LivecellX: A Scalable Deep Learning Framework for Single-Cell Object-Oriented Analysis in Live-Cell Imaging

Ke Ni[1,2], Gaohan Yu[3], Zhiqian Zheng[3], Yong Lu[2], Dante Poe[1,2], Yanshuo Chen[8], Mia Sanborn[7], Zilin Wang[2], Shiman Zhou[2], Xueying Zhan[7], Weikang Wang[4,5*†], Jianhua Xing[2,3,6*†]

[1*]Joint CMU-Pitt Ph.D. Program in Computational Biology, University of Pittsburgh, Pittsburgh, 15232, PA, USA.
[2]Department of Computational and Systems Biology, University of Pittsburgh, Pittsburgh, 15232, PA, USA.
[3]Department of Physics and Astronomy, University of Pittsburgh, Pittsburgh, 15232, PA, USA.
[4]Institute of Theoretical Physics, Chinese Academy of Sciences, Beijing, 100190, China.
[5]School of Physical Sciences, University of Chinese Academy of Sciences, Beijing, 100049, China.
[6]UPMC-Hillman Cancer Center, University of Pittsburgh, Pittsburgh, 15232, PA, USA.
[7]Ray and Stephanie Lane Computational Biology Department, Carnegie Mellon University, Pittsburgh, 15213, PA, USA.
[8]Department of Computer Science, University of Maryland, College Park, 20742, MD, USA.

*Corresponding author(s). E-mail(s): wangwk@itp.ac.cn; xing1@pitt.edu;
†These authors contributed equally to this work.

Abstract

Quantitative analysis of single-cell dynamics in live-cell imaging is pivotal for understanding cellular heterogeneity, disease mechanisms, and drug responses. However, this analysis demands stringent accuracy in cell segmentation and

tracking. A single segmentation error can significantly impact trajectory analyses, leading to error cascades, despite recent advances that have improved segmentation precision. To tackle these challenges, we introduce LivecellX, a deep-learning-based, object-oriented framework designed for scalable analysis of live-cell dynamics. We have defined a new task: segmentation correction for both over-segmentation and under-segmentation errors, and developed innovative evaluation metrics and machine learning techniques to address this issue. Our work includes annotating a novel imaging dataset from two distinct microscope types and training a Corrective Segmentation Network (CS-Net). The network leverages normalized distance transforms and synthetic augmentation to rectify segmentation inaccuracies. We also propose trajectory-level correction algorithms that use temporal consistency and CS-Net to resolve errors at the trajectory level. After tracking, LivecellX facilitates biological process detection, diverse feature extraction, and lineage reconstruction across different datasets and imaging platforms. Its object-oriented architecture enables efficient data management and seamless integration across multiple datasets. Enhanced by Napari GUI support and parallelized computation, LivecellX offers a robust and extensible infrastructure for high-throughput single-cell imaging analysis, paving the way for future developments in live-cell foundation models.

Keywords: live-cell imaging, corrective segmentation network, lineage construction, single cell trajectory

1 Introduction

Live-cell imaging is a fundamental technique for studying the spatial and temporal dynamics of single cells. Compared to fixed imaging and single-cell sequencing, live-cell imaging offers the advantage of capturing the heterogeneity in cellular dynamics [1, 2]. However, there are several major challenges in processing high-throughput live-cell imaging data. The first challenge is extracting single-cell trajectories from the massive datasets generated by live-cell imaging microscopes. Analyzing single-cell dynamics requires both segmentation and tracking. To obtain various types of quantitative information, such as cell morphology and protein distribution across individual cells, accurate single-cell segmentation is crucial, but it remains one of the most challenging steps [3]. With the rise of deep learning methods, particularly deep convolutional neural networks, has made it feasible to segment millions of single cells from large live-cell imaging data [4–8]. Nonetheless, no segmentation methods can achieve perfect accuracy, especially when training data is limited for label-free live-cell datasets on various cell lines imaged with different types of microscopes. Segmentation errors, such as over-segmentation (o-seg) and under-segmentation (u-seg), are significantly amplified during processing long-term temporal data. For instance, consider a simplified scenario in which if cells are tracked over 100 frames with a segmentation accuracy of 98%, the probability of maintaining a trajectory without any segmentation errors drops to approximately $0.98^{100} \approx 0.13$. Long trajectories and high cell-density

datasets make human visual check and manual correction for millions of cells impractical. Therefore, identifying and correcting improperly segmented masks is essential [9]. The challenge is further aggravated by cell cycle. Lineage tracing is a critical task in analyzing long-term live-cell imaging data [10], but cell proliferation significantly complicates segmentation and tracking [11]. First, the appearance of cells undergoing division differs markedly from regular cells, complicating the training of deep learning models for segmentation. Second, cell division increases cell confluence, further complicating segmentation. Third, cell division adds complexity to tracking. For example, tracking algorithms often rely on a score function based on displacement or overlap between cells in consecutive frames and solve the linear assignment problem to link the cells [12]. The generation of new cell instances from divisions can create undesired linkages.

The next bottleneck in processing live-cell imaging data is the extraction of high-dimensional features. It is possible to extract hundreds of features from snapshot experimental data using labeling. However, extracting high-dimensional features from live-cell imaging data poses significant challenges due to the technical limitations of fluorescence labeling. First, the number of fluorescence channels is limited by the capabilities of fluorescent microscopy and labeling techniques. Secondly, fluorescence labeling can induce significant photo-toxicity in cells during long-term live-cell imaging [10]. In contrast, transmitted light imaging maintains cells in a healthier physical state, significantly enhancing the reliability and repeatability of experiments, particularly for long-term live-cell imaging studies. To overcome the limitations of fluorescence microscopy in live-cell imaging, various methods have been developed. These include deep learning approaches [13, 14] and the extraction of morphological or textural features [15–17] from transmitted light images, enabling the extraction of multi-dimensional features without relying on fluorescence.

Increasingly, researchers recognize the importance of extracting quantitative single-cell information from live-cell imaging data. While several software solutions exist for single-cell tracking and trajectory analysis, none fully address the associated challenges simultaneously [11, 18–21]. For instance, most of these tools are designed for cells with stained nuclei [22], relying primarily on manually curated feature-based methods, with only a few incorporating deep learning techniques [11, 23].

In this study, we propose a comprehensive workflow for analyzing single-cell trajectories from time-lapse imaging datasets. The contribution of LivecellX is threefold. First, we frame a new task specifically designed for live-cell imaging to resolve under-segmentation (u-seg) and over-segmentation (o-seg) errors. We develop a deep-learning-based method that effectively addresses these issues, incorporating novel techniques such as the focus mechanism, EDT normalization, and synthetic data sampling. An annotated dataset was collected from our laboratory for the segmentation correction task. We provide the community with an annotation module to generate training data for weakly-supervised learning efficiently from microscopy live-cell imaging data and report the performance of the effectiveness of our protocol. Second, we design the LivecellX architecture, transforming the analysis task from an image-matrix-oriented processing paradigm to a single-cell object-oriented paradigm.

Finally, we integrate LivecellX with the Napari graphical user interface, allowing users to interactively analyze live-cell data objects.

2 Results

2.1 An Overview of the LivecellX architecture and workflow

To address the live-cell analysis challenges, we introduce LivecellX, a comprehensive framework for live-cell imaging data analysis (Fig. 1). LivecellX provides an integrated solution for segmentation, tracking, dynamics analysis by adopting a single-cell, object-oriented architecture. Our architecture not only enhances segmentation and tracking accuracy, but also simplifies the extraction of trajectory dynamics, making it easier for users to analyze complex biological processes.

Besides standard segmentation and tracking procedures (Fig. 1a), we incorporate a specifically-designed module to correct errors in temporal imaging datasets (Fig. 1b). Central to the LivecellX correction module is the corrective segmentation network (CS-Net), a context-aware, multi-scale machine learning architecture designed to reduce segmentation inaccuracies automatically. The CS-Net models has been trained on both publicly available datasets and live single-cell datasets acquired and annotated in our laboratory provided in this study (Fig. 1b). It identifies and rectifies segmentation errors by analyzing irregular overlap relationships between single-cell masks across consecutive frames (Fig. 1b).

For the cell tracking stage, we developed an extensible module with a well-defined programming interface, enabling easy integration of multiple tracking algorithms. In our implementation, we incorporated the SORT tracker [24] and b-track [25, 26]. To address post-tracking trajectory-level errors, we developed trajectory-level algorithms to improve tracking quality using the CS-Net and temporal information. These algorithms significantly reduce the number of required CS-Net inferences compared to applying CS-Net to every single cell. The corrected single-cell trajectories are updated within the LivecellX data structures for downstream analysis.

In LivecellX, we also implemented a biological process detection module that can identify important biological processes, such as cell mitosis, automatically or manually (Fig. 1c). LivecellX incorporates various feature extraction functions (Fig. 1d) and parallelizes the computation process efficiently. For each segmented cell, the features are extracted and stored in single-cell objects. Within this module, we implement several widely-used feature extractors, including morphology features and texture features. The high-dimensional features are utilized in the dynamics analyses such as trajectory embedding and manifold mapping (Fig. 1e).

To ensure robustness and user accessibility, we developed an asynchronous graphical user interface (GUI) based on Napari, allowing seamless interaction with the data visually via a graphical user interface and via coding at any stage of the analysis.

We demonstrate our framework's capabilities through analyses of live-cell imaging data obtained from different types of microscope: differential interference contrast(DIC) images acquired with the Nikon Ti2-E microscope, and hologram images taken by a Nanolive CX-A tomography microscope. By combining automated methods with interactive GUI, LivecellX provides a comprehensive solution for precise,

large-scale live-cell imaging analysis, empowering researchers to obtain more accurate biological insights.

