HHS Public Access

Author manuscript

Eurograph IEEE VGTC Symp Vis. Author manuscript; available in PMC 2023 January 16.

Published in final edited form as:

Eurograph IEEE VGTC Symp Vis. 2022 ; 2022: 115–119. doi:10.2312/evs.20221103.

CellTrackVis: analyzing the performance of cell tracking algorithms

W. Li[1], **X. Zhang**[1], **A. Stern**[2], **M. Birtwistle**[3], **F. Iuricich**[1]

[1]School of Computing, Clemson University, United States

[2]Icahn School of Medicine at Mount Sinai, New York, United States

[3]Department of Chemical and Biomolecular Engineering, Clemson University, United States

Abstract

Live-cell imaging is a common data acquisition technique used by biologists to analyze cell behavior. Since manually tracking cells in a video sequence is extremely time-consuming, many automatic algorithms have been developed in the last twenty years to accomplish the task. However, none of these algorithms can yet claim robust tracking performance at the varying of acquisition conditions (e.g., cell type, acquisition device, cell treatments). While many visualization tools exist to help with cell behavior analysis, there are no tools to help with the algorithm's validation.

This paper proposes CellTrackVis, a new visualization tool for evaluating cell tracking algorithms. CellTrackVis allows comparing automatically generated cell tracks with ground truth data to help biologists select the best-suited algorithm for their experimented pipeline. Moreover, CellTackVis can be used as a debugging tool while developing a new cell tracking algorithm to investigate where, when, and why each tracking error occurred.

1. Introduction

The ability to analyze moving cells in a video sequence is a fundamental investigative tool for biologists. Formally, cell tracking is the problem of tracking cells in a video sequence acquired from a microscope Field of View (FOV). Cells are first identified and classified according to their state: moving (if the cells are simply migrating), dying, or dividing. Then, cell tracks are iteratively computed in the *cell linking* phase by pairing the moving cells in a frame with the moving cells in the consecutive frame. While tracking cells by hand is extremely time-consuming, fully automated approaches can be leveraged to produce cell tracks automatically. Tracking methods can be classified in local and global linking methods [UMM*17,HNB20]. *Local methods* establish cell tracks by finding an optimal bijection between two sets of detections in consecutive images, while *global methods* compute cell tracks by working on the entire image set simultaneously. The key problem for both local and global approaches is how to measure the likelihood of two cells being linked. Common measures include handcrafted appearance features [DAS11, HZG*18], Euclidean distance [JLM*08], overlapping area [LLD20] or, more recently, feature vectors automatically generated by deep learning models [PŠF*19, HNB20]. Although many measures have been proposed to link cells, all these techniques still lack generality. This makes it hard for

biologists to identify the approach most suited for their experimental conditions (e.g., cell type, microscopy, frame rate). The visualization tool proposed in this paper aims to facilitate the comparison of cell tracking algorithms from a spatial and temporal point of view to easily identify the best tracking algorithm for a given experimental condition.

2. Related Works

In live-cell data processing, most visualization tools focus on the analysis and understanding of cellular behavior [PKE17]. In particular, the goal is to understand how cells are impacted by different experimental conditions (e.g., drugs or new treatments) [PKE15, LPJ*22, JKW*08,DSG*17]. This requires the analysis of many characteristics of cells population such as, cell movement and shape [PM07,HLLK09], cell proliferation [CBI*08,FHWL12], or cell progeny [GLHR09, PKE15]. Recent efforts have been concentrated on the visual analysis of large datasets produced by high-throughput cell imaging experiments. Most recently, Loon was introduced to process and analyze drug screening datasets [LPJ*22]. Loon allows to select and visualize single representative cells to avoid the time-consuming manual inspection of the entire dataset. ScreenIt [DSG*17], and CellProfilerAnalyst [JKW*08] are other two examples of visualization tools focusing on the analysis of cell behaviors and cell characteristics. Compared to these approaches, CellTrackViz puts its focus on cell tracking performance rather than cell analysis. In a way, CellTrackVis is more similar to annotation tools used to correct the results obtained in the pre-processing phase [HWKT09, KHC*07, WWR*11,WWB*14]. While most of these approaches focus on the cell identification phase [HWKT09], a few methods have been proposed for cell tracking [KHC*07,WWR*11,WWB*14]. These allow users to interact with the results of one cell tracking algorithm, and one FOV, and fix mistakes in the tracks computed. The typical visual interface used by these tools is the *lineage diagram* [PKE15], a hierarchical representation of the development history of each cell which includes events such as cell division or cell death. Different from these tools, CellTrackViz allows the analysis of multiple FOV and multiple algorithms at once.

3. Overview

Requirements

Our tool is designed to facilitate the comparison of ground truth cell tracks and predicted cell tracks. Cell links are computed by iteratively pairing cells in two consecutive frames of a video. A cell linking algorithm assigns every cell in a frame to a cell in the consecutive frame. Then, an *error link* is a link computed by a tracking algorithm between two frames, which does not exist in the ground-truth cell track data.

Targeted *users* of our tools are biologists interested in evaluating existing cell tracking algorithms to select the one best suited for their dataset (task i), or computer scientists evaluating the performance of a new algorithm being developed (task ii). To this end, the main requirements for our analysis approach are:

- *Algorithms comparison (R1).* It is required to visualize an overview of the algorithms' performance in order to establish, at a glance, which algorithm produces the least errors.
- *Spatial analysis (R2)* It is required to analyze how errors are spatially distributed on the field of view so as to identify spatial patterns in error committed.
- *Temporal analysis (R3)* It is required to analyze how errors are temporally distributed on the field of view. Healthy cell cultures tend to proliferate over time. Since a higher number of cells increases the possibility of linking errors, it is important to include this component so to analyze when errors are being committed.
- *Error link cause (R4)* A close-up visualization of the field of view is required to inspect the cause of a given error link. To this end, the user needs to select an error link (committed at a specific frame) to observe the cell movement pattern.

Dataset

Our study dataset consists of 3,468 images acquired at 15-minute intervals using a GE IN Cell Analyzer 2500 HS. Images are acquired at 20x magnification, and each image has a resolution of 2040x2040 pixels. Images were taken from 12 FOVs for a total of 289 images per FOV and 72 hours time course. Given the input image set, we have manually generated all cell tracks using the open-source software tool Image Fiji [SAF*12] and the Track- Mate [TPS*17] plugin. In addition, the dataset includes cell linking results for four cell tracking algorithms. We associated each method with a unique label (and color). CNN, indicates a new algorithm developed in-house. LAP [JLM*08] is a simple approach which computes links by solving a linear assignment problem. TRACKMATE, provided by TrackMate [TPS*17], uses a Kalman filter [Kal60] and a linear motion model to estimate a cell future position. TRACKPY [CG96] is provided by Trackpy [ACK*21] and computes cell tracks by using a Brownian motion model.

