Large-Scale Joint Segmentation and Tracking

A thesis submitted in fulfilment of the requirements
for the degree of Doctor rerum naturalium (Dr. rer. nat.)

by

Jordão Okuma Barbosa Ferraz Bragantini

Faculty of Computer Science
TUD | Dresden University of Technology, Germany

Research conducted at Chan Zuckerberg Biohub, San Francisco, USA

February 2026

Day of defense: 18.05.2026

Examination committee:

Prof. Dr. rer. nat. Stefan Gumhold (chair of the committee)
Technische Universität Dresden, Dresden, Germany

Prof. Dr. sc. techn. Ivo Fabian Sbalzarini (reviewer)
Technische Universität Dresden, Dresden, Germany

Prof. Dr. rer. nat. Fred A. Hamprecht (external reviewer)
Universität Heidelberg, Heidelberg, Germany

Prof. Dr. rer. nat. Loïc A. Royer (subject expert)
Chan Zuckerberg Biohub, San Francisco, USA

Prof. Dr. Martin Weigert (committee member)
Technische Universität Dresden, Dresden, Germany

Certificate

It is certified that the work contained in this thesis entitled **"Large-Scale Joint Segmentation and Tracking"** by **Jordão Okuma Barbosa Ferraz Bragantini** has been carried out under my supervision and that it has not been submitted elsewhere for a degree.

Prof. Ivo Sbalzarini
Professor
Faculty of Computer Science
TUD | Dresden University of Technology,
Germany

Dr. Löic A. Royer
Senior Group Leader & Dir. of Imaging AI
Royer Group
Chan Zuckerberg Biohub,
San Francisco, USA

Declaration

This is to certify that the thesis titled **"Large-Scale Joint Segmentation and Tracking"** has been authored by me. It presents the research conducted by me under the supervision of **Dr. Löic A. Royer** and **Prof. Ivo Sbalzarini**.

To the best of my knowledge, it is an original work, both in terms of research content and narrative, and has not been submitted elsewhere, in part or in full, for a degree. Further, due credit has been attributed to the relevant state-of-the-art and collaborations with appropriate citations and acknowledgments, in line with established norms and practices. Generative AI was used to proofread grammar and style of this thesis.

Jordão Okuma Barbosa Ferraz Bragantini
Reference no. 23921D1
Faculty of Computer Science
TUD | Dresden University of Technology,
Germany

Abstract

The reconstruction of cellular trajectories across 2D, 3D, and multi-channel time-lapse recordings is fundamental to understanding tissue-scale biological processes. Despite advancements in imaging technology, accurately tracking cells remains challenging, particularly in complex and crowded tissues where cell segmentation is often ambiguous. This thesis investigates whether temporal information can be leveraged to resolve this ambiguity, with the premise that segmentation is a primary bottleneck in current tracking-by-detection paradigms.

We present Ultrack, a versatile and scalable framework for joint segmentation and tracking that addresses these challenges by formulating cell tracking as an optimal hierarchical cut problem. Unlike traditional methods that assume perfect prior detections, Ultrack considers a high-recall pool of candidate segmentations derived from multiple algorithms, parameter settings or the image intensity itself. By evaluating these hypotheses through an integer linear programming (ILP) formulation, the method utilizes temporal consistency to select the segments most likely to represent true biological entities, ensuring robust performance even under high segmentation uncertainty.

The proposed framework is validated on diverse and challenging datasets, including terabyte-scale developmental time-lapses of zebrafish, fruit fly, and worm embryos. To address the lack of validation data at the terabyte scale, we introduce a dual-channel sparse labeling protocol that enables the generation of "platinum-level" ground truth, facilitating the assessment of long-term tracking in multi-terabyte datasets.

The proposed technique to compute segments with temporal information is not exclusive to tracking and is extended to the general domain of data analysis by showing a proof of concept for a novel clustering method by translating the tracking into a non-parametric stability clustering problem, we propose an algorithm that identifies globally optimal stable clusters in arbitrary data. This provides a unified approach to model selection that evaluates combinations of candidate partitions in a single pass.

Acknowledgements

This thesis is the result of many years of research at the Royer Lab at the Biohub and would not have been possible without the support of many.

There is no other way to start this than by expressing my gratitude to Loïc Royer. I would not have pursued a doctorate if it were not for him. He suggested and encouraged this possibility before we worked closely together when I was interviewing for his team at the beginning of my Master's degree. I am more than grateful for his mentorship and kindness. I am also incredibly thankful to Ivo Sbalzarini. I could not have asked for another remote supervisor. I am indebted to you. My sincere thanks to Sandy Schmidt, who has shaped the Biohub research environment into what it is today and always open to provide advices. I would not be here if it were not for the napari community, I am grateful for everyone I met through it.

I am thankful to Martin Weigert, Fred Hamprecht and Stefan Gumhold for accepting being part of my thesis committee.

My profound appreciation goes to my first cohort in the Royer Lab: Merlin Lange, Bin Yang, Hirofumi Kobayashi, Shruthi Vijay Kumar, Bruno Moretti, Xiang Zhao, and Ahmet Can Solak, who welcomed me to the US during the pandemic and office closures, and taught me so much about biosciences. A special thank you goes to Merlin Lange, with whom I worked closely on multiple projects and received invaluable guidance.

I also owe special thanks to the current cohort of the Royer Lab: Ilan Theodoro, Teun Huijben, Alex Hillsley, Seth Hinz, Reinier Grona, Sheng Xiao, Thibault Goldsborough, Yuening Liu, Vera Janssen, Javier Carmona, Jia Le Lim, Gordon Leary and Giovanni Palla, who renewed the team spirit and have contributed so much to making this an excellent research environment.

This work would not have been possible without the support of the Biohub and its staff, particularly operations, the scientific computing, and experimental science teams, who have kept the lights on, the servers running and the data flowing. I am deeply thankful to them.

I am grateful to Eduardo Hirata and Talon Chandler, who have been close friends and have taught me a great deal about optics and physics.

Acknowledgements

Although it was a long time ago, I am fortunate to have had Alexandre Falcão as my supervisor, who initiated my scientific journey in Brazil.

I would never have achieved what I did without the support of my parents, Mauro and Karyn, who raised me to be the person I am today.

My final thanks go to my wife, Bárbara, this thesis would not have been finished without her support. Her love, companionship, and determination motivate me every day, and I am eternally grateful for her patience and sacrifices when we were living apart. I could not have been luckier in finding a partner for life.

Contents

List of Figures

List of Tables

List of Publications

Publications from Thesis

1. J. Bragantini, I. Theodoro, X. Zhao, T. A. Huijben, E. Hirata-Miyasaki, S. VijayKumar, A. Balasubramanian, T. Lao, R. Agrawal, S. Xiao, *et al.*, "Ultrack: pushing the limits of cell tracking across biological scales," *Nature Methods*, pp. 1–14, 2025

2. J. Bragantini, M. Lange, and L. Royer, "Large-scale multi-hypotheses cell tracking using ultrametric contours maps," in *European Conference on Computer Vision*, 2024

3. M. Lange, A. Granados, S. VijayKumar, J. Bragantini, S. Ancheta, Y.-J. Kim, S. Santhosh, M. Borja, H. Kobayashi, E. McGeever, *et al.*, "A multimodal zebrafish developmental atlas reveals the state-transition dynamics of late-vertebrate pluripotent axial progenitors," *Cell*, vol. 187, no. 23, pp. 6742–6759, 2024

Other Publications

4. T. A. Huijben, A. G. Anderson III, A. Sweet, E. Hoops, C. Larsen, K. Awayan, J. Bragantini, M. Lange, C.-L. Chiu, and L. A. Royer, "inTRACKtive: a web-based tool for interactive cell tracking visualization," *Nature Methods*, pp. 1–3, 2025

5. E. Eck, B. Moretti, B. H. Schlomann, J. Bragantini, M. Lange, X. Zhao, S. VijayKumar, G. Valentin, C. Loureiro, D. Soroldoni, *et al.*, "Single-cell transcriptional dynamics in a living vertebrate," *bioRxiv*, 2024

6. J. Moore, D. Basurto-Lozada, S. Besson, J. Bogovic, J. Bragantini, E. M. Brown, J.-M. Burel, X. Casas Moreno, G. de Medeiros, E. E. Diel, *et al.*, "Ome-zarr: a cloud-optimized bioimaging file format with international community support," *Histochemistry and Cell Biology*, vol. 160, no. 3, pp. 223–251, 2023

7. B. Yang, M. Lange, A. Millett-Sikking, X. Zhao, J. Bragantini, S. Vijay Kumar, M. Kamb, R. Gómez-Sjöberg, A. C. Solak, W. Wang, *et al.*, "Daxi—high-resolution, large imaging volume and multi-view single-objective light-sheet microscopy," *Nature Methods*, vol. 19, no. 4, pp. 461–469, 2022

Symbols

G^I	A regular grid image graph
V^I	A set of nodes (pixels or voxels) in the image graph G^I
E^I	A set of edges in the image graph G^I
w^I	An edge weight function on the image graph G^I
P	A set of segments (partition) of a graph G
S_i	A set of connected nodes (segment, subgraph) in a graph G
P_λ	A set of segments (partition) of a graph G at level λ
S_i^λ	A segment at level λ
H	A hierarchy, sequence of increasing λ partitions
$\mathcal{H}$	A set of hierarchies
Φ	A hierarchical clustering algorithm
$anc_H(p)$	The path from p to the root of the hierarchy H
w^S	An weight (similarity) function between segments
π	A path (tracklet) in the graph G
η	A hierarchy perturbation function
C	A set of contraints of a constrained optimization problem

To my beloved wife and parents
for their support, encouragement and patience

Chapter 1

Introduction

Microscopy is concerned with the study of the world through the magnification of images beyond what our naked eye can see. Beginning with the invention of the first microscope in the late 16th and early 17th century, which enabled magnification beyond what could be achieved with a simple magnifying glass, this breakthrough led to the first illustration of the microscopic world in *Micrographia* by Robert Hooke in 1665 [8]. This seminal work revealed a hidden universe of intricate structures, from the compound eyes of insects to the cellular structure of cork, fundamentally transforming our understanding of nature.

Since then, the study of living organisms (biology) has evolved in conjunction with the field of microscopy. The 19th century brought crucial innovations, such as improved lens design, which reduced optical aberrations, and the development of sample preparation techniques, which made targeted cellular structures visible. These advances enabled enduring discoveries, including the formulation of cell theory and the identification of bacteria as disease agents. The 20th century witnessed the emergence of fluorescence microscopy [9], which revolutionized biological imaging by allowing specific cellular components to be selectively labeled and visualized. The invention of the electron microscope [10] further pushed the boundaries, revealing subcellular structures at nanometer resolution, and enabling the study of the molecular basis of life. More recently, super-resolution techniques have overcome the diffraction limit of light [11, 12, 13], earning the Nobel Prize in Chemistry in 2014, and light-sheet microscopy [14, 15] have enabled three-dimensional imaging of large living specimens with minimal photo damage.

As microscopy techniques grew more sophisticated, generating increasingly complex and voluminous datasets, computation emerged as an essential tool for addressing multiple challenges across the imaging pipeline. Computational methods now control instruments through adaptive feedback systems [16], reconstruct images from raw acquisitions using deconvolution algorithms [17, 18], enhance image quality through deep learning [19], and quantify biological structures through automated segmentation [20, 21] and tracking [22]. This computational revolution has transformed microscopy from an observational tool into a quantitative analytical platform capable of automatically extracting numerical data from biological samples.

For quantifying dynamical biological processes, light-sheet microscopy is at the forefront, due its gentle and fast acquisition enabling the visualization of long time-lapses. Recent advancements [23, 24, 25, 26, 7, 27] have enabled the visualization of cellular dynamics with unprecedented spatiotemporal resolution [28] over large fields of view [7, 27] encompassing thousands of cells [3, 26], producing vast amounts of multi-dimensional data [29]. These technologies allow researchers to observe biological processes as they unfold in living organisms, from individual cell divisions to the coordinated movements of thousands of cells during embryonic development, Figure 1.1. However, one key challenge remains: the accurate reconstruction of cell trajectories and lineages in complex biological systems [30]. This ability unlocks deeper insights into cellular state and behavior [31, 3, 32], tissue mechanics [33], morphogenesis [34, 23, 35], and regeneration [36, 37].

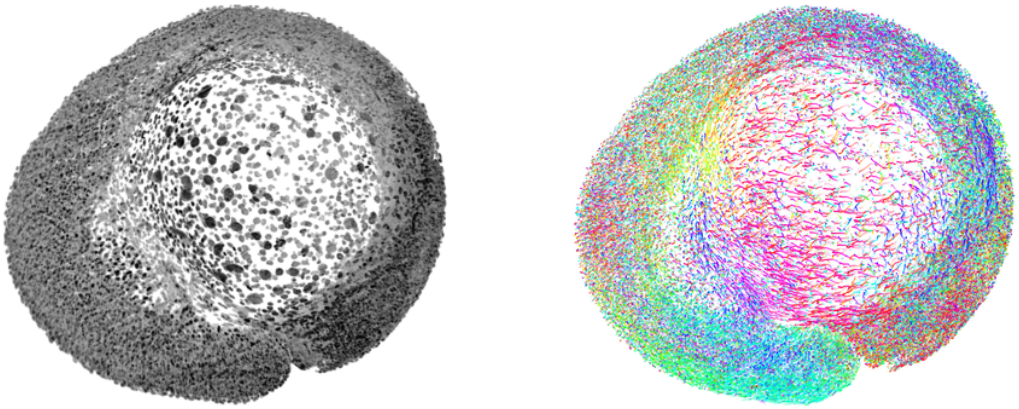


Figure 1.1: Developing zebrafish embryo cell tracking. Maximum intensity projection of 3D volume of zebrafish and its respective cell tracks computed with the method developed in this thesis. Tracks are colored according to the cell movement orientation, highlighting how different regions (tissues) result from distinct migration patterns.

While cell segmentation has advanced rapidly with the advent of deep-learning-based methods [38, 39, 40, 41], cell tracking success is not as widespread [30]. The primary challenge of tracking stems from the association of spurious segmentations over time, hindering long-term tracing, especially in dense and dynamic cellular environments. Most automatic tracking methods employ a two-step approach, known as tracking-by-detection: first segmenting (detecting) cells, then associating (*i.e.* linking) them across time frames [42, 43, 44, 45, 46, 47, 48, 49]. While computationally efficient, this paradigm struggles with the compounding of errors over time, particularly in dense tissues [50] or when cells divide rapidly.

Simultaneous segmentation and tracking offers a promising alternative [51, 52, 53, 54, 55], by providing multiple candidate segments and or inter-segment links, and the final cell tracks and segmentations are obtained by satisfying biological constraints (*i.e.* cells divide but do not merge) and optimizing specific criteria (*i.e.* only keep links best supported by the data). However, existing methods in this category are constrained by their inability to leverage arbitrary segmentation inputs [22, 53], as well as limitations in data scale or dimensionality due to high computational costs [54, 55]. These factors have limited their applicability beyond moderate-sized 2D datasets.

Moreover, recent trends in the field have abandoned this paradigm, focusing on association step of the tracking-by-detection, often assuming that cell segmentations are perfect [56, 48, 57].

In this thesis, we revisit the problem of joint segmentation and tracking of cells, going against the current trend of tracking-by-detection approaches and pose the following question:

Is association or segmentation the bottleneck of cell tracking?

To address this question, we investigated a novel approach, called Ultrack, for a robust and scalable method for large-scale cell tracking that is robust to the presence of segmentation uncertainty. It solves the computational limitations of previous joint segmentation and tracking methods. While being versatile, tracking cells (or nuclei) in 2D, 3D, and multi-channel datasets, in a wide range of biological contexts. Rather than

replacing existing segmentation methods, our algorithm integrates with various segmentation algorithms, including state-of-the-art deep learning-based cell segmentation tools [39, 58, 38, 40, 41, 59], enjoying the benefits of the recent advances in the field. By jointly evaluating candidate segmentations and tracks through an integer linear programming (ILP) formulation, it computes the segments that are most temporally consistent. In other words, the information from adjacent time points resolves which candidate segmentations are more likely to be the true cells' segmentations, enhancing segmentation and tracking performance, particularly in complex, densely packed tissues where cell boundaries are often uncertain.

1.1 Overview

This thesis is organized as follows. Chapter 2, Background, starts with Section 2.1, briefly contextualizing the data generating process of biological images and the challenges they present. Then, the sections from Section 2.2 to 2.5 are divided into two parts. First, we provide a high-level overview of existing work in each field and motivate our work. Next, we formalize and introduce notations of key concepts, which are necessary to understand the proposed cell tracking method. Chapter 3, builds upon these formulations and introduces our cell tracking method. Chapter 4 presents our cell tracking results in various scenarios, from multi-colored cultured cells to a developing zebrafish embryo.

Chapter 5 extends the cell tracking approach to the general problem of clustering arbitrary data, by translating the problem to a stability clustering framework. Therefore, rather than working on time-lapses, it evaluates perturbed views of the data. With this, we propose a new non-parametric noise model for hierarchies, an exact ILP algorithm that finds the most stable clusters in a single pass, while existing stability model selection methods require re-running the algorithm per parameter (*e.g.* number of clusters), and an approximate algorithm to the ILP problem using dynamic programming that can scale to large datasets. This approach enables us to revisit the usefulness of stability for model selection in clustering and assess whether flat clustering is applicable to real-world data.

Chapter 6 provides a summary of the thesis, the limitations of the proposed approach, and future directions of the cell tracking field.

A Note on Reading this Thesis

This thesis is organized to separate the technical aspects of our proposed approach, *Ultrack*, from its practical applications. However, the chapters are designed to be read in a complementary manner, where Chapter 3 formalizes the proposed algorithm for joint segmentation and tracking, Chapter 4 presents the experimental data and applications that validate the algorithmic design choices.

Through the text we will refer to which sections are complementary to each other, as well as in the list below:

- **Core Tracking Algorithm:** Begin with the optimal hierarchical cut formulation, Section 3.1, then, controlled experiments evaluating different ways of providing a contour map for our algorithm, Section 4.2 to Section 4.4, evaluate its performance on standardized benchmarks in the Cell Tracking Challenge, Section 4.6.
- **Hypothesis Construction:** Review the hierarchical pruning strategy, Section 3.2, alongside results demonstrating how multiple hypotheses alleviate the need for manual parameter tuning, Section 4.2.
- **Multi-Channel Integration:** Explore the association of color features, Section 3.4, in conjunction with the multi-colored cells experiments, Section 4.4.
- **Scalability and Distributed Computing:** Study the distributed processing implementation, Section 3.6, alongside the runtime analysis of terabyte-scale zebrafish developmental recordings, Section 4.9.

1.1.1 Contributions

Therefore, the contributions of this thesis are:

- A new algorithm for joint segmentation and tracking of cells, Ultrack, that:
  - Is less dependent on the accuracy of pre-computed segmentations, making use of both spatial and temporal information to reconcile segmentation ambiguity.
  - Supports distributed and out-of-core processing of multi-terabyte datasets with millions of segmentation instances.

  - Leverages existing pre-trained models and supports traditional image processing segmentation as input, achieving reasonable performance even when ground-truth labels are unavailable for deep-learning model training.
- A new benchmark of five-terabyte imaging data of developing zebrafish embryo, to our knowledge, the largest dataset currently available for cell tracking.
- A new method for clustering, to our knowledge, it is the first method that finds the globally optimal stable clusters considering combinations of candidate clustering solutions. Our contributions include:
  - A new non-parametric noise model for hierarchies.
  - An exact ILP algorithm that finds the most stable clusters in a single pass.
  - An dynamic-programming approximate algorithm for the ILP problem that can scale to large datasets.

A huge part of the motivation for this work is to power biological research, where existing methods were not sufficient to solve the problems we were facing. Therefore, making the tools accessible to the biological community is fundamental to accomplish this goal. Appendix A.5 describes additional open-source software developed in this thesis.

Chapter 2

Background

The field of cell tracking relies upon multiple domains of technology development, from computer science to optics and biology. In this section we provide an overview of the relevant areas of these fields. First, Section 2.1 briefly contextualizes the data generating process of biological images and the challenges they present. From then on, we focus on the primary domain of this thesis, computer science. Describing the fundamentals of instance and hierarchical segmentation in Section 2.2 and Section 2.3, respectively. Then, in Section 2.4 we review the general problem of multi-object tracking (MOT) in computer vision applications outside of the context of biology, highlighting their differences. Finally, Section 2.5 introduces existing cell tracking methods and how they relate to the proposed approach.

2.1 Biological Imaging

The process of recording a living specimen into a digital representation is governed by the laws of optics and the constraints of digital sensors. Understanding this data-generating process is crucial, as the computational challenges in cell tracking, such as low signal-to-noise ratios and boundary ambiguity, are often rooted in the physical limitations of the imaging system.

2.1.1 From Specimen to Digital Signal

A biological image is essentially a discrete sampling of a continuous spatial distribution of light. In fluorescence microscopy, the process begins with the excitation of fluorophores (molecules that emit light upon light absorption) attached to specific cellular structures. These emitted photons are collected by an objective lens and focused onto a camera sensor.