2.2 The Corrective Segmentation Network revises label-free image segmentation results

Accurate segmentation is crucial in time-lapse imaging, as even minor errors can significantly impact downstream analyses. For instance, u-seg can lead to misidentification during cell tracking, resulting in incorrect trajectory data in subsequent frames. To address this issue, we propose the CS-Net to rectify o-seg and u-seg in single-cell analyses by leveraging temporal context (see Methods). The CS-Net takes both the incorrect single-cell mask and the corresponding region of interest (ROI) in the original images as input, and outputs the corrected mask (Fig. 2a). The CS-Net data processing pipeline converts both the incorrect input masks and the ground truth masks into Euclidean distance transform (EDT) masks [27]. Post-processing procedures, such as the watershed algorithm [28], are applied to the EDT maps output by the model. Additionally, a classifier is trained jointly to categorize the type of segmentation error (Fig. 2a).

Correct single-cell masks in consecutive trajectory frames generally show a high Intersection over Union (IoU) with one another. Conversely, segmentation errors lead to significantly reduced overlap (Fig. S1). Based on this observation, we propose a pipeline to gather training data by classifying temporal consistencies (see supplement for details). After segmentation is performed, our data preparation scripts detect events with inconsistent IoU values across consecutive frames. Those inconsistencies attributed to segmentation errors are then selected as candidates, and the correct masks are manually labeled.

In addition to regular over-segmentation (o-seg) and under-segmentation (u-seg) errors (Fig. 2b), we define a critical subtype called *o-seg dropout* (Fig. 2c, top), where only part of a single cell is segmented and other parts are missing. This error is difficult to correct during inference because the model lacks cues to associate neighboring fragments with the same cell. To address this, we augment the training and testing datasets with o-seg dropout samples, derived from o-seg training examples (see Section 5.1). These dropout cases are crucial for enabling CS-Net to correctly reconstruct full cells from fragmented masks during inference. Without them, CS-Net must infer cell boundaries solely by merging fragments, which is less effective and requires more negative training examples.

We prepared single-cell image crops at varying scales and trained the model on regions of interest (ROIs) sampled through our augmentation pipeline (Supplementary Fig. 1). This approach enables the CS-Net to effectively handle inputs of different sizes and spatial resolutions. Each training sample consists of an ROI centered on the error mask. The bounding box of the error mask is cropped and padded with different numbers of pixels to enhance training robustness. For u-seg, the bounding box surrounds the under-segmented cell mask. In the case of o-seg, all fragmented pieces of the cell mask should be included within the bounding box. If the bounding box does not encompass the entire cell, the ROI might not contain the ground truth mask, even with added padding. The training curves are shown in Fig. S2.

To improve semantic representation, we implemented a focus mechanism. The focus mechanism is implemented by a normalization technique that scales the EDT values to a range of 1 to 5, resulting in contour-map-style inputs and outputs. This normalization helps the CS-Net focus on pixels requiring correction (Fig. 2d). In contrast, the original EDT has a wide value range, which can complicate the training process and compromise numerical stability. By integrating this focus mechanism, our approach effectively guides the deep learning model to prioritize corrections on specific cell instances while leveraging both visual features and semantic advantages, ultimately improving segmentation accuracy.

To qualitatively demonstrate the effectiveness of CS-Net, we present representative examples of corrected over-segmentation (o-seg) and under-segmentation (u-seg) cases. The performance of CS-Net was evaluated using the following criteria:

- **Intersection over Union (IoU):** This metric quantifies the average overlap between the predicted cell masks and the ground-truth annotations across all samples. A higher IoU score indicates better agreement between predictions and the ground truth.
- **Root Mean Square Error of the Euclidean Distance Transform (EDT):** This metric measures the discrepancy between the predicted and ground-truth scaled EDT masks, providing a quantitative assessment of mask shape similarity.
- **Cell Match:** This criterion computes the percentage of ground-truth cells that are correctly matched to predicted cells, reflecting the accuracy of individual cell identification.
- **Correct Success Rate:** This metric evaluates the proportion of test samples in which all ground-truth cells are successfully matched with cells corrected by CS-Net. A sample is considered a "success" if every ground-truth cell has a corresponding match in the predicted mask.

Statisical analyses with these metrics revealed that the CS-Net significantly increased the segmentation quality. Our observations indicate that the CS-Net successfully corrected nearly all the o-seg cases. For u-seg, even though the IOU was not improved, the CS-Net still showed high performance on the other three metrics(Fig. 2h).

More specifically, we compared the performance of CS-Net with and without focus mechanism on the u-seg and correct cases (Fig. 2i). The CS-Net with focus mechanism showed larger proportion of cell match in both cases. Moreover, this improvement is robust under different values of padding. The accuracy of classifier is also higher in CS-Net with focus mechanism (Fig. 2j).

Additionally, we have implemented a manual correction module based on Napari[29] and LivecellX data structures. Besides o-seg and u-seg cases, cells probably cannot be detected during segmentation, resulting in missing cells. To address wrong segmentation cases including missing cell cases, our GUI includes a tool that allows users to draw a polygon and apply CS-Net to refine the manually drawn cell masks with one click (Supplement Fig. 2).

2.3 Trajectory-level correction algorithms accurately identify and correct segmentation errors impacting tracking

Most current cell tracking methods rely on object tracking algorithms from the computer vision domain. [30] However, these tracking algorithms do not address erroneous segmentation masks, which can lead to inaccurate tracking results. To address this issue, we offer post-tracking correction algorithms designed to rectify specific errors resulting from incorrect segmentation masks.

The typical distortions in linkages between single-cell masks in consecutive frames, caused by o-seg and u-seg, are as follows:

1. U-seg of two cells: When two cells that belong to trajectory A and trajectory B respectively are under-segmented and merged into a single mask, they are mistakenly assigned to the same trajectory, despite belonging to different ones. If the merged object is assigned to trajectory A, a third cell in this frame might be incorrectly assigned to trajectory B (Fig. 3a left). This can lead to a cascade of incorrect linkages in other trajectories, particularly those involving neighboring cells.

2. O-seg of a single cell: If a cell at time t in a trajectory is over-segmented into multiple fragments, but only one fragment is assigned to the trajectory, it results in an o-seg case. Unlike u-seg, o-seg and o-seg dropout do not necessarily cause direct tracking linkage errors most of the time, but can impact subsequent trajectory analysis. In rare cases where the segmentation model introduces frequent o-seg errors, the unassigned fragments may lead to incorrect linkages in the trajectories of neighboring cells (Fig. 3a, right).

We design algorithms based on CS-Net with temporal information to perform tracking corrections (Fig. 3b). By assessing temporal inconsistencies based on the IoU and Intersection over Minimum (IoMin) between consecutive frames of single-cell masks, we can identify candidate segmentation errors.

Typically, u-seg leads to missing frames or fragmented trajectories, whereas o-seg results in abrupt changes along a trajectory in feature space (Methods). The CS-Net is applied to these candidate inconsistent cases.

Comparing with o-seg, u-seg results more complicated tracking errors. We observed one common situation that the u-seg error lasts for several consecutive frames across a single cell trajectory. The consecutive errors may not be solved completely if applying the CS-Net only once. And the remaining u-seg errors still cause incorrect linkage in this trajectory even its neighboring trajectories. Hence we implement a multi-round procedure, with results presented in Fig. 3c. The trajectories undergo multiple iteration of error detection until no new errors can be identified.

In Fig. 3c, we present the cumulative counts of u-seg cases detected and corrected by LivecellX. It should be noted that relying solely on missing or broken trajectories to detect u-seg cases can lead to numerous false positives illustrated by the distribution of expert annotated TD-1 candidates in Fig. 3i. To handle these false positive cases, the CS-Net classifies all the candidates and generates segmentation masks, which will not be adopted if they conflict with trajectory inconsistency detection. After fifty iterations, both the attempted corrections and resolved cases reach a point of saturation.

The number of o-seg cases in our evaluation dataset is relatively low compared to u-seg cases. To address this imbalance, we synthesized 5,000 o-seg samples using the same protocol employed for generating synthetic o-seg training data. We found these synthetic o-seg cases to be significantly more challenging than real ones, often representing rare or edge cases. We evaluated our trajectory correction algorithms on both the synthetic and real o-seg datasets.

To identify initial o-seg candidates for correction, we used the change in cell area between consecutive time points—referred to as the TD-1 feature change in Fig. 3—as the selection criterion. These candidates were then passed to CS-Net for further processing (Fig. 3b). The ROC curve (Fig. 3d) shows that CS-Net, equipped with trajectory inconsistency detection, achieves an AUC of 97%, outperforming the 87% AUC achieved by using the TD-1 feature change alone.

Visual inspection confirmed that all real o-seg cases within trajectories were correctly identified and corrected by our algorithm. These results demonstrate that CS-Net effectively detects and corrects both u-seg and o-seg errors. Additionally, segmentation and tracking errors can lead to missing frames in cell trajectories. We define the proportion of such missing frames as the *trajectory vacancy rate*. Our analysis shows that this vacancy rate is substantially reduced after applying our trajectory-level correction method, particularly following multi-round correction (Fig. 3e).

We further evaluated the effectiveness of IoU and IoMin in identifying true u-seg candidates. Our analysis revealed that IoMin is more inclusive than IoU: all true positive u-seg candidates detected by IoU were also captured by IoMin (Fig. 5f, 5g). Next, our domain experts manually classified and annotated all TD-1 candidates identified by IoMin into five categories: u-seg, m-seg, o-seg, correct, and mitotic. The class distribution of these IoMin candidates is presented in Fig. 5h. We found that no fixed IoMin threshold could effectively distinguish true mitotic events from true u-seg instances. This limitation reveals the need for our CS-Net in trajectory-level correction and candidate classification.

2.4 LivecellX detects biological processes and constructs cell lineages

A key distinction between tracking in live cell imaging and other tracking tasks is cell cycle, which is crucial for analyzing single-cell dynamics and presents new challenges for tracking. To address these challenges, we implemented a lineage reconstruction module in livecellX. In cell cycle, a mother cell divides into two daughter cells during the mitotic phase. This phase is marked by rapid changes in the cell's appearance, which complicates segmentation and tracking. For example, in a 2D culture environment, mitotic cells exhibit distinct features in transmitted light images. Before dividing, the mother cell detaches from the substrate and contracts into a small round shape. Following this, the mother cell divides, creating two daughter cells, each also appearing as a small round shape. Subsequently, the daughter cells begin attaching to the substrate of the Petri dish and spreading out(supplement fig). More significantly, the birth and termination of trajectories exhibit distinct dynamics. The mother cell may be linked with neither initial frames of daughter cell trajectories or be linked with one daughter cells with the other daughter cell trajectory left (Fig. 4a). The aim of

this module is to identify mitosis and clarify the relation between mother and daughter cell trajectories and the relation between sister cell trajectories. Compared to relying solely on static snapshot data, employing temporal information greatly enhances the detection of mitosis and apoptosis.

