2023, 1–16

Manuscript in preparation
General Research Profile

GENERAL RESEARCH PROFILE

Integrating Human Interaction For Improved Accuracy in Cell Tracking Methods

S.I. Pullens[1,2,*]

[1]Faculty of science - Utrecht University and [2]Computational Immunology group - Radboud University

[*]S.I.Pullens@students.uu.nl

Abstract

Tracking cell microscopy videos is a challenging and time-consuming task. We present our attempted improvements to optimize our user-friendly method to perform (semi) automated cell tracking. With this tool the user is able to visualize and correct tracks instantly when needed. Additionally, we also compare our method with free, most commonly utilized tracking software. Our results show that we are not only able to perform on par with the other methods, but can outperform these sophisticated methods with a simpler algorithm. Resulting in faster tracking and automatic updating based on human input. Furthermore, we will address the challenges that where encountered during our research.

Key words: Cell-tracking, Cell-segmentation, Automated tracking, Comparative-analysis, Cell-crowd dynamics, Immunology, Immune cells.

Introduction

Crowd dynamics in immune cells is an interesting research field, as numerous fundamental questions remain unanswered. The specific question that had sparked our interest is 'how does the immune system maintain rapid response times while constantly moving in dense packs and navigating through dense tissues?' While the inverse correlation between density and mobility holds true in numerous scenarios, such as in human crowds or animal herds [1], exceptions like the collective behavior of ants challenge our understanding of crowd dynamics. In the case of ants, an increased number of individuals does not appear to hinder their overall movement [2].

Moreover, advancements in imaging techniques have provided new possibilities to explore the complex immune cell interactions within densely packed tissues. As our knowledge of crowd dynamics in immune cells continues to expand, it may open doors to innovative strategies for enhancing immune responses.

Background

To find out how immune cells interact and move in the human body, We studied two types of specific immune cells - T cells and neutrophils. Most of this is done by microscopic imagery of immune cells. To be able to better visualize and thus understand the dynamics, the cells are placed into micro channels of varying widths (Table 1, Figure 2). This will be discussed in detail in the 'Data Description' section. With the use of these micro channels we are able to analyse the movement of the cells in one or two dimensions.

Tracking cells in the videos is needed for cell movement analysis and model generation. This can either be done manually or with the use of automated cell tracking software. Since manual tracking is very time consuming and labor intensive, automatic cell tracking is the more viable approach. Automated cell tracking has been a common challenge dating back to 1987 [3, 4]. Since then, multiple approaches have been suggested [5, 6, 7, 8, 9, 10, 11]. However, due to the difficulties that arise in automated cell tracking (specifically

Compiled on: September 14, 2023.
Manuscript prepared by the author.

high cell movement, cells moving in and out of frame/focus and high morphological properties, like deformation and cell division) it has proven rather difficult to perform automated cell tracking with perfect accuracy [12].

Tracking Methodology

Cell tracking implies the linking of detections (cells) over multiple frames in a video. Cell tracking consist of broadly two steps. First, the frames needs to be analysed, and scanned for detections. This can be a challenging task due to out of focus images, different shapes of desired detections, multiple object being inside the frame and the presence of artifacts (e.g. debris). The second step is linking the detection that represent the same object (cell) over multiple frames in a video. This can become remarkably complex due to frame gaps, missing detections, overlapping cells and detections moving in and out of frame.

State-of-the-art cell tracking programs are fairly complex, require numerous parameters, and perform poorly on our videos. The in-house algorithm, with less parameters, was developed to address the issues of the state-of-the-art algorithms. Here, we provide an overview of the in-house tracking algorithm, and the two well-known tracking tools Trackmate [13] and CellProfiler [14].

Tracking Software

In-house tracking method

During development of the in-house tracking method, two aspects quickly became apparent. The tracking should be easy to perform, and the user needs to be able to visualize the generated tracks (and correct them where necessary). With this in mind, the method was not only fast, but also easy to use by incorporating a single parameter; the maximum distance. With this, the algorithm specifically searches for the identical detection within the designated maximum distance window in the subsequent frame. A simple euclidean distance and Dijkstra algorithm [15, 16] performed surprisingly well compared to other highly sophisticated methods (Table 2). Because automated tracking is unable to generate perfect tracks, a hybrid approach was chosen. By combining automated track generation with human input to rectify errors in the created tracks, the idea was that the information from human input would automatically improve the creation of subsequent tracks.

The complete method consist of three phases. The first two phases are fully automated and needed to perform initial tracking. All phases can be found online in the repository 'Data availability'.

Phase 1) Videos are downloaded in .CZI format, and consist of two color channels; the nucleus and cytoplasm. This format is then converted to an RGB image, where the nucleus and cytoplasm are recolored according to their corresponding markers: nucleus red and cytoplasm green. This phase also consist of cell detection. This detection was performed using the pre-trained '2D-versatile-fluo' model from the stardist package [17, 18]. A drawback from this method is that is can also falsely identify debris in the videos as cells. We are currently working on an improved version that will be published later. However, since this method still produces feasible results, this improved version is not further discussed in this paper.

Phase 2) The graph-based, frame-to-frame tracking algorithm has been developed in python, and is heavily inspired by the Viterbi algorithm [19, 20]. First, we generate a temporal 3d graph (Figure 1), where 'X' and 'Y' (not shown) represent the resolution of the images and 'Z' represents the frame. Every graph starts with an activation layer, which only contains a single activation node 'A', and a likewise ending layer with ending node 'B'. All the intermediate layers define a single frame. Each detection node 'D' represents a single detection in the image, 'D' nodes can only be selected once. Each layer also contains nodes 'Y' and 'X', these represent cells that still need to come into frame (and thus have not been detected yet), and cells that went out of picture before the video was over. Since we cannot know for certain if a cell that left the frame is the same cell that enters in a later frame. Because of this it is considered a new cell.

After graph generation, the the Dijkstra algorithm is used to add one track at the time, representing a line of nodes and intermediate edges, to which a weight has been assigned. Our approach is an initially greedy approach. Implying that it does not compute the whole graph space but only looks at the current and previous frame to link nodes, making it relative fast. Since the edges to and from the 'Y' and 'X' nodes hold a bigger penalty than a new detection, but have a lower penalty score than a new detection outside of the maximum

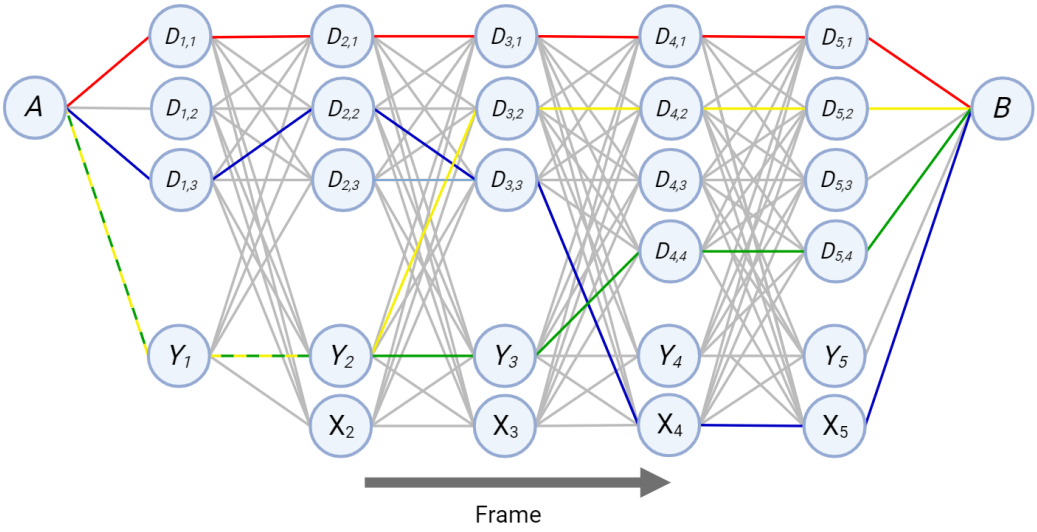

Figure 1. Visual representation of developed graph-based, frame-to-frame tracking algorithm. Every column (with exception of the 'A' and 'B' node) represent a single frame in the video. Every circle represents a node in the network; Activation node (A), Ending node (B), Detection node (D), Not yet detected node (Y) and left frame node (X). All edges (lines between nodes) hold a penalty score (can differ from other edges, not shown). All nodes can be selected multiple times, excluding the detection (D) nodes. Colored lines represent theoretical tracks; red track is identified, started from the first frame one and continues throughout the duration of the video. Blue track is detected in the first three frames and went out of picture in the latter two frames. Yellow track had no detection in the first two frames, but found detection in the latter frames. Green track had no detections in the first three frames, only the final two frames. Note that multiple tracks can go trough the (Y) and (X) nodes. Figure has been simplified for clarity; graph normally contains thousands of detection nodes per layer, and has more frame layers. Figure has been adapted from [20]-Figure 3b. Created with BioRender.com.

distance, it is believed the cell detection has halted. When a cell has been assigned to a track, it is saved and removed from the graph, decreasing the graph density. The Algorithm keeps running until all detection nodes 'D' have been assigned to a track.

Phase 3) The third (optional) phase involves human interaction. A web based environment has been developed to visualize and correct cell tracks (Supp. Figure 10). With this environment, the user cannot only visualize the tracks (Yellow circle for current frame and white lines for future frame) in the images and visualize detections (white circles). But can also manually add detections when they where missed or remove false positive detections, correct tracks by clicking on another detection, or remove specific connections, making it impossible for the Dijkstra algorithm to generate the same path trough the temporal 3D graph. One of the key capabilities of this phase is that the Dijkstra algorithm is running realtime in the background, generating new paths trough the graph whilst the user is making changes to the graph. Because of this, the user does not have to reconstruct the whole track when making a single change. When the user selects a different detection in the next frame than suggested, the algorithm locks in this connection (by setting the score to 1), resulting in a realtime recalculation of the lowest scoring path. This could mean that detections that where previously in this track will be reallocated to a new track. An tutorial can be found in the associated GitHub repository 'Data availability'.