The sensor discretizes this continuous signal into a regular grid of *pixels* (2D). Multiple 2D grids can be stacked to form different channels, 3D volumes, time-lapses, or a combination of them. Each pixel records an intensity value, which is a quantization of the number of photons detected at that spatial coordinate. Consequently, the digital image is represented as a matrix $I \in \mathbb{Z}_{+}^{T \times C \times Z \times Y \times X}$, where T is time, C is the number of channels, Z, Y, X are spatial dimensions.

2.1.2 Physical Limitations and Image Degradation

The recording process from reality to an array of pixels (*i.e.* digital image) is not perfect. Several physical factors degrade the signal, and computational methods to be successful must account for them:

- **The Diffraction Limit and Point Spread Function (PSF):** No optical system can focus light into an infinitely small point. Due to diffraction, a single point source of light is blurred into a finite shape called the Point Spread Function (PSF), Figure 2.1. Mathematically, the observed image $I(x)$ at a pixel x is the convolution of the true object $I_{true}(x)$ with the PSF (h) and a noise term $n(x)$, $I = I_{true} * h + n$. This limits the resolution and can cause cell boundaries to be blurred or completely lost.
- **Noise Sources:** Biological images are inherently noisy. The primary sources are *shot noise*, arising from the discrete nature of photons (governed by Poisson statistics), and *read noise*, which is electronic interference from the sensor itself. In high-speed light-sheet imaging of living samples, researchers often limit their exposure time for faster acquisition or lower photo toxicity, resulting in low signal-to-noise ratio (SNR) that complicates segmentation and tracking.

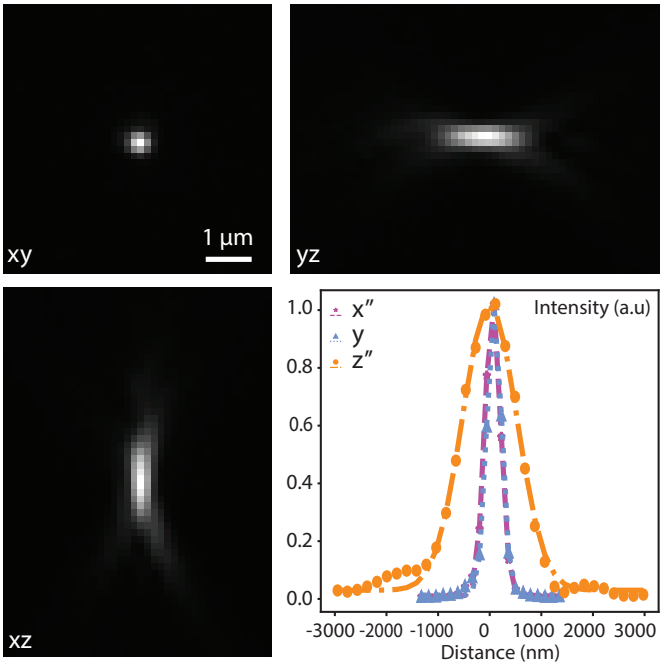


Figure 2.1: Example of a point spread function (PSF) of a diffraction-limited bead (spot) from a light-sheet fluorescence microscope [7]. The first image, the **xy** plane recapitulates the expected spot intensity profile, while the **xz** and **yz** planes presents an spread along the axial direction. The spread degrades our ability to resolve small structures as cell boundaries. The intensity profile plot shows the difference between the axial and the lateral intensity profiles. Results averaged over 6 beads. The narrower the PSF, the higher the resolution. Figure adapted from [7].

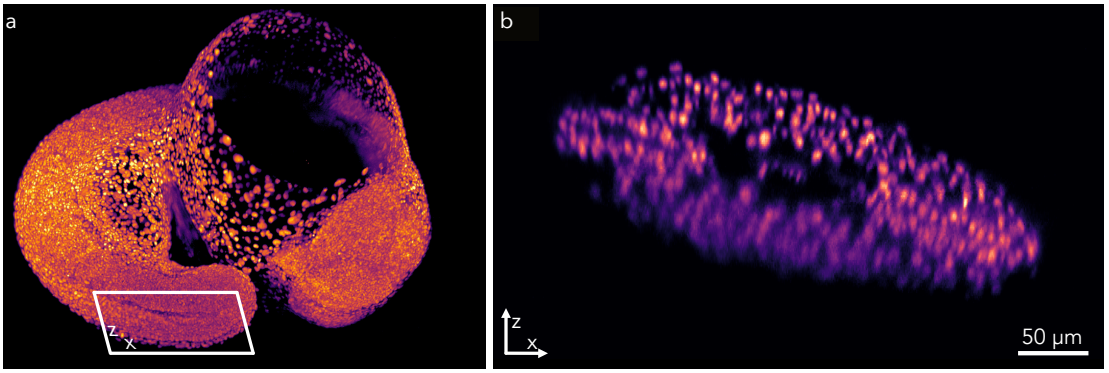


Figure 2.2: Image quality variations in volumetric microscopy image. **a**, Maximum intensity projection of whole zebrafish embryo. **b**, 2D slice of the 3D volume, indicated by plane in **a**. Image quality varies severly along the vertical axis, from bright and sharp nuclei to dimmer and blurrier.

- **Optical Aberrations:** As light travels through the specimen, variations in the refractive index of the tissue cause wavefront distortions. These aberrations, such as spherical aberration or astigmatism, leads to spatially varying blurring and loss of signal intensity, especially when imaging deep within large organisms like whole embryos, where light needs to travel through more distinct structures, Figure 2.2.

2.1.3 Implications for Cell Tracking

These degradations mean that the true boundary of a cell is rarely represented by a sharp change in pixel intensity or even visible due to limited resolution. Instead, boundaries transitions are often diffuse, obscured by noise, or completely lost due to scattering. In the context of this thesis, these experimental constraints motivate the need for a more flexible and robust tracking framework, rather than the rigid, deterministic tracking-by-detection, because the digital representation of a cell is inherently uncertain, segmentation is imperfect and a tracking framework must be able to evaluate multiple possible segmentations across time to find the most physically and biologically plausible interpretation of the data.

2.2 Instance Segmentation

Instance segmentation is the problem of partitioning pixels (voxels[1] in the 3D case) of an image into disjoint groups of pixels (*i.e.* regions, superpixels, segments), where each region represents a different instance of an object. In the general context, segmentation is an ill-posed problem as pixels can belong to multiple regions depending on your definition of an object. In the biological domain, it is almost always well defined, as the objects of interest are often the cells, the smallest living units of an organism, an example is shown in Figure 2.3.

A common approach is to reduce segmentation into a graph partitioning problem, often formulated as an optimization problem where the optimal partition yields the desired segmentation based on the edge weights of the graph. These weights provide cues, such

[1] pixels and voxels will be used interchangeably throughout the thesis

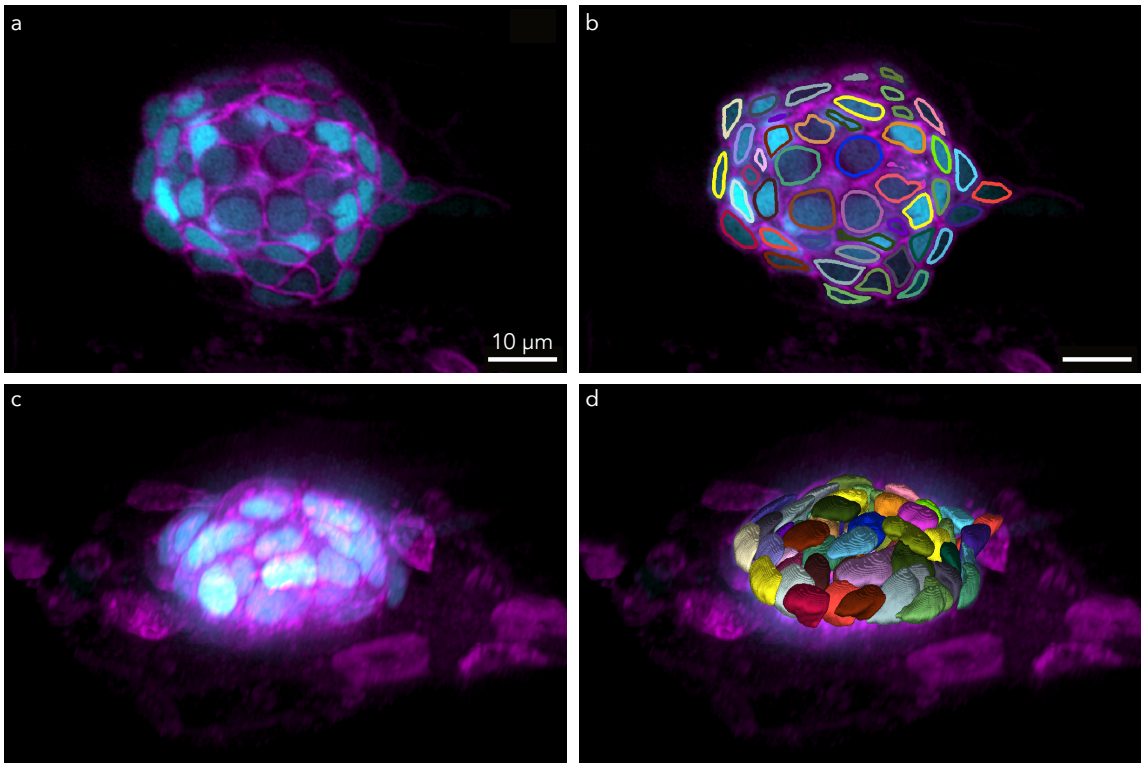


Figure 2.3: Cell instance segmentation of a zebrafish neuromast. Example of 3D instance segmentation of a zebrafish neuromast with fluorescently labeled nuclei and membranes. **a**, Representative 2D slice displaying cell membranes (magenta) and nuclei (cyan). **b**, its respective instance segmentation where uniquely colored contours denote individual cells. **c**, Maximum intensity projection (MIP) of the 3D volume. **d**, Volumetric rendering of the segmented instances.

as affinities (indicating similarity) or image gradients and contours (indicating dissimilarity), which determine whether pixels should belong or not to the same segment.

Notable graph-based segmentation methods include:

- **The watershed transform** [60, 61]: This method takes a topological perspective of the image, viewing the intensity of a grayscale image as a topographic map where brighter values are higher (peaks) and darker values are lower (valleys). If a drop of water were released at each pixel coordinate, it would flow into a local minimum; pixels that drain into the same minimum are grouped into the same segment.
- **The image foresting transform (IFT)** [62]: A dynamic programming algorithm that extends Dijkstra's shortest-path [63] by allowing multiple sources and minimizing the maximum edge weight along each path rather than the cumulative sum. It

supports a diverse set of edge-weight functions to encode dissimilarity between pixels [64, 65]. When using image intensity as a direct edge weights, it is equivalent to the watershed transform [66].

- **The multi-cut** [67, 68]: This approach extends the binary graph-cut (minimum cut) [69, 70] to an unknown number of objects. The objective is to remove the minimum set of edges (in terms of total weight) such that the graph is partitioned into connected components. It is equivalent to sparse correlation clustering, as it uses signed edge weights to control the number of clusters. While the multi-cut is less susceptible to outliers than previous approaches, it is significantly more computationally expensive because it is an NP-hard problem.

Couprie *et al*. [71] demonstrated the theoretical equivalence between these approaches under a unified framework for the binary (two-classes) scenario. Recently, the *mutex-watershed* [72, 73] was proposed to combine the signed edge weights typical of the multi-cut formulation with the greedy nature of the watershed transform, aiming to leverage the benefits computational efficiency of the watershed and the robustness of the multi-cut.

With the rapid development of deep learning for pixel-level prediction[2] particularly through Convolutional Neural Networks (CNNs) [74], instance segmentation has become increasingly more accurate while also becoming more accessible to non-experts, who can now easily download pre-trained models and use them for their own purposes with little knowledge of the underlying algorithms. Among various CNN architectures, the U-Net [75] remains the most prominent example in the biological domain, despite being a decade old — a significant timespan in the fast-evolving field of deep learning.

Deep instance segmentation methods can be broadly categorized into two paradigms: (i) pixel-embedding methods and (ii) affinity-based methods. The former seeks to learn a high-dimensional feature representation where pixels belonging to distinct objects are separated in this abstract space (*i.e*. embedding space), often by estimating spatial offsets (*i.e*. translations) between pixels. These embeddings are subsequently processed by a clustering algorithm (*e.g*. mean-shift [76]) to produce discrete assignments of pixels into segments.

[2]Pixel-level prediction is the task of predicting a value for each pixel in an image.

Notable pixel-embedding methods include:

- **StarDist** [38]: Which predicts the center of each cell and a set of radial distances (rays) that enables the reconstruction of the cell boundaries as star-convex polygons encoded by the endpoint of each ray.
- **Cellpose** [39, 58, 77, 78]: Which uses a dynamical system formulation to predict horizontal and vertical spatial gradients, effectively representing the flow of each pixel toward its respective cell centroid.

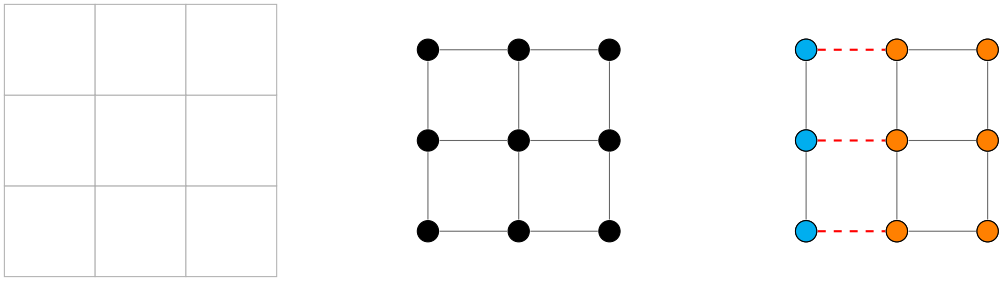
Affinity-based methods instead estimate object boundaries or local connectivity between pixels. The final segmentation is then obtained using the previously mentioned graph-based partitioning methods. Maximum Affinity Learning of Image Segmentation (MALIS) [79] and its later iteration [80] employ a topological loss by minimizing the Rand index [81] between segmentations derived from the predicted boundaries and the ground truth. Similarly, Learned Watershed [82] and Learned Random Walk [83] improve standard boundary prediction by employing objective functions inspired by their respective graph-based counterparts, Watershed [60] and Random Walks [84], respectively.

It is essential to note that both paradigms initially process the image to generate a representation that simplifies the grouping task, followed by a clustering step to achieve the final segmentation. Consequently, clustering remains a critical component of the pipeline, which is a primary reason why segmentation (clustering) is central to our tracking approach.

Furthermore, the distinction between pixel embeddings and affinity-based methods is not absolute. Pixel embeddings can be converted into affinities by computing pairwise distances between pixels, after all, affinities are essentially an inverse measure of the distance between neighboring pixels in a feature or intensity space.

2.2.1 Notations

An image I is represented as a weighted graph $G^I = (V^I, E^I, w^I)$, where the set of nodes V^I corresponds to the pixels (or voxels) of the image. These nodes are arranged in a regular grid defined by their spatial coordinates. Each node is connected to its neighbors within a specified radius r. For a radius of one ($r = 1$), a non-boundary node in



(a) Pixel grid I (b) Graph $G^I = (V^I, E^I)$ (c) Partition $P = \{S_0, S_1\}$

Figure 2.4: Formal representation of an image as a graph. a, The image I viewed as a regular grid of pixels. **b**, The graph G^I, where pixels become nodes V^I and the neighborhood defines the edges E^I. **c**, A graph cut where removing edges (red dashed lines) creates disjoint sets S_0 and S_1, forming a flat clustering.

a 2D image has a 4-connectivity (von Neumann neighborhood), while in 3D, it has a 6-connectivity.

Let $p_i \in \mathbb{R}^d$ denote the spatial position of node $i \in V^I$. The set of edges E^I is defined as:

$$E^I = \{(i, j) \in V^I \times V^I \mid 0 < \|p_i - p_j\| \leq r\}, \tag{2.1}$$

where $\|\cdot\|$ denotes the Euclidean norm of the difference between the spatial coordinates of the nodes (*e.g.* Euclidean distance). The weight function $w^I : E^I \mapsto \mathbb{R}$ assigns a value to each edge (i, j), typically representing the similarity or dissimilarity between pixels based on intensity differences or more sophisticated local gradients. A visual representation of this graph structure is shown in Figure 2.4.

In this framework, segmentation is the process of partitioning the set of nodes V^I into disjoint subsets. From a graph-theoretic perspective, this is formulated as a *graph cut*, Figure 2.4c, where edges are removed to decompose the graph into a set of connected components, each representing a distinct segment.

More formally, a partition P of the graph G^I is a collection of sets $P = \{S_0, S_1, \ldots, S_{k-1}\}$ such that:

1. $\bigcup_{i=0}^{k-1} S_i = V^I$: The union of all segments covers the entire image.
2. $S_i \cap S_j = \emptyset \, \forall \, i \neq j$: Distinct segments are disjoint.
3. $S_i \neq \emptyset \, \forall \, i \in \{0, \ldots, k-1\}$: No segment is empty.

Because every node is assigned to exactly one segment, this result is known as a *flat clustering*. This contrasts with *hierarchical clustering*, where segments are nested across multiple scales, as it will be described next in Section 2.3.

2.3 Hierarchical Segmentation

Hierarchical segmentation extends instance segmentation by identifying a sequence of nested partitions of the data, rather than a single flat partition. This sequence ranges from the finest level, where each pixel constitutes an individual segment, to the coarsest level, where the entire image is represented as a single segment. A primary advantage of hierarchical representations over flat clustering is their ability to reduce the possible ambiguity from assigning an elements into a single category. For example, in a natural image, a single pixel may concurrently belong to multiple nested categories, such as a tooth, a mouth, a head, and a person. Furthermore, the internal nodes of the hierarchy provide a significantly more compact representation of the data than the original graph, enabling more efficient computation of filtering and selection operations — a property our proposed approach exploits.

Hierarchical representations can be broadly categorized into two classes: inclusion hierarchies and partition hierarchies [85]. Inclusion hierarchies, such as component trees [86, 87], represent nested regions originating from local features (*e.g.*, minima or maxima), where regions at finer levels are subsets of regions at coarser levels. These are particularly effective for filtering extrema-oriented structures (*e.g.*, dark or bright objects) or specific shapes [88]. In contrast, partition hierarchies, such as Binary Partition Trees [89], represent nested partitions of the entire image domain at each level. Such tree structure provides a complete segmentation at each level, with regions merging as the hierarchy coarsens. Partition hierarchies are well-suited for representing objects at intermediate intensity levels and for tasks requiring complete image coverage at every scale, as in our use case.

Unlike flat clustering, nested segmentation cannot be easily represented using discrete

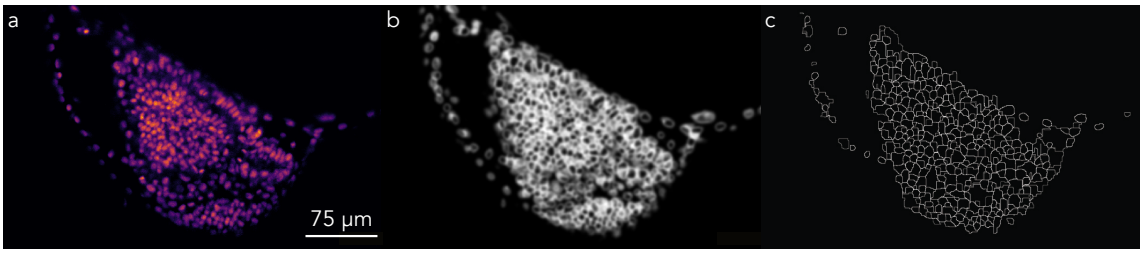


Figure 2.5: Ultrametric Contour Map (UCM) representation. a, A 2D image of fluorescently labeled nuclei. **b**, A nucleus boundary estimation using a convolutional neural network from the same image. **c**, The corresponding UCM of the nucleus boundary estimation from a hierarchical segmentation with altitude corresponding to the contour map dynamics (*e.g.* relative difference between minimum and boundary values).

labels in the image domain. Consequently, hierarchical segmentations are commonly represented using Ultrametric Contour Maps (UCM) [61, 90, 91], where the boundaries between regions are assigned a value λ_i corresponding to the hierarchical level at which the adjacent regions merged. This representation exploits the mathematical duality between fuzzy contour maps (*i.e.* UCMs, edge detection maps, and affinity maps) and hierarchical structures, Figure 2.5.

Several methodologies leverage this duality for hierarchical segmentation. Multi-scale Combinatorial Grouping (MCG) [92] accelerated the computation of normalized cuts [93] on affinity maps through a multi-scale approach, starting from a lower-resolution image and progressively increasing the resolution before aligning and merging results into a contour map. Maninis *et al.* [94] used a convolutional neural network to estimate boundaries at different orientations and subsequently applied the oriented watershed transform [95] to obtain the hierarchical segmentation.

Once a hierarchy is constructed, distinct simplification techniques are applied based on the task, such as removing regions during filtering or selecting regions for content retrieval and object detection. For segmentation specifically, Kiran and Serra [96] proposed a framework for non-horizontal optimal cuts[3] restricted to a specific set of energy functions. More recently, the Generalized Algorithm for Signed Graph Partitioning

[3] A non-horizontal cut is a threshold in the hierarchy's altitude that can vary between branches, in contrast to a horizontal cut which is a fixed threshold across all branches.