In sum, cell mitosis show distinct hehaviors in cell appearance and dynamics. We can utilize this information to detect them. Here we empoloyed machine learning based detection methods to identify cells undergoing mitosis (Methods). In essence, this method frames the task of mitosis identification as a trajectory classification problem at the single-cell level based on the dynamics (Fig. 4b). We integrated two types of detection models: One simply relies on the morphological dynamics and the other one is based on deep learning (Methods). Given the critical role of the cell cycle in various biological processes, understanding cell lineage will help elucidate the influence of the cell cycle on these processes. Here, we present a single-cell lineage spanning three generations, captured by livecellX. Representative images at different time points reveal the heterogeneous dynamics of sister cells and cousin cells (Fig. 4c). As shown in this lineage, the cousin cells in the third generation exhibited different levels of motility and cell cycle durations.

2.5 LivecellX enables efficient trajectory dynamics analysis

With livecellX, multiple trajectories can be extracted from time-lapse imaging data. And we integrated different feature extractors following single cell segmentation. The characteristics of cell morphological features rend them as promising variables for quantifying cell dynamics. Cell morphology associates with cell state intimately and hundreds of features can be extracted from it. Furthermore, morphological features can be readily quantified in transmitted-light images, which does not require fluorescent labeling. There are various approaches to extracting these features. The simplest one is using the predefined features like area and major axis length (MAL). Other representations includes Fourier transform and Zernike polynomials. According to Pincus's report, the active shape model is among the most effective frameworks for representing single-cell morphology [31]. The active shape model, combined with PCA, provides an orthogonal basis set (PCA modes) for describing single-cell shapes (Fig. 1d). Using PCA modes, single-cell shapes can be reconstructed accurately. (Fig. 5e) Furthermore, these modes reflect the dynamic changes in single-cell shapes [2]. In our implementation, we have incorporated several distinct types of morphological features: the active shape model and tens of predefined morphology features such as area, eccentricity and perimeter. In Fig. 5g, we shown one long single cell trajectory in multi-dimensional space with its typical images (left). Specifically, we marked several time points with extreme values in area, MAL and eccentricity (right). Compared to traditional transmitted-light microscopy, holo-tomography microscopy provides more quantifiable features. For example, organelles, particularly the nucleus, can be clearly imaged. The features of these organelles offer different perspectives for describing cell states.

To derive more insights from label-free labeling, we have introduced various multi-dimensional features to quantify the distribution of fluorescence proteins across individual cells, based on single-cell segmentation. These features include Haralick

features and local binary patterns, which are widely used texture features in image analysis. The quantified texture features within a single cell not only encapsulate intensity information but also relate the spatial distribution of fluorescence proteins to cell states. We integrated these texture features in livecellX and used them to represent single cell trajectories (Fig. 5).

Beyond filter-based features from the traditional computer vision domain, we have also incorporated feature extractors based on machine learning methods such as autoencoders (AE) and variational autoencoders (VAE). The latent embeddings derived from the VAE can serve as features for individual cells. For example, the Harmony method utilizes these embeddings to generate original single-cell images (Fig. 5j). Alternatively, a VAE can be employed to generate single-cell images (Fig. 5k, right), with its latent embeddings capturing different features compared to those obtained through the Harmony method.

Dynamic manifold can be reconstructed in a topologically equivalent manner using time-delay embedding, which has been proved by Takens' theorem [32]. As illustrated in Fig. 5c, the time-delay embedding of a cell cycle trajectory reveals a circle-like structure in the embedded space. To quantify the similarity between multi-dimensional single-cell trajectories, dynamic time warping (DTW) analysis is also included in this pipeline (Fig. 5h). DTW can compute the optimal match between trajectories occurring at different speeds. The DTW distance matrix illustrates the heterogeneity of single-cell trajectories throughout the cell cycle (Fig. 5k,l). Common characteristics of these heterogeneous single-cell trajectories can be revealed through barycenter averaging based on DTW distances (Fig. 5k, left).

2.6 LivecellX is a portable single-cell object-oriented framework

Beyond these software-specific discrepancies, a broader issue in live-cell imaging analysis is the lack of a high-level standardized data structure that integrates heterogeneous imaging formats and conceptually represents cells and cell trajectories . Current workflows typically adopt an image-oriented paradigm, which requires loading matrices from disk and iterating multiple nested loops through frames to extract and analyze single-cell dynamics. This approach often results in redundant computations and cluttered code, particularly when handling data with multiple channels and organelles (Fig. 6a). Moreover, the reliance on matrix-based operations makes it challenging to track and manipulate individual cell objects across time points, limiting the efficiency and scalability of downstream analyses.

To address these issues, we developed an object-oriented data structure in LivecellX that frees users from technical details by organizing data around biologically meaningful entities rather than raw image matrices. Instead of iterating over pixel-based representations, LivecellX constructs single-cell objects post-segmentation, assembles cell trajectories post-tracking, and integrates these trajectories into a hierarchical framework that enables efficient group operations (Fig.6b). This design provides an intuitive and scalable approach to managing live-cell imaging data. More specifically, this single-cell object-oriented framework consists of three hierarchical levels. The first level is the *single-cell object*, which encapsulates various information within a cell's

scope, including organelles, cell features and meta information. To ensure broad compatibility, LivecellX offers a unified interface for loading segmentation results from various sources, including Cellpose, StarDist, and Detectron [8, 33, 34]. The second level is the *single-cell trajectory*, which links all single-cell objects belonging to the same tracked cell across frames, along with metadata such as velocity, morphology changes, and lineage relationships. Each trajectory object retains various information, such as the mother cell trajectory, daughter cell trajectories, and sister cell trajectories, to facilitate automated lineage tracing. The third level is the *single-cell trajectory collection*, which organizes all trajectories within an experiment, stores global metadata, and enables large-scale statistical trajectory-level analyses. Single-cell trajectory collection also make analyses across datasets possible. Notably, each single-cell object can contain associated organelle objects. To support this, both organelle and single-cell classes are derived from a shared base class, reflecting their data structural and functional similarities. This unified design ensures flexibility and extensibility when integrating subcellular features into LivecellX single-cell.

Another characteristic of live-cell imaging analysis is its reliance on a complex interplay of diverse image processing libraries, each with its own data structures and conventions. This introduces inefficiencies when dealing with multi-channel images, organelle tracking, and long-term time-lapse data (Fig. 6). OpenCV and scikit-image (`skimage`) follow different conventions for handling image processing tasks, particularly with bounding boxes. OpenCV uses a `(x, y, width, height)` format, where `(x, y)` represents the top-left corner, and width and height define the size of the bounding box. In contrast, `skimage` typically follows the `(min_row, min_col, max_row, max_col)` convention, where coordinates specify the top-left and bottom-right corners of the region. Additionally, OpenCV often works with BGR color ordering by default, whereas `skimage` assumes RGB. Similar subtle assumption differences can lead to potential bugs frequently. LivecellX resolves those issues by standardizing the conventions in the object-oriented framework and integrating the libraries to minimize glitches caused by underlying packages.

By shifting from an image-centric to a cell-centric paradigm, LivecellX provides a flexible and efficient framework for analyzing single-cell dynamics, reducing redundant computations and improving data interpretability. This object-oriented approach ensures that live-cell imaging analysis is both scalable and adaptable to diverse experimental conditions, paving the way for more advanced downstream analyses and enabling large-scale, multi-dataset integration.

2.7 LivecellX provides a GUI utility built on object-oriented framework

To enhance usability and streamline interactive analysis, LivecellX includes a graphical user interface (GUI) tightly integrated with its object-oriented data structures. As illustrated in Fig. 6c, the GUI features a Napari-based visualization window for displaying raw microscopy images alongside derived masks and annotations. It offers interactive controls for key operations such as segmentation correction, tracking refinement, manual annotation, and lineage tracing.

The GUI is fully programmatically accessible—users can launch or close the interface from within Python scripts or Jupyter notebooks without interrupting code execution. This design allows users to inspect and modify intermediate results dynamically during pipeline execution. Built directly on top of the LivecellX object system, the GUI leverages the underlying single-cell and trajectory objects to ensure consistency and interoperability with all other modules.

In practice, if a user identifies an error in a specific trajectory (e.g., due to segmentation drift or mitotic mislabeling), the GUI can be used for both verification and correction. Several types of trajectory-level operations supported in LivecellX are summarized in Fig. 6d.

2.8 LivecellX data structure encapsulates I/O operations and reduce repetitive temporal data retrieval

LivecellX abstracts input and output handling through a dynamic dataset class, enabling on-demand retrieval of large-scale imaging data. To optimize efficiency, the dataset class implements a least recently used (LRU) cache, reducing redundant disk access when processing the same or adjacent time-frame images.

Another significant challenge is computational complexity issues. Analyzing thousands of cells on a single core can take hours to days. To alleviate this, LivecellX implements modular and extensible operations that significantly speed up the analysis process. By parallelizing feature computation across multiple cores and computing nodes, we achieve a linear speedup. LivecellX facilitates large-scale analysis of cells from multiple experiments and sources.

3 Discussion

In this study, we introduced a comprehensive pipeline for analyzing single-cell dynamics from time-lapse imaging data. LivecellX offers a single-cell, object-oriented framework that facilitates cell segmentation, movement tracking, feature extraction, and trajectory analysis. Our aim is to free researchers from the cumbersome technical details and error-prone coding structures that often accompany these tasks.