4. Design and implementation

The tool is organized in three distinct views, designed to address the requirements and tasks described before. In this section, we refer to Figure 1(a–h) to indicate the components of our tool's layout.

4.1. Performance view

The first dashboard provides an overview of the algorithms' performance on all the fields of views in the dataset. Each field of view is represented by a card (Figure 1(a–c)). Cards are organized (left-to-right top-to-bottom) in descending order of links number.

Each card is formed by three components. The same categorical color map indicates an algorithm in all components. The FOV ID is indicated at the top of the card (a) and colored according to the best-performing algorithm. A bar chart indicates the total number of error links committed by each algorithm (b). A line chart (c) indicates, for each algorithm, the cumulative number of error links committed over time. Namely, the x-axis of the line chart

indicates the frame number and the y-axis indicate the total number of errors committed up to that point. Interactions with the line chart allow the user to precisely quantify the error links committed by the various algorithms up to a specific frame.

Discussion—Comparing the overall performance is a central need for both biologists and computer scientists (R1). CellTrackViz avoids normalized measures (e.g., precision, recall, track fraction score, or complete track score [UMM*17]) to maintain an explicit reference to the cell population's cardinality. This is enforced by the card's ordering that drags the user's attention on the most challenging datasets (the one with more links and consequently more cells). The possibility to analyze performance over time (R3) exposes characteristics of the algorithms not visible with static views. For example, in Figure 2(a), we notice that the performance of TRACKMATE in *FOV*#1 are competitive early on with those of the other algorithms and degrade over time. The same is observed in *FOV*#11 (see Figure 2(b)). Exposing this information can help the user formulate a hypothesis regarding the cause of these errors. For example, TRACKMATE may be confounded by the total number of cells in the FOV, so a growing population may cause low performance. This is important for biologists that may decide to avoid TRACKMATE if their experiments require high cell density. Tracking errors over time can also expose differences in the FOVs characteristics rather than just in an algorithm. For example, for *FOV*#1 (see Figure 2(a)) we notice that linking errors steadily increase for all algorithms while in *FOV*#11 (see Figure 2(b)) the errors sharply increase only at the beginning until reaching a plateau. This effect may be due to cell populations with different mobility patterns or different cell densities.

4.2. Algorithm's view

The second dashboard allows the user to focus on a single algorithm to either inspect its performance or to refine the hypothesis formulated with the first view. Also in this case, the layout uses small multiples to represent each FOV with a card. Each card represents the error links spatially organized in the FOV domain (R2). Each error is represented by a dot indicating the cell position before the error was committed and a line indicating the wrong link created by the algorithm. The length of the line, in particular, provides visual clues regarding the types of spatial errors committed by a specific algorithm. For example, longer lines indicate errors committed by an algorithm that tends to classify cells as fast (traveling long distances between frames); shorter lines indicate that the algorithm tends to connect close occurrences of cells. In addition to the error links, each card shows a line chart (e) indicating the population at each time step. Namely, the x-axis of the line chart indicates the frame number and the y-axis indicate the total number of cells in that frame. By interacting with the line chart, a user can select a specific time step/frame and visualize only error links up to that frame.

Discussion—This view provides a unique spatio-temporal visualization of a cell population. From the temporal standpoint, analyzing the cell population over time comes from an explicit requirement of biologists to track the cell population's health. For example, we may notice that only a few error links are committed on a population of dying cells. While overall, this may be numerically reported as a good tracking score, visually, we can recognize that the reason is due to a reduced number of cells (and not the algorithm's

ability to track cells). At the same time, this view allows to confirm or reject the hypothesis formulated while looking at the overview results. For example, by focusing on TRACKMATE, we can notice that also the cell number has generally sharp increases at the beginning of each FOV (see Figure 4). This supports the hypothesis that TRACKMATE suffers when the total cell number in a FOV is higher. Another important information provided by this view is the type of links created by an algorithm. For example, comparing TRACKPY and TRACKMATE on FOV 1 (see Figure 3), we notice that TRACKMATE commits errors on longer links which indicates the algorithm assumes cells will travel long distances between one frame and the other. Instead, the errors committed by TRACKPY are depicted with shorter lines.

4.3. FOV view

The third view displays *detailed information* about a selected algorithm and a specific FOV. This view is organized into two components. The first component (Figure 1(g)) is a scatter plot representation showing all error links. Errors are organized according to the cell track and the time frame they occurred in. Errors on the same horizontal position belong to the same cell track. Errors on the same vertical position occurred in the same time frame (indicated by a label). The scatter plot representation is juxtaposed to a visualization of the frames (Figure 1(f)). At loading time, the last frame of the sequence is shown. Every time the user interacts with the slider (Figure 1(h)), or the scatterplot (Figure 1(g)), the frame and the displayed frame number are updated to align with the selection.

The scatterplot is used as a selection tool. By clicking on a horizontal line, the user can select a cell path. As a result, all error links involving that path will be displayed. Selecting a single error link will trigger a close-up view of the error (see Figure 5). Specifically, the window will be automatically zoomed in the neighborhood of the error, and the error link visualization will be augmented, showing the path of the cell before and after the error. If the user interacts with the slider, the path will get automatically updated to show the new position of the cell in the frame.

Discussion—Biologists can use this view to confirm or reject hypotheses about general algorithm behavior. Computer scientists can use this view to improve a linking algorithm under development. Figure 5 shows an error link as depicted by the FOV view. Upon selection, the view is centered at the location where the error occurred. The white line indicates the ground-truth path of the selected cell. Dashed lines indicate the path before the error link, while the solid white line indicates the ground-truth path after the error link. The orange point indicates where the error link occurred, while the orange line indicates the wrong path. Bright-field images can be hard to read for humans due to the low contrast. For this reason, we used a white line (in high contrast with the background) to help the user identify the cell's path in the sequence before and after the error link occurred. On multiple occasions, this detailed analysis of the error links helped us identify and correct the input annotations (i.e., what we assumed as ground-truth data).

5. Conclusion

CellTrackVis shows detailed information about cell tracking errors. The General view allows domain scientists to identify the best tracking algorithm for their experimental conditions. The algorithm's and FOV views are useful debugging tools to identify common mistakes committed by specific tracking algorithms. CellTrackVis currently focuses on linking errors only. We are interested in extending the tool so as to include errors committed during the identification phase.

Acknowledgment

We wish to thanks the reviewers for their insightful comments. W. Li was partially supported by Clemson University's R-Initiatives. M. Birtwistle was partially supported from NIH/NIGMS R35GM141891. Our code is available at https://github.com/DaVisLab/CellTrackVis.