Trackmate

Trackmate [13] is available as a plugin for the Fiji - ImageJ program [21]. ImageJ is a java-based image processing program developed at the National Institutes of Health and the Laboratory for Optical and Computational Instrumentation (LOCI, University of Wisconsin) [22]. With this, it gains the advantage of utilizing the integrated graphical user interface (GUI), provided by the ImageJ program. The major pitfall of ImageJ is the non-programmability (excluding macro's), meaning that it cannot be implemented into an automatic pipeline.

CellProfiler

CellProfiler [14] is a stand-alone python program that originated from MATLAB [23]. The program is currently maintained by the Cimini [24] Lab at the Imaging Platform of the Broad Institute. CellProfiler is written in a module-based manner, making it easy for users to developed their own cell-microscopy pipelines, and export these to a multitude of different output formats. Even though it is possible to execute the program trough the command-line, it remains difficult to implement CellProfiler into personal algorithms, due to the high amount of mandatory parameters, making it difficult to automate. Used versions and parameters for both programs can be found in the associated GitHub repository 'Data availability'.

Table 1. List of all used videos for comparison analysis, including their meta-data.

Video ID	Number of Frames	Number of Tracks	Number of Collisions
neuts-6um-1	271	526	469
neuts-6um-2	271	1777	31867
tcells-6um-1	181	73	598
tcells-15um-1	271	139	4402
tcells-30um-1	271	1115	21003

All video's have a resolution of 1376 by 1104 pixels and a temporal resolution of 20 seconds between each frame. All videos where generated by the Mandl JN lab. - McGill University and can be found at https://computational-immunology.org/hfsp/

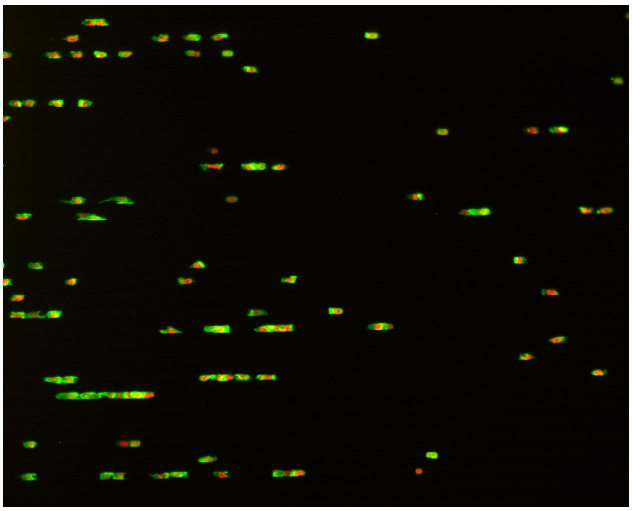

Figure 2. Frame 157 of the Neutrophils-6μm-1 video. Nuclei are dyed using TdTomato and f-actin was dyed using enhanced green fluorescent protein (EGFP). Cells are trapped in 6μm stacked channels, simulating one dimensional movement (left and right). Frames from other movies can be found in Supp. Figure 1.

Data Description

The data that was used to compare the cell tracking programs consist of five videos (Table 1), with their corresponding cell positions file, tracks generated from the different algorithms and a manually tracked datafile. The latter is considered the ground truth. All videos contain either activated mice T cells or neutrophils cells. To be able to distinguish the cell nucleus from its cytoplasm (and thus being able to record its dynamics), two dyes were used. TdTomato was used for the nucleus and f-actin was dyed using enhanced green fluorescent protein (EGFP) (Figure 2, Supp. Figure 1). These videos where selected to cover all different properties of the cells.

All videos have a resolution of 1376 x 1104 pixels, with an interval of 20 seconds between each frame. Three of the five videos contain activated, mice T cells moving in stacked 6μm, 15μm and 30μm lanes. These distances where selected to simulate 1-dimensional, limited 2-dimensional and unrestricted 2-dimensional cell dynamics respectively. The other two videos contain activated mice neutrophils cells, moving in stacked 6μm lanes. The second video was added to the comparison list, due to a rupture in three of the lanes, making the cells being able to move between lanes, resulting in a high number of cell collisions (Supp. Figure 2). The reason this was added to the comparison was to see if a unexpected change in movement would effect the cell-tracking algorithm results. to make sure no directional bias would occur, the cells were seeded on both sides of the lanes.

In this study, we will provide an overview of the existing algorithm, suggest an improvement (tracking with intensity), and discuss the extra features that will improve the user's experience, compared to the other software. Furthermore, we will provide a comparison between the in-house method and the previously mentioned tracking software.

Analyses

The primary objective of this research was to track the introduced cell videos, enabling us to perform analysis on the movement of the immune cells. Even though there's a lot of literature and developed methods about cell tracking [5, 6, 7, 8, 9, 10, 11], no alternative method generated tracks of a sufficient quality for the required analysis. Because of this, an in-house methods was developed, which is mainly focused to track the cells from the provided videos (Supp. Figure 1). In this paper, we will discuss this new method and

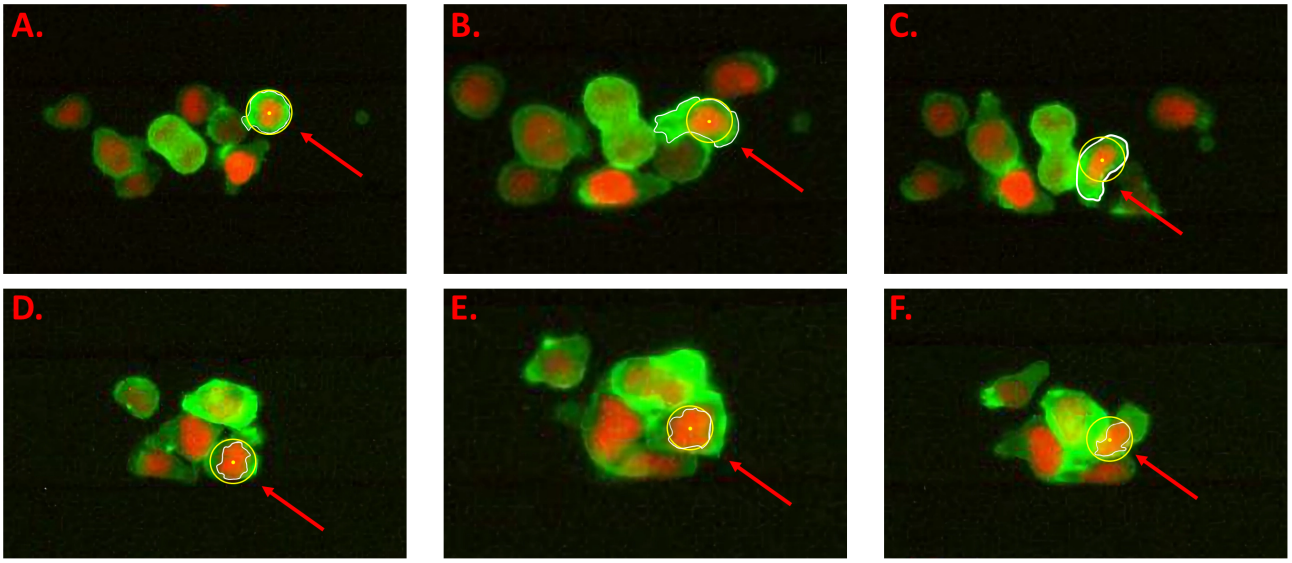

Figure 3. Frames from the Tcells-30μm-1 video. Indicated cells in the top (A-C) and bottom (D-F) row are the same cell, over different frames in the video. The highlighted cell in the top row (A-C) shows high cell membrane deformation, resulting in high fluctuations in the cell membrane intensity. The highlighted cell in the bottom row (D-E) shows high nucleus deformation, resulting in high fluctuations in nucleus intensity. A) cells from frame 181. B) cells from frame 186. C) cells from frame 189. D) cells from frame 189. E) cells from frame 188. F) cells from frame 187.

show a comparative analysis between our method, and the two most commonly used leading-edge, free tracking programs: Trackmate and CellProfiler. A brief explanation of cell tracking involves two primary elements: cell detection, detecting the cells in the image and cell tracking, linking the (correct) cells that have been detected between different frames.

With our initial testing, it became apparent that using only the euclidean distance to link cells in different frames performed unsatisfactory on cell clusters. These densely packed cells have relative similar distances from the initial cell, which was causing the generated track to 'jump' between cells (Supp. Figure 3). To improve this, we tried not only using the distance information from the cells, but also utilize the color information of the cells. This was done by also saving the average color intensities from the nucleus and the cell membrane, over a specified area centering the cell position (Figure 3) ('Methods' section.). After intensive testing we became aware that this method would not prove durable, mainly because the high morphological rates of both the cell membrane (Figure 3 A-C) and nuclei (Figure 3 D-F). This ended up causing more harm than good, resulting in the method being sidelined. This is discussed in detail in the 'Discussion' section. For the comparative analysis we tested the two well-known tools for automated cell tracking - (Trackmate [13] and CellProfiler [14]).

Since we are only interested in the tracking performance of the different programs and not their segmentation (image detection) capabilities, we imported masked videos instead of the real videos. (Supp. Figure 5). This was done to ensure that differences in segmentation performance would not affect the cell tracking performance. A mask image is an image that represents another image, where points of interest in the original image are represented with an non-zero value (most common values are 1 and 255). Whilst the non interesting points are represented with pixel value 0. With this method, we can easily represent points of interest (in our case cells) from the original image.

Results

In our analysis, we generated the output of these programs (including our own method), and compared these with our manually tracked outputs (which are considered the ground truth). Next, these results are quantified using the well established Cell Tracking Benchmark (CTB) [25] measurements (see 'Methods' section). Our main findings where that, even though all cell-tracking programs scored high on the technical (DET & TRA) measurements, CellProfiler and Trackmate performed poorly on the biological measurements. Especially compared to our in-house methods, which have not been manually corrected yet (Table 2). Even though our method has considerable room for improvement, it did outperform the other methods. We observed that all tracking algorithms performed worse on videos with wider lanes. This can be explained by the added complexity that wider lanes produce. This complexity would mainly consist of an extra dimension for the cells to move in, and the added deformation rate of cells trying to squeeze past each other.