We tackled several significant challenges commonly encountered in this field, including inaccurate segmentation, tracking errors, and lineage reconstruction. By leveraging advances in deep learning, we have developed specialized functional modules to effectively address these issues. By using the information of inaccurate mask as focus, the CS-Net can correct over-segmentation and under-segmentation errors, thereby enhancing segmentation quality. Furthermore, we developed algorithms to rectify tracking errors following the correction of segmentation issues. Additionally, we implemented a transformer-based module that utilizes both temporal and spatial information to accurately identify mitotic cells.The integration of these modules significantly improves the accuracy and reliability of extracted single-cell trajectories. This pipeline offers substantial benefits for research into the dynamics of single cells, providing a robust toolset for analyzing live-cell imaging data.

However, it's crucial to recognize that algorithms in the field are not capable of perfectly addressing every challenge encountered in the analysis of time-lapse imaging

data. There are certain scenarios where manual intervention is necessary. To accommodate this need, we have developed a graphical user interface (GUI) that allows researchers to manually rectify segmentation errors or other issues that the algorithm is unable to resolve. For instance, if a segmentation error leads to incorrect trajectory linkages, researchers can utilize the GUI to make manual corrections.

The application of machine learning was pivotal in attaining accurate results. However, supervised learning approaches heavily rely on training data, and the process of labeling this data can be labor-intensive. To alleviate this burden, we have employed several strategies to minimize manual labeling efforts. We utilized publicly available cell image datasets to pre-train the segmentation neural network, significantly reducing the need for extensive manual labeling. Additionally, we developed semi-automatic algorithms to assist in labeling incorrect segmentations as well as mitosis and apoptosis events.

Time-lapse imaging is an essential technique for studying cell dynamics, offering insights that are not accessible through snapshot data. However, challenges related to fluorescence labeling and imaging can limit the extraction of detailed information from time-lapse data. Our pipeline addresses this limitation by integrating modules that extract multidimensional features from both transmitted light and fluorescence images. We acknowledge, though, that this approach still provides a constrained view of the high-dimensional dynamics of cells. Fortunately, advancements in deep learning have opened new possibilities for extracting additional information from transmitted light images. For instance, researchers have successfully trained neural networks to identify organelles within these images. In future iterations of LivecellX, we aim to incorporate more organelle features to further enhance its functionality, thereby offering a more comprehensive analysis of cellular dynamics.

4 Figures

5 Methods

5.1 Corrective segmentation network

Initially, the original segmentation masks for each frame are generated. The overlap relationships between single-cell masks in consecutive frames are then calculated. If a cell mask M_i^t in frame t overlaps with two cell masks M_j^{t+1} and M_k^{t+1} in frame $t+1$, this suggests a potential o-seg in frame $t+1$ or u-seg in frame t. Similarly, if a cell mask M_i^{t+1} in frame t+1 overlaps with two cell masks M_j^t and M_k^t in frame t, it indicates potential o-seg in frame (t) or u-seg in frame (t+1). This method filters out candidate segmentation errors by identifying temporal inconsistencies.

The ROI containing the potentially incorrect segmentation mask, along with its neighboring area, is cropped from the original image and the corresponding mask. Unlike typical segmentation tasks, the aim of CS-Net is to correct these original segmentation masks. As the pre-trained weights of CS-Net were derived from computer vision datasets, the model expects input data formatted with three channels. Consequently, the input tensor shapes are defined as 3 x H x W, where H and W represent the height and width dimensions, respectively. The first channel in the inputs contains

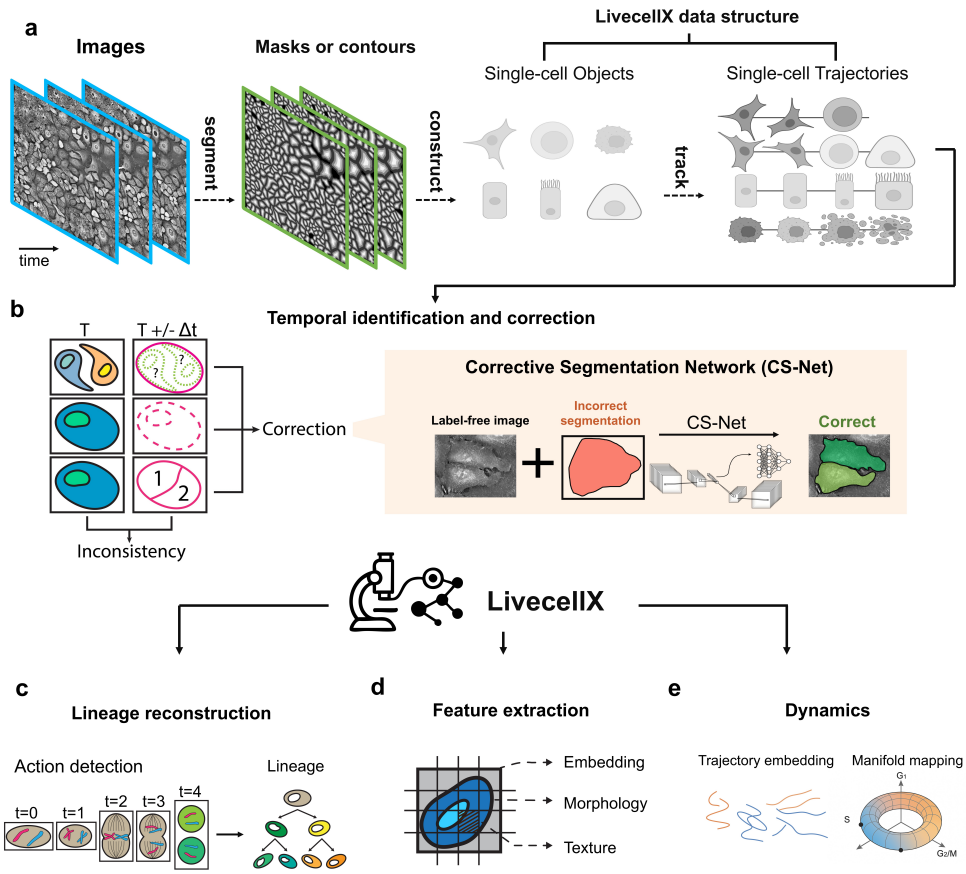

Fig. 1 Overview of the livecellX pipeline for analyzing single-cell dynamics in live-cell imaging data. (a) This comprehensive pipeline encompasses single-cell segmentation, tracking, detection of biological processes, and trajectory analysis. Instead of directly working with matrix data, the LivecellX framework utilizes specifically designed single-cell and trajectory class objects for enhanced analysis. (b) O-seg and u-seg can be detected by utilizing the inconsistency of cell masks in consecutive frames and corrected by CS-Net which takes both the original single cell images and incorrect cell masks as input. (c) LivecellX is integrated with a lineage reconstruction module that can label single-cell trajectories as mother and daughter trajectories based on mitosis detection, facilitating dynamics analyses. (e) LivecellX supports various types of single-cell feature extractors. (f) LivecellX implemented multiple methods for analyzing cell dynamics.

the raw images, serving as the primary source of information for our baseline model. The second channel features either a binary mask, a label mask, an Euclidean distance transform (EDT) mask, or a normalized EDT mask, as highlighted in the focus mechanism section5.1.2. The third channel is a duplication of the second channel.

CS-Net generates three distinct masks as predictions: a segmentation mask, an u-seg mask, and an o-seg mask. The segmentation mask provides a continuous probability distribution across the image, indicating the likelihood of each pixel belonging

to the target segmentation class. The u-seg mask identifies regions where the segmentation boundaries extend beyond the actual object boundaries, effectively serving as a probability map of pixels that should be removed. Conversely, the o-seg mask highlights areas where the model fails to capture fine details or boundaries within objects, acting as a probability map for the inclusion of pixels.

network structure?

5.1.1 Corrective segmentation evaluation metrics

We define **cell match rate** as the percentage of predicted masks that accurately correspond to ground-truth cells. Mathematically, let N represent the number of samples. For the i-th sample, let M_i denote the total number of ground-truth cells, and let $\hat{M}_i$ represent the number of predicted masks that correctly match a ground-truth cell based on a defined matching criterion. In our scenario, this criterion is IoU is greater than 0.5. The cell match rate is then defined as:

$$\text{Cell Match Rate} = \frac{1}{N} \sum_{i=1}^{N} \frac{\hat{M}_i}{M_i}$$

The **correction success rate** we introduced for the segmentation correction task is more stringent than the cell match rate. This metric requires that the predicted cells and ground-truth cells be *bijectively matched*, meaning there must be a one-to-one correspondence between predicted masks and ground-truth masks for a given sample.

Each pair must satisfy a predefined matching criterion, which in our case is IoU is greater than 0.5, consistent with the criterion used in the cell match rate. A sample is considered successfully corrected only if every ground-truth cell is accurately matched to a predicted cell, and vice versa.

For the i-th sample, let G_i represent the set of ground-truth masks, and P_i denote the set of predicted masks. Let $\mathcal{M}_i \subseteq G_i \times P_i$ be the set of matched pairs that satisfy the matching criterion.

We define a successful correction if $|\mathcal{M}_i| = |G_i| = |P_i|$, i.e., a perfect one-to-one matching exists.

Then, the correction success rate is given by:

$$\text{Correction Success Rate} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\left[|\mathcal{M}_i| = |G_i| = |P_i|\right]$$

where $\mathbb{1}[\cdot]$ is the indicator function, equal to 1 if the condition is true and 0 otherwise.

Root mean square error (RMSE) is a metric to quantify the difference between the predicted mask and the ground-truth mask. For a predicted mask M_{predict} and ground-truth mask M_{GT}, both represented as images of the same dimensions, RMSE is defined as:

$$\text{RMSE}(M_{\text{predict}}, M_{\text{GT}}) = \sqrt{\frac{1}{|M|} \sum_{x \in M} \left(M_{\text{predict}}(x) - M_{\text{GT}}(x)\right)^2}$$

where $|M|$ is the total number of pixels in the mask, and x indexes over all pixels. A lower RMSE signifies better alignment between predicted and ground-truth masks. RMSE is appropriate for evaluating both binary masks and EDT masks. However, when dealing with label masks, caution is advised, as the ground-truth and predicted labels may not align due to potential differences in label assignment.