(GASP) [97] was proposed to unify correlation and hierarchical clustering. GASP provides a framework that combines the benefits of both strategies, returning a flat clustering while simultaneously computing the hierarchical segmentation. Given this, one of our primary contributions in this thesis is a novel segmentation selection method that jointly optimizes segmentation and tracking by identifying the most temporally consistent segments within the hierarchy, as it will be described in Chapter 3.

2.3.1 Notations

As described above, a hierarchical segmentation comprises a sequence of nested partitions rather than a single flat partition. We denote a hierarchy as H, which is defined by a set of partitions $\{P_\lambda\}_{\lambda \in [0,\Lambda]}$, where λ represents the scale or merging level.

At the finest level, $\lambda = 0$, the partition consists of the set of all individual pixels, such that $P_0 = \{\{n\} \mid \forall\ n \in V^I\}$. Conversely, at the coarsest level, $\lambda \geq \Lambda$, the partition converges to a single region encompassing the entire image domain, $P_\Lambda = \{V^I\}$.

For any two levels λ_0 and λ_1 where $0 \leq \lambda_0 \leq \lambda_1 \leq \Lambda$, the partitions satisfy a nesting property. Such that, for every segment $S_i^{\lambda_0} \in P_{\lambda_0}$, there exists a segment $S_j^{\lambda_1} \in P_{\lambda_1}$ such that:

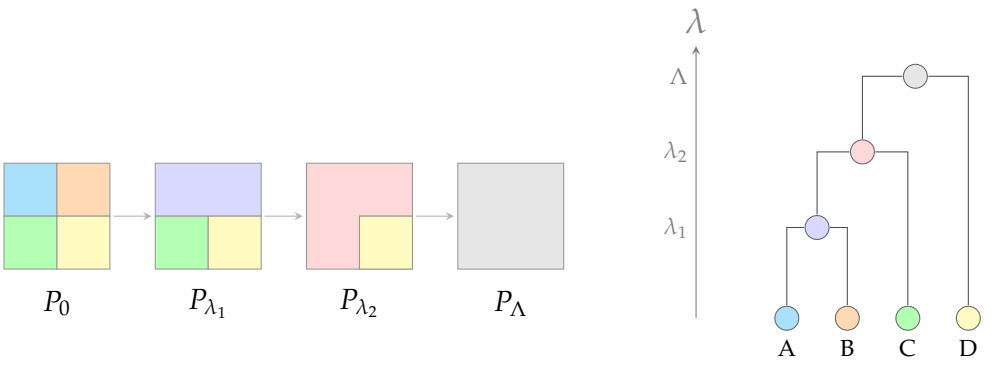
$$S_i^{\lambda_0} \subseteq S_j^{\lambda_1}. \tag{2.2}$$

Therefore, for any two segments $S_i^{\lambda_0} \in P_{\lambda_0}$ and $S_j^{\lambda_1} \in P_{\lambda_1}$, they are either nested, $S_i^{\lambda_0} \subseteq S_j^{\lambda_1}$, or completely disjoint, $S_i^{\lambda_0} \cap S_j^{\lambda_1} = \emptyset$.

A hierarchy H can be represented as a tree (or dendrogram) where each node represents a segment S_i^λ. The leaves of the tree represent the segments at the finest level (P_0). An internal node $S_i^{\lambda_1}$ at level λ_1 is connected to its children $\{S_j^{\lambda_0}\}$ from a finer level λ_0 if and only if the parent is the union of its maximal descendants:

$$S_i^{\lambda_1} = \bigcup_{S_j^{\lambda_0} \in \text{children}(S_i^{\lambda_1})} S_j^{\lambda_0}. \tag{2.3}$$

As illustrated in Figure 2.6, this structure allows for a compact representation of the image data, where internal nodes encapsulate regional information at varying levels of abstraction.



(a) Sequential nested partitions P_λ (b) Dendrogram representation

Figure 2.6: Hierarchical segmentation and its tree representation. **a**, A sequence of partitions evolving from the finest level (P_0) to the coarsest (P_Λ) as segments merge. **b**, The corresponding dendrogram where nodes represent segments at different scales λ.

2.4 Multi-Object Tracking

Object tracking is the process of maintaining object correspondences across a temporal sequence. One of the earliest and most enduring approaches is the Kalman filter [98], developed in the 1960s. It estimates the state (position and velocity) of a single object over time by predicting its state in the next frame and subsequently updating these estimates based on new observations.

With the widespread availability of high-resolution sensors (*e.g.* digital cameras), the scope of the field has expanded from tracking single entities to Multi-Object Tracking (MOT), which has become an integral part of modern computer vision. Traditionally, MOT was approached as a two-step *tracking-by-detection* pipeline: objects are first detected in each frame and then associated across time [99, 100, 101].

Recently, there has been a resurgence in joint detection and tracking frameworks. These contemporary approaches generally fall into two categories:

(i) **Graph-based approaches** [102, 103, 104]: These methods formulate tracking as a clustering problem on a graph. Bounding boxes or low-level keypoints are represented as nodes, with spatial and temporal edges encoding potential associations. The problem is often solved using variants of the multi-cut (correlation clustering) algorithm [105, 68], where the resulting connected components represent individual tracks.

(ii) **Sequence prediction approaches** [106, 107]: These methods extend Transformer-based architectures for image detection [108] by treating the association as a sequence prediction task and simultinously detecting (segmenting) objects. While compelling, these models require vast amounts of training data and are computationally intensive, as the attention mechanism scales quadratically with the number of objects. Furthermore, they often struggle with cell division (*i.e.* mitosis), as tracks are typically associated with a single token. Consequently, their translation to the biological domain remains limited [109].

MOT applications span from surveillance [110, 111] to autonomous vehicle navigation [112]. Challenges include tracking fast-moving objects in dense environments, managing occlusions, and handling missed detections. Unlike cell tracking, objects in natural scenes often possess distinct features (*e.g.* color, texture, or shape) that facilitate re-identification despite frequent occlusions, Figure 2.7.

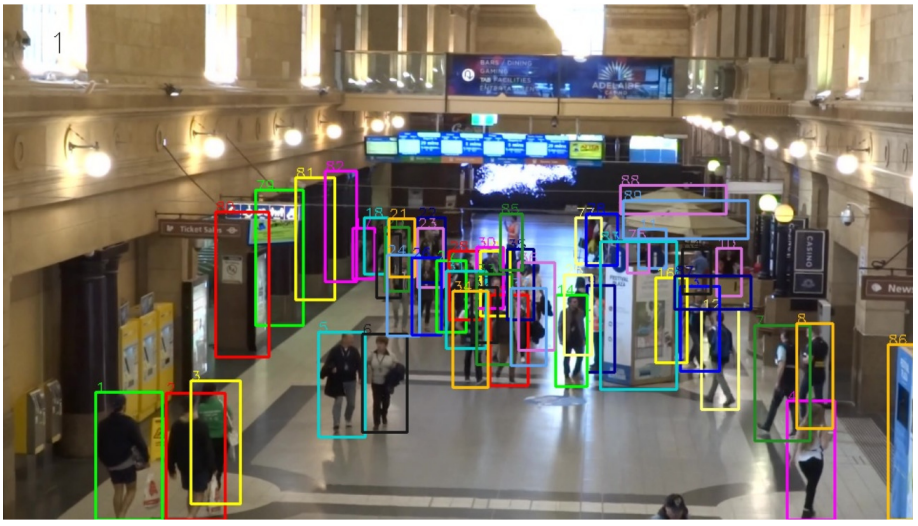


Figure 2.7: A frame from the MOT20 dataset with object bounding boxes overlayed, each unique color represents a distinct object. Image adapted from MOT20 [111].

Translating these approaches into the biological domain highlights several fundamental differences between traditional MOT and cell tracking that are often overlooked, such as:

- **High Object Density**: Microscopy datasets often contain several orders of magnitude more objects than standard MOT benchmarks. For example, the videos from

MOT17 [110] and MOT20 [111] average 30 and 139 objects per frame, respectively. In contrast, a simple 2D cell culture video, Figure 4.3, may contain over 300 objects, while 3D time-lapses of developing embryos, Figure 4.11, can exceed 27,000 objects per frame — a nearly 200-fold increase.

- **Limited Data Availability**: While natural imaging data can be crowdsourced or captured via vehicle-mounted cameras, biological experiments require specialized equipment and expensive reagents, leading to significantly smaller datasets.
- **Annotation Complexity**. Despite the success of self-supervised learning [113, 114], supervised methods remain superior for instance segmentation [40, 115, 41]. Additionally, annotating 3D biological data is particularly arduous due to the massive volume size and the limitations of 2D interface interactions.
- **Morphological Similarity**: In fluorescence microscopy, cells often appear nearly identical, with only specific structures (like nuclei) visible. Consequently, cell tracking cannot rely heavily on texture-based features for object identification (association).
- **Signal Quality Trade-offs**: Experiments are often pushing the boundaries of what can be seen in the microscope, and trade-offs over the different aspects of imaging are required. For example, for a longer imaging session, exposure must be minimized, resulting in low signal-to-noise ratios or low frame rates.
- **Cell Proliferation**: Unlike most objects in natural scenes, cells undergo division. This rapid change in appearance and the resulting increase in object count represent a significant challenge that traditional MOT algorithms are rarely designed to handle.

These distinctions highlight the critical need for specialized cell tracking algorithms. In the following section, we formalize the Integer Linear Programming (ILP) formulation, which is a valuable part of object tracking algorithms and lies at the core of our proposed tracking approach, before reviewing cell tracking methods in Section 2.5.

2.4.1 Network Flow for Multi-Object Tracking

Integer Linear Programming (ILP) is a powerful tool for solving constrained optimization problems, where the constraints and the objective function are composed of linear equations and inequalities. The *integer* part of the formulation requires the resulting solution to be a discrete value, in our case, often a binary value, $\{0, 1\}$. Network flow is

a special case of ILPs where the restrictions can be abstracted to network capacities (*i.e.* bandwidth) and flow conservation constraints.

Hungarian Matching

When a disjoint set of segmentations (or detections) is available for each frame, a standard approach for establishing correspondences between adjacent frames is the Hungarian algorithm [116].

Let P and P' be two sets of segments representing objects in adjacent frames. We define an association score function $w^S : P \times P' \mapsto \mathbb{R}^+$, which quantifies the similarity or likelihood of a link between any pair of segments $(S_i, S_j) \in P \times P'$. The Hungarian algorithm seeks to maximize the total sum of these association scores by determining an optimal assignment such that each segment is assigned to at most one other segment in the adjacent frame.

More formally, x_{ij} is defined as a binary indicator variable such that $x_{ij} = 1$ if segment $S_i \in P$ is linked to $S_j \in P'$, and $x_{ij} = 0$ otherwise.

Efficient solvers for this linear assignment problem exist, most notably the Jonker-Volgenant algorithm [117], which achieves a time complexity of $O(n^3)$, where n is the maximum number of segments between the two sets, $\max(|P|, |P'|)$. While these specialized algorithms are more efficient, the problem can be expressed as a general ILP. This formulation is didactic and serves as a building block for the more complex tracking models introduced later in this thesis.

The ILP formulation for the Hungarian algorithm is:

$$\arg\max_{\mathbf{x}} \quad \sum_{S_i \in P} \sum_{S_j \in P'} x_{ij} w^S(S_i, S_j) \tag{2.4}$$

$$\text{s.t.} \quad \sum_{S_i \in P} x_{ij} \leq 1 \quad \forall\, S_j \in P' \tag{2.5}$$

$$\sum_{S_j \in P'} x_{ij} \leq 1 \quad \forall\, S_i \in P \tag{2.6}$$

$$x_{ij} \in \{0, 1\} \quad \forall\, (S_i, S_j) \in P \times P' \tag{2.7}$$

Equation 2.5 and Equation 2.6 enforce the one-to-one matching constraint, ensuring that each segment in one frame is associated with no more than one segment in the next. To track an entire time-lapse, this matching process is applied sequentially between every pair of consecutive frames.

The limitations of this approach become more pronounced when the number of objects varies between frames, as it occurs when elements enter or leave the field of view. While the linear assignment problem can be adapted to handle such imbalances, for instance, by augmenting the cost matrix with dummy nodes or introducing slack variables. A more general framework that avoids stiching consecutive frames is provided by a single global *network flow* formulation, which will be described next.

Network Flow

Zhang *et al*. [99] extended the bipartite matching framework with a more sophisticated formulation that optimizes all assignments across the entire sequence of frames with a single ILP formulation while accounting for the costs of track initialization (entry) and termination (exit).

In this network flow formulation, the tracking problem is modeled as a directed acyclic graph where nodes represent detected segments and edges represent potential temporal associations. The objective is to maximize the total score of the selected paths and entry and exit penalizations. The ILP formulation for a time-lapse of T frames is defined as:

$$\arg\max_{\mathbf{x},\mathbf{y}} \quad \sum_{t=1}^{T-1} \sum_{i \in P_t} \sum_{j \in P_{t+1}} w_{ij} x_{ij} + \sum_{t=1}^{T} \sum_{i \in P_t} (w_i y_i + w_\alpha x_{\alpha i} + w_\beta x_{i\beta}) \tag{2.8}$$

$$\text{s.t.} \quad x_{\alpha j} + \sum_{i \in P_{t-1}} x_{ij} = y_j \quad \forall\, j \in P_t, \forall\, t \in \{1, \ldots, T\} \tag{2.9}$$

$$x_{i\beta} + \sum_{j \in P_{t+1}} x_{ij} = y_i \quad \forall\, i \in P_t, \forall\, t \in \{1, \ldots, T\} \tag{2.10}$$

$$x_{ij}, x_{\alpha i}, x_{i\beta}, y_i \in \{0, 1\} \tag{2.11}$$

Here, y_i is a binary variable indicating whether segment i is included in a track, and x_{ij} indicates an association between segments i and j. The weights w_{ij} is a simplified

notation of the association scores $w^S(S_i, S_j)$ previously defined. The variables $x_{\alpha i}$ and $x_{i\beta}$ represent entry and exit events, respectively, with their associated weights w_α and w_β acting as penalties (typically $w_\alpha, w_\beta < 0$). These penalties prevent the model from creating trivial, short-lived tracks to maximize the objective function.

Equation 2.9 and Equation 2.10 are the *flow conservation constraints*. They ensure that for any selected segment $y_i = 1$, there must be exactly one incoming flow (either a birth $x_{\alpha i}$ or a link from a previous frame) and exactly one outgoing flow (either a death $x_{i\beta}$ or a link to a subsequent frame).

To gain an intuitive understanding of this formulation, consider a single-object scenario where a path π consists of a sequence of associations $\{x_{01}, x_{12}, \ldots, x_{(n-1)n}\}$. The total score for this path is:

$$\text{score}(\pi) = w_\alpha + w_\beta + \sum_{t=0}^{n-1} w_{t,t+1} + \sum_{t=0}^{n} w_t \tag{2.12}$$

Therefore, any path π is included in the optimal solution only if $\text{score}(\pi) > 0$ because an empty solution with a score of zero is preferred over a negative score. In other words, a track is maintained only when the cumulative association and node weights exceed the combined cost of entering and leaving the field of view:

$$\sum_{t=0}^{n-1} w_{t,t+1} + \sum_{t=0}^{n} w_t > -(w_\alpha + w_\beta) \tag{2.13}$$

This property demonstrates how the network flow formulation inherently filters out noise by requiring that valid tracks demonstrate sufficient evidence of temporal persistence.

2.5 Cell Tracking

As highlighted when reviewing **Multi-Object Tracking (MOT)**, Section 2.4, cell tracking is a particularly challenging problem due to the massive number of objects, the limited availability of annotated training data, poor image quality, and the high morphological similarity between individuals. Furthermore, the varying number of objects caused by

cell division requires specialized algorithms. While some paradigms are borrowed from natural image MOT, the most successful biological methods are often specifically engineered to handle these constraints.

Consistent with the broader MOT context, cell tracking approaches can be categorized into: (i) *tracking-by-detection*, which first identifies objects in each frame and subsequently associates them over time; and (ii) *joint segmentation and tracking*, which simultaneously optimizes both segmentation and association selection. However, unlike in MOT, joint segmentation and tracking have not been widely adopted due to the computational complexity of large microscopy datasets and limited annotated data availability.

Tracking-by-Detection and Embedding-based Methods

Recent tracking-by-detection methods leverage deep neural networks to estimate associations between objects. Caliban [118] and the work of Ben-Haim and Raviv [56] use Graph Neural Networks (GNNs) to classify edges within cell lineages. Trackastra [48] employs a Transformer-based architecture to learn high-dimensional embeddings for each object, using their high-dimensional similarity to score temporal links.

Another common strategy involves using a single convolutional network to concurrently estimate object locations and association scores. Because a single model predicts both segmentation and association, it may appear to be joint segmentation and tracking. However, tracking is solved in a two step approach, where the detections are fixed and not influenced by the tracking step, except in cases where they are discarded for failing to contribute during the associations. EmbedTrack [44] adapted the instance segmentation framework of Neven *et al*. [119] to estimate segmentation masks alongside an embedding map used for inter-frame association. CELLECT [57] avoids explicit segmentation, instead estimating centroids and processing pixel-level embeddings with a multi-layer perceptron to compute association scores.

Flow-based methods [120, 121, 43] operate similarly but focus on estimating the motion of objects rather than object-level feature embeddings. This motion is used to advect locations from one frame to the next, with scores based on the proximity of the advected location to actual next frame detections. Malin-Mayor *et al*. [45] and Hirsch *et al*. [122]

extended these concepts by proposing loss functions that permit sparse annotation of centroids and links. Notably, they employ an Integer Linear Programming (ILP) solver to obtain lineages, providing a more robust alternative to the heuristics used in earlier flow-based work [120, 121, 43].

The use of sparse annotations or data curation suggestion systems [123, 124, 125] is a recurring theme, as manual annotation of hundreds of thousands of cells is rarely feasible. In Section 4.8, we describe a protocol for obtaining high-quality sparse annotations for 3D datasets using sparse fluorescence labeling.

Joint Segmentation and Tracking

Most similar to our proposed approach are joint segmentation and tracking methods, where the final cell segments are determined during the tracking process. In [53], the authors compute a hierarchical segmentation by fitting ellipsoids to a binary map of cell contours, estimate association scores using gradient boosting [126], and resolve tracks via a network flow ILP. TGMM [22] computes segmentation and tracking by fitting a mixture of Gaussians (GMM) to image intensities representing each nuclei, employing a hierarchical segmentation of supervoxels and initializing the GMMs with the previous frame's parameters for an accelerated model fitting.

Conservation tracking [51] departs from the traditional binary ILP formulation and uses an ILP to estimate the number of objects within each connected components of a foreground map, effectively handling common under- and over-segmentation errors before splitting multi-object components via Gaussian mixture fitting as a post-processing step. Mother Machine Tracking [52] employs parametric max-flow graph-cuts [127] to obtain an inclusion hierarchy of segments, distinguishing it from our partition-tree approach [85], and a specialized ILP for one-dimensional plus time tracking.

Jug *et al*. [54] introduced (and later improved [55]) the Moral Lineage Tracing (MLT) problem, where tracking is formulated as a minimum-cost multi-cut (sum-cut) subject to biological constraints: cells can divide but not merge, and may enter or leave the field of view. This extends both the spatial multi-cut literature [67, 68, 128, 20] and the MOT multi-cut approaches [102, 103, 104] by incorporating lineage constraints. Despite

its mathematical elegance, the NP-hardness of the multi-cut problem plus the biological constraints typically limits its application to medium-sized datasets.

Scalability remains a critical gap in the literature. With few exceptions, such as [45], deep learning methods often struggle to scale to massive 3D datasets due to memory limitations or the requirement for unavailable 3D annotations. Similarly, many non-deep-learning methods cannot scale to hundreds of gigabytes because of algorithmic inefficiencies [54, 53, 55], with notable exceptions being TGMM [22] and the work of Magnusson *et al*. [129]. The work in this thesis addresses this gap by providing a scalable and versatile approach to cell tracking.

Summary and Conclusions

This chapter has established the foundational concepts for large-scale cell tracking in biological imaging. We began by defining the **instance segmentation** problem, transitioning from graph-based approaches like the watershed transform to modern deep learning architectures. We then explored **hierarchical segmentation**, specifically partition hierarchies and their visual representation, Ultrametric Contour Maps (UCM), which preserve nested object relationships and provide a multi-scale representation of the data.

In reviewing **Multi-Object Tracking (MOT)**, we identified that specialized solutions for the biological domain are beneficial to the nature of microscopy data. Finally, we analyzed current **cell tracking** methodologies, noting a critical trade-off between the joint optimization of lineages and the computational scalability required for massive 3D datasets, such as whole embryos.

The gaps identified in this literature, particularly the need for versatile and scalable joint segmentation and tracking, form the primary motivation for this thesis. In the following chapter, we introduce our proposed approach, which can take arbitrary segmentation as input and jointly optimizes segment selection from hierarchies while maximizing temporal association.

Chapter 3

Large-Scale Joint Segmentation and Tracking

The preceding chapter established the specific requirements for biological cell tracking, emphasizing the limitations of adapting traditional computer vision algorithms to large-scale microscopy. This chapter describes the proposed approach, Ultrack, an algorithm designed for the large-scale joint segmentation and tracking of cells.