CellProfiler performed considerably worse on the detection of cells in the Neutrophils-6μm-2 video compared to the others. This is a particularly hard video to track, since three lanes suffered a rupture, resulting in cells flowing between different lanes (Supp. Figure 2). Even though all methods where only provided with the masks (meaning the added segmentation complexity can be discarded), it appears CellProfiler has difficulties with cells that move unanticipated.

We can see that Trackmate seems to have a worse DET and TRA score compared to CellProfiler, even though all programs received the same masks as input (Supp. Figure 5). Trackmate seemed to perform worse overall on the detection (DET: 0.993-0.801, average 0.919), compared to CellProfiler (0.999-0.983, average 0.991). Whilst performing notably better on the biological measurements (TRA: 0.992-0.801, average 0.907, CT: 0.508-0.038, average 0.194, TF: 0.898-0.521, average 0.687) than CellProfiler: (TRA: 0.967-0.926, average 0.949, CT: 0.043-0.000, average 0.018, TF: 0.601-0.223, average 0.416). Additionally, we observed that our method (on average) outperformed all other methods (DET: .993-0.928, average 0.971, TRA: 0.992-0.925, average 0.966, CT: 0.342-0.049, average 0.190, TF: 0.929-0.668, average 0.765). Showing that a simple method cannot only compete, but outperform other complex

Table 2. Tracking performance results of the different algorithms based on different videos. Measurements are adapted from the Cell Tracking Benchmark; DET: Detection accuracy, TRA: Tracking performance, CT: Complete Tracks, TF: Track Fractions. Number of required, tunable parameters (NP) is stated below the name of the software. Measurements are normalized between [0, 1] interval, higher is better.

Software	Measurements	T-cells 6μm	T-cells 15μm	T-cells 30μm	Neutrophils 6μm 1	Neutrophils 6μm 2	Average
In-house	DET	0.992	0.972	0.928	0.993	0.971	0.971
(NP: 1)	TRA	0.992	0.968	0.925	0.989	0.957	0.966
	CT	0.342	0.049	0.161	0.281	0.119	0.190
	TF	0.929	0.668	0.729	0.816	0.682	0.765
Trackmate	DET	0.993	0.911	0.905	0.957	0.827	0.919
(NP: 7)	TRA	0.992	0.899	0.894	0.950	0.801	0.907
	CT	0.508	0.038	0.176	0.194	0.053	0.194
	TF	0.898	0.521	0.688	0.763	0.563	0.687
CellProfiler	DET	0.999	0.993	0.985	0.996	0.983	0.991
(NP: 61)	TRA	0.963	0.953	0.934	0.967	0.926	0.949
	CT	0.000	0.004	0.020	0.043	0.025	0.018
	TF	0.223	0.294	0.453	0.601	0.509	0.416

Values in table are rounded to three decimal places (0.001 precision).

methods for cell tracking.

Because we expected the detection metric to be a perfect score (due to providing the methods with a perfect mask of the detections), we where quite surprised to find out the scores where close, but not totally perfect. When looking into this further, we performed an intersection of union comparison [26] (Supp. Table 1), which revealed the near perfect scores weren't caused by sub-pixel rounding errors, but where being caused by the algorithm misinterpreting the detection masks, an example can be found in (Supp. Figure 4). Due to small location errors, the detection positions where not identical to the ground truth, resulting in the detection metric being near perfect.

Since most of the biological scores were surprisingly low, we investigated this further. The complete Tracks (CT) and track fractions (TF) metrics only consider tracking correctness, regardless of the accuracy of the segmentation (See 'Methods', Equation 3&4) [25, 27], Because the CT and TF measurements are directly influenced by the number of tracks and the ground truth, we examined the number of tracks compared to the track length (Figure 4). With this new information, it became clear why the CT and TF measurements performed so poorly. CellProfiler produced consistently 8 to 28 times more tracks compared to the ground truth (whilst Trackmate and the in-house method produced 1.4 to 5.7 and 2.4 to 7.1 more respectively). As mentioned in [27], tracks with a low CT and TF score still contain errors that need to be corrected in order to enable realistic biological conclusions.

We also briefly inspected the run time of the different methods. Whilst Trackmate and our in-house method where finished within a couple of seconds, CellProfiler could take up 15 to 30 minutes for each video. (all methods where ran on a 11th Gen Intel(R) Core(TM) i5-1135G7 laptop). In order to comment on the software's usability complexity, we counted the number of required, tunable parameters (NP), adapted from [27]. Our method uses a single parameter: the maximum distance. Trackmate requires a minimum of 7 parameters, whilst CellProfiler requires the most parameters by far: 61 parameters. Note that input and output file locations have not been taken into account for this measurement.

Discussion

In this paper we presented our attempted improvements and an in-house cell tracking method, which is able to perform fast tracking, and embracing human interaction by developing a clear interface for the user, making it easy to manually see and correct the tracks where necessary. We also showed that we can outperform more complex, state-of-the-art cell tracking software [13, 14] by only using a simple Dijkstra algorithm [15, 16] on the provided movies.

It became abundantly clear, that an automated method with improved tracking accuracy needed to be developed. Primarily because manually tracking the tracking output could take over 50 hours to complete for a video. The main attempted improvement, was the implementation of the color intensity information into the scoring function. After testing however, it became apparent that the outcomes were not sufficient. The main reason for this was the overlapping pixel area that was being used to find the average color values for the nucleus and cell membrane in the cells (Figure 3). To tackle this problem, we initially tried narrowing the scanned area. This however, resulted in even worse results due to cells being cutoff. This can be explained by the difference in cell sizes and shapes (Figure 3). We also attempted to look at nuclei colors exclusively. However, not all frames from a specific cell have the same intense nuclei staining, resulting in skewed performances.

Even though the initial idea looked promising, we soon found out that due to the high morphological properties and different cell sizes, interfering with other scan-areas was inevitable. Especially since the initial reason for this added logic was to improve tracking in cell clusters (Supp. Figure 3). This problem would only be fixed if there would be pixel perfect masks available, showing the perfect outline of the cells. With this information however, cell tracking in clusters would be easier to perform, making the added intensity logic obsolete. Even without this implemented logic, we are still able to perform on par with the other cell tracking software.

The reason we didn't pursue this improvement further, is because multiple improvements where being developed at the time. During this time, we also started experimenting with fully convolutional neural networks, which showed promising results. This ultimately led to us not pursuing the color intensity check further. These added features however are still in active development, and have not been used to generate the listed tracks. This resulted in these improvements being out of scope for this analysis.

Though our methods demonstrate a considerable advantage, it is essential to consider that our method was exclusively developed using the provided data, whilst Trackmate and CellProfiler are developed to use on a plethora of different imagery datasets. At the

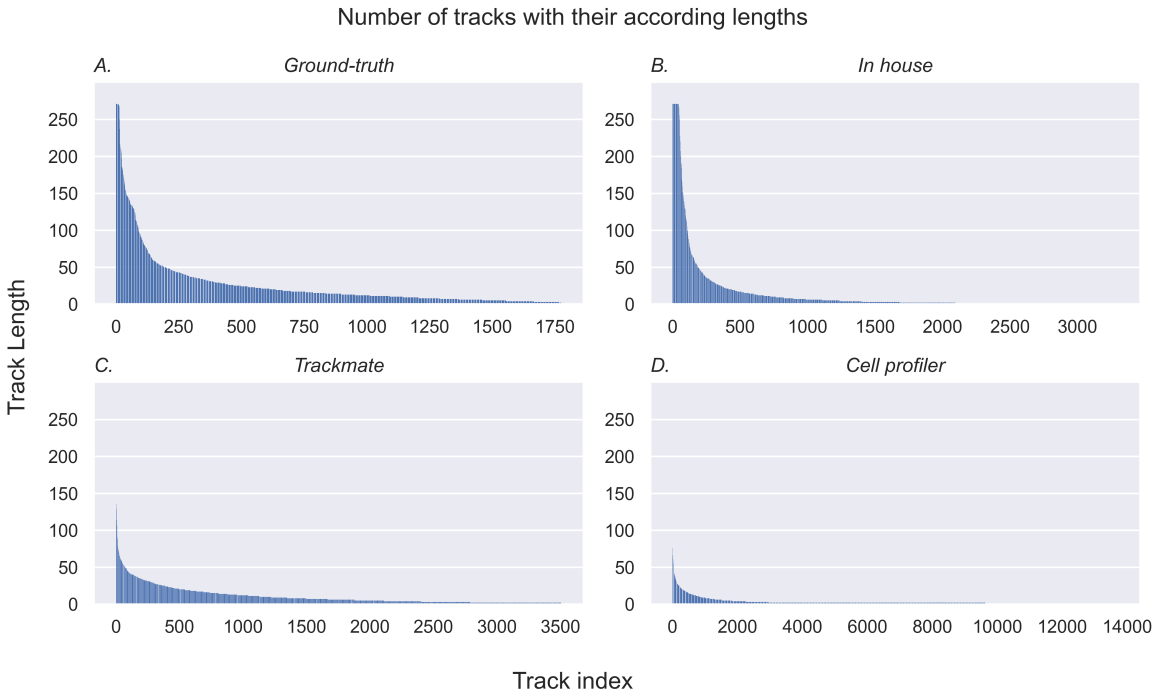

Figure 4. Number of tracks with their according track lengths, generated from the Neutrophils-6μm-2 masked video. A) Ground truth. B) in-house method. C) Trackmate method. D) CellProfiler method. Track lengths from other videos can be found in Supp. Figure 6-9. Note: x-axis from CellProfiler appears empty, but actually holds high amount of small length tracks (< 5 frames).

time of writing this paper, we have not tested our methods on other benchmark videos. There is a likelihood of subpar performance on different videos, since currently, all high performing tracking algorithms demonstrate poor generalizability [25].

The paper mainly compares the first stage (without human interaction) of our method with the other software, whilst using the manually corrected tracks as a ground truth. Of course, for further analysis with benchmark videos, the provided ground truth needs to be used. The main goal of the software was not centered on achieving a flawless, 100% accurate tracking algorithm. Instead, it aimed to establish a user-friendly platform where users could refine tracks for further analysis.