5.1.2 Focus mechanism

To enhance the semantic awareness of the correction task, we introduce a *focus mechanism* that guides deep learning models to concentrate on a specific cell during a single forward pass. This mechanism is implemented by incorporating a dedicated focus channel, which supplies the model with the target cell mask,

Formally, given an input image tensor $X \in \mathbb{R}^{H \times W \times C}$ and a focus mask $F \in \mathbb{R}^{H \times W \times 1}$, we construct the focused input $\tilde{X}$ by concatenating F to X along the channel dimension:

$$\tilde{X} = \text{concat}(X, F),$$

where $\text{concat}(\cdot)$ denotes the channel-wise concatenation operation. The resulting tensor $\tilde{X} \in \mathbb{R}^{H \times W \times (C+1)}$ includes both the original image features and an additional semantic cue, thereby facilitating improved learning and inference.

Focus mask types

Different choices for F impact the effectiveness of the focus mechanism. We examine two primary strategies:

Binary Focus Masks A binary mask F highlights the target cell by setting the relevant pixels to 1 while marking all others as 0:

$$F_{i,j} = \begin{cases} 1, & \text{if } (i,j) \in \mathcal{C}_{\text{target}}, \\ 0, & \text{otherwise.} \end{cases}$$

Here, $\mathcal{C}_{\text{target}}$ denotes the region occupied by the target cell. One issue with this method is that the focus input mask for over-segmentation (o-seg) cases may be indistinguishable from correct masks, particularly in scenarios where two o-segmented pieces of a cell are joined together. A straightforward approach is to use label masks, where each integer label represents a distinct cell, instead of binary masks. However, since these label values lack semantic meaning, they can lead to training instability. In such instances, neither the binary focus mask nor the label focus mask proves effective. Inspired by recent advances in the field [6, 8], we explore the potential of EDTmaps as F to resolve these issues.

Euclidean Distance Transform Maps EDT map F encodes the spatial relationship between pixels and the cell boundary:

$$F_{i,j} = \begin{cases} d_{i,j}, & \text{if } (i,j) \in \mathcal{C}_{\text{target}}, \\ 0, & \text{otherwise.} \end{cases}$$

where $d_{i,j}$ represents the Euclidean distance of pixel (i,j) to the closest boundary of the target cell. EDT maps offer a continuous gradient of focus intensity, helping

to mitigate the instability issues often seen with label-based masks. Additionally, to prevent numerical and semantic problems arising from the variation in cell sizes, as illustrated in Fig. 2, we implement a normalization strategy to confine the EDT values of a cell within the following range:

$d_{i,j}$ is normalized by first computing the scaling factor as

$$\text{factor} = \frac{\max(d_{i,j})}{\text{edt}_{\max} - 1}$$

and then transforming all nonzero values ($d_{i,j} \geq 1$) using

$$\tilde{E}_{i,j} = \frac{d_{i,j}}{\text{factor}} + 1.$$

This approach ensures that the normalized EDT values are scaled relative to the maximum distance, with a minimum value of 1 for positive distances. This scaling preserves numerical stability and retains semantic meaning for both the target cell and the background regions.

5.1.3 Training and testing setup

Loss functions

For model training, we use Binary Cross Entropy (BCE) as loss function, specifically tailored to our requirements. Because each pixel may belong to multiple channels, standard cross-entropy across all classes is inappropriate. Therefore, we also compute the Mean Squared Error (MSE) alongside BCE, incorporating Euclidean distance transformation to facilitate comparative evaluation.

During ablation tests, we explored various strategies for augmenting the other two channels. One approach involved removing backgrounds from raw images to create background-free representations of single cells in one of both channels. Additionally, we examined the effectiveness of incorporating segmentation masks and single-cell masks as two additional channels. This methodology enables us to assess the impact of these extra channels on the model's performance.

5.1.4 Choosing padding sizes for inference

Typically, a u-seg sample includes all pixels from two or more cells, whereas o-seg and o-seg dropout cases might lack parts of a cell. Since the CS-Net does not have access to the ground-truth category of an incorrect single cell mask during inference, selecting an appropriate padding size for the cropped ROI is crucial. To ensure the model gets sufficient contextual information to accurately correct all samples, we pad the inputs with an additional 50 pixels around the single cell mask based on scale of single cell in pixel.

5.2 Experimental data collection and livecell imaging setup

5.2.1 Cell Culture

MCF10A cells: Human breast epithelial MCF10A cells (ATCC, CRL-10317) were cultured in DMEM/F-12 medium supplemented with 5% horse serum, 20 ng/mL epidermal growth factor (EGF), 0.01 mg/mL insulin, 500 ng/mL hydrocortisone, and 100 ng/mL cholera toxin. Cells were maintained at 37 °C in a humidified incubator with 5% CO_2 and used within 10 passages. For live-cell imaging with the Nanolive CX-A microscope, 500–5,000 cells were seeded into 35 mm glass-bottom imaging dishes (Ibidi, 80137) two days prior to imaging. Label-free images were acquired every 5 minutes, and TRITC channel images (Ex 555 nm / Em 587 nm) every 30 minutes using a 60× objective (N.A. = 1.45). Imaging was performed over 57 hours, resulting in 684 frames. For experiments lasting over 48 hours, 300 µL of fresh medium was added every 48 hours.

A549 cells: Human lung adenocarcinoma A549 cells (ATCC, CCL-185) were cultured in F-12K medium supplemented with 10% fetal bovine serum (FBS) and maintained under the same incubation conditions (37 °C, 5% CO_2). Cells within 10 passages were used. Imaging was conducted for 27 hours at 5-minute intervals, yielding 323 frames.

A549/VIM-RFP cells: A549 cells stably expressing Vimentin-RFP (ATCC, CCL-185EMT) were cultured in F-12K medium (Corning) supplemented with 10% FBS. Cells were seeded into 35 mm glass-bottom culture dishes (Cellvis D35-20-1.5-N) and maintained at 37 °C in a humidified atmosphere with 5% CO_2. Cells from passages 3–10 were used.

For live-cell imaging with the Nikon Ti2-E microscope, 5,000–30,000 cells were plated two days before imaging. Differential interference contrast (DIC) images were acquired every 5 minutes, and TRITC channel images (Ex 555 nm / Em 587 nm) every 30 minutes using a 20× objective (N.A. = 0.75). Imaging was performed for 24–120 hours. For time courses exceeding 48 hours, 500 µL of fresh culture medium was added every 48 hours.

Raw microscopy data and processed datasets for all cell types are available at the following locations:

5.3 Initial segmentation and preprocessing

We used the pretrained `cyto2` model from Cellpose [8] as the initial segmentation backbone. To adapt the model to our specific imaging conditions, we employed Cellpose's human-in-the-loop training pipeline for fine-tuning. A total of 15 time frames from A549 cells and 35 time frames from MCF10A cells were manually annotated to create a small training set. This fine-tuned model was then applied across all video frames to generate segmentation masks.

Following inference, we post-processed the predicted masks using OpenCV. Contours were extracted with `cv2.findContours` using the parameters `mode=cv2.RETR_EXTERNAL` and `method=cv2.CHAIN_APPROX_SIMPLE`. In cases where multiple contours were detected for a single object, only the largest-area contour was retained to mitigate over-segmentation artifacts.

5.3.1 Dataset annotation

We annotated the dataset and generated training samples, which include instances of both o-seg and u-seg. We mapped single objects from the original segmentation masks to their corresponding objects in the ground truth, as defined by human experts. If one cell object maps to multiple ground-truth objects, it is categorized as a u-seg sample. Conversely, if multiple cell objects map to a single ground-truth cell object, it is identified as an o-seg case.

5.4 Trajectory correction algorithms

When the majority of cells are accurately segmented, applying CS-Net to verify and correct every segmented cell is unnecessary, especially given the potential processing of millions of single-cell objects. Instead, we utilize spatial-temporal information from live cell tracking to refine single-cell trajectories, a method termed "trajectory inconsistency detection." O-seg typically causes abrupt fluctuations in the IoU along single-cell trajectories, often accompanied by an extra mask at the current frame or a short trajectory consisting of a partial mask that persists over several frames. Errors resulting from o-seg are identified and corrected using this approach.

Addressing u-seg is more challenging than dealing with o-seg, as u-seg involves multiple objects and often leads to cascading tracking errors among neighboring trajectories. The main types of errors include the omission of specific frames in adjacent single-cell trajectories and the premature termination of trajectories before the experiments finish. As a result, u-seg cell masks can be identified by detecting these errors. Understanding the nature of these errors is also crucial for accurately mapping the corrected masks back into the single-cell trajectories.

To filter out missing and early-ending events, we use IoU and IoMin metrics between single-cell masks across consecutive frames. The corrected masks are then integrated into single-cell trajectories based on the identified error types. For instance, if an under-segmented cell mask at frame t is corrected into two distinct cell masks, and one of these can be mapped to a neighboring single-cell trajectory that is missing a frame at t, then trajectory inconsistency detection remains self-consistent. This approach increases the likelihood of maintaining accurate trajectories.

If the results from CS-Net are not consistent with the spatial-temporal information derived from the local imaging data, the proposed corrections should not be accepted. Additionally, some errors may persist across multiple frames. To address this, we have implemented an iterative strategy that allows both the CS-Net and the trajectory correction logic to run through multiple iterations until no further corrections are possible.

5.4.1 Trajectory correction evaluation metrics

We introduce **trajectory vacancy** as a straightforward yet informative metric for evaluating the quality of a single-cell trajectory. For a given trajectory, the vacancy rate quantifies the extent to which it is fragmented over time. Specifically, we define the trajectory vacancy rate as follows::

$$\text{Trajectory Vacancy Rate} = 1 - \frac{\hat{n}}{t_{\max} - t_{\min} + 1}$$

where $t_{\max}$ and $t_{\min}$ are the maximum and minimum time frames of the cells in the trajectory after tracking, and $\hat{n}$ is the number of tracked cells in the trajectory. A lower vacancy rate indicates a more complete and continuous trajectory.