References

[ACK*21]. Allan DB, Caswell T, Keim NC, van der Wel CM, Verweij RW: Soft-matter/trackpy: Trackpy v0.5.0. Zenodo, Apr. 2021. doi:10.5281/zenodo.4682814. 2

[CBI*08]. Cedilnik A, Baumes J, Ibanez L, Megason S, Wylie B: Integration of information and volume visualization for analysis of cell lineage and gene expression during embryogenesis. In Electronic Imaging 2008 (San Jose, CA, Jan. 2008), Börner K, Gröhn MT, Park J, Roberts JC, (Eds.), p. 68090J. doi:10.1117/12.768014. 2

[CG96]. Crocker JC, Grier DG: Methods of digital video microscopy for colloidal studies. Journal of colloid and interface science 179, 1 (1996), 298–310. 2

[DAS11]. Dewan MAA, Ahmad MO, Swamy MNS: Tracking biological cells in time-lapse microscopy: An adaptive technique combining motion and topological features. IEEE Trans. Biomed. Engineering 58, 6 (2011), 1637–1647. doi:10.1109/TBME.2011.2109001. 2

[DSG*17]. Dinkla K, Strobelt H, Genest B, Reiling S, Borowsky M, Pfister H: Screenit: Visual Analysis of Cellular Screens. IEEE Transactions on Visualization and Computer Graphics 23, 1 (Jan. 2017), 591–600. doi:10.1109/TVCG.2016.2598587. 2 [PubMed: 27875174]

[FHWL12]. Fangerau J, Hockendorf B, Wittbrodt J, Leitte H: Similarity analysis of cell movements in video microscopy. In 2012 IEEE Symposium on Biological Data Visualization (BioVis) (Seattle, WA, USA, Oct. 2012), IEEE, pp. 69–76. doi:10.1109/BioVis.2012.6378595. 2

[GLHR09]. Glauche I, Lorenz R, Hasenclever D, Roeder I: A novel view on stem cell development: Analysing the shape of cellular genealogies. Cell Proliferation 42, 2 (Apr. 2009), 248–263. doi:10.1111/j.1365-2184.2009.00586.x. 2 [PubMed: 19254328]

[HLLK09]. HSU Y-Y, Liu Y-N, Lu W-W, Kung S-H: Visualizing and quantifying the differential cleavages of the eukaryotic translation initiation factors eIF4GI and eIF4GII in the enterovirus-infected cell. Biotechnology and Bioengineering 104, 6 (Dec. 2009), 1142–1152. doi:10.1002/bit.22495.2 [PubMed: 19655339]

[HNB20]. Hayashida J, Nishimura K, Bise R: MPM: Joint Representation of Motion and Position Map for Cell Tracking. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (Seattle, WA, USA, June 2020), IEEE, pp. 3822–3831. doi:10.1109/CVPR42600.2020.00388. 1, 2

[HWKT09]. Hamilton NA, Wang JT, Kerr MC, Teas-dale RD: Statistical and visual differentiation of subcellular imaging. BMC Bioinformaitics 10, 1 (Dec. 2009), 94. doi:10.1186/1471-2105-10-94. 2

[HZG*18]. Hu H, Zhou L, Guan Q, Zhou Q, Chen S: An automatic tracking method for multiple cells based on multi-feature fusion. IEEE Access 6 (2018), 69782–69793. doi:10.1109/ACCESS.2018.2880563. 2

[JKW*08]. Jones TR, Kang IH, Wheeler DB, Lindquist RA, Papallo A, Sabatini DM, Golland P, Carpenter AE: CellProfiler Analyst: Data exploration and analysis software for complex image-

based screens. BMC Bioinformatics 9, 1 (Dec. 2008), 482. doi:10.1186/1471-2105-9-482. 2 [PubMed: 19014601]

[JLM*08]. Jaqaman K, Loerke D, Mettlen M, Kuwata H, Grinstein S, Schmid SL, Danuser G: Robust single-particle tracking in live-cell time-lapse sequences. Nature Methods 5, 8 (Aug. 2008), 695–702. doi:10.1038/nmeth.1237. 2 [PubMed: 18641657]

[Kal60]. Kalman RE: A New Approach to Linear Filtering and Prediction Problems. Journal of Basic Engineering 82, 1 (Mar. 1960), 35–45. doi:10.1115/1.3662552.2. 2

[KHC*07]. Khan IA, Husemann P, Campbell L, White NS, White RJ, Smith PJ, Errington RJ: ProgeniDB: A Novel Cell Lineage Database for Generation Associated Phenotypic Behavior in Cell-based Assays. Cell Cycle 6, 7 (Apr. 2007), 868–874. doi:10.4161/cc.6.7.4045. 2 [PubMed: 17387278]

[LLD20]. Lugagne J-B, Lin H, Dunlop MJ: DeLTA: Automated cell segmentation, tracking, and lineage reconstruction using deep learning. PLOS Computational Biology 16, 4 (Apr. 2020), e1007673. doi:10.1371/journal.pcbi.1007673.2 [PubMed: 32282792]

[LPJ*22]. Lange D, Polanco E, Judson-Torres R, Zangle T, Lex A: Loon: Using Exemplars to Visualize Large-Scale Microscopy Data. IEEE Transactions on Visualization and Computer Graphics 28, 1 (Jan. 2022), 248–258. doi:10.1109/TVCG.2021.3114766. 2 [PubMed: 34587022]

[PKE15]. Pretorius AJ, Khan IA, Errington RJ: Cell lineage visualisation. Computer Graphics Forum 34, 3 (June 2015), 21–30. doi:10.1111/cgf.12614. 2

[PKE17]. Pretorius AJ, Khan IA, Errington RJ: A Survey of Visualization for Live Cell Imaging: Visualization for Live Cell Imaging. Computer Graphics Forum 36, 1 (Jan. 2017), 46–63. doi:10.1111/cgf.12784. 2

[PM07]. Prasad M, Montell DJ: Cellular and Molecular Mechanisms of Border Cell Migration Analyzed Using Time-Lapse Live-Cell Imaging. Developmental Cell 12, 6 (June 2007), 997–1005. doi:10.1016/j.devcel.2007.03.021. 2 [PubMed: 17543870]

[PŠF*19]. Payer C, Štern D, Feiner M, Bischof H, Urschler M: Segmenting and tracking cell instances with cosine embeddings and recurrent hourglass networks. Medical Image Analysis 57 (Oct. 2019), 106–119. doi:10.1016/j.media.2019.06.015. 2 [PubMed: 31299493]

[SAF*12]. Schindelin J, Arganda-Carreras I, Frise E, Kaynig V, Longair M, Pietzsch T, Preibisch S, Rueden C, Saalfeld S, Schmid B, Tinevez J-Y, White DJ, Harten-Stein V, Eliceiri K, Tomancak P, Cardona A: Fiji: An open-source platform for biological-image analysis. Nature Methods 9, 7 (July 2012), 676–682. doi:10.1038/nmeth.2019. 2 [PubMed: 22743772]

[TPS*17]. Tinevez J-Y, Perry N, Schindelin J, Hoopes GM, Reynolds GD, Laplantine E, Bednarek SY, Shorte SL, Eliceiri KW: TrackMate: An open and extensible platform for single-particle tracking. Methods 115 (Feb. 2017), 80–90. doi:10.1016/J.YMETH.2016.09.016. 2 [PubMed: 27713081]