Our future plans mainly consist of improving our initial, phase 1 tracking, due to it directly resulting in smaller correction workload for the user. Currently, we are working on a neural network that will be trained on the human cleaned data. This is however, still in active development and out of scope for this comparative analysis.

Methods

Only the initial phase tracking results have been used for comparative analysis. This was done because manually corrected tracks where taken as a ground truth for the Cell Tracking Benchmark measurements. Comparing only phase one results still shows viable results, compared to the other programs (Table 2).

Implementing Color Intensity

To improve the tracking accuracy (especially in clustered cell groups, like Supp. figure 3) in our method, we tested the use of adding color intensity measurements to the tracking algorithm. This was accomplished not only by saving the cell locations in the detection nodes during graph generation (Figure 1), but also by assessing the color intensities of nuclei and cell membranes for each cell. We tested multiple configurations in the cost function. Evenly weighted scores for both the distance and color had little effect; tracks didn't differ compared to only using the euclidean distance. We hypothesize this to be caused by the overlapping cells in combination with high morphological changes the cells endure (Figure 3). Upon raising the weight of the euclidean distance (proportional to the color intensity scores) we observed no change compared to only using distance metrics. We also experimented with various states where the color intensity scoring function weighted higher in proportion to the distance metric. In low proportions, the tracks showed no direct improvement in comparison with the distance only - tracks. A rise in the weight resulted in diminished results: tracks started to neglect the positions of the cells and jump between different lanes (not shown) which is practically impossible. For this reason, we chose to exclusively incorporate Euclidean distance as a metric within our scoring function.

Comparative Analysis Method

Our comparative analysis consist of tracking five different videos (Supp. Figure 1) with our in-house method, Trackmate [13] and CellProfiler [14]. Since the videos have already been tracked manually, it is clear that the cells do not experience any cell division. With this in mind, the LAP detection algorithm [28] was used for both Trackmate and CellProfiler, due to it not considering mitosis or meiosis. To compare the cell tracking results with other methods, we primarily used the widely-adopted benchmark measurements from the Cell Tracking Benchmark (CTB) community [25]. Because CTB processes the tracks in a different format, conversions needed to be performed. These format conversions mainly consisted of converting .csv track files, to a combination of .txt and .tiff files. Scripts for these conversions can be found in the associated GitHub repository 'Data availability'.

Below we list the measurement formulas used by the CTB, adapted from [25, 27]. These comparison measurements are performed by measuring how much the generated tracks differ from the ground truth, and how much the cost will be to convert the generated track to the ground truth, compared to the cost of restarting from the beginning. All measurements are normalized between

a [0, 1] interval, meaning that a score of 0 means that restarting from the beginning would have a lower cost than correcting the generated track. A score of one means that no cost is needed due to the track already being identical to the ground truth.

The detection accuracy (DET) metric (Equation 1) is derived from object identification, and is computed by the cost it would take to achieve the same detection as the ground truth (GT).

$$DET = 1 - \frac{min(AOGM - D, AOGM - D_0)}{AOGM - D_0} \tag{1}$$

where:

DET = Detection accuracy.
$AOGM - D$ = Cost of transforming a set of nodes provided by the participant, into a set of GT nodes.
$AOGM - D_0$ = Cost of creating the set of GT nodes from scratch.

The tracking accuracy (TRA) metric (Equation 2), is derived from how accurately each detection has been correctly detected in the current and sequential frames. The acyclic Oriented Graph Matching (AOGM) measure [29] is the normalized measurement between the generated Acyclic Oriented Graph (AOG) and the AOG from the ground truth graph.

$$TRA = \frac{min(AOGM, AOGM_0)}{AOGM_0} \tag{2}$$

where:

TRA = Tracking accuracy.
$AOGM$ = Normalized version of the Acyclic Oriented Graph Matching measure.
$AOGM_0$ = AOGM value required for recreating the reference graph from scratch.

The complete tracks (CT) metric (Equation 3), is derived from the F1 score of the cost to reconstruct the ground truth tracks with the candidate tracks.

$$CT = \frac{2T_{rc}}{T_c + T_{gt}} \tag{3}$$

where:

T_{rc} = Number of completely reconstructed ground truth tracks.
T_{gt} = Number of all ground truth tracks.
T_c = Number of all computed tracks.

The track fractions (TF) metric (Equation 4), is based on finding the longest, uninterrupted, correct fraction of a detected reference track, and averaging these over all detected reference tracks.

$$TF = \frac{1}{n}\sum_{i=1}^{n} \mathrm{TF}_i \tag{4}$$

where:

TF = Longest, correctly reconstructed, continuous fraction of a detected reference track.
n = Number of detected reference tracks in the video.
TF_i = Track fraction for the i-th detected reference track.

As mentioned before, we wanted to look at the differences in generated track lengths (Figure 4). This has been performed on all candidate track videos (Sup. Figure 6-9). Even though the difference fluctuates between multiple videos, it still shows a trend where CellProfiler makes a big amount of small length tracks, and Trackmate -even though making longer tracks than CellProfiler- still has smaller tracks than our simplified in-house method.

Furthermore, we also made a comparison between the number of required tunable parameters (NP), which has been adopted from [27]. For this measurement, we went over all steps in all three tracking methods, and counted the number of required parameters that where needed for results. Input- and output file related parameters (e.g. file locations) have not been taken into account.

Availability of source code and requirements

- Project name: Cell tracking comparison
- Project home page: https://github.com/Sjonnie404/Cell-tracking-comparison
- Operating system(s): Platform independent
- Programming language: Python, Jupyter Notebook, Vue, Flask, Javascript, HTML
- Used versions: Python-3.9[1], Java-8.0, HTML-5.0, Vue-3.2.37, Fiji-1.54f, Trackmate-v.7.11.1, CellProfiler-4.2.5, Cell-Tracking-Challenge plugin-1.0
- License: GPLv3+

Data Availability

The data sets supporting the results of this article are available in the Cell tracking comparison repository, https://github.com/Sjonnie404/Cell-tracking-comparison.

Declarations

Contributions

The videos that have been used in this research are generated by the Mandl Lab[2]. In-house tracking algorithm has been developed by Mihaela Mihaylova[3] and Johannes Textor[3]. Intersection over union comparison script has been developed by Mihaela Mihaylova[3]. Initial collision detection script has been made by Inge Wortel[3]. The initial convertion scripts to covert the data to the cell tracking challenge benchmark format has been developed by Lukas Wiebke[4]. The Comparison analysis and manually correcting tracking videos has been done by Shane Pullens[3,5].

List of abbreviations

- AOG, Acyclic Oriented Graph
- AOGM, Acyclic Oriented Graph Matching
- CP, CellProfiler
- CT, Complete tracks (metric)
- CTB, Cell Tracking Benchmark
- DET, Detection accuracy (metric)
- EGFP, Enhanced Green Fluorescent Protein
- GT, Ground truth
- GUI, Graphical user interface
- IH, In-House
- NP, Number of required, tunable parameters
- TF, Track fraction (metric)
- TM, Trackmate
- TRA, Tracking accuracy (metric)

Consent for publication

Not applicable.

1 Used package versions can be found in the 'requirements.txt' file in the GitHub repository.
2 Faculty of Medicine - McGill University, Montreal, Canada.
3 Computational Immunology group at the data science group at the Institute for Computing and Information Sciences, Radboud University, Nijmegen, The Netherlands.
4 Institure for Tecnology, Buenos Aires ITBA, CONICET, Buenos Aires, Argentina.
5 Faculty of science - Utrecht University.

Competing Interests

The author(s) declare that they have no competing interests.

Acknowledgements

I would like to acknowledge Johannes Textor for the guidance and supervision for this project, Mihaela Mihaylova for the discussions, leading to better analysis and results for this project, Lucas Wiebke for helping with the CTB conversion scripts and Lukas Kapitein for the external supervision.

Authors' Information

This paper was written by S.I. Pullens, a student who is currently working on finishing his Bioinformatics & Biocomplexity Master's at Utrecht University, who did this research in collaboration with the Computational Biology group at Radboud University.

References

1. Fang Z, Lo SM, Lu JA. On the relationship between crowd density and movement velocity. Fire Safety Journal 2003;38(3):271–283.
2. John A, Schadschneider A, Chowdhury D, Nishinari K. Trafficlike collective movement of ants on trails: Absence of a jammed phase. Physical Review Letters 2009;102(10):1–4.
3. Midtvedt B, Helgadottir S, Argun A, Pineda J, Midtvedt D, Volpe G. Quantitative digital microscopy with deep learning. Applied Physics Reviews 2021;8(1).
4. Geerts H, De Brabander M, Nuydens R, Geuens S, Moeremans M, De Mey J, et al. Nanovid tracking: a new automatic method for the study of mobility in living cells based on colloidal gold and video microscopy. Biophysical Journal 1987;52(5):775–782.
5. Loffler K, Mikut R. EmbedTrack - Simultaneous Cell Segmentation and Tracking Through Learning Offsets and Clustering Bandwidths. IEEE Access 2022;10:77147–77157.
6. Ben-Haim T, Raviv TR. Graph Neural Network for Cell Tracking in Microscopy Videos. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 2022;13681 LNCS:610–626.
7. Lefebvre AEYT, Ma D, Kessenbrock K, Lawson DA, Digman MA. Automated segmentation and tracking of mitochondria in live-cell time-lapse images. Nature Methods 2021;18(9):1091–1102.
8. Boukari F, Makrogiannis S. Automated Cell Tracking Using Motion Prediction-Based Matching and Event Handling. IEEE/ACM Transactions on Computational Biology and Bioinformatics 2020;17(3):959–971.
9. Nishimoto S, Tokuoka Y, Yamada TG, Hiroi NF, Funahashi A. Predicting the future direction of cell movement with convolutional neural networks. PLoS ONE 2019;14(9):1–14.
10. He T, Mao H, Guo J, Yi Z. Cell tracking using deep neural networks with multi-task learning. Image and Vision Computing 2017;60:142–153.
11. Lou X, Schiegg M, Hamprecht FA. Active Structured Learning for Cell Tracking. Methods 2014;33(4):849–860.
12. Maška M, Ulman V, Svoboda D, Matula P, Matula P, Ederra C, et al. A benchmark for comparison of cell tracking algorithms. Bioinformatics 2014;30(11):1609–1617.
13. Tinevez JY, Perry N, Schindelin J, Hoopes GM, Reynolds GD, Laplantine E, et al. TrackMate: An open and extensible platform for single-particle tracking. Methods 2017;115(2017):80–90. http://dx.doi.org/10.1016/j.ymeth.2016.09.016.
14. Stirling DR, Swain-Bowden MJ, Lucas AM, Carpenter AE, Cimini BA, Goodman A. CellProfiler 4: improvements in speed,