The framework is based on the fundamental principle that tracking accuracy is inherently capped by the quality of the underlying segmentation, as inaccurate segmentations inevitably lead to a cascading effect of errors throughout the tracking process. To mitigate this, Ultrack uses hierarchical segmentations to provide a diverse pool of candidate segments while avoiding an overwhelming number of possible combinations.

From the clustering-centric perspective of instance segmentation, this methodology addresses the following question:

> *Can temporal information across frames be leveraged to improve the selection of spatial segments?*

As illustrated in Figure 3.1, individual frames can contain insufficient information for a definitive segmentation, even for highly accurate segmentation models or human experts. However, the temporal sequence provides additional constraints that clarify the true underlying structure.

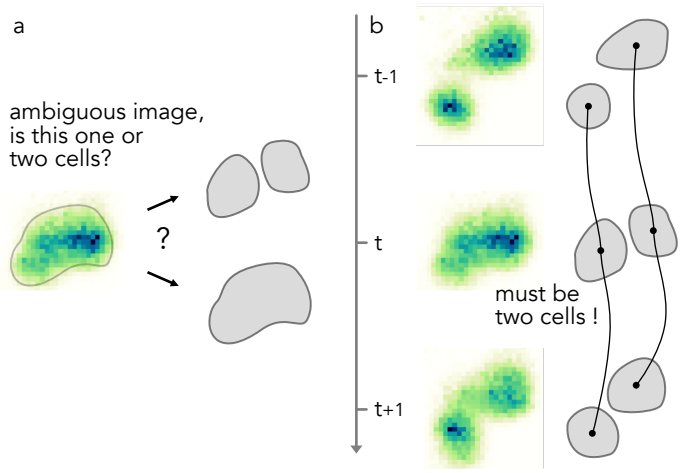


Figure 3.1: Example of segmentation ambiguity resolved by temporal information. **a**, An ambiguous nuclear-labeled region where determining the segmentation and the number of cell instances is difficult from a single frame. **b**, The temporal sequence of the same region $(t-1, t, t+1)$ where the existence of two distinct cells is resolved by the information of the neighboring frames.

To address this, we formulate the joint segmentation and tracking problem as an optimal hierarchical cut. Unlike methods that attempt to enumerate every possible segment combination, our hierarchical formulation constrains the search space while maintaining a high-recall pool of hypotheses. This allows the system to identify the correct segmentation by evaluating temporal evidence over multiple frames, even on terabyte-scale datasets.

While this chapter focuses on the formal methodology, Chapter 4 provides the experimental results that validate these technical choices. To maintain clarity for the reader, each section within this chapter concludes with a reference to the corresponding empirical assessment in the results chapter.

The remainder of this chapter is organized as follows:

- Section 3.1, presents the novel formulation for performing non-horizontal cuts in hierarchical segmentations and its translation into a joint tracking problem.
- Section 3.2, details the method for constructing segmentation hypotheses.
- Section 3.3, describes the specific approach used for contour map generation during the Cell Tracking Challenge.

- Section 3.4, and Section 3.5, cover the association of hypotheses between frames and their registration using flow fields, respectively.
- Section 3.6, describes the distributed implementation of the algorithm for terabyte-scale datasets.

3.1 Cell Tracking as an Optimal Hierarchical Cut Problem

To avoid committing to a single potentially erroneous segmentation, we employ hierarchies of candidate segmentations. Where, ideally, the true cell instance will be one of these candidate segments. Therefore, the candidate segments should provide *high recall* even at the potential expense of *precision*. This approach differs from previous joint segmentation methods that combinatorially combine superpixels during tracking [54, 55] because the set of possible segments is much smaller than all possible partitions, which enables the application of the framework to extremely large datasets.

While it is optimistic to assume that the hierarchy will always perfectly match the true segments, this approach can always identify a single region at a finer level that belongs to a unique true segment, even if it is at the cost of incomplete coverage.

To introduce our approach, and the motivation for developing the algorithm, we formulate a hypothetical problem of finding the instances in the hierarchy as an optimization problem to maximize the overlap between the *unknown* true instances and a selection of disjoint segments from the hierarchy. We refer to this as the maximum total intersection over union (MTIoU). Given a hierarchy H and a set of ground-truth segments $S^{gt} = \{S_1^{gt}, S_2^{gt}, \ldots, S_n^{gt}\}$, the set of disjoint segmentations with MTIoU can be determined

with the following ILP:

$$\arg\max_{\mathbf{x},\mathbf{y}} = \sum_{i \in H} \sum_{j \in S^{gt}} w_{ij} x_{ij} \tag{3.1}$$

$$\text{s.t.} \quad y_i = \sum_{j \in S^{gt}} x_{ij} \quad \forall\, i \in H \tag{3.2}$$

$$\sum_{i \in H} x_{ij} \leq 1 \quad \forall\, j \in S^{gt} \tag{3.3}$$

$$y_i + y_j \leq 1 \quad \forall\, i, j \in H \mid i \subset j \tag{3.4}$$

$$y_i \in \{0, 1\} \quad \forall\, i \in H \tag{3.5}$$

$$x_{ij} \in \{0, 1\} \quad \forall\, i, j \in H \times S^{gt} \tag{3.6}$$

where the association weight function w_{ij} is the intersection over union (IoU) between segment i and j. The variable x_{ij} indicates if i is the optimal match to j, while y_i indicates if segment i was selected, $y_i = 1$. Equation Equation 3.2 forces a single assignment per selected segment, Equation 3.3 forces at most one assignment per ground-truth segment, and Equation 3.4 restricts the solution to disjoint segments belonging to different branches of the hierarchy.

In practice, information about the true instances (*i.e.* ground-truth) is unavailable. We approximate this scenario by solving the segmentation selection between hierarchies of adjacent frames under the assumption of fine temporal resolution, where cell segments will overlap at specific nodes across hierarchies while spurious segments will not, under the assumption that mistakes are stochastic. By formulating the ILP across the entire temporal sequence and including biological constraints, such as cell division and track appearing or disappearing, the resulting segment selection and temporal association reconstruct the complete cell lineages.

Combining the previously described global multi-object tracking and MTIoU formulations, we define the following ILP for hierarchical multi-hypothesis tracking:

$$\arg\max_{\mathbf{x},\mathbf{y}} = \sum_{t\in[2..T]} \sum_{i\in H_{t-1}} \sum_{j\in H_t} w_{ij}x_{ij} + \sum_{i\in\mathcal{H}} (w_\alpha x_{\alpha i} + w_\beta x_{i\beta} + w_\delta x_{\delta i}) \tag{3.7}$$

$$\text{s.t.} \quad y_j = x_{\alpha j} + \sum_{i\in H_{t-1}} x_{ij} \;\; \forall\, j \in H_t, \forall\, t \in [1..T] \tag{3.8}$$

$$y_i + x_{\delta i} = x_{i\beta} + \sum_{j\in H_{t+1}} x_{ij} \;\; \forall\, i \in H_t, \forall\, t \in [1..T] \tag{3.9}$$

$$y_i \geq x_{\delta i} \;\; \forall \;\; i \in \mathcal{H} \tag{3.10}$$

$$y_i + y_j \leq 1 \quad \forall\, i,j \in H_t \mid i \subset j, \forall\, t \in [1..T] \tag{3.11}$$

$$y_i \in \{0,1\} \;\; \forall\, i \in \mathcal{H} \tag{3.12}$$

$$x_{ij} \in \{0,1\} \;\; \forall\, i,j \in H_{t-1} \times H_t, \forall\, t \in [2..T] \tag{3.13}$$

where H_t represents the candidate segments in frame t and $\mathcal{H} = \cup_t H_t$. Penalization parameters $w_\alpha, w_\beta, w_\delta$ correspond to appearance, disappearance, and division events, respectively. Boundary conditions are set such that $w_\alpha = 0$ at $t = 1$ and $w_\beta = 0$ at $t = T$, so appearing and disappearing events are not penalized at the first and last frame, respectively. Equations Equation 3.8 and Equation 3.9 enforce constant flow, allowing up to two outgoing segments for divisions, Equation 3.10. Finally, Equation 3.11 ensures only disjoint segments are selected within any frame. Figure 3.2 illustrates a visualization of the variables and constraints of this ILP formulation.

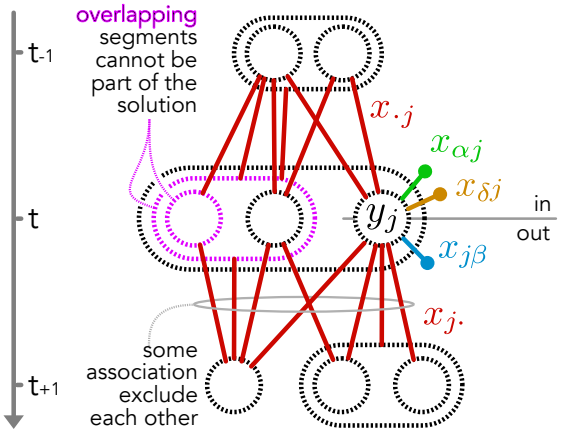


Figure 3.2: Hierarchical multi-hypothesis tracking ILP diagram. Candidate edges ($x_{\cdot\cdot}$) are represented in red. Nodes ($y_{\cdot}$) are represented in a nested manner, where leaves are circles and internal nodes are oblongs. Magenta lines represent a pair of competing (overlapping) hypotheses that exclude each other. Each node has a set of slack variables, shown only for node y_j for clarity. Variables above the node center ($x_{\alpha j}, x_{\delta j}, x_{\cdot j}$) represent incoming flow and below ($x_{j\beta}, x_{j\cdot}$) represent outgoing flow.

This formulation is the core of the algorithm, the following sections will describe how the individual components of the ILP are obtained.

3.2 Hypotheses Construction

Although our tracking formulation is not tied to a particular hierarchical segmentation method, we compute segmentation hierarchies using the hierarchical watershed framework [130, 131]. This framework constructs hierarchies in log-linear time (dominated by sorting) with respect to the number of edges in the underlying image graph, using a minimum-spanning-tree construction [132] and a union-find data structure [133].

For each frame t, we generate a set of candidate segments H_t from a binary foreground map F_t and a fuzzy contour map C_t provided by the user. The foreground F_t indicates the location of cells against the background, while C_t provides boundary evidence (high values indicate likely cell boundaries). Since connected components of F_t are disjoint by definition, we compute a hierarchy for each component independently and then combine them into a single set H_t, ignoring the background.

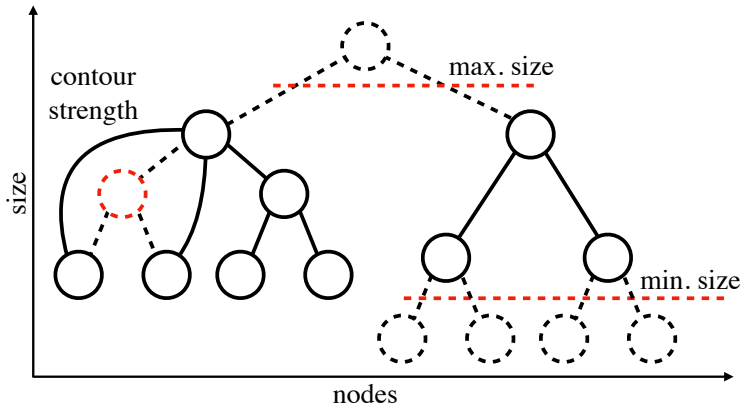


Figure 3.3: Hierarchy filtering by minimum size, maximum size, and contour strength. Dashed lines indicate removed elements. Leaves of the hierarchy are omitted for clarity.

Each hierarchy is constructed using the watershed hierarchy by area [134, 131] on an undirected weighted graph restricted to the corresponding connected component in F_t (the image graph construction and weighting are described in Section 2.2.1). In brief, edge weights w_{uv} are the average contour intensity values associated with the incident nodes.

Using the full hierarchies in the tracking ILP, Equation 3.7, can easily make it take a prohibitively long time to solve while including segments that are not relevant to the tracking problem. Therefore, we apply three criteria to remove irrelevant segments [135] and reduce the number of candidate solutions:

1. **Minimum size:** prune the hierarchy below a minimum size threshold;
2. **Maximum size:** cut the hierarchy above a maximum size threshold;
3. **Non-relevant contours:** fuse segments whose average contour strength is below a threshold.

The minimum and maximum cell sizes are typically known for a given experiment, while the non-relevant-contours threshold can be tuned by visually inspecting under- and over-segmentation. Figure 3.3 shows a cartoon of how a hierarchy is pruned.

Algorithm 1 summarizes the single-pass procedure used to generate segmentation hypotheses for one time point. While the overall construction follows the Kruskal-style watershed hierarchy formulation [132], our implementation must satisfy an additional

Input: F_t: foreground map of frame t
C_t: contour map of frame t
min_size: size lower threshold
max_size: size upper threshold
min_contour: contour strength threshold
Output: H_t: set of segmentation hypotheses for frame t

$H_t \leftarrow \{\}$
$CC \leftarrow$ connected components of F_t
for $cc \in CC$ **do**
  $G_{cc} \leftarrow$ edge weighted graph from C_t for pixels in cc
  $MST_{cc} \leftarrow$ Minimum Spanning Tree of G_{cc}
  $E'_{cc} \leftarrow$ edges of MST_{cc} ordered by attribute (area) [132]
  `// Kruskal-like tree construction`
  $UF \leftarrow$ Tarjan's union-find data structure [133]
  $BPT \leftarrow$ binary partition tree data structure
  $count \leftarrow 0$
  $tree\_map \leftarrow \{v : v \; \forall \; v \in V_{cc}\}$
  for $(u, v) \in E'_{cc}$ **do**
    $root\_u \leftarrow UF.find(u)$
    $root\_v \leftarrow UF.find(v)$
    if $root_u == root_v$ **then**
      continue
    end
    $tree\_new \leftarrow BPT.merge(tree\_map[root_u], tree\_map[root_v])$
    if *tree_new is a watershed and* $w_{uv} > min\_contour$ **then**
      for $n \in \{u, v\}$ **do**
        if *n is valid according to min_size, max_size* **then**
          $H_t \leftarrow H_t \cup UF.component(n)$
          $count \leftarrow count + 1$
        end
      end
    end
    $root\_new \leftarrow UF.union(root\_u, root\_v)$
    $tree\_map[root\_new] \leftarrow tree\_new$
  end
  if $|cc| < max\_size$ *or* $count = 0$ **then**
    $H_t \leftarrow H_t \cup \{cc\}$
  end
end
return H_t

Algorithm 1: Ultrack's hypotheses construction pseudo code for a single time point

requirement that is not addressed by the standard minimum-spanning-tree with union-find construction: we need to *materialize* a large number of intermediate regions as explicit masks for downstream tracking, while keeping the overall runtime close to the log-linear complexity of the original algorithm.

To meet this requirement, we maintain three synchronized views of the evolving partitions. First, the union-find structure (UF) enables fast disjoint set queries during the Kruskal sweep (testing whether two nodes already belong to the same component). Second, because union-find does not retain merge history, we explicitly store the merge tree using a binary partition tree (BPT), which encodes the hierarchical relationships between regions as edges are processed. Third, we keep an associative map ($tree\_map$) from current union-find representatives to the corresponding nodes in the BPT, so that each union operation updates both the connectivity structure and the hierarchy consistently.

Crucially, hypotheses masks must be extracted *online* during the sweep because at the last iteration they have all been merged into the root. Many of the validity checks (*e.g.* whether a node corresponds to a watershed event and whether its children would exist given the contour criteria) can only be evaluated once the parent merge is observed, consequently, candidate segments are emitted from within the main edge iteration loop, when the required information becomes available. This design turns hypotheses generation into part of the hierarchy construction itself, rather than a separate post-processing step, and avoids an additional traversal of the full tree. In practice, this coupling between (i) union-find connectivity, (ii) explicit merge-tree bookkeeping, and (iii) online mask extraction is what allows us to produce a compact set of high-quality hypotheses without sacrificing the efficiency of Kruskal-style watershed hierarchies.

Because each foreground, F_t, connected component is processed independently, the resulting per-component hierarchies are then combined into H_t. Finding the connected components is linear in the number of pixels in the foreground map. For each component cc, the algorithm proceeds as follows. In Line 5, we compute the minimum spanning tree of the component graph, with runtime $O(|E_{cc}| \log |E_{cc}|)$. In Line 6, we derive the attribute used by the watershed-by-area construction and sort the MST edges accordingly, which takes $O(|V_{cc}|)$ to compute the attribute values and $O(|V_{cc}| \log |V_{cc}|)$ for sorting the

edges with the new attribute value as we have $V_{cc} - 1$ edges at this point after the MST construction.

The subsequent lines initialize the required data structures and iterate over the sorted edges in a Kruskal-like sweep [136]. At each step, UF is used to skip edges whose endpoints already belong to the same component, and otherwise to merge two components. To avoid generating non-informative candidates, we only emit segments at watershed events (Line 18), *i.e.* merges whose parent combines multiple minima. Importantly, whether a region is a valid watershed is determined at the time its parent merge is formed, therefore, hypotheses are emitted for the children at the moment the parent is processed (Line 19), rather than emitting the parent itself. After any hypotheses are recorded, the corresponding union and tree merge are performed (Line 26) and the sweep advances to the next edge. Finally, after all edges have been processed, it checks if the root is a valid hypothesis or if filtering was too strict to omitting all hypotheses ($count = 0$), if so, we add the connected component mask itself to H_t (Line 30).

The dominant cost per component is the log-linear edge sorting. Union-find operations ($find$ and $union$) run in quasi-constant amortized time (inverse Ackermann) [133], and maintaining the explicit merge tree and $tree\_map$ adds only constant-time bookkeeping per processed edge. To support online mask materialization, we augment union-find with an explicit representation of each set's members: each component maintains a linked list of its elements, and the union operation concatenates these lists when merging two sets in constant time. As a result, $UF.union(\cdot)$ retains its intended amortized behavior, querying a component with $UF.component(\cdot)$ takes constant time, $O(1)$, and materializing the mask takes time proportional to the component size, $O(|V_{cc}|)$.

Therefore, the main additional cost relative to the standard hierarchy construction comes from repeated calls to $UF.component(\cdot)$. In the worst case, with several minima and a completely unbalanced tree, component extraction would be triggered many times and yield a quadratic upper bound, $O(|V_{cc}|^2)$, $O(|V_{cc}|)$ for the MST edge iteration times $O(|V_{cc}|)$ for the component extraction. In practice, however, the size and contour filters, the balanced structure induced by the watershed-by-area attribute substantially reduce the number and the size of emitted candidates. Consistent with this, profiling shows runtimes dominated by the sorting term $O(|E_{cc}| \log |E_{cc}|)$, matching the expected behavior

of an efficient hierarchical watershed construction.

3.3 Contour Map Generation

The hierarchy construction, and therefore, Ultrack, relies on two complementary image-derived signals to generate segmentation hypotheses: A binary *foreground* map F_t that indicates pixels belonging to cells against the background, and a fuzzy *contour* (boundary evidence) map C_t that highlights likely cell boundaries. Where, for the hierarchy construction, F_t defines the domain of interest, while C_t determines the nesting structure of the candidate segments. In this section, we describe (i) how these maps are obtained from existing instance segmentation labels and (ii) how they are predicted directly from microscopy images using a neural network when dense annotations are available.

3.3.1 From instance labels to foreground and contour maps

Let L_t denote an instance-labeled segmentation at time t, where $L_t(x) \in \{0, 1, \dots, M_t\}$, M_t is the number of labels at time t, and 0 denotes background. When multiple label maps are available, $\{L_t^{(0)}, L_t^{(1)}, \dots, L_t^{(K-1)}\}$ (*e.g.* from different algorithms or varied parameters), they are converted into a unified foreground map F_t and contour map C_t.

Foreground map. For each label map, a binary foreground indicator is formed by thresholding non-background pixels:

$$F_t^{(k)}(x) = \mathbb{1}[L_t^{(k)}(x) \neq 0]$$

If several label maps are provided, they are combined via a pixelwise logical OR, which is equivalent to an element-wise maximum:

$$F_t(x) = \max_k F_t^{(k)}(x)$$

This ensures that any pixel assigned to an object in at least one input map is treated as foreground.

Contour map. A binary contour map is derived from each label map by marking pixels whose neighborhood contains a different label, using a fixed neighborhood $\mathcal{N}(x)$ (*e.g.*, 8-connectivity in 2D or 26-connectivity in 3D) such that:

$$C_t^{(k)}(x) = \mathbb{1}\left[\exists\, y \in \mathcal{N}(x) \text{ such that } L_t^{(k)}(y) \neq L_t^{(k)}(x)\right]$$

Multiple maps are aggregated by averaging,

$$C_t(x) = \frac{1}{K}\sum_k C_t^{(k)}(x)$$

yielding values in $[0, 1]$. Since the downstream hierarchy construction depends only on the relative ordering of contour strengths, with the exception of the contour strength threshold, averaging provides a bounded dynamic range while preserving the consensus of the input labels.

Both aggregation strategies are used in Section 4.2 to combine segmentations from different parameter settings for the same deep learning model and in Section 4.4 to integrate multi-channel and multiple algorithms results.

In the applications with intensity-based tracking, as in Section 4.3, a function of the image intensities can be used as the contour map C_t directly, effectively bypassing discrete labels and intermediate segmentation steps.

