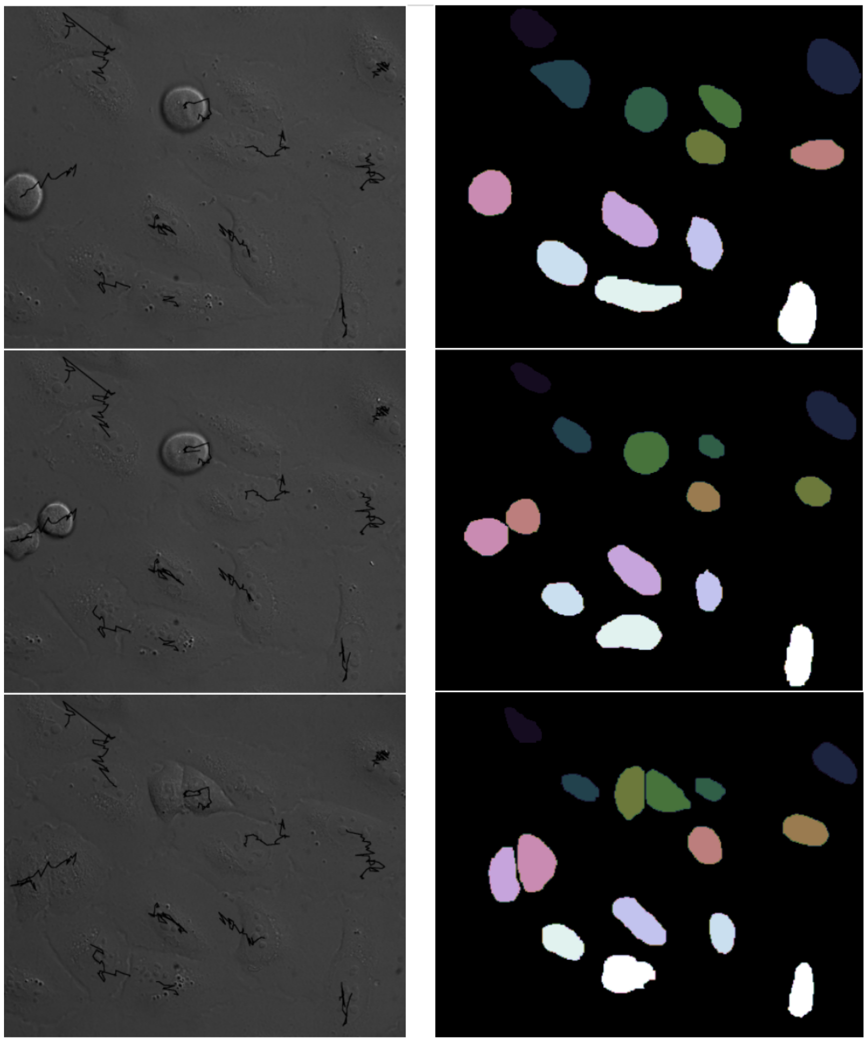


Figure 9: Cell tracks and respective watershed (30th, 40th and 50th frame)

4 Discussion

Overall, our group managed to successfully develop a robust cell tracking algorithm and applied it to images with live cellular dynamics. Our cell tracking algorithm can be divided into two main parts, image segmentation and cell tracking. In the first part, image segmentation, we deal with identifying cells from images. We approached this problem by applying the watershed method and managed to obtain markers. With these markers, we can set up the second part of our algorithm, cell tracking. We track cells by solving the linear assignment problem through the use of the Hungarian method and achieved robust tracks for motile cells. The watershed method proved to be particularly effective for handling challenges associated with low-contrast and noisy microscopy images, enabling us to delineate cell boundaries and distinguish overlapping cells. The Hungarian method further ensured precise tracking across frames by optimizing cell-to-cell associations, even under dynamic conditions. This combination of methods provided robust results, as evidenced by the accurate tracking trajectories presented in our results section. Despite these successes, some limitations remain. Our current approach does not explicitly account for biological events such as cell division or apoptosis, which can lead to interruptions in tracking or incorrect assignments. Addressing these phenomena will be a critical step in improving the algorithm's applicability to more complex cellular behaviors. Although our method performed well on the dataset at hand, future work will involve extending the approach to more complex datasets, such as biofilms or other environments where cells exhibit highly irregular and dynamic behaviors. Such data sets will likely require further refinement of both segmentation and tracking components to handle increased complexity.

5 Ties to Class

We felt that this was a good project to do for this class primarily because it involves a classic example of biological modeling in the biological imaging domain. On the segmentation side, watershed involves randomly sampling seeds from within a region of interest, and growing those regions outwards such that the regions have distinct boundaries where the intensity gradients are high. Additionally, cell tracking is a classic biological modeling problem where we try to understand cellular dynamics by understanding the trajectory that a cell takes, or how fast the cell is going. We model this as a graph optimization problem, where we take the minimum weighted bipartite matching between cells within consecutive frames.

References

[1] Kircher, M. F., Gambhir, S. S., & Grimm, J. (2011). Noninvasive cell-tracking methods. Nature reviews Clinical oncology, 8(11), 677-688.

[2] Ulman, V., Maška, M., Magnusson, K. E., Ronneberger, O., Haubold, C., Harder, N., ... & Ortiz-de-Solorzano, C. (2017). An objective comparison of cell-tracking algorithms. Nature methods, 14(12), 1141-1152.

[3] Maška, M., Ulman, V., Svoboda, D., Matula, P., Matula, P., Ederra, C., ... & Ortiz-de-Solorzano, C. (2014). A benchmark for comparison of cell tracking algorithms. Bioinformatics, 30(11), 1609-1617.

[4] Emami, N., Sedaei, Z., Ferdousi, R. (2021). Computerized cell tracking: Current methods, tools and challenges. Visual Informatics, 5(1), 1-13.

[5] Antonelli, L., Polverino, F., Albu, A., Hada, A., Asteriti, I. A., Degrassi, F., ... & Guarracino, M. R. (2023). ALFI: Cell cycle phenotype annotations of label-free time-lapse imaging data from cultured human cells. *Scientific Data, 10*(1), 677.

[6] Manfredi MG, Ecsedy JA, Chakravarty A, Silverman L, Zhang M, Hoar KM, Stroud SG, Chen W, Shinde V, Huck JJ, Wysong DR, Janowick DA, Hyer ML, Leroy PJ, Gershman RE, Silva MD, Germanos MS, Bolen JB, Claiborne CF, Sells TB. Characterization of Alisertib (MLN8237), an investigational small-molecule inhibitor of aurora A kinase using novel in vivo pharmacodynamic assays. Clin Cancer Res. 2011 Dec 15;17(24):7614-24. doi: 10.1158/1078-0432.CCR-11-1536. Epub 2011 Oct 20. PMID: 22016509.

[7] Vairalkar, M. K., & Nimbhorkar, S. U. (2012). Edge detection of images using Sobel operator. International Journal of Emerging Technology and Advanced Engineering, 2(1), 291-293.

[8] Canny, J. (1986). A Computational Approach To Edge Detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on. PAMI-8. 679 - 698. 10.1109/TPAMI.1986.4767851.

[9] Medina-Acosta, G.A., Delgado-Penín, J.A. On the feasibility of a channel-dependent scheduling for the SC-FDMA in 3GPP-LTE (mobile environment) based on a prioritized-bifacet Hungarian method. J Wireless Com Network 2011, 71 (2011). https://doi.org/10.1186/1687-1499-2011-71