5.5 Biological process detection module and lineage reconstruction

LivecellX supports the application of machine learning models to classify cell trajectories based on temporal features and time-lapse image sequences. This task is framed as a multiclass classification problem, where the objective is to assign each single-cell trajectory to a specific biological process class. LivecellX is compatible with a wide array of machine learning frameworks, enabling users to apply various methods. We demonstrate this capability using SciPy, MMDetection, and PyTorch as representative tools.[35, 36]. We offer pre-trained transformer-based and CNN-based deep-learning models tailored for DIC and holotomographic datasets, along with utility functions to seamlessly apply these models to our data structures. LivecellX outputs temporal features, images, and videos that are specifically formatted for direct use in machine learning tasks. Users can easily extract these inputs from LivecellX data structures, such as single-cell objects and trajectory objects, with minimal code. The detection module is designed to be extensible, enabling users to integrate custom methods for feature extraction through LivecellX data structures.

Lineage reconstruction in LivecellX is carried out after detecting biological processes. The framework provides utility functions to split or merge trajectories based on common cellular events, such as mitosis and cell fusion. In the case of mitosis, the mother and daughter trajectories are separated at the time point of division. For cell fusion, two individual cell trajectories are combined into a single trajectory at the fusion time point. The relationships between cell trajectories are stored within the LivecellX data structures. Users have the flexibility to define new types of relationships either programmatically or through the graphical user interface. The trajectory object's input also encompasses information about neighboring cells and the background. To ensure that the Timesformer model focuses on potential mitosis events, the normalized EDT mask of each cell along the trajectory is included as a focus channel. This additional channel significantly enhances the efficiency of both the correction and detection processes.

5.6 Feature extraction and trajectory dynamics

We offer a dedicated module for feature extraction, including the active shape model, Haralick features, local binary pattern, variational auto-encoder, Harmony [37], and diffusion models [38, 39]. LivecellX integrates the active shape model implemented in Pincus's Celltool[31]. Haralick features and local binary patterns are extracted using the Mahotas package[40]. The variational auto-encoder is implemented through Harmony. The time delay embedding and DTW distances are computed by tslearn [41].

6 Code Availability

The `LivecellX` package is publicly available at:
https://github.com/xing-lab-pitt/livecellx. The repository includes source code, installation instructions, and example notebooks to facilitate reproduction and further development.

Acknowledgements. This work was partially supported by National Institute of General Medical Sciences (R01GM148525), National Institute of Diabetes and Digestive and Kidney Diseases (R01DK1192, R56DK119232) and National Science Foundation (DMS2325149) to JX, National Institute of Biomedical Imaging and Bioengineering (T32EB009403) to DP, National Natural Science Foundation of China Grants (No.12247104) to WW.

Declarations

The authors declares no conflict of interest.

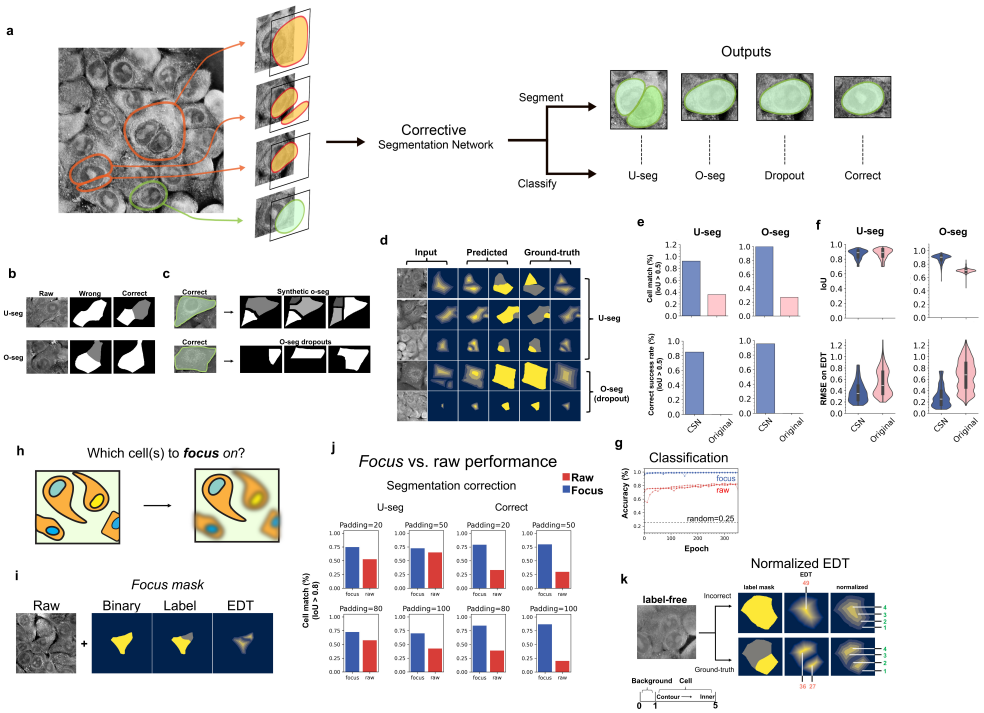

Fig. 2 Corrective Segmentation Network (CS-Net). (a) The CS-Net module takes the segmented mask from the initial step and the cropped region of interest (ROI) from label-free imaging results as inputs and outputs the corrected cell masks as well as error types. The error types include u-seg, o-seg and o-seg dropout. And it is possible that correct cell mask is identified as segmentation error when simply rely on temporal inconsistency. (b) Tyical u-seg (top) and o-seg (bottom) cases in training dataset. (c) Synthetic o-seg samples in training dataset (top) and typical o-seg dropout(right) which require the CS-Net to fill in missing mask regions, making the correction task more challenging. (d) Examples of typical outputs from the CS-Net, showing successful corrections of o-seg and u-seg errors. (e-f) Statistical analysis of the CS-Net's accuracy, focusing on the sample-level metrics we designed to measure the correction rates for o-seg and u-seg. (g) Accuracy of classifier in CS-Net with and without the focus mechanism. (h-i) Illustration of focus mechanism in CS-Net. Different type of cell masks can be used in the focus mechanism. (j) Comparing with cell match (IoU over 80 percent)for models with and without the focus mechanism under different padding pixels in ROI : As the number of padding pixels increases, leading to more cells in the input, the performance gap between the model with the focus mechanism and the one without becomes increasingly evident. (k) Example of a training sample demonstrating the superior numerical and semantic stability of normalized EDT masks compared to the unnormalized version. An incorrect mask (top) is provided as input, and the CS-Net is tasked with producing the ground truth (bottom). The normalized and unnormalized input-output variants are shown, with min/max values displayed for comparison.

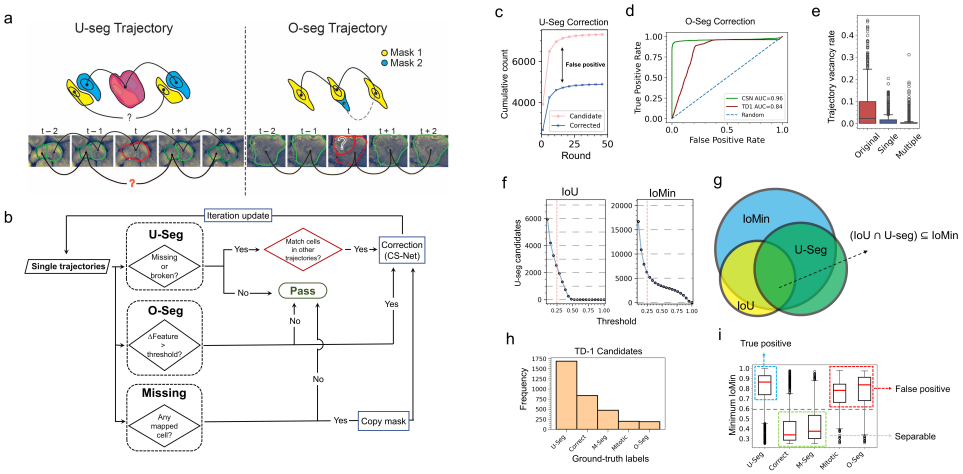

Fig. 3 Figure 3 Temporal Correction on Trajectories (a) Schematic illustration of tracking errors caused by under-segmentation (u-seg) and over-segmentation (o-seg) (top), with real examples (bottom) for illustration. A u-seg case typically involves two trajectories, resulting in a missing cell time point in one cell trajectory and an extra cell in the other trajectory at the same time point. In contrast, o-seg usually creates a one snapshot tracklet and leads to sharp morphological changes within the trajectory. (b) Flowchart of tracking correction: For u-seg cases, the algorithm first identifies candidate trajectories that match the scenarios depicted in (a), specifically those with missing time points or abrupt endings. If the cell mask at the end or missing time point matches a cell mask from another trajectory, the pair is selected as a correction candidate. Single-cell trajectories with o-seg errors are filtered based on feature changes. This module corrects the selected candidates with CS-Net and updates the single-cell trajectory objects accordingly. (c) Cumulative counts of candidate u-seg and fixed u-seg in multiple round correction. Both numbers become saturated after multiple rounds. The difference is due to the false positive cases detected by temporal inconsistency. (d) ROC curve for classifying o-seg cases based on CS-Net correction results and a simple two-way feature change score, as defined in the Methods. (e) Statistical results showing that the tracking correction algorithms significantly reduce the trajectory vacancy rate. Original represents trajectories without tracking correction. Single refers to the single-round correction algorithm, while Multiple refers to the multi-round algorithm. (f) Number of under-segmentation (u-seg) candidates identified using IoU and IoMin criteria. (g) Venn diagram showing the overlap between (1) u-seg candidates identified by IoMin, (2) candidates identified by IoU, and (3) ground truth u-seg cases. (h) Distribution of annotated classes for TD-1 candidates identified by IoMin. (i) Box plot showing the minimum IoU values for each TD-1 sample, indicating that no single IoMin threshold reliably separates u-seg cases from non-u-seg cases.