3.3.2 Predicting foreground and contours with neural networks

When dense segmentation annotations are available, F_t and C_t can be predicted directly from microscopy images using a neural network. This supervised approach was employed for the Cell Tracking Challenge submissions, Section 4.6, and the terabyte-scale zebrafish time-lapse tracking, Section 4.9.

We employed an U-Net [75] architecture that maps a single-channel grayscale input to a two-channel output: a foreground probability map and a contour probability map. The model is a four-level encoder-decoder design with depths of $32, 64, 128$, and 256, using 5×5 convolutional kernels. In the decoder, feature maps are upsampled via linear interpolation and fused with encoder activations through skip connections.

Training is performed by minimizing a Dice loss [137] on both output channels because the Dice score handles class imbalances directly, while traditional binary cross-entropy loss does not. To improve boundary localization, auxiliary losses are added at three intermediate decoder resolutions, following the holistically-nested edge detection framework [138]. The final output are transformed to $[0, 1]$ range through sigmoid activations. Figure 3.4 shows an example of the predicted foreground and contour probability maps.

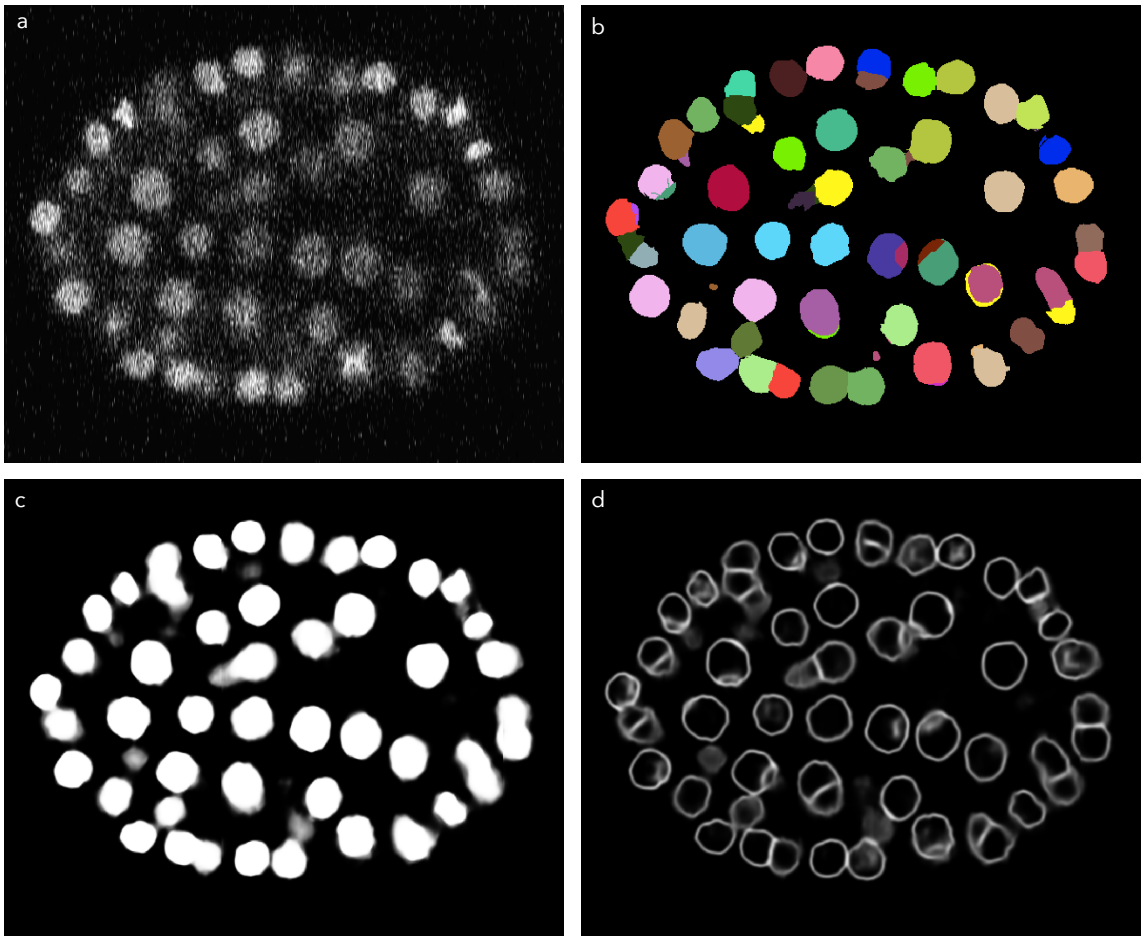


Figure 3.4: U-Net for foreground and contour prediction. **a**, Input image. **b**, Ground truth labels. **c**, Predicted foreground probability map. **d**, Predicted contour probability map.

3.4 Hypotheses Inter-Frame Association

Following the generation of per-frame segmentation hypotheses, Section 3.2, the next stage of the pipeline involves proposing plausible temporal correspondences between hypotheses in consecutive frames. These correspondences define the set of candidate edges used by the tracking ILP. They constrain the search space to biologically and geometrically feasible associations while assigning a similarity score to each candidate edge.

3.4.1 Candidate association graph

Let $\mathcal{H} = \bigcup_t H_t$ denote the set of all segmentation hypotheses across the temporal sequence, and let $t(i)$ be the time index of hypothesis $i \in \mathcal{H}$. We construct a directed set of candidate temporal edges $E^T \subseteq \{(i,j) \in H_t \times H_{t+1}\}$, where candidate edge $(i,j) \in E^T$ indicates that hypothesis i at time t may be associated with hypothesis j at time $t+1$. Each candidate edge is assigned a weight w_{ij} quantifying the similarity between the two hypotheses.

To avoid the $|H_t| \times |H_{t+1}|$ comparisons between all hypotheses, we assume a limited displacement of cells between adjacent frames. For each hypothesis $j \in H_t$, we compute its centroid and query its $2k$ nearest neighbors among centroids in H_{t-1} within a predefined radius using a KDTree [139]. This yields at most $2k$ candidate predecessors per hypothesis, which may be optionally refined using additional cues, Section 3.4.2. Finally, we retain up to k candidate edges with the highest similarity score (with ties broken by smaller centroid distance) for inclusion in E^T.

The association score is computed using the Intersection-over-Union (IoU) between the binary masks of the two hypotheses for the $2k$ candidates identified in the initial search. Therefore, let S_i denote the pixel or voxel set of hypothesis i, the association score is defined as:

$$\mathrm{IoU}(i,j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}, \qquad w(i,j) = \mathrm{IoU}(i,j)^{\gamma},$$

where $\gamma = 4$ is used in all experiments. Raising the IoU to a power greater than one, lets the additive cost function favor fewer, higher-confidence links than several low-confidence links. This effect is illustrated in Figure 3.5, where for $\gamma = 1$, the $A - C$

and $B - D$ would be preferred over $AB - CD$, despite $AB - CD$ being a perfect match with $IoU(AB, CD) = 1.0$.

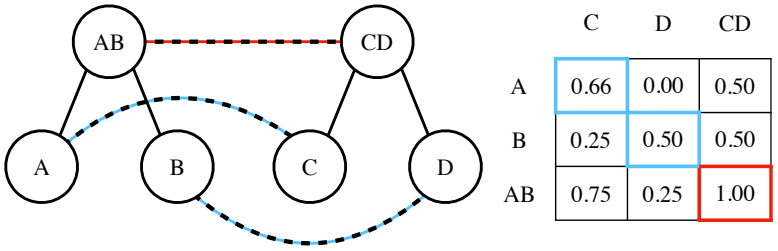


Figure 3.5: ILP power IoU association example. Two simple hierarchies are represented on the left. The right figure shows the weight matrix w containing association scores w_{ij}. Blue edges represent the optimal matching obtained with $\gamma = 1$ (linear IoU). Red edges represent the solution obtained with $\gamma \geq 2$, which accentuates high-overlap candidates and favors fewer, higher-confidence associations.

3.4.2 Optional intensity and color feature filtering

In multi-channel acquisitions, intensity information provides a supplementary cue to reject improbable connections. Ideally, hypotheses representing the same cell across consecutive frames exhibit similar intensities statistics in each channel, whereas links between distinct cells typically yield larger discrepancies. This cue is applied as a *hard filter* on the candidate edges produced by the nearest neighbor search.

Let $I_p \in \mathbb{R}^c$ denote the image intensity vector at pixel p across c channels. For a hypothesis i with mask S_i, we compute the per-channel mean and standard deviation:

$$\mu_i = \frac{1}{|S_i|} \sum_{p \in S_i} I_p, \qquad \sigma_i = \sqrt{\frac{1}{|S_i|} \sum_{p \in S_i} (I_p - \mu_i)^2},$$

Note that, μ_i and $\sigma_i \in \mathbb{R}^c$. For a candidate association (i, j), the edge is accepted only if:

$$\max \left| \frac{\mu_i - \mu_j}{\sigma_j} \right| \leq z,$$

where j is treated as the reference node at time t. The normalization by σ_j provides an intensity-invariant criterion, the threshold z represents the maximum allowed z-score in

units of intensity standard deviations between the mean color of the candidate predecessor and the reference. The maximum across all channels is conservative, such that a substantial deviation in any single channel results in the rejection of the connection.

This filtering step is computationally efficient, as it is restricted to the $O(|H_t|2k)$ candidate pairs from the initial search rather than the full set of cross-frame pairs. In all multi-color evaluations, $z = 3.0$ is used, corresponding to a symmetric 3σ criterion (values outside this range are equivalent to a 0.0027 probability under a normal distribution). The utility of this color-feature association for resolving ambiguous scenarios is demonstrated in Section 4.4.

3.5 Hypotheses Warping with Flow Fields

Our association score assumes fine temporal resolution, when that is not the case, its accuracy is compromised because cells can move significantly resulting in zero overlap between the two hypotheses. Despite this, the hypotheses locations can be warped to compensate for motion using local image features without modifying the original image data.

Motion estimation in large 3D volumes is frequently performed using non-linear registration to account for local tissue deformations. While traditional particle image velocimetry strategies use phase cross-correlation on local patches [140], we implemented a continuous optimization approach that directly minimizes intensity discrepancies in the coordinate space. This formulation leverages automatic differentiation to iteratively refine a displacement field, allowing for the inclusion of arbitrary differentiable regularizers [141]. The registration objective is defined by the following loss function:

$$L(\theta) = \|I_{t-1} - \text{grid\_sample}(I_t, coords + \theta)\|_1 + \lambda \text{TV}(\theta), \tag{3.14}$$

where I_t is the image at time t, $coords$ represents the identity coordinate grid, θ is the displacement field, and `grid_sample` is a differentiable function that queries the image at the given coordinates, interpolating the values if necessary. The first term represents the data

fidelity term, while $\mathrm{TV}(\theta)$ denotes the Total Variation (TV) penalty, which regularizes the flow field to be piecewise smooth by penalizing abrupt changes in the field.

To improve convergence and ensure robustness to large displacements, θ is optimized using a multi-resolution pyramid approach. Starting the optimization at a low resolution, then its resulting field is used to initialize the optimization at the following higher resolution iteration.

We evaluate the effectiveness of this approach in Section 4.5.

3.6 Distributed Cell Tracking

To process terabyte-scale datasets in a reasonable time, it is necessary to distribute computation across multiple machines. Below we describe how we structure the pipeline so that all computationally intensive stages are either embarrassingly parallel or decomposed into independent subproblems with minimal coordination.

The hypotheses generation and association steps are independent for each frame or pair of frames, allowing for straightforward distribution in an embarrassingly parallel manner. The primary challenge lies in the ILP computations, as large datasets can contain millions of segmentation hypotheses and an even greater number of ILP variables. While tiling the data is necessary to manage this complexity, naively tiling across the temporal axis leads to disconnected tracks at window boundaries.

To address this, previous approaches such as Malin-Mayor *et al*. [45] used a custom distributed computing scheduler to process spatiotemporal chunks in parallel while avoiding concurrent processing of overlapping regions. We implemented a simpler two-pass interleaved scheduling scheme for the ILP computations:

(i) The time-lapse is split along the time-axis into W overlapping windows, where the overlap is smaller than the window size.
(ii) The ILPs for all even-numbered windows are solved.
(iii) The solutions from these even-numbered windows are added as fixed constraints to the remaining subproblems.
(iv) The ILPs for the odd-numbered windows are solved.

This strategy ensures that no two overlapping windows are processed simultaneously and allows for trivial parallel execution, provided that odd-numbered executions wait for the completion of even-numbered windows.

The overall distributed pipeline is organized as follows, where T is the number of frames in the time-lapse and W is the number of temporal windows, indicating the number of concurrent processes:

1. $1\times$: Instantiate a database in its own node to store intermediate data and solutions.
2. $T\times$: Spawn workers to create the hypotheses and the overlaps between them (ILP constraints), returning the results to the database.
3. $(T-1)\times$: Spawn workers to query the database for hypotheses at frames t and $t-1$, compute association weights, and return them to the database.
4. $(W/2)\times$: Spawn workers to query association weights and previous solutions, solve ILPs for *even*-numbered windows, and store the results in the database.
5. $(W/2)\times$: Spawn workers to query existing solutions and weights, solve ILPs for *odd*-numbered windows, and store results in the database.
6. $1\times$: Spawn a worker to query all solutions and segmentation data to export the global tracking solution.

The compute graph of this distributed pipeline is illustrated in Figure 3.6.

The performance and runtime efficiency of this distributed approach on multi-terabyte datasets are analyzed in Section 4.9.

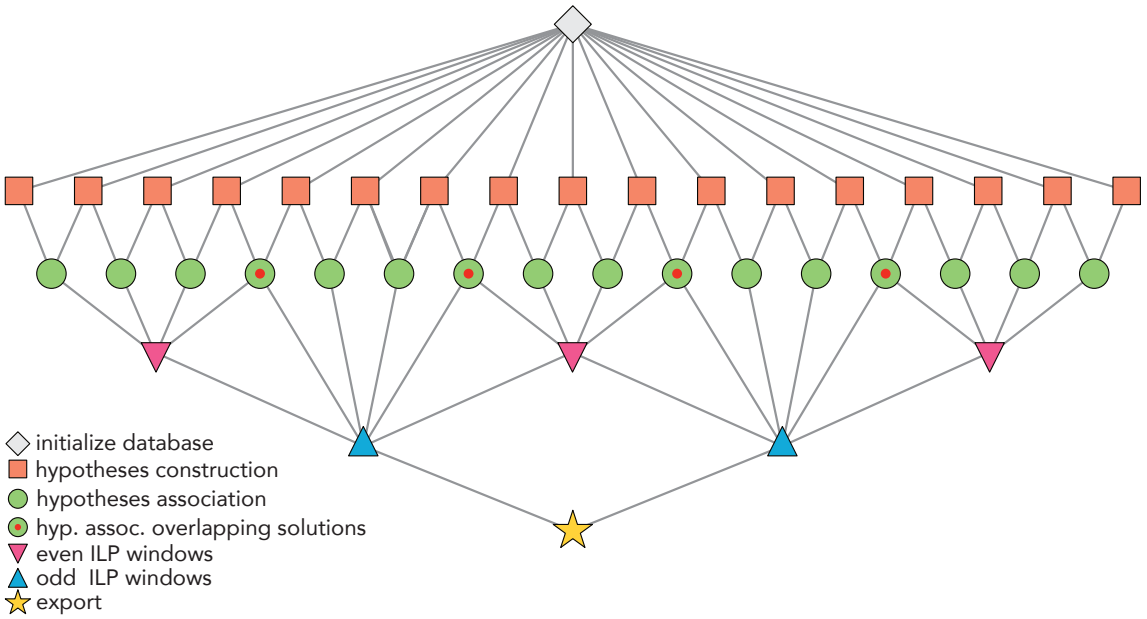


Figure 3.6: Distributed computation workflow for large-scale cell tracking. The compute graph illustrates the dependencies between the processing routines starting from top to bottom. The process begins with the initialization of a centralized database to manage intermediate states. Hypotheses construction and inter-frame association are performed as embarrassingly parallel tasks across the temporal sequence. The ILP optimization is executed using a two-pass interleaved scheduling scheme to ensure temporal consistency across overlapping windows without race conditions. The even-numbered ILP windows are solved first, providing boundary constraints for the subsequent optimization of odd-numbered windows. Finally, the optimal segments and tracks are queried and exported from the database as a unified global solution.

Chapter 4

Results & Applications

In this chapter, we evaluate the performance and practical utility of the Ultrack framework across a diverse range of biological and computational scenarios. The results presented here are synthesized from several previously published works [2, 3, 1] and have been restructured to provide a comprehensive assessment of the methodology developed in this thesis.

The chapter begins by defining the evaluation metrics used throughout our experiments, Section 4.1. We prioritize the standard Cell Tracking Challenge (CTC) metrics for comparative analysis. In cases involving sparse annotations, we adopt the error-rate metrics established by Malin-Mayor *et al*. [45].

Following the establishment of these metrics, we present a series of experiments designed to validate the core advantages of using multilevel contour maps for joint segmentation and tracking:

- Hierarchical Parameter Integration, Section 4.2, We demonstrate how aggregating multiple segmentation hypotheses from varying algorithm parameterizations yields superior accuracy compared to any single optimal setting.
- Label-Free Tracking, Section 4.3, We explore the feasibility of bypassing discrete segmentation labels by tracking image intensities directly as multilevel contours, a study conducted in collaboration with Eduardo Hirata.

- Multi-Channel and Hybrid Workflows, Section 4.4, We evaluate the integration of single-channel deep learning models with classical image processing for multi-channel data, using samples prepared and imaged by Richa Agrawal.
- Motion Compensation, Section 4.5, We assess the efficacy of hypothesis translation for tracking highly motile cells through temporal registration.

Next, the proposed approach is benchmarked against current state-of-the-art methods using standardized datasets from the Cell Tracking Challenge (CTC), Section 4.6, and the Epithelial Cell Benchmark (ECB), Section 4.7. These benchmarks represent distinct tracking challenges: the CTC focuses on sparsely distributed, nuclear-labeled embryonic cells, while the ECB evaluates densely packed, membrane-labeled tissues. To our knowledge, the proposed method is the first to achieve top-tier performance on both benchmarks.

Finally, we demonstrate the scalability of our approach through large-scale zebrafish embryo tracking. We introduce a new validation protocol using sparse fluorescence labeling to obtain high-fidelity ground truth , a collaborative effort with Xiang Zhao (sample preparation and imaging) and Ilan Theodoro (model training). We conclude with a runtime analysis of terabyte-scale datasets and a near-perfect lineage reconstruction of a developing zebrafish neuromast, Section 4.10, completed in collaboration with Teun Huijben.

4.1 Evaluation Metrics

4.1.1 Cell Tracking Challenge Metrics

The evaluation scores were calculated using the official binaries from the Cell Tracking Challenge (CTC) for all datasets submitted to the competition [30]. Additional calculations were performed using the `traccuracy` Python package [142]. The *TRA* (tracking) metric quantifies the cost of editing a predicted lineage into the reference (ground-truth) lineage, normalized by the cost of constructing the lineage from scratch, $AOGM_0$. This is

mathematically defined as:

$$AOGM = w_{NS}n_{NS} + w_{FN}n_{FN} + w_{FP}n_{FP} + w_{ED}n_{ED} + w_{EA}n_{EA} + w_{EC}n_{EC} \tag{4.1}$$

$$TRA = 1 - \frac{\min(AOGM, AOGM_0)}{AOGM_0} \tag{4.2}$$

where n_{NS} represents the number of node splits (*i.e.* under-segmented cells), n_{FN} and n_{FP} are false negative and false positive nodes, respectively, n_{ED} and n_{EA} denote false positive and negative edges, and n_{EC} represents nodes with incorrect semantics. The weights defined by the competition are: $w_{NS} = 5$, $w_{FN} = 10$, $w_{FP} = 1$, $w_{ED} = 1$, $w_{EA} = 1.5$, and $w_{EC} = 1$, emphasizing the greater complexity in correcting missing components compared to deleting extra ones. For further details, refer to the work of Matula *et al*. [143].

The primary CTC ranking used by the competition and in this thesis is the average of the *TRA* and *SEG* scores. The *SEG* score calculates the average Jaccard index (intersection over union) across all segmentation instances. A ground-truth (gt) segmentation instance, S_j^{gt}, is paired with a predicted instance S_i if:

$$|S_i \cap S_j^{gt}| > 0.5|S_j^{gt}| \tag{4.3}$$

Note that the intersection $\cap$ is used here to ensure the predicted segment covers more than half of the ground truth. This criterion ensures a single match per gt instance[1]. This same matching logic is applied to find the correspondences between nodes in the $AOGM$ metric.

From the detections identified by the CTC procedure, we also computed the detection F1-score:

$$\text{F1-score} = \frac{2n_{TP}}{2n_{TP} + n_{FN} + n_{FP}} \tag{4.4}$$

This provides a performance assessment less sensitive to segmentation quality or class imbalance [144]. For multi-color, Section 4.4, and label-free, Section 4.3, experiments where only a subset of lineages is annotated, we applied the standard CTC protocol: only lineages present in the first frame of the ground truth are retained for evaluation.

[1]Predicted instances with multiple ground-truth matches contribute to the n_{NS} count in the $AOGM$ calculation.