utility and usability. BMC Bioinformatics 2021;22(1):1–11. https://doi.org/10.1186/s12859-021-04344-9.
15. Dijkstra EW. Dijkstra. Numer Math 1959;271:269–271.
16. Sniedovich M. Dijkstra's algorithm revisited: The dynamic programming connexion. Control and Cybernetics 2006;35(3):599–620.
17. Schmidt U, Weigert M, Broaddus C, Myers G. Cell detection with star-convex polygons. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 2018;11071 LNCS:265–273.
18. Weigert M, Schmidt U, Haase R, Sugawara K, Myers G. Star-convex polyhedra for 3D object detection and segmentation in microscopy. Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020 2020;p. 3655–3662.
19. Viterbi A. Viterbi algorithm. Scholarpedia 1973;4(1):6246.
20. Magnusson KEG, Jalden J, Gilbert PM, Blau HM. Global linking of cell tracks using the viterbi algorithm. IEEE Transactions on Medical Imaging 2015;34(4):911–929.
21. Schindelin J, Arganda-Carreras I, Frise E, Kaynig V, Longair M, Pietzsch T, et al. Fiji: An open-source platform for biological-image analysis. Nature Methods 2012;9(7):676–682.
22. Schneider CA, Rasband WS, Eliceiri KW. NIH Image to ImageJ: 25 years of Image Analysis HHS Public Access. Nat Methods 2012;9(7):671–675.
23. Inc TM, Statistics and machine learning toolbox documentation. Natick, Massachusetts, United States: The MathWorks Inc.; 2022. https://www.mathworks.com/help/stats/index.html.
24. Cimini Lab;. Accessed: 2023-03-08. https://cimini-lab.broadinstitute.org/.
25. Maška M, Ulman V, Delgado-Rodriguez P, Gomé z-de Mariscal E, Peña FAG, Ren TI, et al. The Cell Tracking Challenge: 10 years of objective benchmarking [under review]. Nature Methods 2022;20(July).
26. Rezatofighi H, Tsoi N, Gwak J, Sadeghian A, Reid I, Savarese S. Generalized intersection over union: A metric and a loss for bounding box regression. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2019;2019-June:658–666.
27. Ulman V, Maška M, Magnusson KEG, Ronneberger O, Haubold C, Harder N, et al. An objective comparison of cell-tracking algorithms. Nature Methods 2017;14(12):1141–1152. http://dx.doi.org/10.1038/nmeth.4473.
28. Jaqaman K, Loerke D, Mettlen M, Kuwata H, Grinstein S, Schmid SL, et al. Robust single-particle tracking in live-cell time-lapse sequences. Nature Methods 2008;5(8):695–702.
29. Matula P, Maska M, Sorokin DV, Matula P, Ortiz-De-Solórzano C, Kozubek M. Cell tracking accuracy measurement based on comparison of acyclic oriented graphs. PLoS ONE 2015;10(12).

Layman Summary

The goal of this study was to understand and improve the in-house automated cell tracking method that has been developed by the Computational Immunology group (Radboud University). Besides this, we also evaluated the performance of this method, by comparing it to other, known tracking software (CellProfiler and Trackmate).
Tracking means to follow a specific object or detection over time. In our case, these objects are cells in a microscopy video. Because there are so many cells to track, it is very hard (and takes a lot of time) to track these cells manually. This is complicated even further because the cells move, deform and move in- and out of frame of the picture. To decrease the time that was spent on manually tracking the cells, an automated tracking method was developed.
A new method needed to be developed, because the previous mentioned software didn't produce good enough tracks. The tracks that where given by the software contained so many mistakes, that manually tracking the cells was faster to do than correcting all the errors.
The main reason to track these cells is to be able to perform analysis on the movement of cells, which can give new insights. Since our group is especially interested in the immune system, Neutrophil cells and T cells where analysed. These cells where put into stacked lanes of various widths: 6μm, 15μm and 30μm. These widths where selected to mimic 1-dimensional, limited 2-dimensional and unrestricted 2-dimensional cell movement respectively.
The developed tracking method only makes use of the distances between detections to track the cell. For every cell, it looks ahead one frame, and links the cell from the next frame that is closest to the cell in the current frame. It repeats this for every frame and every cell, resulting in tracks. Even though this relative method makes some mistakes, it still seems to outperform the other software (based on our provided videos).
Because this method was not perfect, we tried improving it by incorporating the color information of the cells. This unfortunately did not increase the quality of the tracks like we hoped for, in some cases it performed even worse. Because other improvement methods were also being developed and showed better results, we stopped implementing the color intensity.
To measure the performance of the in-house tracking method (and compare it with the others), we compared features of the generated tracks, and computed a penalty score based on how far it was off compared to the ground truth (these are the tracks that have been corrected by hand).
Even though the in-house method scores where average, they did outperform the other methods. This shows that even a simple tracking method (using only a single parameter) can outperform highly sophisticated tracking methods on specific videos.
Future goals of this research will be to improve the tracking accuracy of the in-house method. This is important because when the generated tracks are made better, less correction needs to be done, which speeds up the process.

Supplementary data

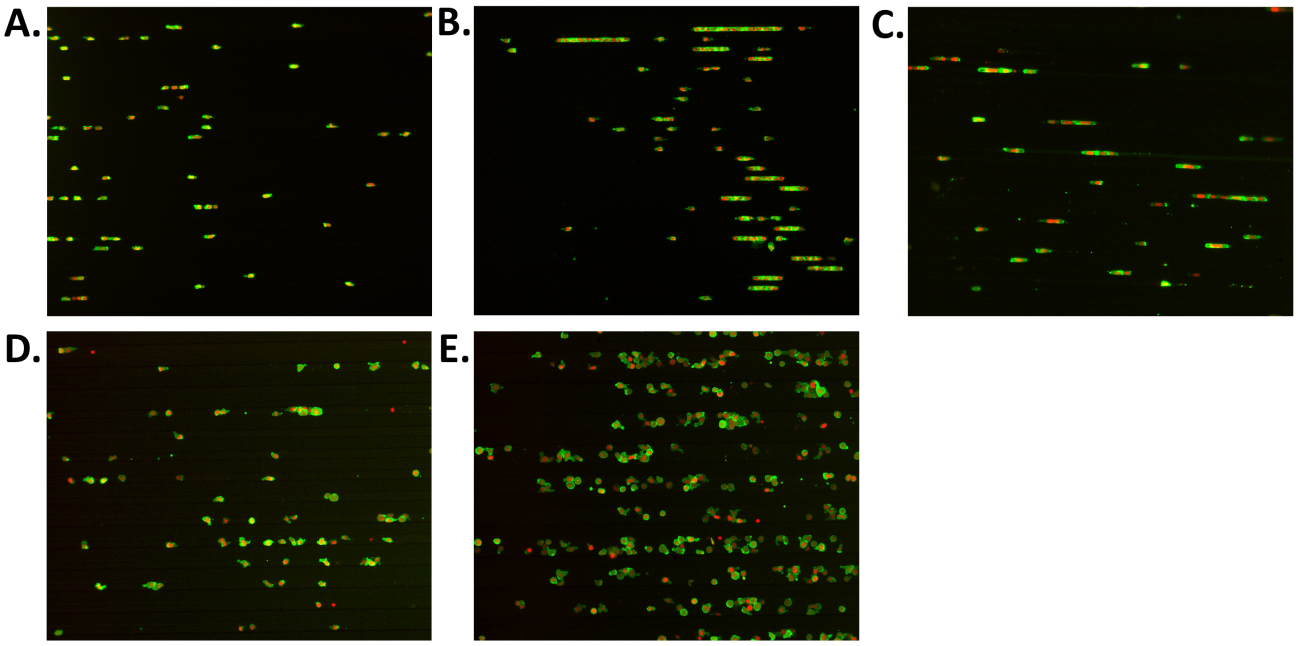

Figure S1. First frame of every analysed video. Videos contain microscopic imagery from activated Neutrophils and T-cells that are placed in stacked 6- , 15- and 30 μm lanes. Videos can be downloaded from https://computational-immunology.org/hfsp/. A) frame 0 from Neuts-6μm-1 video. B) frame 0 from Neuts-6μm-2 video. C) frame 0 from Tcells-6μm-1 video. D) frame 0 from Neuts-15μm-1 video. E) frame 0 from Neuts-30μm-1 video.