[UMM*17]. Ulman V, Maška M, Magnusson KE, Ron-Neberger O, Haubold C, Harder N, Matula P, Matula P, Svoboda D, Radojevic M, et al. : An objective comparison of cell-tracking algorithms. Nature methods 14, 12 (2017), 1141. 1, 3 [PubMed: 29083403]

[WWB*14]. Wait E, Winter M, Bjornsson C, Kokovay E, Wang Y, Goderie S, Temple S, Cohen AR: Visualization and correction of automated segmentation, tracking and lineaging from 5-D stem cell image sequences. BMC Bioinformaitics 15, 1 (Dec. 2014), 328. doi:10.1186/1471-2105-15-328. 2

[WWR*11]. Winter M, Wait E, Roysam B, Goderie SK, Ali RAN, Kokovay E, Temple S, Cohen AR: Vertebrate neural stem cell segmentation, tracking and lineaging with validation and editing. Nature Protocols 6, 12 (Dec. 2011), 1942–1952. doi:10.1038/nprot.2011.422.2 [PubMed: 22094730]

CCS Concepts

• *Applied computing* → *Bioinformatics;* • *Human-centered computing* → *Visualization toolkits;*

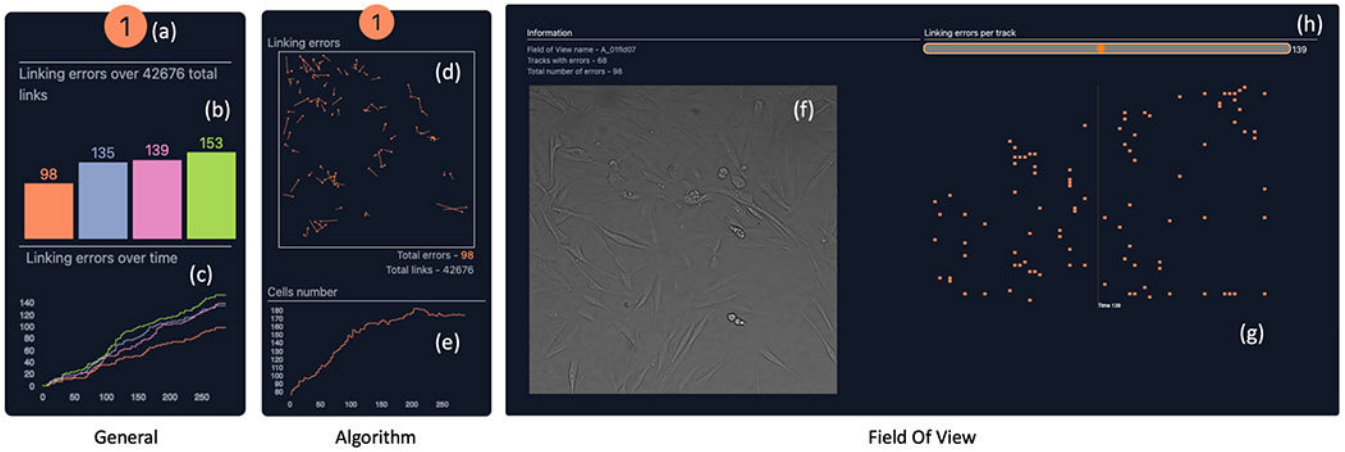
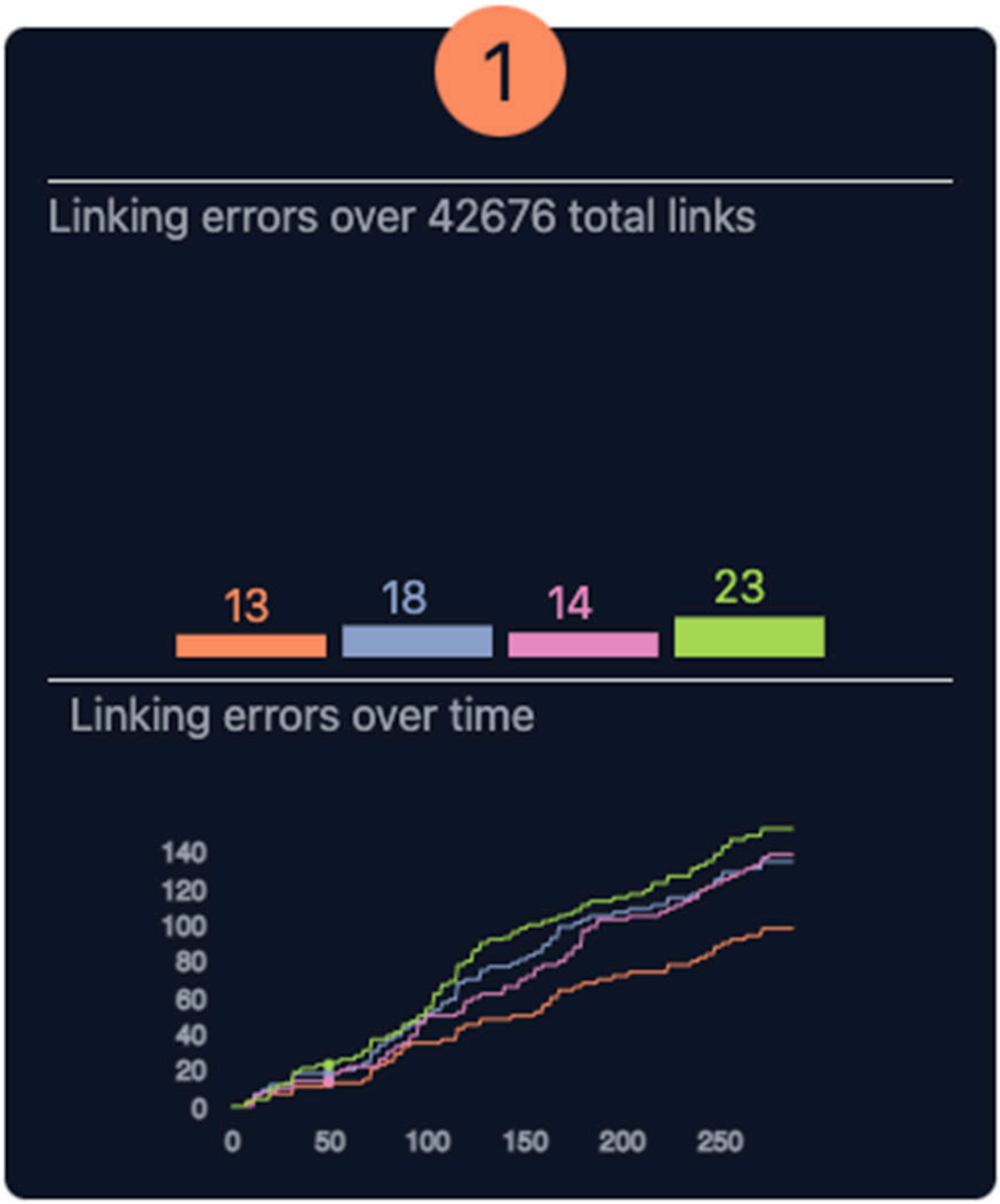
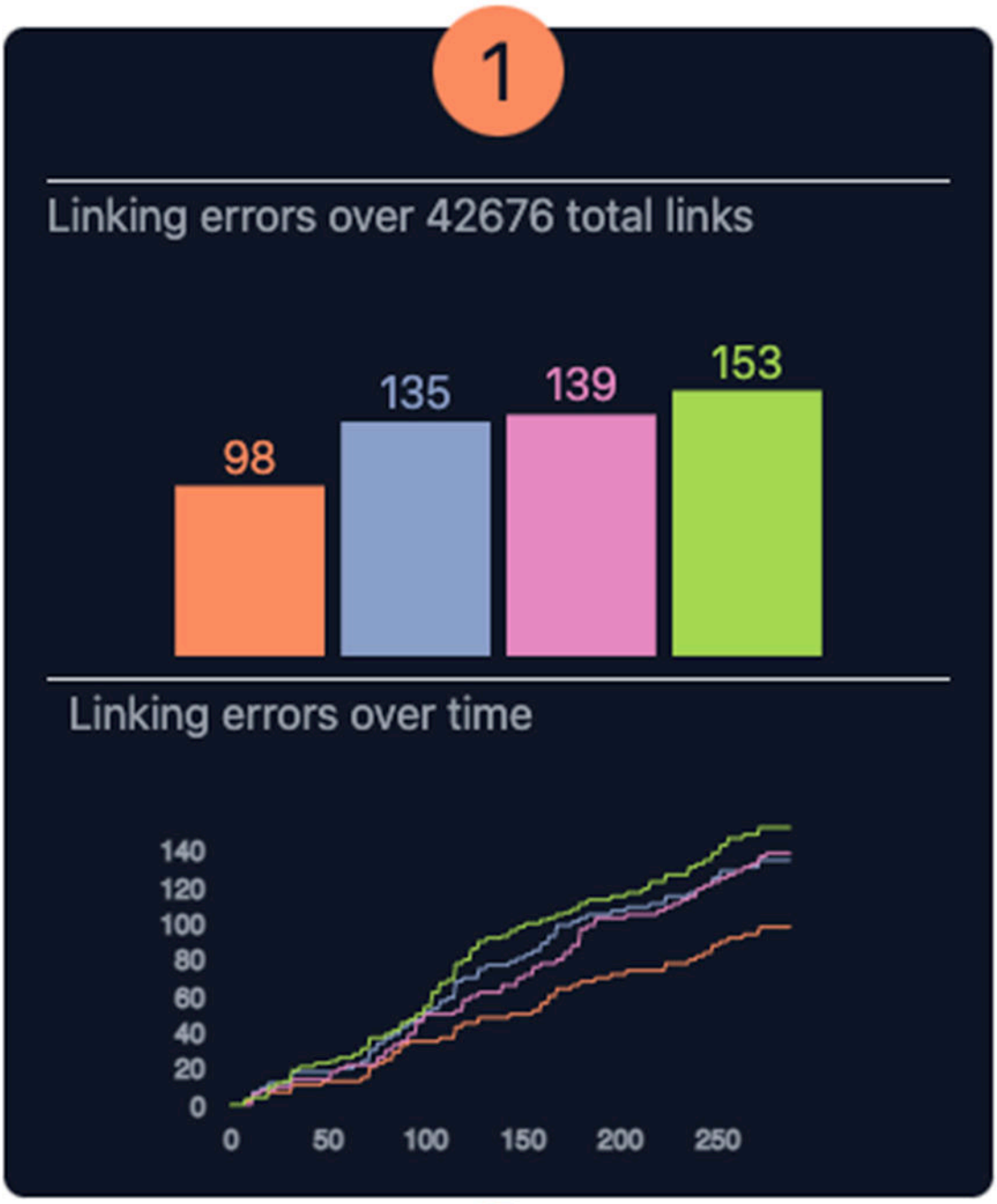
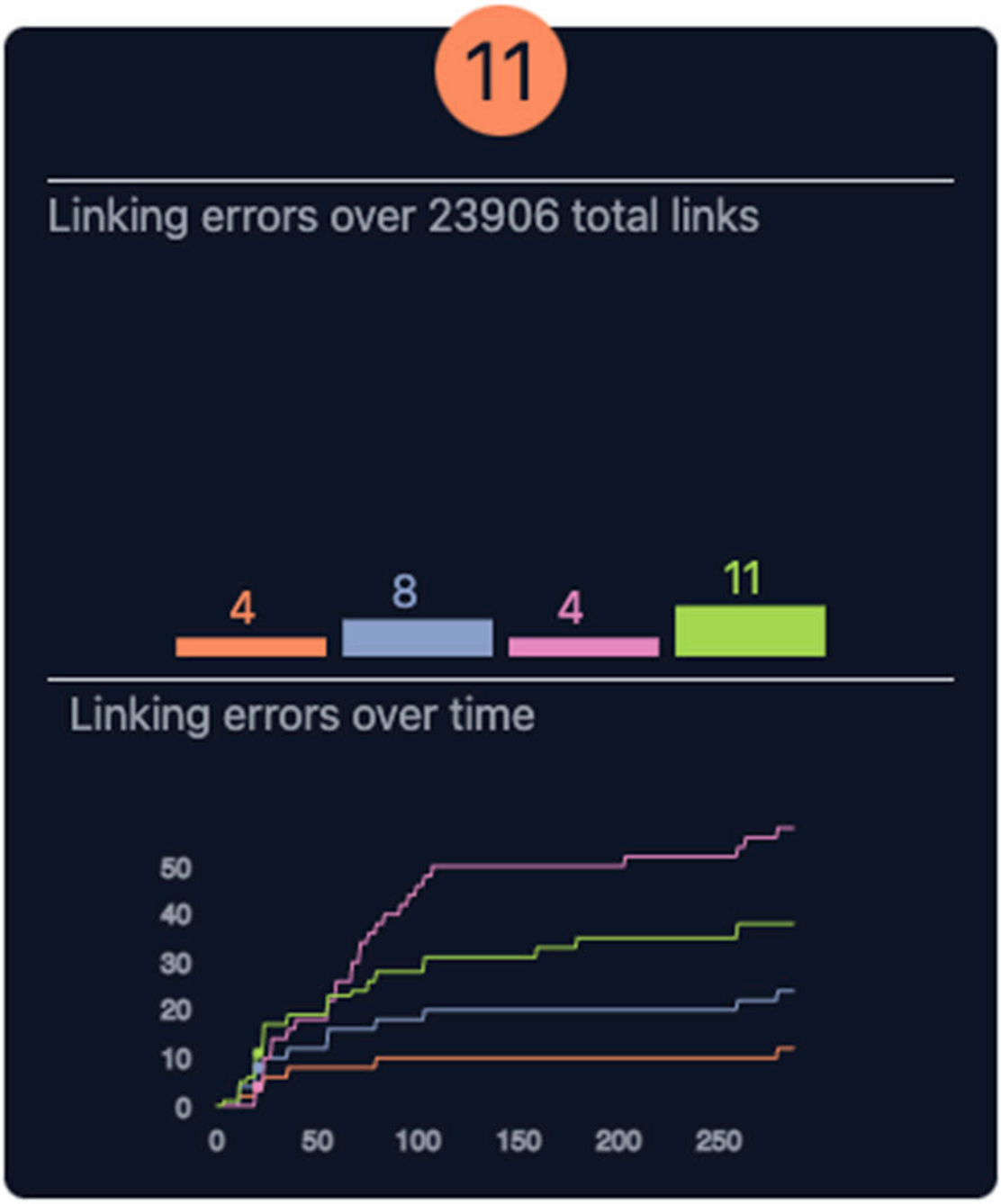