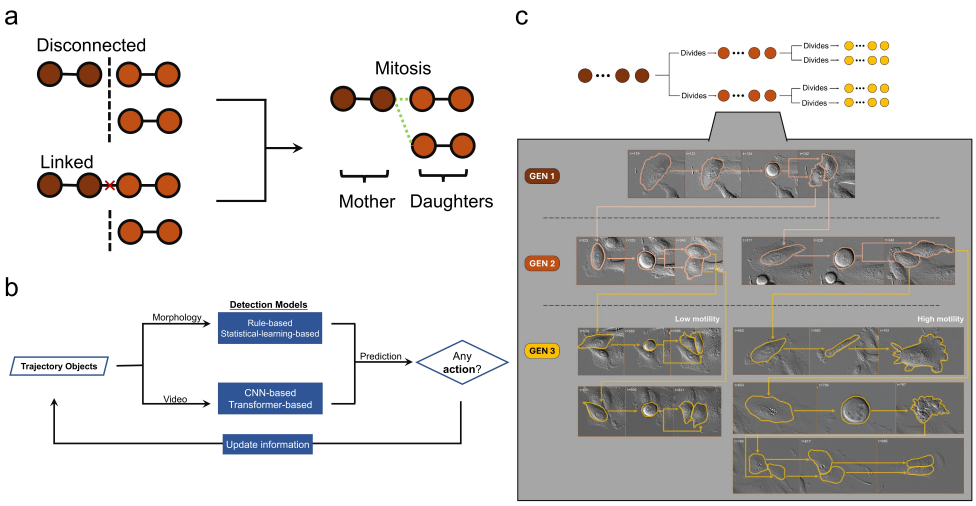

Fig. 4 Lineage reconstruction (a) Typical linkage issues caused by mitosis events. LivecellX detects mitotic cells, splits the detected trajectories and adds lineage information to single cell trajectory objects. (b) Workflow of machine-learning based module for mitosis detection. (c) Typical examples of a cell lineage, with mitosis-related time frames displayed. Half of the third generation (bottom right) exhibits increased motility.

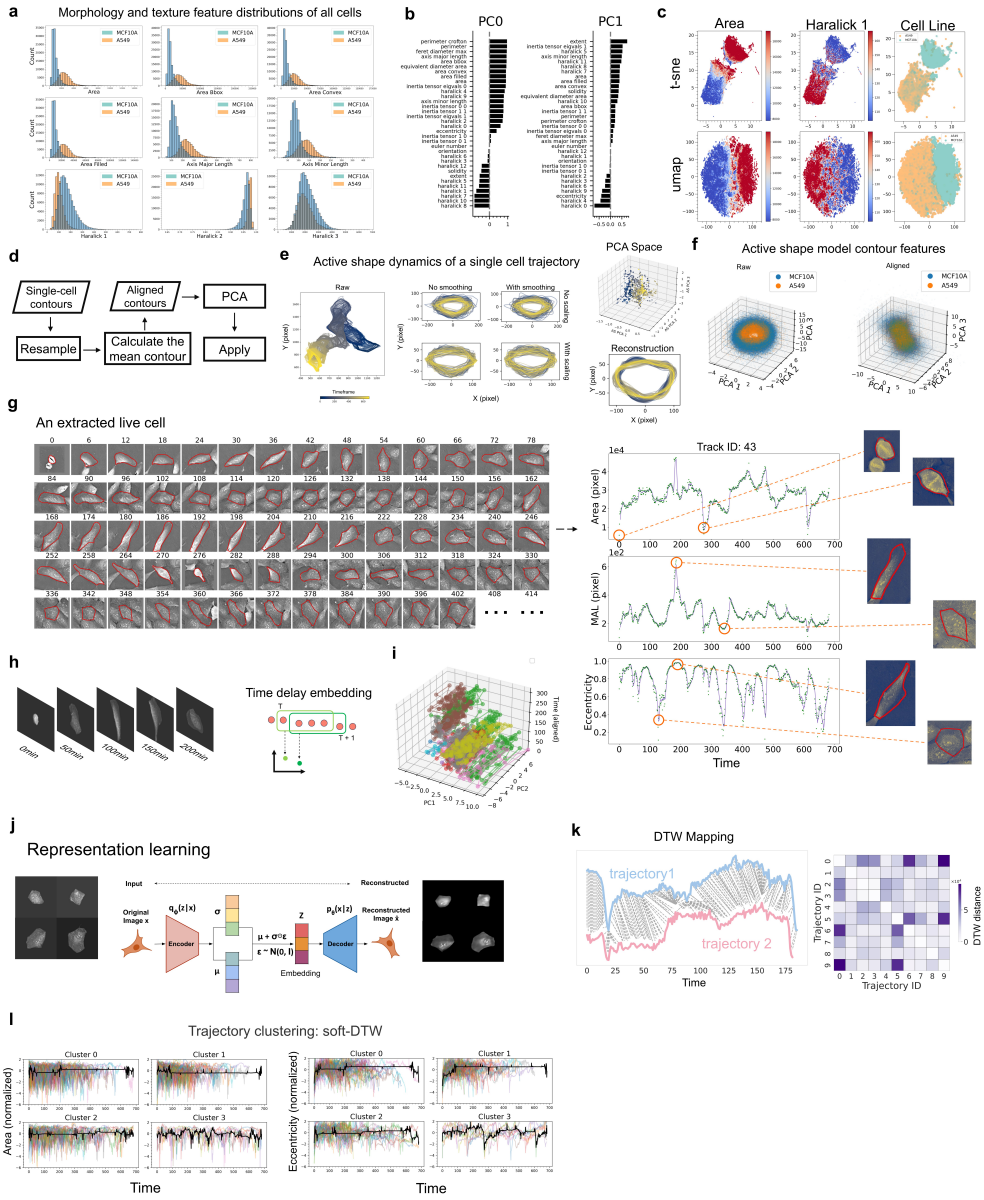

Fig. 5 Morphological feature extraction, temporal embedding, and trajectory analysis of single cells. (a) Distribution of morphology features extracted from A549 and MCF10A cells. (b) Correlation between morphological features and principal components. (c) Two-dimensional projections of morphology and texture features using UMAP and t-SNE. (d) Schematic of the LivecellX pipeline for contour alignment and active shape feature extraction. (e) Active shape dynamics of a single cell trajectory: contours are aligned using an active shape model, projected to PCA space, and reconstructed from principal components. (f) PCA projection of all cell contours from A549 and MCF10A datasets. Left: projection without alignment, highlighting differences such as cell size. Right: projection after size normalization via alignment, emphasizing shape-based variation independent of scale. (g) Left: label-free imaging visualization of an extracted live cell. Right: morphology feature change show the cell cycle and cell shape dynamics. (h) Schematic of time-delayed embedding applied to single-cell trajectories. (i) Time-delayed embedding of 10 randomly selected trajectories, with time points indexed from the first frame of each trajectory. (j) Architecture of the Harmony network used to extract latent embeddings from morphology and texture features. The embeddings in (k) and (l) incorporate Harmony-derived features. (k) Analysis of an ensemble of single-cell trajectories using Dynamic Time Warping (DTW). Left: schematic of DTW alignment; middle: pairwise DTW distance matrix across multiple single-cell trajectories. (l) Trajectory clustering based on soft-DTW distances.

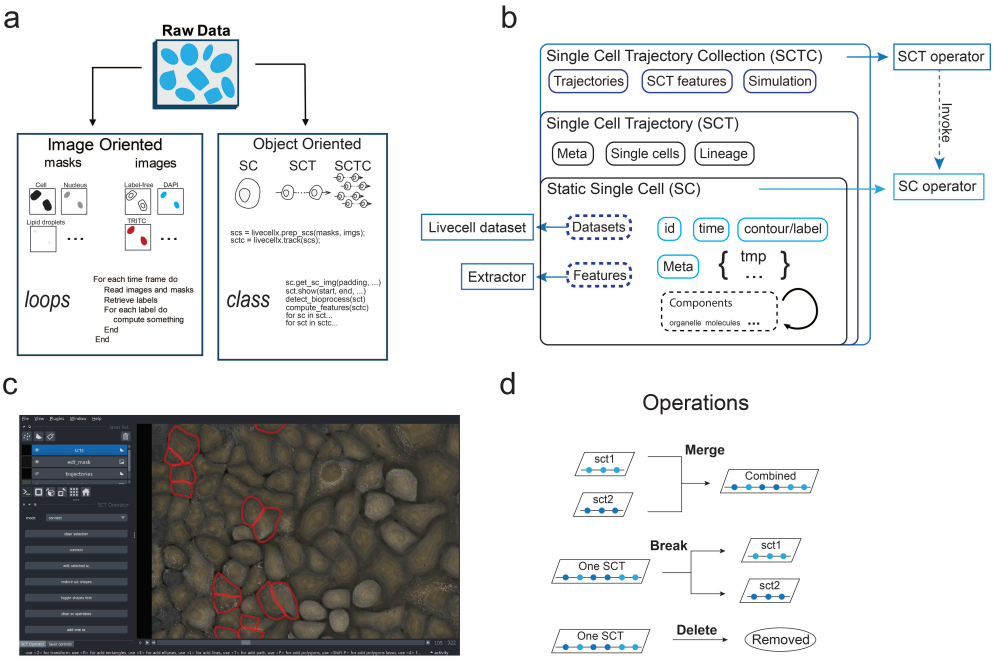

Fig. 6 LivecellX architecture and Napari user interface integration. (a) Comparison between image-oriented and single-cell object-oriented data structures. (b) Hierarchical data structures for static single cells, single-cell trajectories, and trajectory collections, forming the core of the LivecellX framework. (c) Integration of LivecellX with the Napari graphical user interface. The screenshot shows the user interface invoked from a Jupyter Notebook used to visualize and review CS-Net under-segmentation results. (d) Available trajectory-level operations in LivecellX, which can be performed either programmatically in Python or interactively through the Napari interface.