4.1.2 Sparse Annotation Evaluation Metrics

For the evaluation of sparse tracking data in Section 4.8, we applied the metrics implemented in the `linajea` package [45]. These metrics represent the proportion of various error types relative to the total number of edges (temporal links) in the ground-truth data. Because the annotations are sparse, we report a subset of metrics that provide valid estimates for the unlabeled regions of the dataset:

- **FN**: Proportion of false negatives (*i.e.* missing edges).
- **IS**: Proportion of identity switches (*i.e.* associations between incorrect cells).
- **FP-D**: Proportion of false positive divisions (*i.e.* spurious splits) within a 1-frame tolerance.
- **FN-D**: Proportion of false negative divisions (*i.e.* missed splits) within a 1-frame tolerance.
- **sum**: The cumulative error rate of the metrics above.

4.2 Optimal Segmentation Selection by Temporal Consistency

Image segmentation techniques, ranging from classical morphological operators to contemporary deep-learning models, frequently encounter significant challenges during parameter optimization. This tuning process is often hindered by three primary factors: (i) the computational cost of exploring high-dimensional parameter spaces; (ii) the specialized expertise required to calibrate models effectively for novel data; and (iii) the inherent technical limitation where a single set of parameters may fail to encompass the morphological diversity (*e.g.* varying cell sizes or intensities) present within a single time-lapse.

Ultrack addresses these challenges by integrating a set of heterogeneous segmentation hypotheses into a unified multilevel contour representation. Using the temporal consistency provided by the tracking objective to select the most plausible segments across frames. In principle, this allows the system to resolve local segmentation ambiguities that cannot be settled using spatial information alone. An overview of this integrative approach is provided in Video 2.

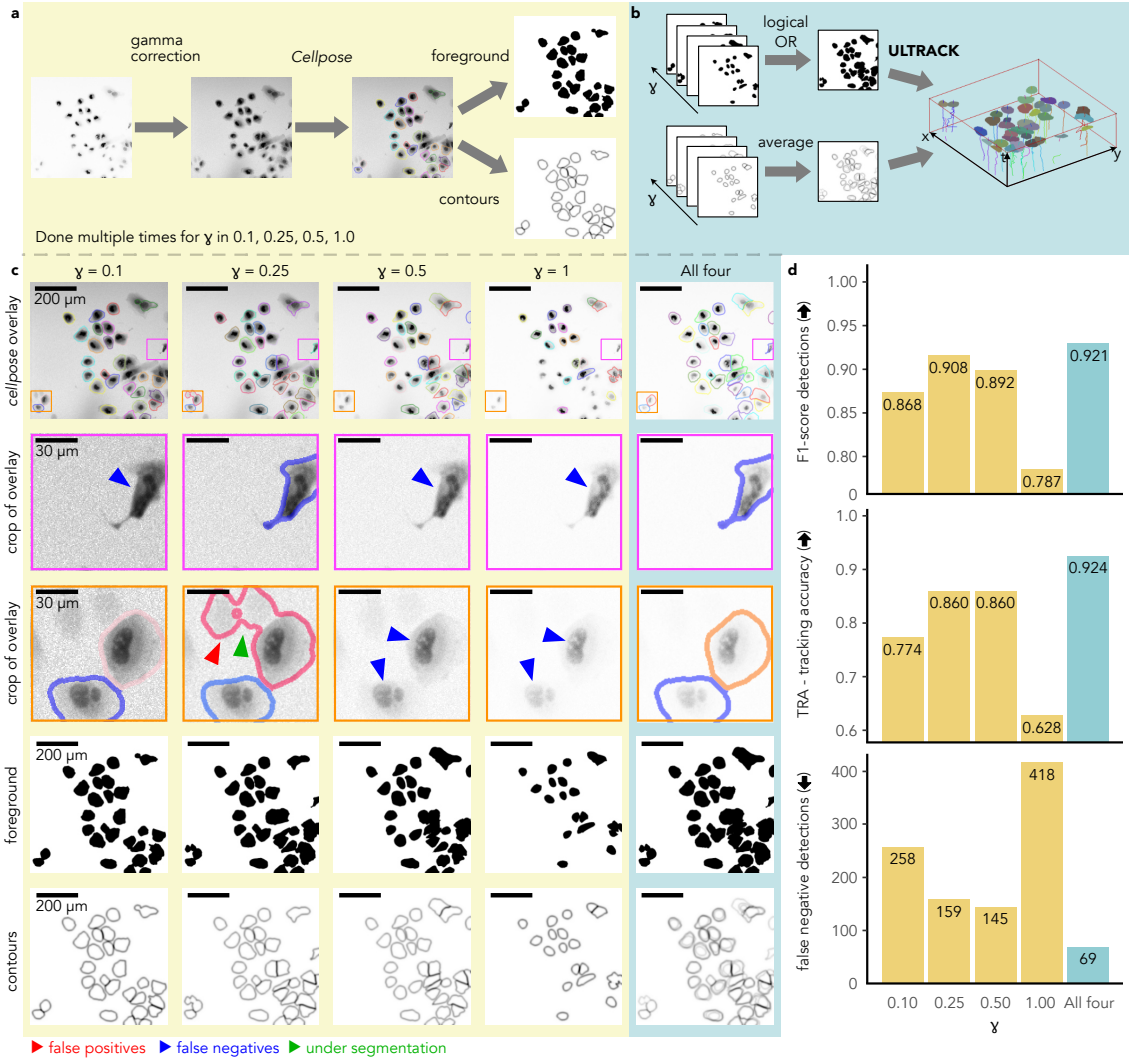


Figure 4.1: Multiple segmentation hypotheses alleviate the curse of parameter tuning. **a**, Segmentation pipeline using gamma correction with varying settings ($\gamma = 0.1, 0.25, 0.5,$ and 1.0) to generate diverse segmentation hypotheses. **b**, Ultrack's approach: integrating multiple foreground and contour maps for joint segmentation and tracking. **c**, Impact of gamma correction on segmentation quality: original images (top row), segmentation results (second and third rows) with errors highlighted by colored arrows, foreground maps (fourth row), and contour maps (fifth row). **d**, Quantitative comparison of F1-score of detections, tracking accuracy (TRA) and false negative cell detections for individual gamma settings and Ultrack's combined approach (All four). Arrows within parenthesis indicate whether lower (↓) or higher (↑) values are better.

To evaluate this approach, we used human hepatocarcinoma-derived cells expressing YFP-TIA-1 [145]. This experiment specifically tests the "high-recall pool" hypothesis established in Section 3: that a diverse set of candidate segments can be narrowed down to a high-precision lineage by evaluating temporal evidence.

We applied the Cellpose "cyto2" model [58] without dataset-specific fine-tuning to simulate a common user workflow. This dataset is characterized by high variance in cell intensities, which induces specific trade-offs in standard segmentation models. For example, inference on the original image intensities ($\gamma = 1.0$) resulted in a high false-negative rate for dim cells, Figure 4.1c, second column from right. Conversely, enhancing the dynamic range ($\gamma < 1.0$) improved the detection of dim entities but led to over-saturation and segmentation errors in brighter cells, Figure 4.1c, rightmost column.

To consolidate these outputs, we extracted binary foreground and contour maps from each of the four parameterizations, Figure 4.1a, and aggregated them into a single multilevel contour map for tracking, Figure 4.1b. We then benchmarked the performance of each individual setting against the proposed multi-hypothesis approach using the official CTC ground truth.

The quantitative results, summarized in Figure 4.1d, indicate that the integrated approach achieves higher accuracy across all evaluated metrics: detection F1-score, tracking accuracy (TRA) [143], and false negative rate. Notably, the multi-hypothesis solution outperformed the best single parameter setting ($\gamma = 0.5$ for TRA; $\gamma = 0.25$ for F1). These findings suggest that the joint optimization framework effectively uses temporal evidence to select the correct candidate from a pool of contradictory hypotheses. This capability reduces the sensitivity of the tracking results to initial segmentation parameters and provides a more robust interpretation of datasets with heterogeneous signal properties.

4.3 Intensity-based Tracking from Label-free Virtual Staining

Off-the-shelf segmentation models frequently underperform when applied to datasets that deviate significantly from their training distribution. In such instances, researchers typically resort to the labor-intensive task of generating manual annotations for model

fine-tuning or implementing custom algorithms tailored to the specific imaging modality. Given the vast diversity of biological imaging techniques, data annotation represents a significant bottleneck in data analysis. In this section, we demonstrate that the Ultrack framework can circumvent the requirement for explicit segmentation labels by tracking image intensities directly as multilevel contours. This approach operates similarly to the traditional grayscale watershed transform [60, 66] but leverages the global temporal consistency of the tracking objective to select valid segments.

This methodology relies on the assumption that the input intensity map provides sufficient evidence of cell boundaries. This can be achieved through membrane labeling, which directly delineates cell perimeters, or via nuclear labeling, which highlights cell interiors. In the latter case, inverting and blurring the image produces a representation where boundaries correspond to high intensity values and cell interiors to local minima.

To evaluate the efficacy of label-free tracking, we used A549 cells imaged via quantitative phase imaging (QPI) acquired with the *recOrder* system [146]. QPI data is particularly challenging for standard segmentation models like Cellpose, as it was not represented in their original training sets [147]. We generated an initial intensity map using VSCyto2D [147], a convolutional neural network designed for virtual staining (VS) that predicts nuclear and membrane markers from label-free acquisitions.

A unified multilevel contour map was constructed by subtracting the normalized and filtered nuclear channel from the filtered membrane channel, Figure 4.2a. The corresponding foreground map was derived by manual thresholding and morphological filtering to ensure connectivity and remove noise. Ultrack was then applied directly to this foreground and the raw combined multilevel contour.

For benchmarking purposes, we tracked the virtual stained nuclear channel and manually curated a set of 247 tracklets of lineages spanning from the first frame to the last frame. The centroids of the tracked cells were used to create reference gold-standard datasets following the Cell Tracking Challenge (CTC) format.

As shown in Figure 4.2b-c, the off-the-shelf Cellpose model failed to recognize the majority of cells in the QPI data. In contrast, the VS-intensity-based approach using Ultrack successfully identified the bulk of the population, achieving a significantly higher F1-score (0.812 vs. 0.347). Specifically, the proposed workflow resulted in 2,802 fewer missed

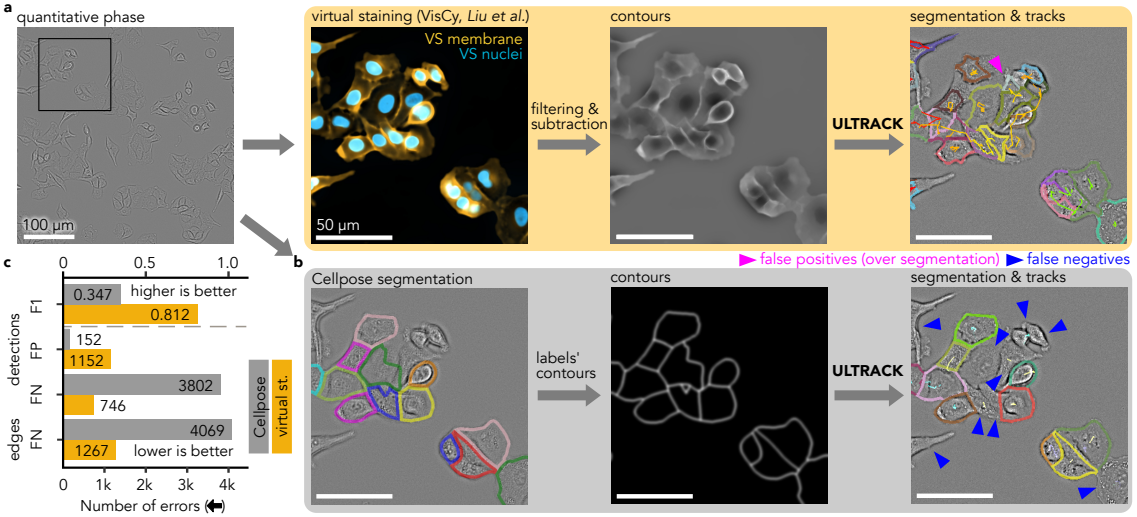


Figure 4.2: Direct intensity-based joint segmentation and tracking from virtual stains. **a**, Processing pipeline for intensity-based joint segmentation and tracking from label-free quantitative phase imaging: Quantitative phase image (top left) is processed by VisCy [147] to generate virtual staining of nuclei and membrane (second from left). These virtual stains are used to derive contours (third from left) and foreground maps (not shown) through filtering, subtraction, and thresholding. Ultrack uses these inputs to perform tracking (right). **b**, Competing approach: first applying a Cellpose segmentation to the quantitative phase image (bottom left), then deriving contours (and foreground map, not shown), and then applying Ultrack (right). False positives (over-segmentation) are indicated in magenta, and false negatives in blue. **c**, Quantitative comparison of tracking errors using Cellpose [77] segmentations (gray) versus virtually stained membranes (orange) as input. The bar chart shows the F1-score of detections, upper y-axis and the number of false negative (FN) edges, false negative (FN) detections, and false positive (FP) detections for each method, lower y-axis, from 247 tracklets spanning 48 frames.

associations (FN edges) and 3,014 fewer missed cell detections (FN detections) compared to the baseline. While the intensity-based method did introduce approximately 1,000 false-positive detections, the overall accuracy gain demonstrates the robustness of joint optimization when faced with unconventional imaging modalities. Qualitative results of the virtual staining and tracking are provided in Video 3 .

These results indicate that tracking directly from image intensity—leveraging virtual staining when necessary, offers a viable alternative to supervised methods, particularly for imaging modalities like polarization, phase contrast, or DIC where annotated training data is scarce. By treating the intensity map cues for a hierarchy of possible segments, Ultrack effectively uses temporal consistency to filter out noise, reducing the dependence on highly accurate, modality-specific segmentation models.

4.4 Enhanced Tracking Through Multi-Channel Integration

Multi-color labeling has been shown to enhance cell segmentation and tracking capabilities, particularly in complex cellular environments [148, 149]. However, applying state-of-the-art deep-learning models to such data often requires training on dataset-specific annotations, as pre-trained weights are typically only available for single-channel data. Ultrack addresses this challenge by combining multiple segmentation inputs in two ways: by varying parameters, as demonstrated in Section 4.2, and by integrating outputs from different segmentation algorithms applied to separate color channels. This flexibility allows for the construction of the high-recall hypothesis pool introduced in Section 3.

We demonstrate this feature on a three-channel "multi-color" dataset of metastatic breast adenocarcinoma cells (MDA-MB-231) with RGB-markings from a lentiviral gene ontology (LeGO) vector system [150]. This experiment shows that pre-trained models, such as Cellpose's "cyto2" [58], can be used alongside classical algorithms by applying them independently to each channel and then combining the outputs into a single multilevel contour representation. This approach circumvents the need for the extensive retraining typically required by other frameworks [121, 151].

For this evaluation, we combined six segmentation outputs from three image channels—three from Cellpose and three from Otsu [152] with watershed [66, 62] into a single contour and detection map for Ultrack, Figure 4.3a. While classical image processing approaches like watershed are generally less precise at resolving adjacent cell instances, they provide greater control and allow for the detection of dimmer cells, complementing the more sophisticated Cellpose output. Furthermore, Ultrack can use the three color channels as features when associating segments between frames, aiding the connection of correct segments across time, Figure 4.3b. See Section 3.4 for the technical implementation of color feature association.

We assessed the effectiveness of our multi-channel, multi-algorithm approach using progressively more sophisticated strategies: (i) individual pipelines using Cellpose and watershed algorithms on a grayscale version (maximum intensity across multiple channels); (ii) each algorithm applied independently on multiple channels; (iii) combination of outputs from both algorithms in the multi-color configuration (Figure 4.3a); and (iv)

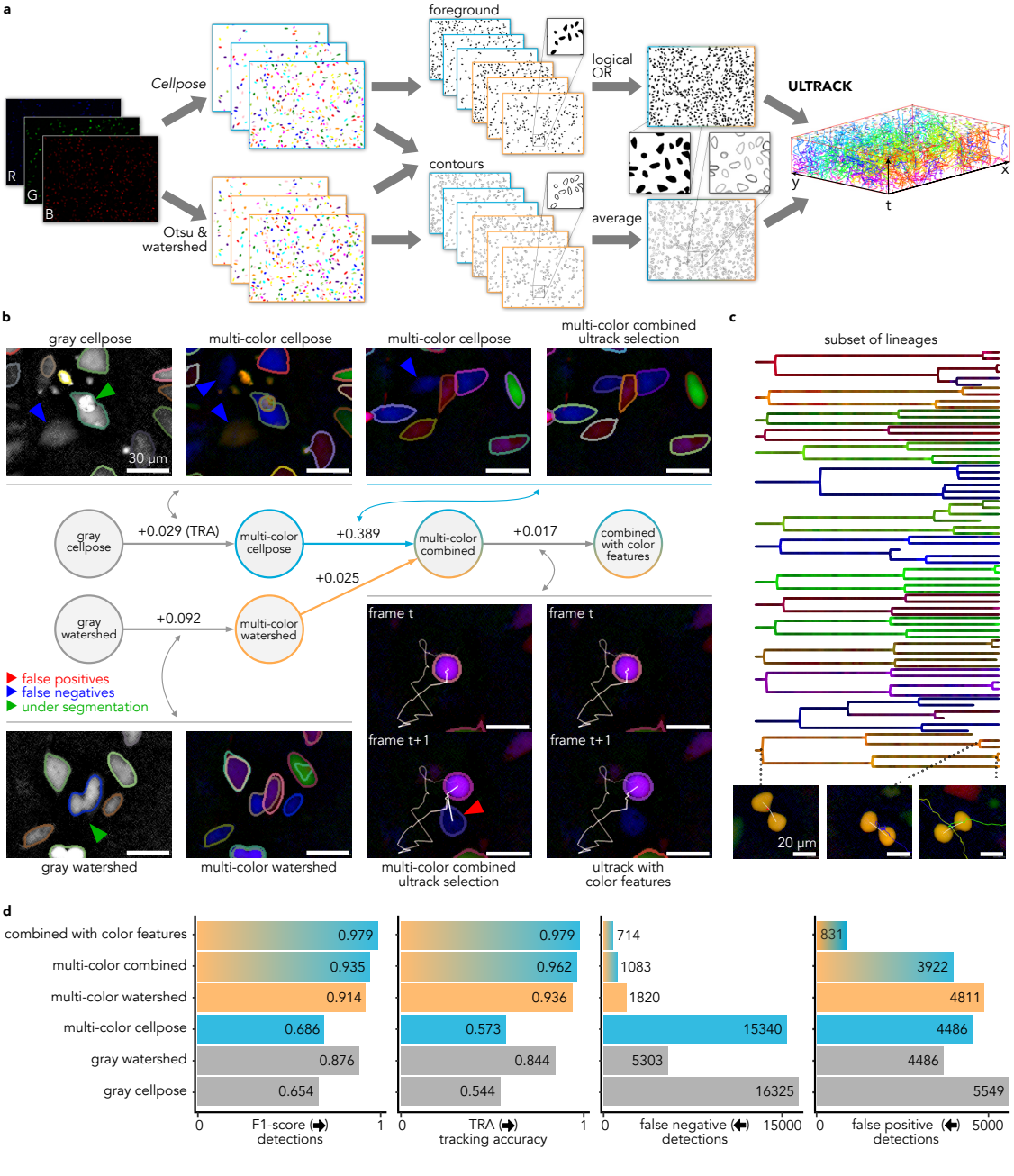


Figure 4.3: Enhanced multi-color cell tracking by integrating diverse segmentation algorithms. **a**, Multi-channel, multi-algorithm tracking pipeline: Individual labels are generated for each color channel, from which foregrounds and contours are derived and subsequently integrated for joint segmentation and tracking. **b**, Stepwise improvement of segmentation strategies (left to right): (i) individual algorithms applied to grayscale images, (ii) applications to multi-color data, (iii) combination of labels from both algorithms and colors, and (iv) enhancement with color features in association scores. The central diagram illustrates the progression of methods, with numbers indicating improvements in the TRA metric. Colored arrows highlight specific error types. **c**, Representative subset of lineages recovered by Ultrack, showcasing cells' respective colors and examples of captured division events. **d**, Quantitative evaluation of tracking accuracy for the approaches described in **b**.

integration of color features into the association scores to improve linking of segments. An example of the resulting tracks is presented in Video 4 .

For validation, we manually curated 413 cell tracks to create a gold-standard dataset. The experimental results demonstrate three key findings: (i) incorporating color channels enhances cell distinction compared to grayscale versions, notably reducing under-segmentation, Figure 4.3b, green arrow; (ii) the combination of watershed and Cellpose algorithms is effective at improving the detection of dimmer cells while maintaining accurate segmentation of brighter ones; and (iii) color-feature linking improves tracking in ambiguous scenarios, Figure 4.3b, bottom center.

Overall, the tracking accuracy increased as additional features—such as diverse segmentation labels and color information—were incorporated. These results indicate that Ultrack can effectively reconstruct valid lineages, Figure 4.3c, by selecting from a large pool of segmentation hypotheses, achieving robust performance without the need for dataset-specific model fine-tuning.

4.5 Improved Tracking with Temporal Registration

Imaging systems must balance multiple factors, including field of view size, spatial resolution, imaging speed, and depth [153]. A critical limitation often encountered is acquisition speed (*i.e.*, temporal resolution), which can hamper cell tracking performance. This is particularly relevant for the proposed framework, as the key assumption for joint optimization is the temporal coherence of segmentation hypotheses across frames. Rapid cell movement and high morphological similarity can complicate inter-frame associations, leading to identity switches or broken tracks.