Figure 1:

CellTrackVis uses three different views to analyze and compare cell tracking algorithms at the general (a-c), single algorithm (d-e), and field of view (f-h) levels. The general view allows comparing the performance of multiple algorithms at once. The Algorithm's view allows focusing on the spatial patterns and errors committed by an algorithm on multiple fields of views (i.e., image sequences). By selecting an algorithm and a single field of view, the user can analyze the causes of each error by studying the original bright-field images composing the video sequence.







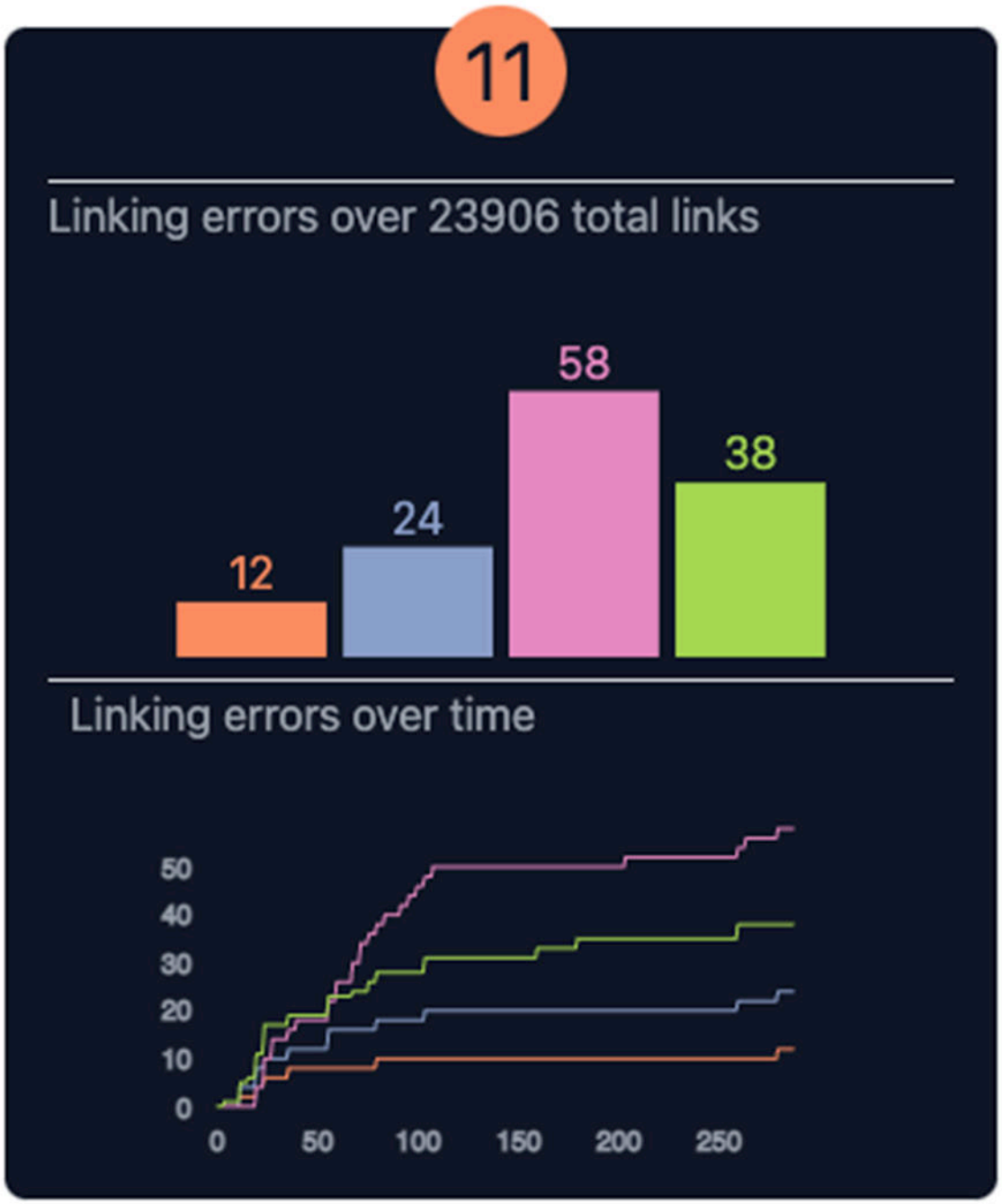
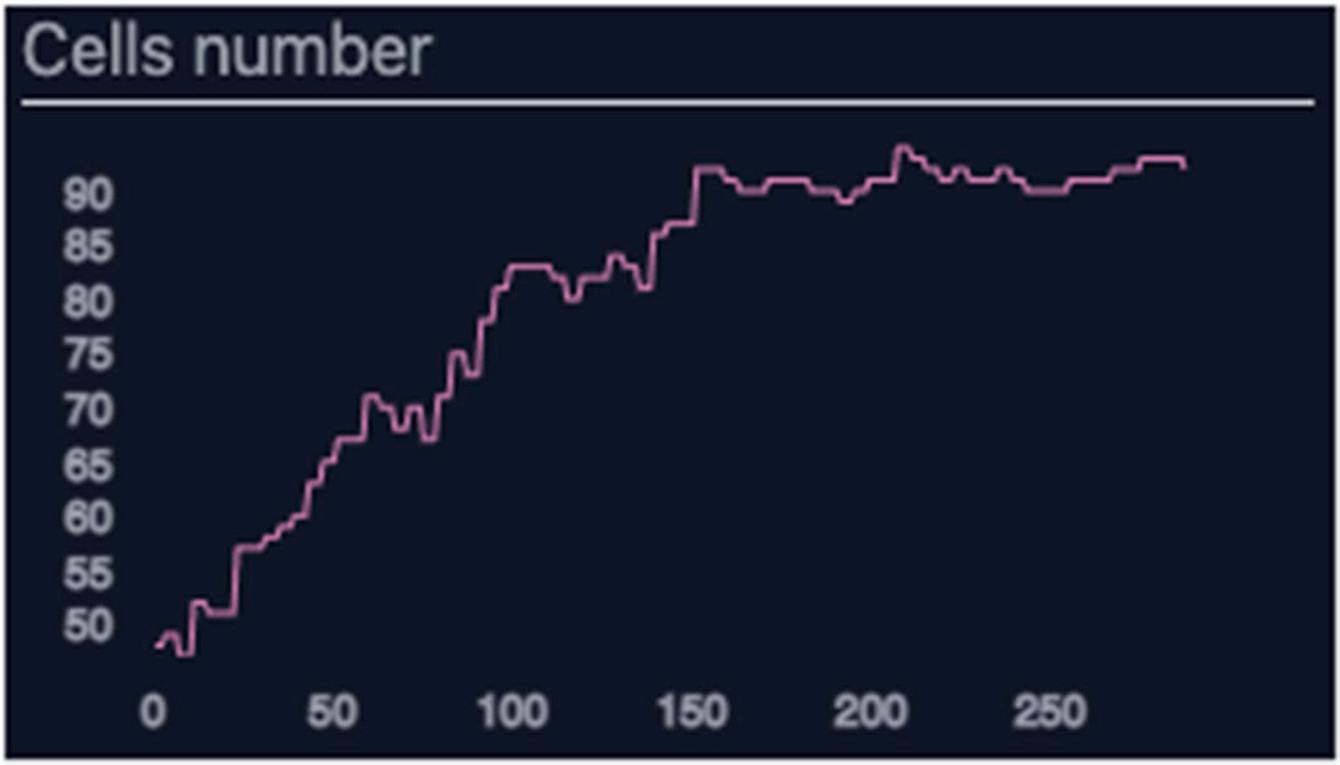