References

[1] Gordonov, S., Hwang, M.K., Wells, A., Gertler, F.B., Lauffenburger, D.A., Bathe, M.: Time series modeling of live-cell shape dynamics for image-based phenotypic profiling. Integr Biol (Camb) **8**(1), 73–90 (2016) https://doi.org/10.1039/C5IB00239A

[2] Wang, W., Douglas, D., Zhang, J., Kumari, S., Enuameh, M.S., Dai, Y., al.: Live-cell imaging and analysis reveal cell phenotypic transition dynamics inherently missing in snapshot data. Science Advances **6**(36), 9319 (2020) https://doi.org/10.1126/sciadv.aba9319

[3] Uchida, S.: Image processing and recognition for biological images. Dev Growth Differ **55**(4), 523–549 (2013) https://doi.org/10.1111/dgd.12033

[4] Shelhamer, E., Long, J., Darrell, T.: Fully convolutional networks for semantic segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(4), 640–651 (2017) https://doi.org/10.1109/TPAMI.2016.2572683

[5] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention – MICCAI, pp. 234–241. Springer, ??? (2015). https://doi.org/10.1007/978-3-319-24574-4_28

[6] Wang, W., Taft, D.A., Chen, Y.-J., Zhang, J., Wallace, C.T., Xu, M., al.: Learn to segment single cells with deep distance estimator and deep cell detector. Computers in Biology and Medicine **108**, 133–141 (2019) https://doi.org/10.1016/j.compbiomed.2019.03.011

[7] Van Valen, D.A., Kudo, T., Lane, K.M., Macklin, D.N., Quach, N.T., DeFelice, M.M., al.: Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments. PLOS Computational Biology **12**(11), 1005177 (2016) https://doi.org/10.1371/journal.pcbi.1005177

[8] Stringer, C., Wang, T., Michaelos, M., Pachitariu, M.: Cellpose: a generalist algorithm for cellular segmentation. Nature Methods **18**, 100–106 (2021) https://doi.org/10.1038/s41592-020-01018-x

[9] Han, H., Wu, G., Li, Y., Zi, Z.: edetect: A fast error detection and correction tool for live cell imaging data analysis. iScience **13**, 1–8 (2019) https://doi.org/10.1016/j.isci.2019.04.005

[10] Skylaki, S., Hilsenbeck, O., Schroeder, T.: Challenges in long-term imaging and quantification of single-cell dynamics. Nature Biotechnology **34**(11), 1137–1144 (2016) https://doi.org/10.1038/nbt.3664

[11] Moen, E., Borba, E., Miller, G., Schwartz, M., Bannon, D., Koe, N., al.: Accurate

cell tracking and lineage construction in live-cell imaging experiments with deep learning. bioRxiv, 803205 (2019) https://doi.org/10.1101/803205

[12] Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics **5**(1), 32–38 (1957)

[13] Christiansen, E.M., Yang, S.J., Ando, D.M., Javaherian, A., Skibinski, G., Lipnick, S., al.: In silico labeling: Predicting fluorescent labels in unlabeled images. Cell **173**(3), 792–80319 (2018) https://doi.org/10.1016/j.cell.2018.03.047

[14] Ounkomol, C., Seshamani, S., Maleckar, M.M., Collman, F., Johnson, G.R.: Label-free prediction of three-dimensional fluorescence images from transmitted-light microscopy. Nature Methods **15**(11), 917–920 (2018) https://doi.org/10.1038/s41592-018-0151-5

[15] Bray, M.-A., Singh, S., Han, H., Davis, C.T., Borgeson, B., Hartland, C., al.: Cell painting, a high-content image-based assay for morphological profiling using multiplexed fluorescent dyes. Nature Protocols **11**(9), 1757–1774 (2016) https://doi.org/10.1038/nprot.2016.105

[16] Way, G.P., Natoli, T., Adeboye, A., Litichevskiy, L., Yang, A., Lu, X., al.: Morphology and gene expression profiling provide complementary information for mapping cell state. Cell Systems **13**(11), 911–9239 (2022) https://doi.org/10.1016/j.cels.2022.10.014

[17] Gut, G., Herrmann, M.D., Pelkmans, L.: Multiplexed protein maps link subcellular organization to cellular states. Science **361**(6401), 7173 (2018) https://doi.org/10.1126/science.aar7173

[18] Hu, T., Xu, S., Wei, L., Zhang, X., Wang, X.: Celltracker: an automated toolbox for single-cell segmentation and tracking of time-lapse microscopy images. Bioinformatics **37**(2), 285–287 (2021) https://doi.org/10.1093/bioinformatics/btaa835

[19] Amat, F., Lemon, W., Mossing, D.P., McDole, K., Wan, Y., Branson, K., al.: Fast, accurate reconstruction of cell lineages from large-scale fluorescence microscopy data. Nature Methods **11**(9), 951–958 (2014) https://doi.org/10.1038/nmeth.2985

[20] Padovani, F., Mairhörmann, B., Falter-Braun, P., Lengefeld, J., Schmoller, K.M.: Segmentation, tracking and cell cycle analysis of live-cell imaging data with cell-acdc. BMC Biology **20**(1), 174 (2022) https://doi.org/10.1186/s12915-022-01484-1

[21] Piltti, K.M., Cummings, B.J., Carta, K., Manughian-Peter, A., Worne, C.L., Singh, K., al.: Live-cell time-lapse imaging and single-cell tracking of in vitro cultured neural stem cells - tools for analyzing dynamics of cell cycle, migration, and

lineage selection. Methods **133**, 81–90 (2018) https://doi.org/10.1016/j.ymeth.2017.12.003

[22] Tian, C., Yang, C., Spencer, S.L.: Elliptrack: A global-local cell-tracking pipeline for 2d fluorescence time-lapse microscopy. Cell Reports **32**(5), 107984 (2020) https://doi.org/10.1016/j.celrep.2020.107984

[23] Hernandez, D.E., Chen, S.W., Hunter, E.E., Steager, E.B., Kumar, V.: Cell tracking with deep learning and the viterbi algorithm. In: 2018 International Conference on Manipulation, Automation and Robotics at Small Scales (MARSS), pp. 1–7 (2018)

[24] Upcroft, A., Zaga, G., La, O., Fry, B.: Simple online and realtime tracking (2016)

[25] Ulicna, K., Vallardi, G., Charras, G., Lowe, A.R.: Automated deep lineage tree analysis using a bayesian single cell tracking approach. Frontiers in Computer Science **3**, 92 (2021) https://doi.org/10.3389/fcomp.2021.734559

[26] Bove, A., Gradeci, D., Fujita, Y., Ban erjee, S., Charras, G., Lowe, A.R.: Local cellular neighborhood controls proliferation in cell competition. Molecular Biology of the Cell **28**(23), 3215–3228 (2017) https://doi.org/10.1091/mbc.E17-06-0368 http://www.molbiolcell.org/content/28/23/3215.full.pdf+html

[27] Felzenszwalb, P.F., Huttenlocher, D.P.: Distance transforms of sampled functions. Theory of Computing **1**(1), 1–48 (2004)

[28] Vincent, L., Soille, P.: Watersheds in digital spaces: An efficient algorithm based on immersion simulations. IEEE Transactions on Pattern Analysis and Machine Intelligence **13**(6), 583–598 (1991)

[29] napari contributors: napari: a multi-dimensional image viewer for python. https://doi.org/10.5281/zenodo.3555620 (2019). https://doi.org/10.5281/zenodo.3555620

[30] Maška, M., Ulman, V., Delgado-Rodriguez, P., Gómez-de-Mariscal, E., Nečasová, T., Guerrero Peña, F.A., Ren, T.I., Meyerowitz, E.M., Scherr, T., Löffler, K., Mikut, R., Guo, T., Wang, Y., Allebach, J.P., Bao, R., Al-Shakarji, N.M., Rahmon, G., Toubal, I.E., Palaniappan, K., Lux, F., Matula, P., Sugawara, K., Magnusson, K.E.G., Aho, L., Cohen, A.R., Arbelle, A., Ben-Haim, T., Riklin Raviv, T., Isensee, F., Jäger, P.F., Maier-Hein, K.H., Zhu, Y., Ederra, C., Urbiola, A., Meijering, E., Cunha, A., Muñoz-Barrutia, A., Kozubek, M., Ortiz-de-Solórzano, C.: The cell tracking challenge: 10 years of objective benchmarking. Nature Methods **20**(7), 1010–1020 (2023) https://doi.org/10.1038/s41592-023-01879-y

[31] Pincus, D., Pincus, S.: Comparison of quantitative methods for cell-shape analysis. Journal of Microscopy **227**(2), 140–156 (2007)

[32] Takens, F.: Detecting strange attractors in turbulence. In: Rand, D., Young, L.-S. (eds.) Dynamical Systems and Turbulence, Warwick 1980, pp. 366–381. Springer, Berlin, Heidelberg (1981)

[33] Weigert, M., Schmidt, U., Haase, R., Sugawara, K., Myers, G.: Star-convex polyhedra for 3d object detection and segmentation in microscopy. In: The IEEE Winter Conference on Applications of Computer Vision (WACV) (2020). https://doi.org/10.1109/WACV45572.2020.9093435

[34] Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019)

[35] Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: MMDetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155 (2019)

[36] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library (2019). https://arxiv.org/abs/1912.01703

[37] Uddin, M.R., Howe, G., Zeng, X., Xu, M.: Harmony: A generic unsupervised approach for disentangling semantic content from parameterized transformations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 20614–20623. IEEE, ??? (2022). https://doi.org/10.1109/cvpr52688.2022.01999

[38] Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 6840–6851 (2020)

[39] Fick, A.: On liquid diffusion. Annalen der Physik **170**(1), 59–86 (1855)

[40] Coelho, L.P.: Mahotas: Open source software for scriptable computer vision. Journal of Open Research Software **1**(1), 3 (2013) https://doi.org/10.5334/jors.ac

[41] Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., Payne, M., Yurchak, R., Rußwurm, M., Kolar, K., Woods, E.: Tslearn, a machine learning toolkit for time series data. Journal of Machine Learning Research **21**(118), 1–6 (2020)