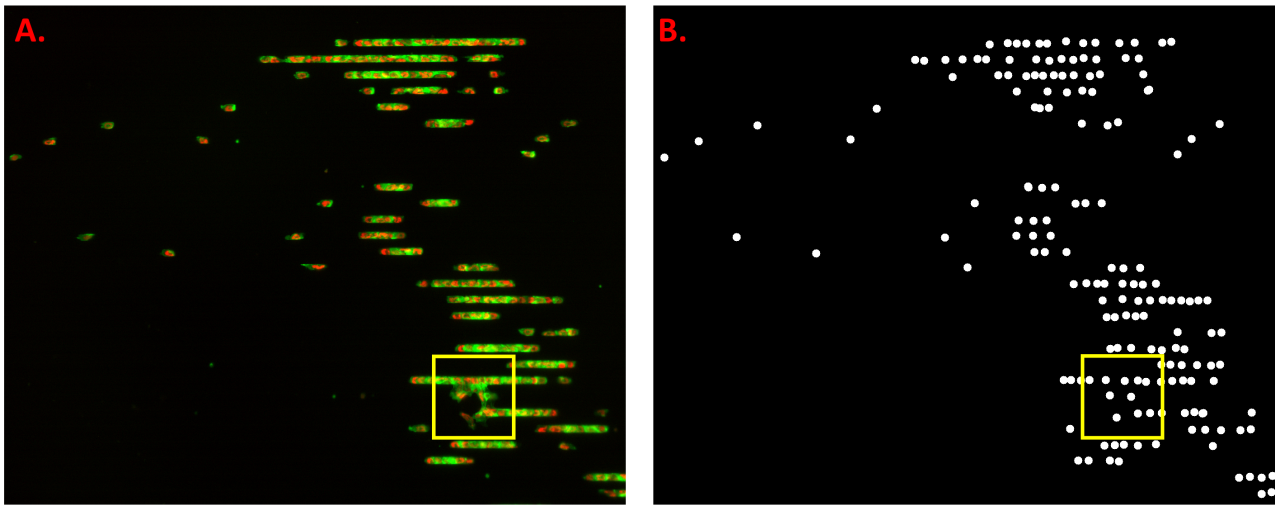

Figure S2. Frame 143 of the Neutrohils-6μm-2 video. Note the rupture in the lanes in the bottom right as denoted by the yellow square. A) Frame 143 of the Neutrohils-6μm-2 video in RGB. B) Masked frame 143 of the Neutrohils-6μm-2 video.

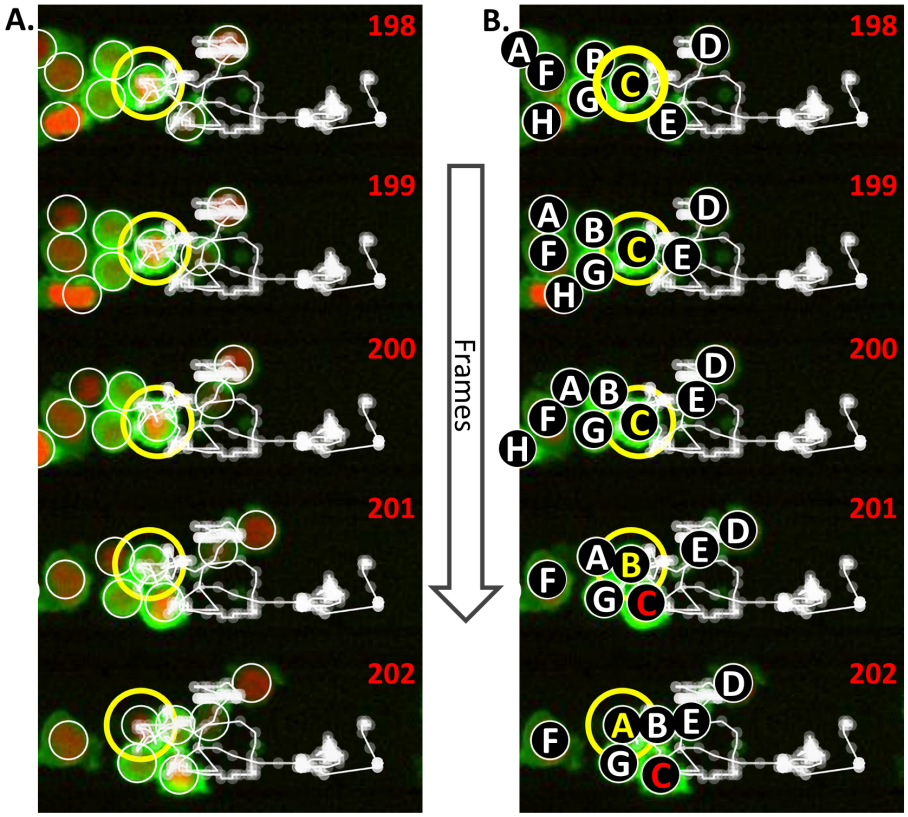

Figure S3. Visualization of track 'jumping' between cells due to being densely packed. Tracks where taken from the Tcells-30 μm-1 video, more specifically frame 198 to 202. A) unedited track from mentioned video. B) Manually edited track from mentioned video. Each letter represents a manually tracked cell. As visualized, the track 'jumps' in frame 2021 from cell 'C' to cell 'B', and in frame 202 again from cell 'B' to cell 'A', which is incorrect.

Table S1. Intersection over Union scores for detection masks, for all three compared methods. Measurements are normalized between [0, 1] interval, higher is better.

Software	T-cells 6μm	T-cells 15μm	T-cells 30μm	Neutrophils 6μm 1	Neutrophils 6μm 2	Average
In-house	0.979	0.984	0.989	0.973	0.988	0.983
Trackmate	0.989	0.984	0.992	0.984	0.987	0.987
CellProfiler	0.950	0.978	0.985	0.979	0.981	0.975

Values in table are rounded to three decimal places (0.001 precision).

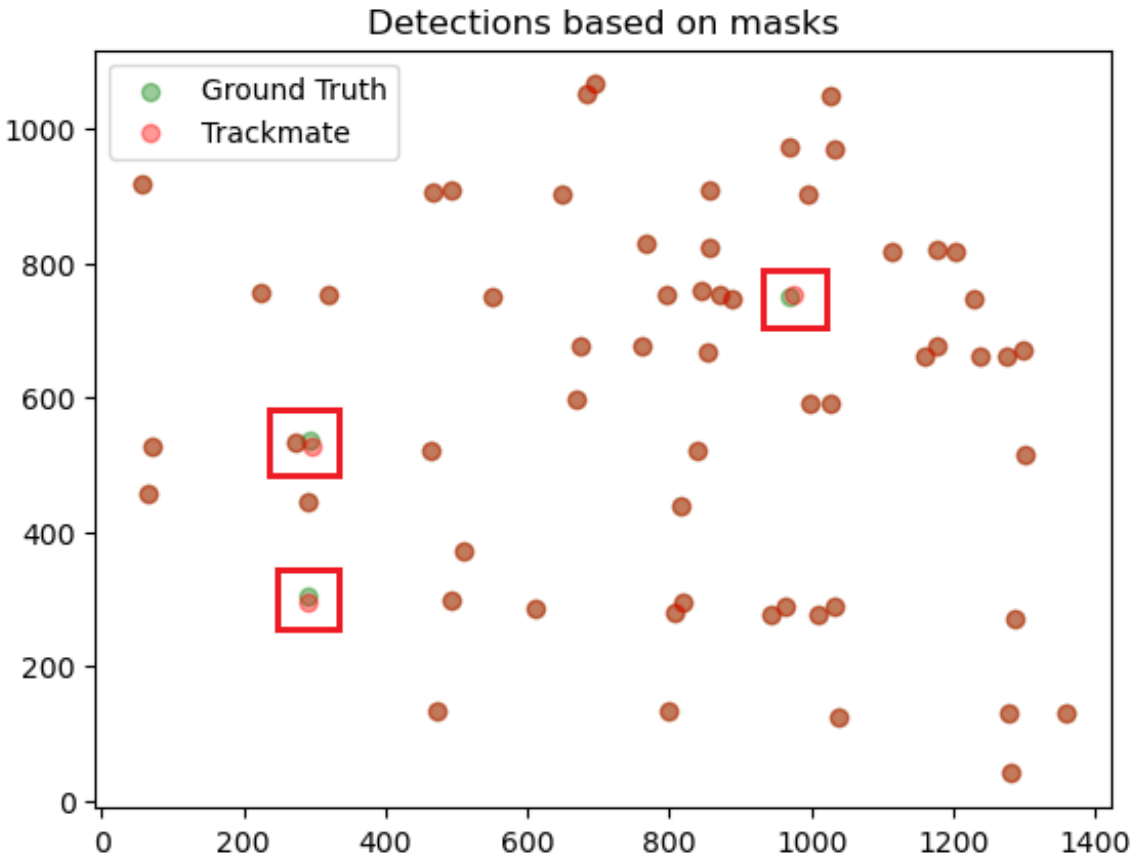

Figure S4. Representation of frame 13 of the masked Tcells-15μm-1 video. Each dot represents a detection, where the green dots are the ground truth, and the orange dots are generated by the Trackmate algorithm. Red squares highlight non-identical detection locations, causing the intersect of union scores to slightly decline. Note that these detection errors happen for all methods, and seem to average between 0.008 and 0.025%, which a single outlier of 0.05%.

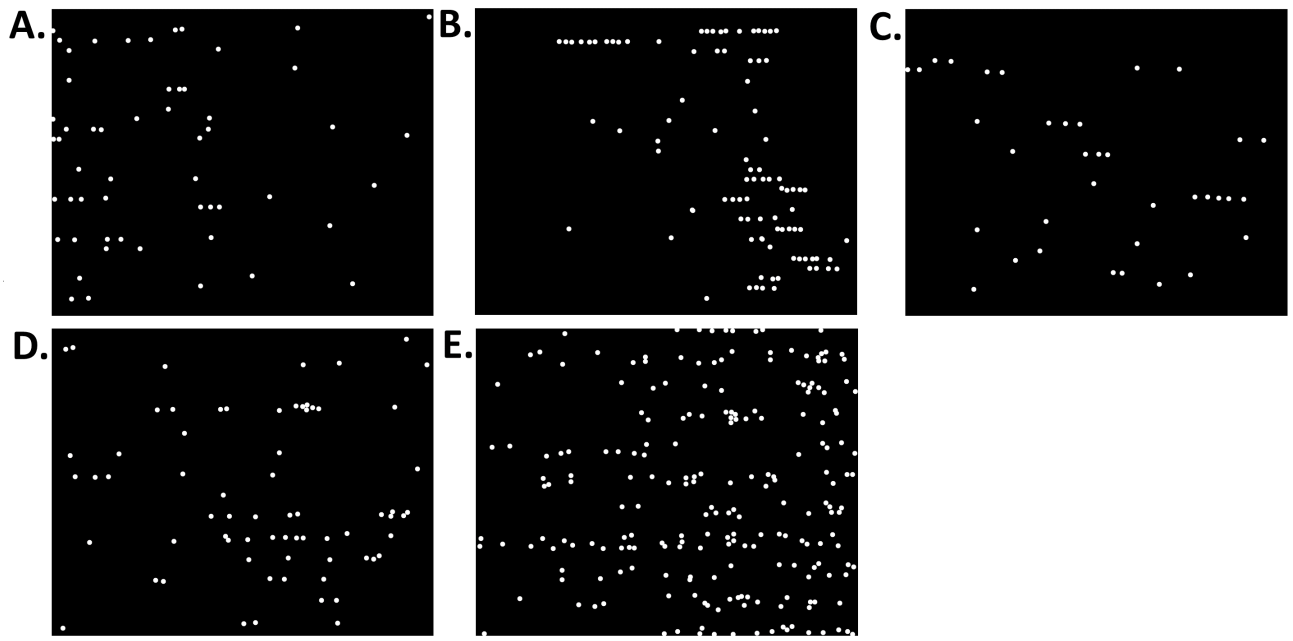

Figure S5. First frame of every masked video. Masked videos contain detection masks from the analysed videos in supplementary Figure 1. Every white pixel represents a cell. Black pixels represent the background. Other images can be can be downloaded from https://github.com/Sjonnie404/Cell-tracking-comparison. A) frame 0 from masked Neuts-6μm-1 video. B) frame 0 from masked Neuts-6μm-2 video. C) frame 0 from masked Tcells-6μm-1 video. D) frame 0 from masked Neuts-15μm-1 video. E) frame 0 from masked Neuts-30μm-1 video.