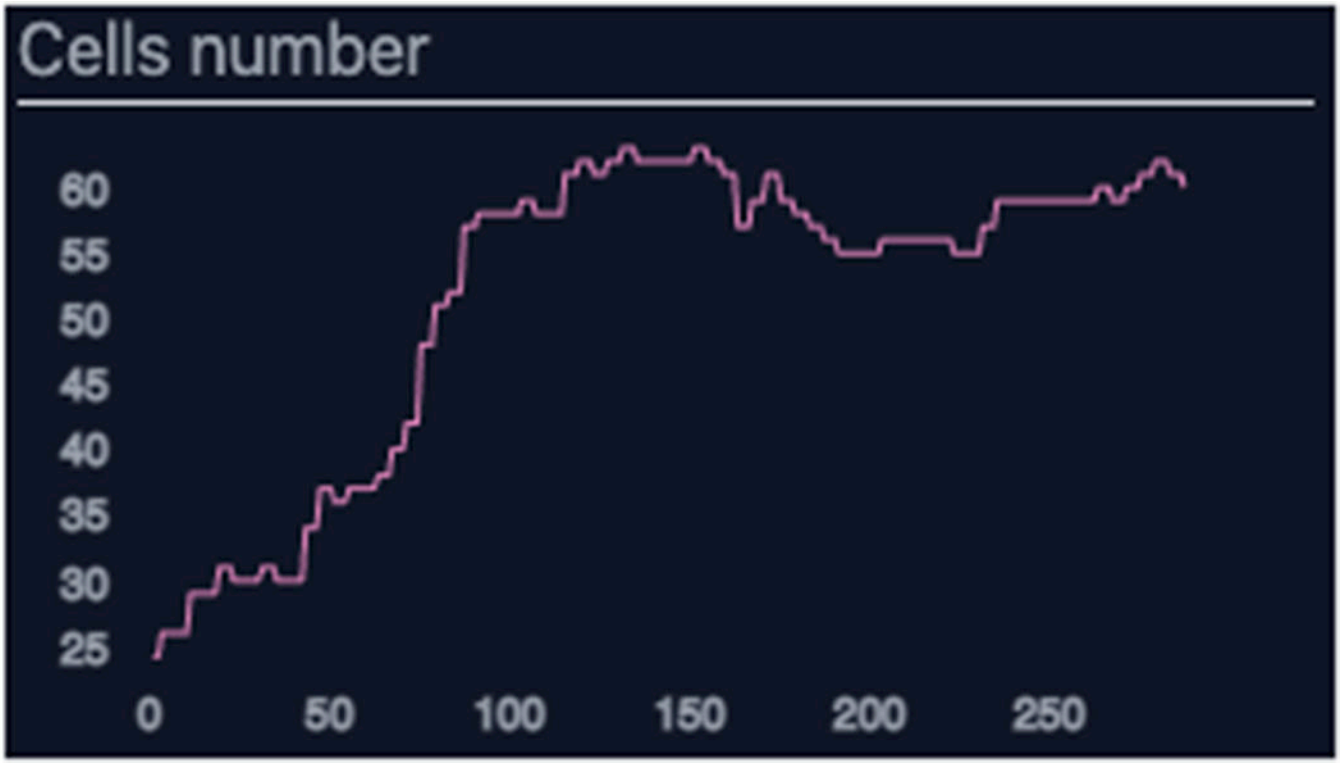
Figure 2:

Performance view for (a) FOV#1 and (b) FOV#11. By interacting with the line chart the user can investigate the number of error links committed up to a certain frame.

Figure 3:

Comparison of the error links' spatial patterns produced by (a) TRACKMATE *and (b)* TRACKPY *on FOV*#1. *The longer lines representing* TRACKMATE*'s errors indicate the algorithm assumes cells are moving very fast between consecutive frames. As opposed,* TRACKPY *produces errors depicted with shorter lines, which means it assumes cells are mostly static.*