Registration of adjacent frames is a common strategy to compensate for limited temporal resolution. However, collective cell motion within growing or deforming tissues frequently demonstrates local coherence that does not conform to simple linear or affine transformations due to the complex dynamics of biological processes [154]. In such cases, non-linear registration, such as movement vector fields [140, 155], is more appropriate for minimizing apparent motion between frames.

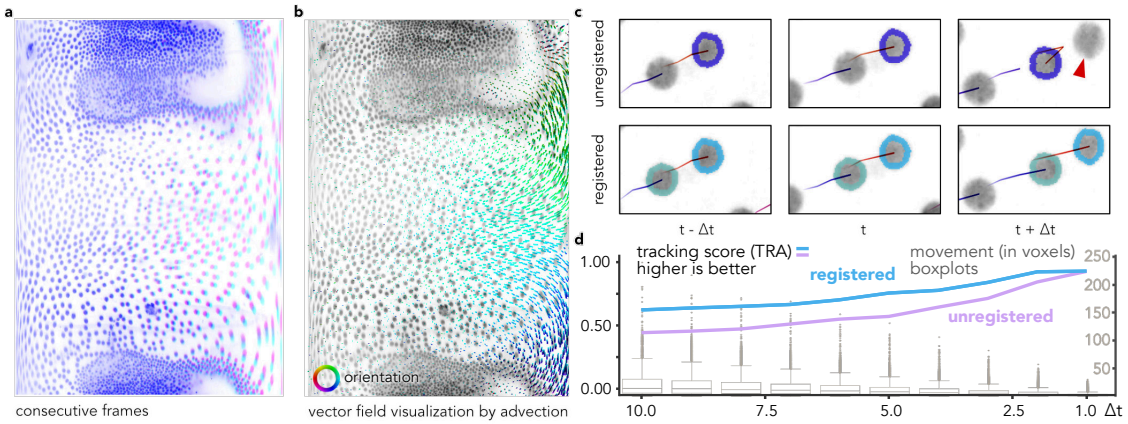


Figure 4.4: Enhancing tracking accuracy through temporal registration. **a**, Consecutive frames, t (magenta) and $t+1$ (cyan), highlighting divergent motion along the right boundary. **b**, Vector field generated by non-linear registration, colored by orientation. **c**, Qualitative comparison of cell identity preservation: the top row shows a cell identity switch in unregistered frames (indicated by red arrow), while the bottom row demonstrates correct associations with flow registration. **d**, Comparative analysis of tracking accuracy (TRA) for registered and unregistered data across different effective velocities (Δt). The right y-axis and boxplots show cell movement distributions in voxels. At $\Delta t = 10$, some cells move more than 150 voxels between frames—a distance significantly exceeding the median cell diameter of ≈ 19.64 voxels. Boxplots follow Tukey's definition [157]. Analysis with an average population of $N = 10,091.5$ edges per group (Δt).

Computing non-linear registration for large volumes can be computationally intensive. To address this, we implemented GPU-accelerated routines to compute vector fields for 2D and 3D time-lapses. These routines use automatic differentiation from deep-learning frameworks [141], providing efficient computation without sacrificing accuracy. As described in Section 3.5, these movement vector fields are integrated into the tracking step by applying local advection to each candidate segmentation, thereby improving association scores without altering the original image data.

We evaluated the effectiveness of this registration approach on the *Tribolium castaneum* cartography dataset [154] from the Cell Tracking Challenge (CTC). This dataset consists of nuclear-labeled embryo cells imaged via selective plane illumination microscopy (SPIM). The ellipsoidal surface of the epithelial cells was transformed into a planar representation using the ImSAnE toolbox [156] to facilitate analysis; however, this transformation introduced significant distortion and excessive motion near the polar boundaries (Figure 4.4a).

We compared tracking accuracy with and without flow registration (Figure 4.4b) across various temporal resolutions, simulated by subsampling frames from $\Delta t = 1$ (original rate) to $\Delta t = 10$. The results, illustrated in Figure 4.4c-d, show: (i) at $\Delta t = 10$, tracking accuracy improved from 0.443 to 0.623 with registration. In this scenario, motile cells moved more than 150 voxels between frames, which is substantial given the median cell diameter of approximately 19.64 voxels. (ii) At $\Delta t = 5$, accuracy increased from 0.571 to 0.756. (iii) At the original resolution ($\Delta t = 1$), performance was comparable (0.927 vs. 0.929). The crops in Figure 4.4c provide a specific instance where the flow field maintains the correct identity, whereas an identity switch occurs in the unregistered baseline. The estimated flow field is visualized in Video 5 .

These findings indicate that temporal registration effectively mitigates the impact of large displacements between frames, enhancing tracking robustness in datasets with high motility or low temporal resolution.

The features presented thus far — the ability to leverage multiple segmentation hypotheses, Section 4.2, integrate multi-channel data, Section 4.4, track directly from intensities, Section 4.3, and apply temporal registration—collectively form a flexible framework for robust cell tracking in complex biological environments. These capabilities address the specific challenges posed by dense, three-dimensional tissues. To assess the performance of this unified approach against existing state-of-the-art methods, we next evaluate Ultrack on standardized benchmarks from the Cell Tracking Challenge and the Epithelial Cell Benchmark.

4.6 Cell Tracking Challenge

The Cell Tracking Challenge (CTC) is a long-standing benchmark for evaluating cellular segmentation and tracking algorithms under a standardized framework [30]. The challenge provides participants with training data for parameter optimization; however, the test set annotations used for competitive assessment and final rankings remain concealed by the organizers. This double-blind structure ensures an unbiased evaluation of each method's performance and prevents information leakage between the training and testing phases. We assessed our framework using five distinct 3D datasets: human breast

cancer cells, *C. elegans* (worm) [158], *D. melanogaster* (fly) [22], *T. castaneum* (beetle), and a 2D cartographic projection of *T. castaneum* development [154].

The worm and breast cancer cell datasets include manually curated lineage data and silver-standard segmentation annotations derived from a consensus of prior challenge editions [159]. We trained a U-Net model [75, 2] using these annotations to predict two outputs per voxel: (i) the probability of a voxel belonging to a cell contour and (ii) the probability of it being in the foreground. Training was conducted by optimizing three Dice losses [137]: a binary foreground prediction loss, a binary cell contour prediction loss, and auxiliary losses on intermediate U-Net decoder layers for multi-scale edge detection [138], see Section 3.3 for technical details. These predicted maps served as input for the Ultrack. As shown in Table 4.1, our approach achieved the highest combined segmentation and tracking score (CTB) of 0.844 for the worm dataset. However, for the breast cancer cells, Ultrack ranked 7th (CTB: 0.683). This performance gap is likely due to over-fitting on the limited 3D segmentation data provided for that dataset as only 12 frames are available per time-lapse, and our internal cross-validation reported an average tracking accuracy (TRA) of 0.905, while the hidden test set score was significantly lower at 0.818.

A common challenge in biological imaging is the absence of dense segmentation annotations for model training, which is the case for the fly, beetle, and beetle projection datasets. While contemporary deep-learning-based submissions typically bypass this by relying solely on point coordinates and motion prediction, we investigated whether the joint optimization framework could produce accurate lineages using alternative inputs. As detailed in Appendix A.2, we evaluated two distinct regimes for these datasets: a deep model trained on pseudo-labels and a straightforward classical image-processing pipeline. Our cross-validation results revealed that the classical approach achieved superior accuracy across these datasets, likely because it avoided the noise associated with pseudo-label generation. For the final classical image-processing pipeline, we detected foreground voxels using a Difference of Gaussians filter followed by Otsu thresholding [152]. Inverted and normalized image intensities were then used as a proxy for cell boundaries, where lower intensities (representing potential contours) were mapped to the highest values.

Kind	Dataset	Rank	combined (CTB)	tracking (TRA)	segmentation (SEG)
Whole 3D embryo	Worm *C. elegans*	1st	**0.844**$^{\text{ultrack}}$	0.987$^{\text{MPI}}$	0.759$^{\text{MU}}$
		2nd	0.829$^{\text{KTH}}$	0.979$^{\text{JAN}}$	0.729$^{\text{KIT}}$
		3rd	0.808$^{\text{KIT}}$	0.975$^{\text{IGFL}}$	0.722$^{\text{KTH}}$
		4th	-	**0.967**$^{\text{ultrack}}$	**0.722**$^{\text{ultrack}}$
	Fly *D. melanogaster*	1st	**0.708**$^{\text{ultrack}}$	**0.802**$^{\text{ultrack}}$	**0.613**$^{\text{ultrack}}$
		2nd	0.617$^{\text{KTH}}$	0.785$^{\text{JAN}}$	0.613$^{\text{KTH}}$
		3rd	0.591$^{\text{JAN}}$	0.785$^{\text{MPI}}$	0.397$^{\text{JAN}}$
	Beetle *T. castaneum*	1st	**0.841**$^{\text{ultrack}}$	0.955$^{\text{MPI}}$	**0.746**$^{\text{ultrack}}$
		2nd	0.804$^{\text{MPI}}$	**0.936**$^{\text{ultrack}}$	0.684$^{\text{RWTH}}$
		3rd	0.785$^{\text{RWTH}}$	0.886$^{\text{RWTH}}$	0.654$^{\text{MPI}}$
Embryo projection	Beetle projection *T. castaneum*	1st	0.867$^{\text{KTH}}$	0.952$^{\text{MPI}}$	0.791$^{\text{KTH}}$
		2nd	0.816$^{\text{MPI}}$	0.942$^{\text{KTH}}$	0.766$^{\text{KIT}}$
		3rd	0.787$^{\text{KIT}}$	**0.854**$^{\text{ultrack}}$	0.680$^{\text{MPI}}$
		4th	**0.754**$^{\text{ultrack}}$	-	**0.654**$^{\text{ultrack}}$
Cells in a dish	Breast cancer cells	1st	0.797$^{\text{KIT}}$	0.884$^{\text{KIT}}$	0.710$^{\text{KIT}}$
		2nd	0.761$^{\text{LEID}}$	0.882$^{\text{KTH}}$	0.642$^{\text{LEID}}$
		3rd	0.757$^{\text{KTH}}$	0.880$^{\text{LEID}}$	0.632$^{\text{KTH}}$
		7th	**0.683**$^{\text{ultrack}}$	**0.818**$^{\text{ultrack}}$	**0.549**$^{\text{ultrack}}$

Table 4.1: Segmentation and tracking scores for 3D datasets of the Cell Tracking Challenge. Ultrack results, as of **August 2024**, are highlighted in **bold**. Ultrack obtains the highest combined segmentation and tracking score (CTB) for three out of five datasets. Ultrack achieved 1st place more often than any other method, excelling for large and complex 3D whole-embryo cell tracking datasets. See Table 4.2 for methods descriptions.

Combining this straightforward image-processing pipeline with Ultrack's ability to resolve multiple hypotheses within the contour map yielded results that compared favorably to methods requiring supervised training. As summarized in Table 4.1, the framework achieved a CTB score of 0.708 for the fly dataset and 0.841 for the beetle dataset, outperforming the respective second-best methods (0.617 and 0.804). The lower score obtained on the beetle projection dataset, 0.754, is likely due to the inherent distortions introduced by the 2D planar transformation, which violates some of the assumptions regarding consistent cell morphology. Qualitative results of the segmentation and tracking for these 3D embryonic datasets are shown in Figure 4.5 and Video 6 .

Label	Reference	Description
KTH	[129]	Uses the Gaussian Mixture Probability Hypothesis Density (GM-PHD) [160] filter to detect and estimate the motions of centroids with the Viterbi algorithm to optimally link segmentations across frames while assigning different cell states (*e.g.* migration, mitosis, appearance, disappearance).
KIT	[44]	*EmbedTrack* combines pixel-embedding segmentation with motion estimation for joint segmentation and tracking by embedding each pixel in a centroid-offset space and incorporating optical flow.
MPI	(unpublished)	Although a detailed description is undisclosed, the source code provided by the challenge organizers suggests a deep-learning-based centroid detection strategy followed by nearest-neighbor linking.
JAN	[122, 45]	Builds upon *Linajea* [45], which predicts cell locations as Gaussians' maxima and their motion offset to their position on the previous frame, by classifying cell states (parent, daughter, continuing cells), improving their ILP formulation to predict these states, while asserting that parent cells can only link to daughter cells. A structured SVM is used to learn optimal ILP weights, enhancing automation linking cell trajectories.
LEID	[161]	Joint segmentation and tracking by model evolution with level sets and specialized routines for cell division and cell appearances.
MU	[162]	Integrates H-minima watershed segmentation with a U-Net model for cell contour and foreground prediction. Subsequent tracking uses an intersection-based heuristic, which pairs cells across frames if they share sufficient overlap.
IGFL	[43]	*ELEPHANT* leverages deep-learning-based cell detection, coupled with predicted flow fields (*i.e.* motion offset) for cell advection and linking. This flow-driven approach resembles the MPI and JAN strategy, using learned motion information to guide inter-frame associations.
RWTH	[163]	Detects cells via a Laplacian of Gaussian filter and applies nearest-neighbor linking in subsequent frames; detections are then used as seeds for segmentation. Manual curation post-processing is performed to rectify missed detections.

Table 4.2: Description of leading competitive methods in the Cell Tracking Challenge. This table summarizes the algorithmic approaches used by competing participants whose scores are reported in Table 4.1.

Overall, these evaluations demonstrate that the joint segmentation and tracking framework maintains high performance across diverse imaging conditions, even when using classical image-processing inputs instead of deep-learning-based contour maps. This versatility is highlighted by the fact that Ultrack achieved more first-place CTB rankings on 3D datasets than any other method currently listed. Notably, the next best performer, MPI, secured first place twice in tracking accuracy (TRA) but never ranked higher than third in segmentation scores (SEG) across the same datasets, suggesting that Ultrack's joint optimization provides a better balance between consistent object identification and lineage reconstruction.

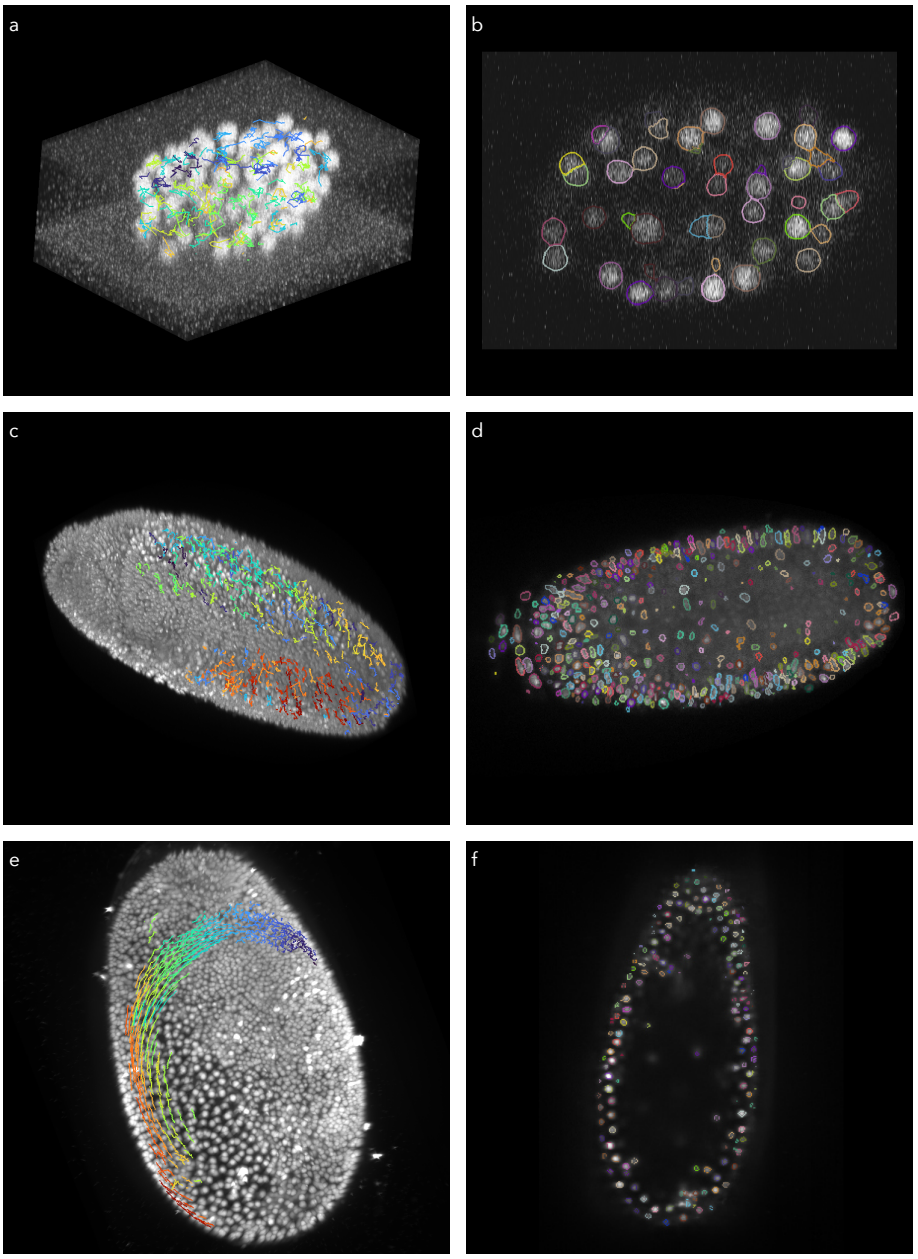


Figure 4.5: 3D whole-embryo tracking results. Rows from top to bottom: *C. elegans* (Worm), *D. melanogaster* (Fly), and *T. castaneum* (Beetle). Left column (**a**, **c**, **e**): Maximum intensity projections with Ultrack cell lineages overlaid. Right column (**b**, **d**, **f**): Representative YX-slices showing the corresponding instance segmentation results visualized as contours. For the fly and beetle datasets, only the subset of tracks starting from the initial coordinates provided by the competition organizers are displayed.

4.7 Epithelial Cell Benchmark

The Epithelial Cell Benchmark [164] evaluates tracking performance in a distinct modality characterized by densely packed cells where membranes are visible, rather than nuclei. The benchmark consists of eight 2D time-lapses divided into two tissue types: three peripodial (PER) and five proper discs (PRO) datasets [165]. In this high-density context, graph-cut methodologies adapted from electron microscopy segmentation are standard and represent the most successful contemporary approaches [80].

We benchmarked Ultrack against the original baselines, including MTL [55], FFN [166], TA [165], MALA [80], and PlantSeg [40]. Additionally, we performed a reproduction of the MALA experiment[2], which achieved competitive results, although slightly lower than the original reported scores, as shown in Table 4.4. Detailed descriptions of these competing methods are provided in Table 4.3.

Method	Variation	PER	PRO
FFN	Default	0.879	0.796
MTL	Default	0.904	**0.818**
TA	Default	-	0.758
PlantSeg	Default model	0.787	0.761
	Trained	0.885	0.800
MALA	Original	**0.907**	**0.817**
	Our repeat	0.902	0.810
	Our tracking†	**0.909**	**0.817**
Blur	Our tracking†	0.853	0.804

Table 4.4: Epithelial Cell Benchmark segmentation (SEG metric) results, top scores in bold, higher is better. Our tracking algorithm is indicated by †

We evaluated Ultrack using two distinct types of input, marked with a † in Table 4.4: (i) original frames filtered with a simple Gaussian kernel ($\sigma = 1.0$) and (ii) affinities predicted by MALA. Qualitative results for the Peripodial (PER) and Proper discs (PRO)

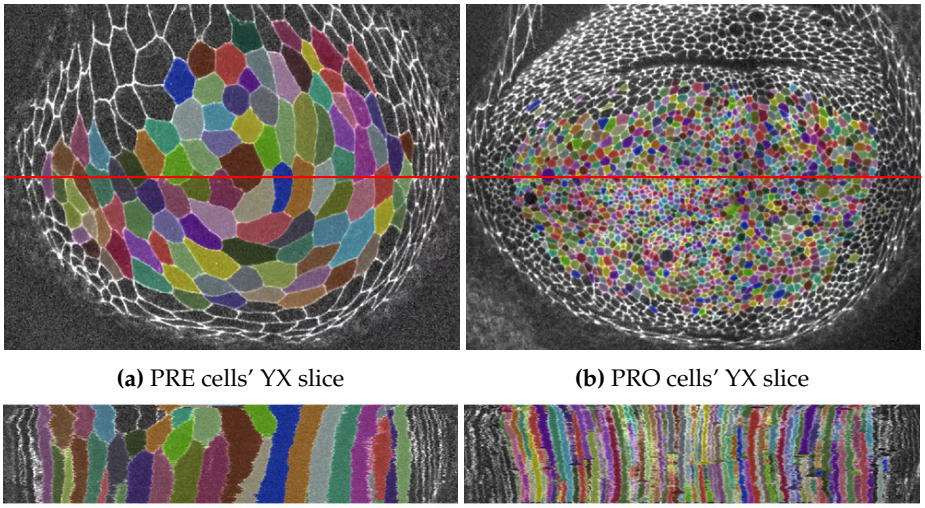
[2] `https://github.com/funkey/flywing`

Label	Reference	Description
FFN	[166]	*Flood Filling Network* is a deep learning model that iteratively fills objects to perform foreground vs. background segmentation starting from a seed point. It was adapted for cell tracking viewing, seeing the flooding as the tracking process of extending the cells limits over space and time starting from seeds at the last frame and constraing such that flooding can only happens towards earlier frames.
MTL	[55]	*Moral Lineage Tracing* solves the tracking as a 3D (time + 2D spatial dimensions) segmentation using the multi-cut algorithm with additional "morality" constraints. These constraints ensure that cells can divide but not merge to be a valid cell lineage.
TA	[165]	*Tissue Analyzer* is a specialized software for single-layered epithelia cell tracking. It uses watershed from minima for segmentation.
MALA	[80]	*MALA* is an extension of Maximum Affinity Learning of Image Segmentation (MALIS) [79], it predicted 3D (T, Y, X) affinities between pixels and obtain the cell segmentation and tracks in two steps: (i) obtaining a 2D segmentation for each cell by aglomerating watershed superpixels (ii) connections the resulting segments using a greedy tracking algorithm with the affinities along the T-dimension of overlapping segments as the cost function.
PlantSeg	[40]	*PlantSeg* is a convolutional neural network developed to detect cell membranes, originally developed for plant cells, but as shown here it can be used for other organisms. They reported two results in their paper: (i) the default off-the-shelf model trained on *Arabidopsis thaliana ovules*, and (ii) a fine-tuned model trained on the epithelial cells dataset. The tracking problem was treated as a 3D segmentation problem without "morality" constraints. The PER dataset were segmented using GASP [97], while the PRO dataset were segmented using Multicut from ilastik [167].