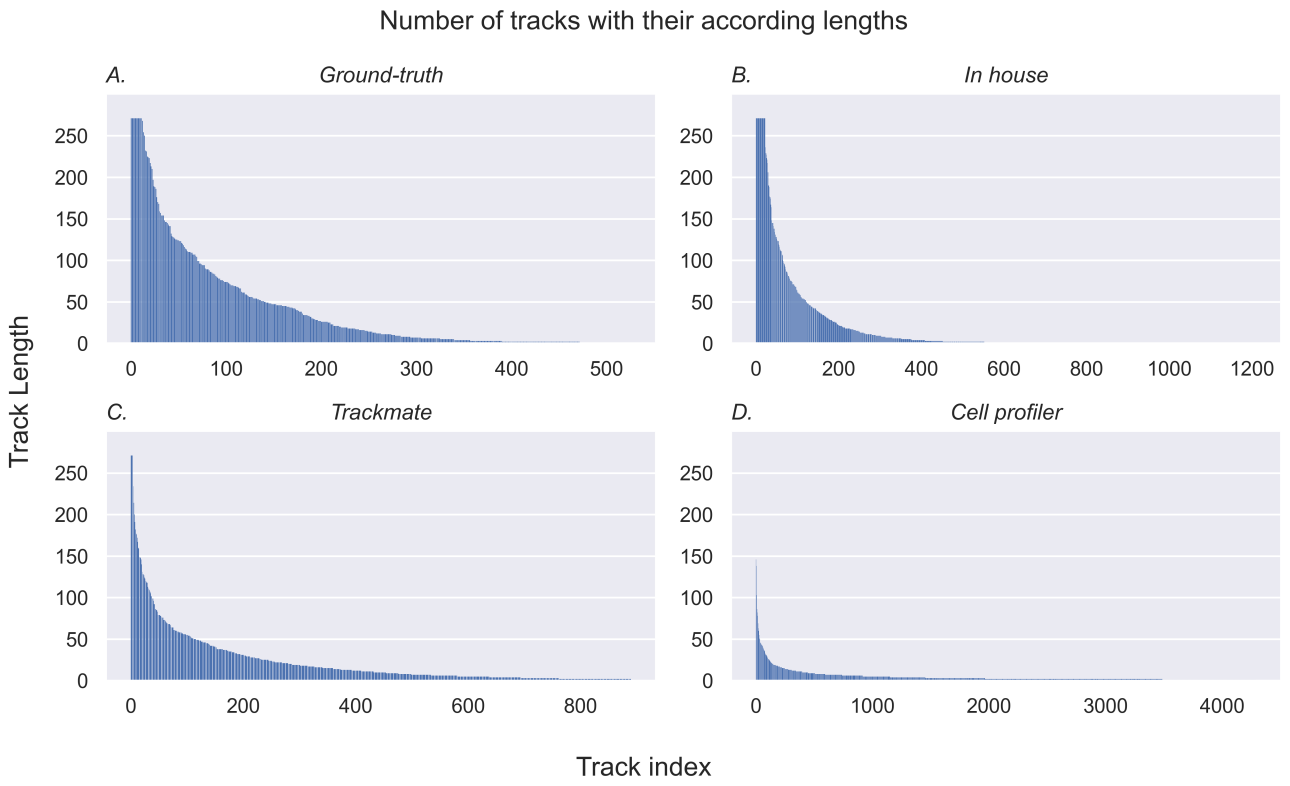

Figure S6. Number of tracks with their according track lengths, generated from the Neutrophils-6μm-1 masked video. A) Ground truth. B) in-house method. C) Trackmate method. D) CellProfiler method.

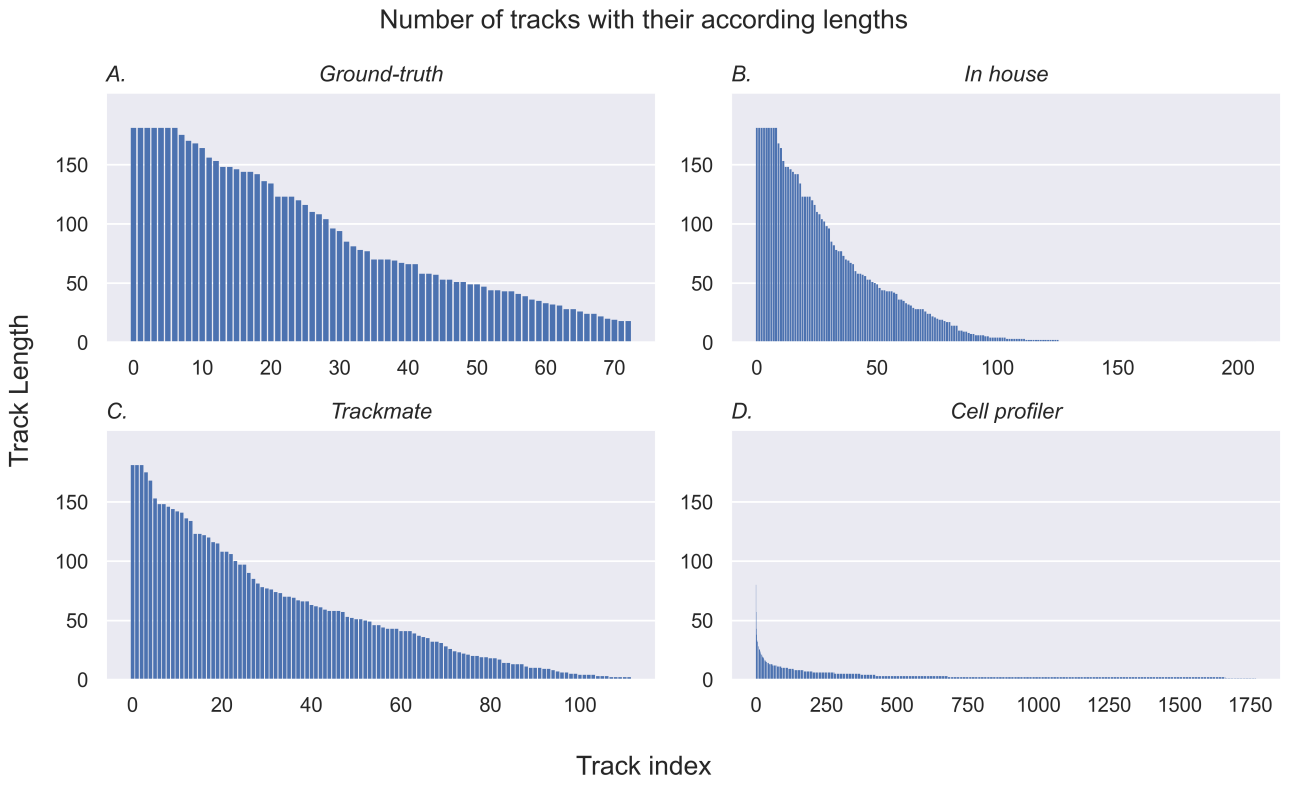

Figure S7. Number of tracks with their according track lengths, generated from the Tcells-6μm-1 masked video. A) Ground truth. B) in-house method. C) Trackmate method. D) CellProfiler method.