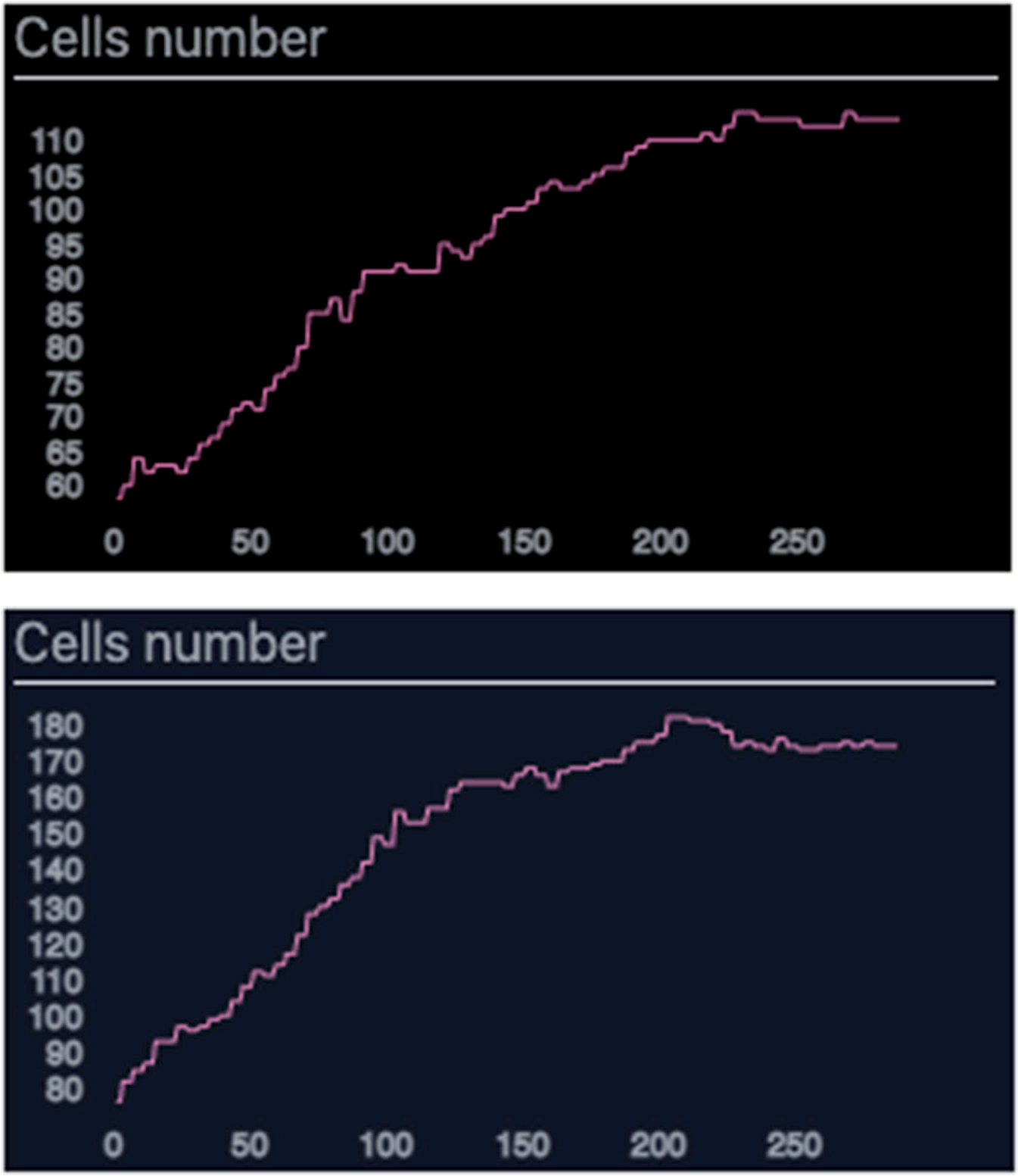


Figure 4:

Line chart showing the total number of cells in (a) FOV#3, *(b) FOV*#7, *(c) FOV*#8 *and (d) FOV*#1.

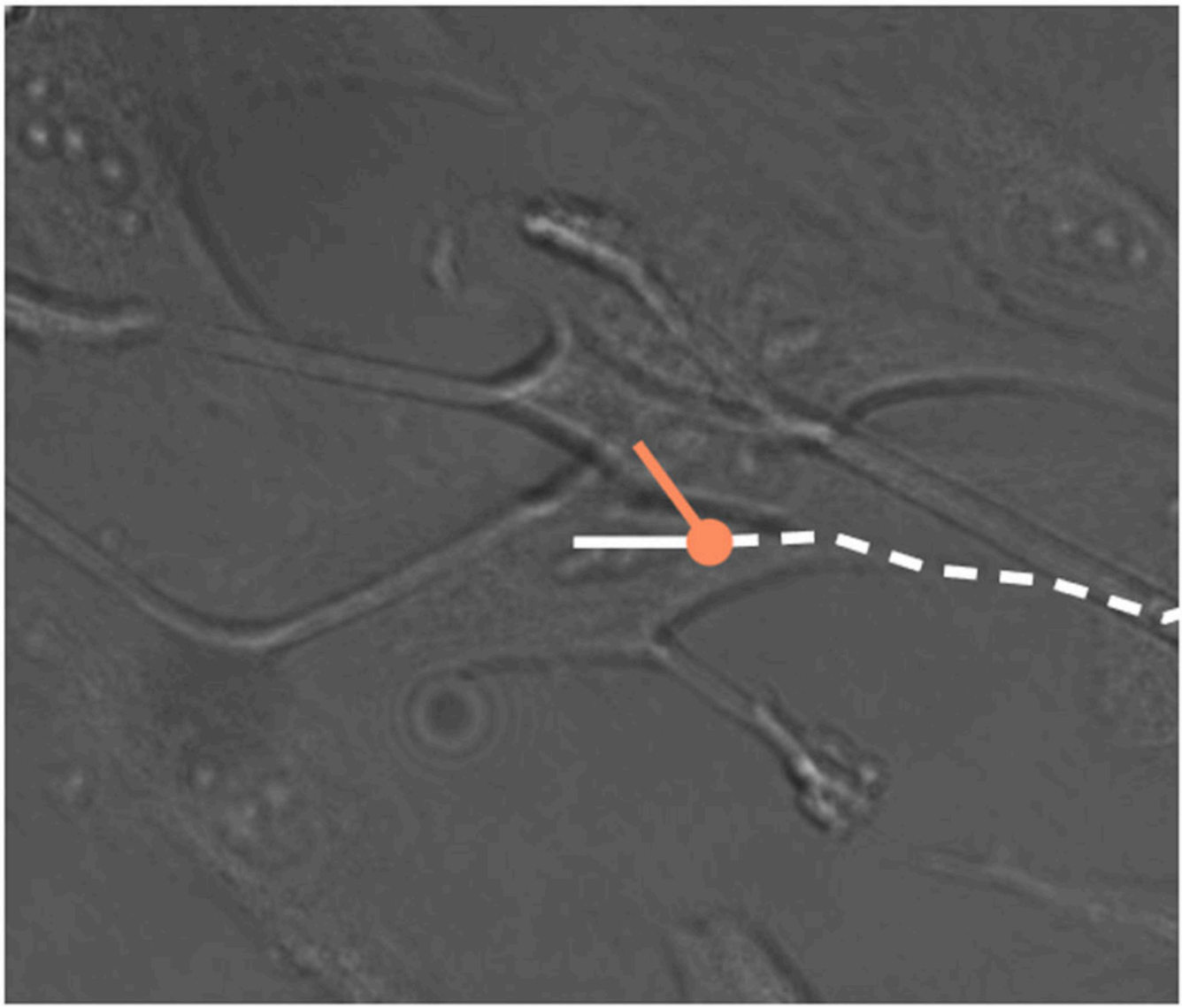


Figure 5:

Zoomed view of an error link. The orange line indicates the error link committed by the CNN *algorithm. The white line indicates the path of the cell before (dashed line) and after (solid line) the error link.*