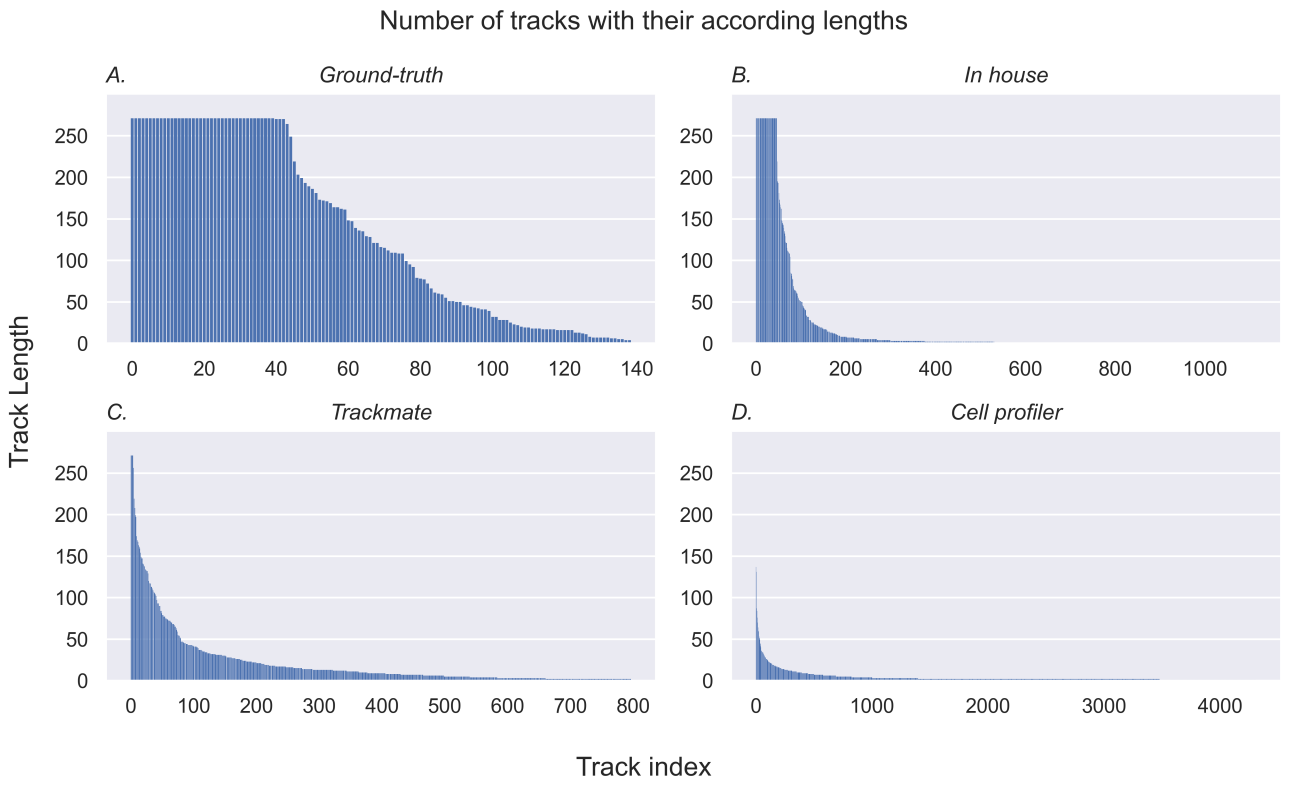

Figure S8. Number of tracks with their according track lengths, generated from the Tcells-15μm-1 masked video. A) Ground truth. B) in-house method. C) Trackmate method. D) CellProfiler method.

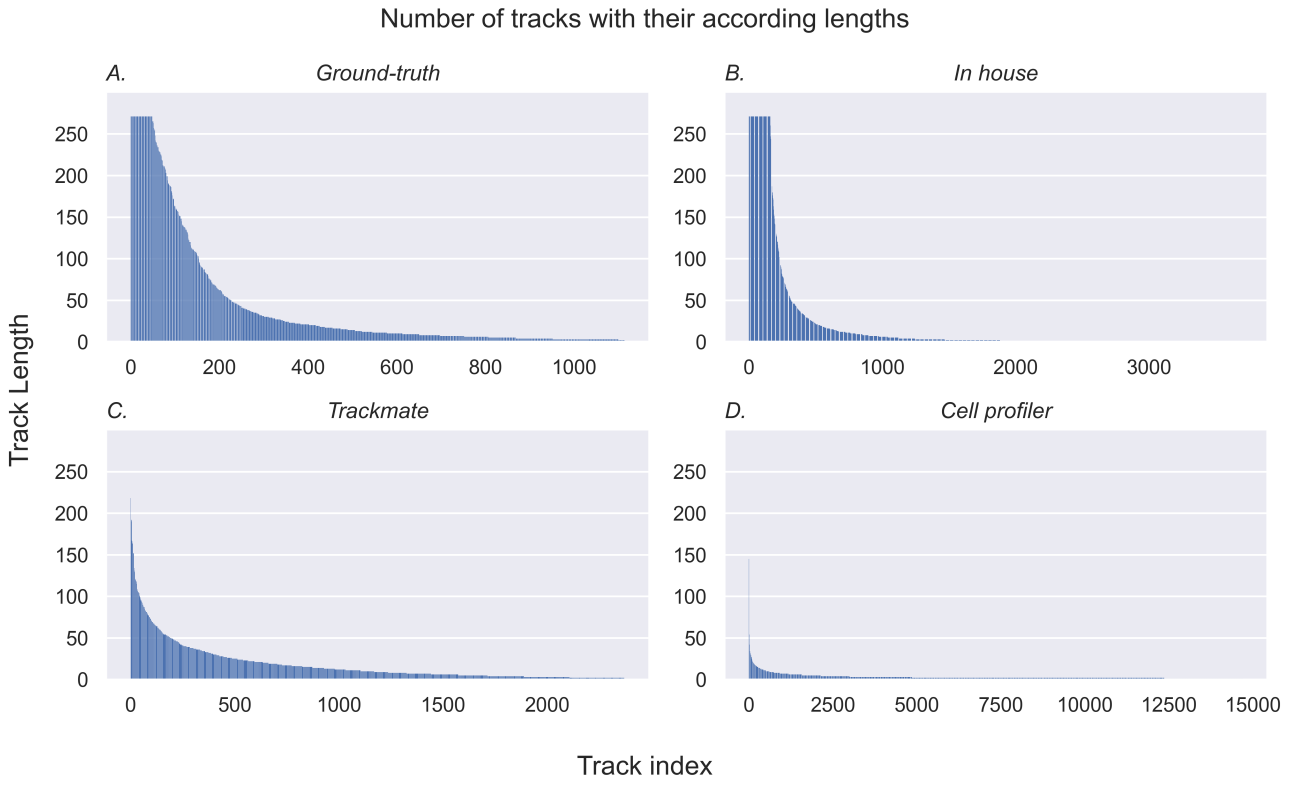

Figure S9. Number of tracks with their according track lengths, generated from the Tcells-30μm-1 masked video. A) Ground truth. B) in-house method. C) Trackmate method. D) CellProfiler method.

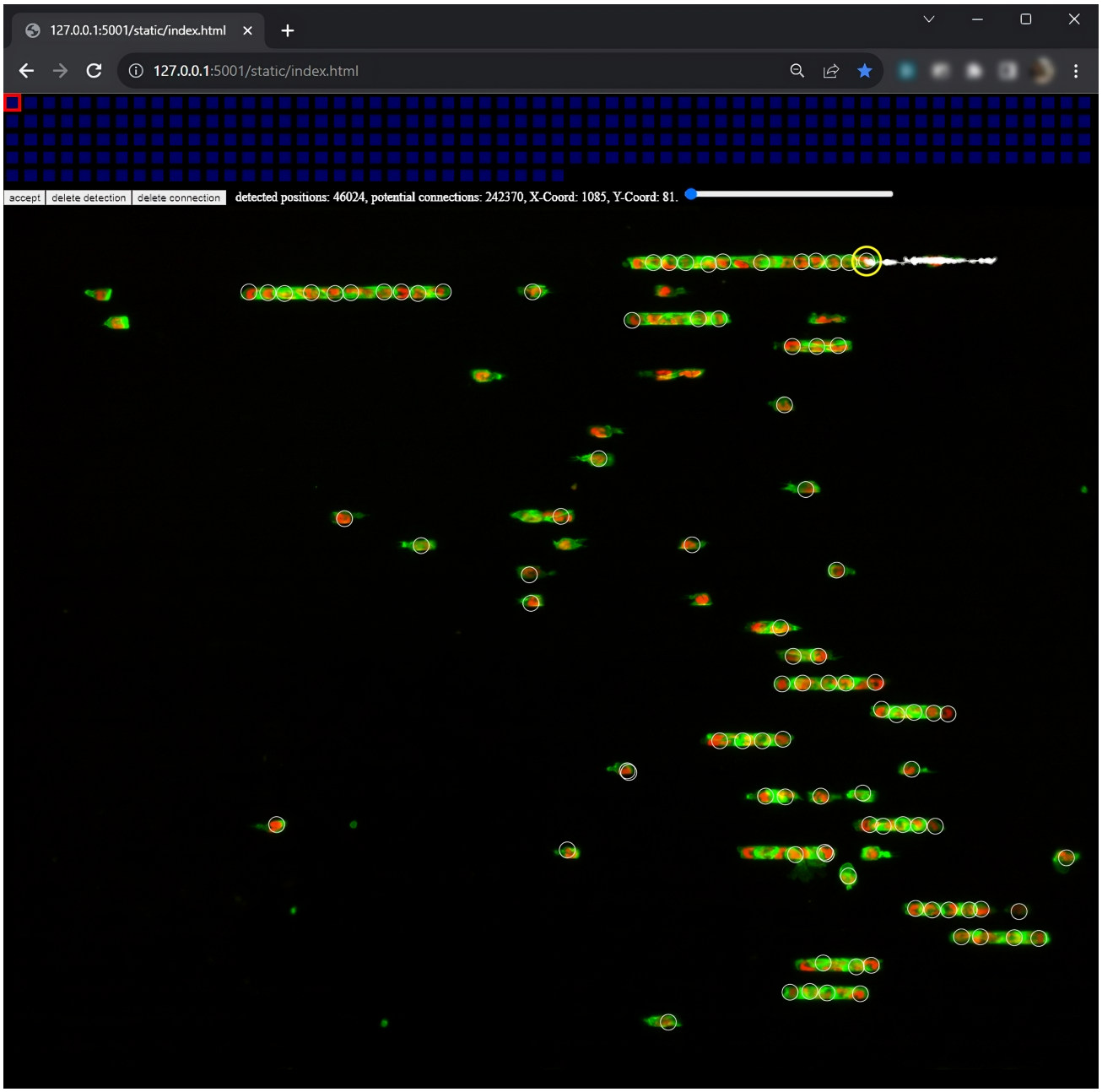

Figure S10. Screenshot of cell tracking web interface, blue squares represent frames of the video and change dynamically with the loaded video. These squared can be clicked to jump to a specific frame. The user can also use the arrow keys to move a single frame left or right. Below the squares are the buttons; 'accept', to accept the currently selected track and save it to the output file (key-shortcut 'a') , 'delete detection', to delete currently selected detection (yellow outlined circle, key-shortcut 'd') and 'delete connection', to delete the edge between the currently selected detection (yellow outline) and the next detection (key-shortcut 'c'). User can also create new detections by holding the 'shift' key and clicking on the frame (no on screen button shown). Next to the button, information about the graph and currently selected detection can be found. The frame with according track of the selected video is displayed below. Big, outlined, white circles represent all detections in the frame (that have not been accepted yet), yellow outline represents currently selected detection and small, white, filled circles, linked with a white edge represent future detections of the currently selected track. Only the currently selected track is displayed for clarity.