Table 4.3: Description of competing methods in the Epithelial Cell Benchmark. This table summarizes the algorithmic approaches used by competing participants whose scores are reported in Table 4.4.

datasets are visualized in Figure 4.6, Video 11 and Video 12 .

The results indicate that the minimal solution using only Gaussian-blurred intensities is comparable to several deep-learning-based approaches, particularly in the PRO datasets where cell sizes are diverse. When using MALA's affinities, Ultrack matches or surpasses state-of-the-art methods, despite our local reproduction of MALA's affinities having lower fidelity than the original published version. These results demonstrate that the joint segmentation and tracking framework effectively identifies optimal segments from the hierarchy, producing results competitive with specialized deep learning and graph segmentation algorithms.



(a) PRE cells' YX slice **(b)** PRO cells' YX slice

(c) PRE cells' TX slice **(d)** PRO cells' TX slice

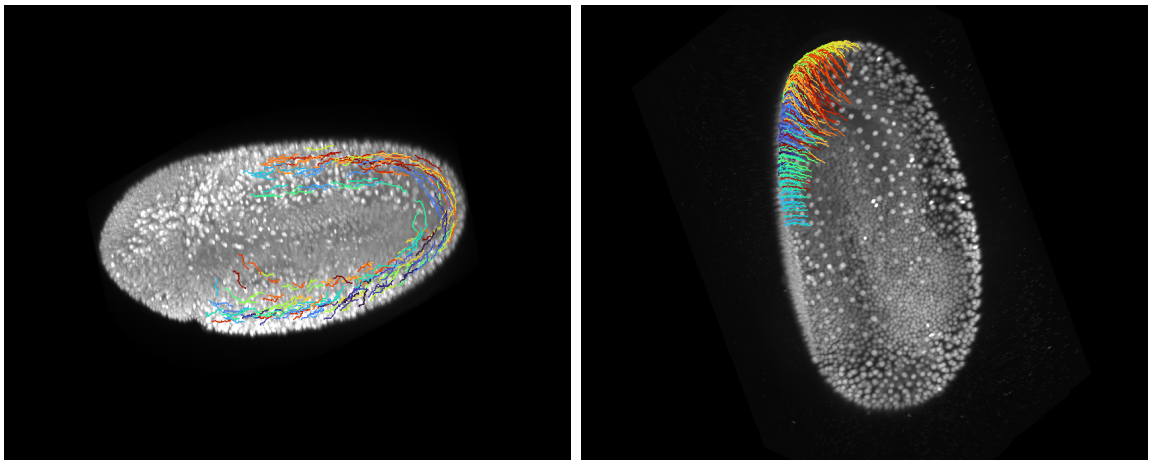
Figure 4.6: Original images overlaid with Ultrack instance segmentation results on the Epithelial Cell Benchmark using affinities predicted by MALA. Only segments within the dataset's region of interest are shown. The red line marks the y-position used to extract the TX slices in panels (c) and (d); panels (a) and (b) show PRE and PRO cells in a YX slice, respectively.

4.8 High-Quality Cell Tracking Ground-Truth via Sparse Fluorescence Labeling.

While the Cell Tracking Challenge (CTC) provides a valuable benchmark for evaluating segmentation and tracking, it relies primarily on human-generated annotations. These annotations are often driven by the convenience of tracking specific cell types studied

by experimentalists during data collection, rather than a comprehensive or random sampling of the entire dataset that would provide a more objective assessment of algorithmic performance. Figure 4.7 illustrates images from the two largest datasets in the CTC, the *Drosophila* embryo and the *Tribolium* embryo.

This limitation is further exacerbated by latest-generation 3D live microscopy datasets, [15, 7, 27, 34, 23, 35, 3]. Such datasets often span hundreds of gigabytes to terabytes, recording thousands of cells over hundreds or more time points, making manual annotation a task that is not only time-consuming but often beyond human capability. Moreover, in densely labeled or multi-dimensional datasets, distinguishing individual cells can be highly uncertain and sometimes impossible, [22, 34, 154, 45, 35].



(a) Drosophila embryo **(b)** Tribolium embryo

Figure 4.7: Example of sparse annotations provided by the Cell Tracking Challenge. For each embryo, a maximum intensity projection is shown overlaid with the annotated cell tracks.

Encouraged by our method's performance on the CTC, we developed a higher-fidelity ground truth dataset to potentially provide a more challenging benchmark of a crowded and dynamic cellular environment from a developing zebrafish embryo. We applied a dual labeling protocol that combines ubiquitous fluorescence nuclear labeling Tg(ef1α:H2B-mNeonGreen) with sparse random nuclear labeling at a distinct wavelength (pMTB-ef1-H2B-mCherry). Sparse labeling was achieved via early-stage microinjection of a marker Tol2 transposase-mediated integration that propagates to daughter cells upon division,

Source	Dataset	T, C, Z, Y, X	Size (GB)	Usage	Ref.
CTC	YFP-TIA-1	30, 1, 1, 1024, 1024	0.063*	Figure 4.1	[145]
	Beetle proj.	135, 13, 2447, 1719	14.76*	Figure 4.4	[154]
	Worm	165, 31, 512, 712	3.73*	Table 4.1	[158]
	Fly	50, 125, 603, 1272	9.59*	Table 4.1	[22]
	Beetle	105, 983, 1846, 983	376*	Table 4.1	[154]
Our work	MDA-MB-231	300, 3, 1, 1440, 1920	2.5	Figure 4.3	[1]
	Sparse zebrafish	522, 2, 505, 2217, 2170	5072	Figure 4.8	[1]
	Late zebrafish	791, 1, 448, 2174, 2423	3733	Figure 4.11	[3]
	Neuromast	500, 1, 73, 1024, 1024	71.3	Figure 4.14	[1]
Other	Label-free	48, 1, 8, 2046, 2003	5.85†	Figure 4.2	[147]

Table 4.5: List and references of datasets grouped by source (CTC, our work, or other). †: size before 3D to 2D virtual staining. * indicates averaged size and shape.

effectively recapitulating their lineage, Figure 4.8a. Video 7 provides a visual overview. For reagents and details, see the Methods section of the original publication, [1].

Compared to other publicly available datasets, such as those from the CTC, this dataset is more than a magnitude larger, reaching over five terabytes of imaging data compared to less than 400 gigabytes for the Beetle dataset in the CTC, Table 4.5.

The sparsity of this second channel significantly simplifies tracking and allows for rapid curation into a set of high-quality, or "platinum-level" annotations, Figure 4.8b. Cells in the sparse channel were automatically tracked using Ultrack and manually curated to ensure high quality standards, Figure 4.8b-top. This curation process involved selecting long lineages that overlapped with green labeled cells, visual inspection, and necessary corrections, resulting in 152 annotated tracklets spanning 85 to 521 frames. This methodology simplifies manual curation and is effective for capturing trajectories that are otherwise challenging to annotate manually in dense datasets.

As evident in Figure 4.9, tracking is easier at earlier stages of development when cell density is lower and optical properties are more favorable. Despite these challenges, nuclei in the sparse channel are consistently easier to track than those in the dense channel, Figure 4.9,.

We using these high-quality sparse-channel labels to evaluate Ultrack's accuracy on the ubiquitously labeled green channel. We employed two segmentation approaches as

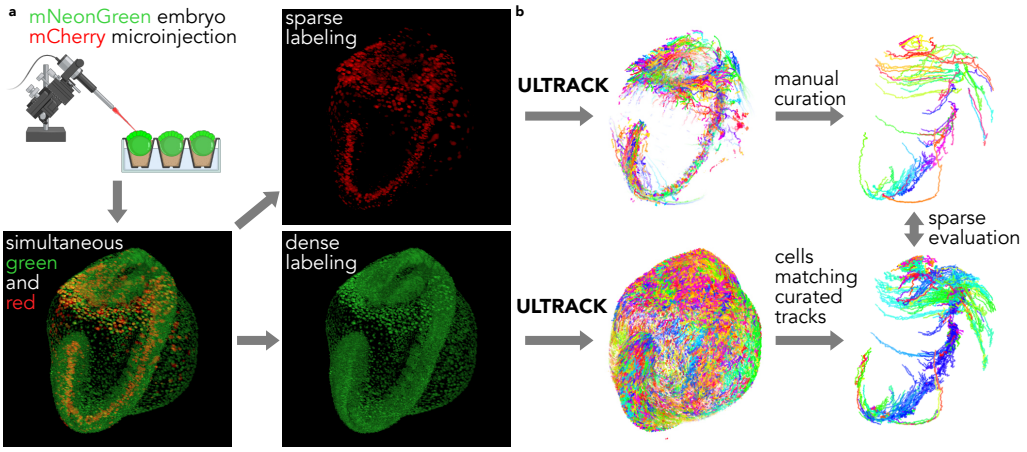


Figure 4.8: Sparse fluorescence labeling for high-fidelity tracking validation. Sparse fluorescence labeling enables high-fidelity tracking validation over extended time-lapses. **a**, Validation pipeline using sparse fluorescence labeling: (i) Embryos with ubiquitous green nuclear labeling Tg(ef1α:H2B-mNeonGreen) are injected with DNA plasmids expressing red fluorescent protein pMTB-ef1-H2B-mCherry; a single embryo is then selected for imaging. (ii) Simultaneous imaging of red and green channels [168] **b**, (iii) Independent tracking of cell nuclei in both channels. (iv) Selection and manual curation of a subset of cell nuclei tracks from the red channel. (v) Comparison of curated tracks to those obtained from ubiquitous labeling.

in [2]: (i) image processing to produce foreground maps and nuclei boundaries directly and (ii) a convolutional neural network to generate foreground and contour maps. The Ultrack setup was consistent, with only the input maps modified. Figure 4.9 compares tracking results using the sparse channel (red) and the ubiquitous channel (green) with deep-learning-based contour maps, showcasing the disproportionate difficulty of tracking cells in dense versus sparse channels.

To assess performance, we evaluated automatic dense tracking results against curated sparse tracking over two periods, Figure 4.10: the first 150 frames and an extended period of 500 frames. Performance was measured using five metrics from [45], described in Section 4.1.2,: (i) FN: false-negative edges, (ii) IS: identity switches, (iii) FP-D: false-positive divisions, (iv) FN-D: false-negative divisions, and (v) the sum error rate.

Tracking accuracy decreases over longer imaging periods due to increasing organism complexity, Figure 4.10 and Figure 4.9. Deep learning inputs benefited from available training data and consistently outperformed image processing maps, maintaining comparable accuracy over 500 frames to what the intensity-based approach achieved in 150

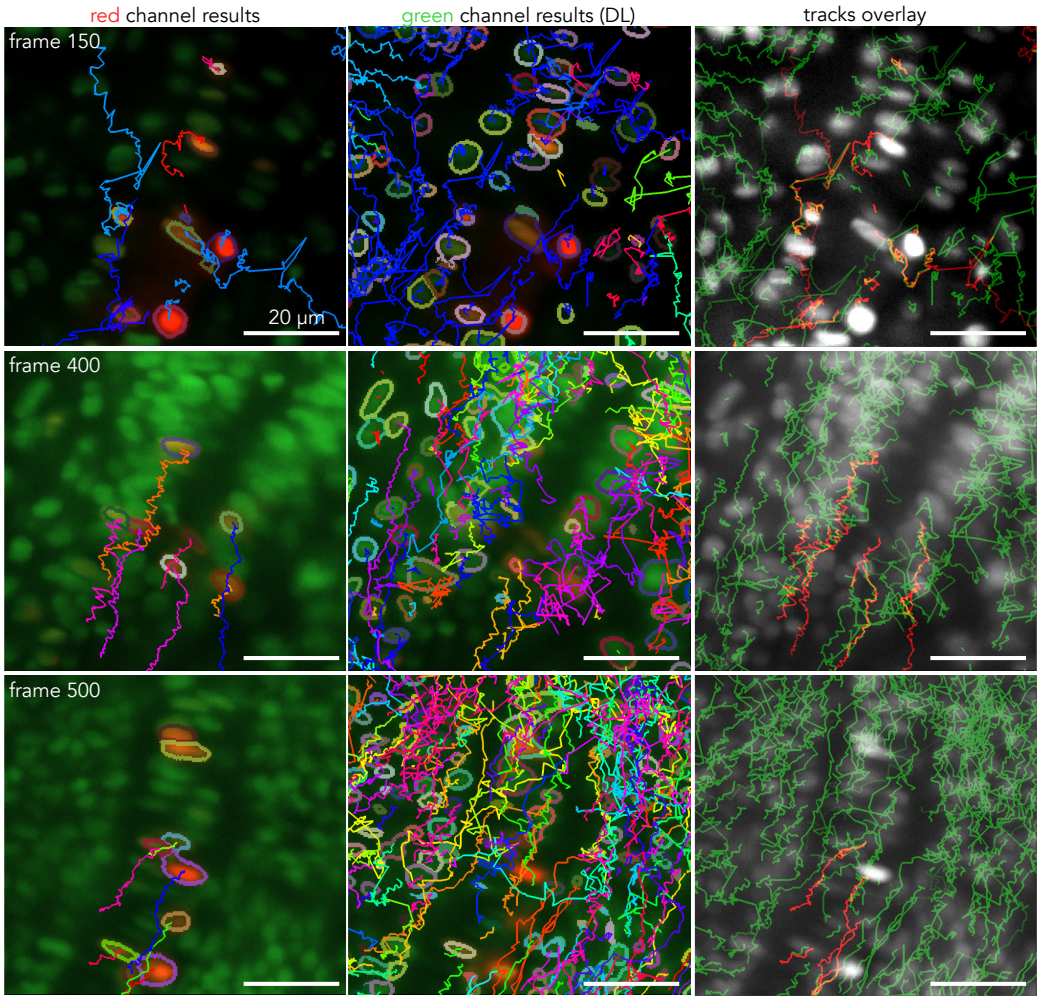


Figure 4.9: Comparative visualization of sparse (first column) and ubiquitous (second column) channel segmentation and tracking results at different time points, and combined results overlaid grayscale intensity colored in red and green (last column); Cell density and tracking difficulty increase over time (top to bottom).

frames.

To contextualize these results, we compared our findings to those reported by Malin *et al*. [45]. Their study established a baseline sum error rate of 0.078 for tracking neuronal cells in zebrafish embryos over 150 frames. As shown in Figure 4.10, Ultrack achieved lower sum error proportions for both deep learning (0.049) and intensity-based (0.070) contour maps over 150 frames. Notably, with deep learning, accuracy remained at 0.073 over a challenging 500-frame period, which is comparable to the 150-frame baseline of

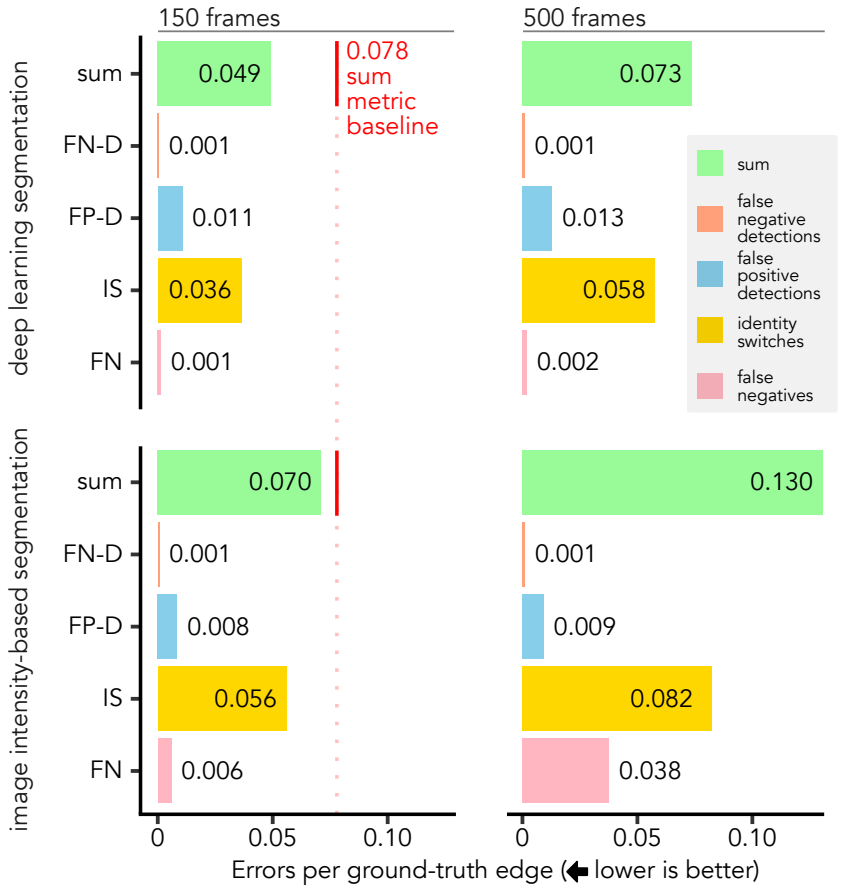


Figure 4.10: Quantitative evaluation of tracking accuracy under different conditions and durations: using deep learning for cell contour estimation (first row) vs. direct image-intensity-based segmentation (second row); measurements taken up to the first 150 frames (first column) and 500 frames (second column). Metrics shown are proportions of false-negative edges (FN), identity switches (IS), false-positive divisions (FP-D), false-negative divisions (FN-D), and their sum. The red vertical line indicates the baseline sum error rate for 150 frames of 0.078 for comparison with Malin *et al*. [45]

0.078. This demonstrates the robustness of our method over prolonged developmental periods despite increasing tissue complexity.

4.9 Scaling to Terabytes of High-Resolution Large-FOV Data

To demonstrate the scalability and performance of the proposed method on massive datasets, we employed it to recover cell lineages of nuclear-labeled zebrafish embryos [3].

Data were acquired using the fast (1 stack per minute), high-resolution (20x magnification, 1.0 NA Olympus objective), and large-imaging-volume (≥ 1 mm^3) DaXi light-sheet microscope [7]. These recordings produced uncompressed time-lapse data ranging from 1.7 to 3.7 terabytes, capturing 8.6 to 13.2 hours of zebrafish development, Figure 4.11c. Tracking results are showcased in Video 8 .

The workflow starts with the prediction of nuclei pixels (foreground) and their respective contours, Figure 4.11a,b. Following this, the joint segmentation and tracking pipeline is applied. Using the distributed processing implementation described in Section 3.6, these datasets were processed on a compute cluster managed by SLURM [169], using a tracking window size of 50 frames with an overlap of 5 frames. A 3.7 TB dataset (791×448×2174×2423 voxels, T×Z×Y×X) was processed in approximately 9.5 hours using 100 CPU cores and 20 GPUs, consistent with the resources used in previous benchmarks [45].

The total runtime comprised 1 hour 20 minutes for foreground and contour prediction (GPU), 2 hours 7 minutes for hierarchical segmentation, 40 minutes for segment association (*i.e.* linking), 3 hours for the initial tracking pass, 2 hours 8 minutes for the second pass, and 18 minutes for data export, Figure 4.12. Cumulatively, this effort represented 25 hours of GPU time and 31 days 22 hours of CPU time, resulting in over 21.5 million cell instances. While certain stages, such as hierarchical hypothesis generation and ILP solving, exhibit superlinear scaling with respect to data size, the parallel and distributed computation kept the total runtimes usable.

Through out-of-core processing, the implementation extends across various computing environments, from high-performance clusters to standard consumer laptops. Memory requirements for handling arbitrarily large datasets are constrained primarily by the memory needed for a single 3D image volume. For example, the 3.7 TB zebrafish dataset was successfully processed on a laptop with 32 GB of RAM in 5 days by accessing data remotely via a 1 Gbps connection, Figure 4.13,[3].

The resulting lineages consisted of millions of segments spanning hundreds of time points. To evaluate tracking fidelity, we manually audited time-lapses to determine the

[3]Hardware details available at `https://github.com/royerlab/ultrack_supplementary/tree/main/hardware`

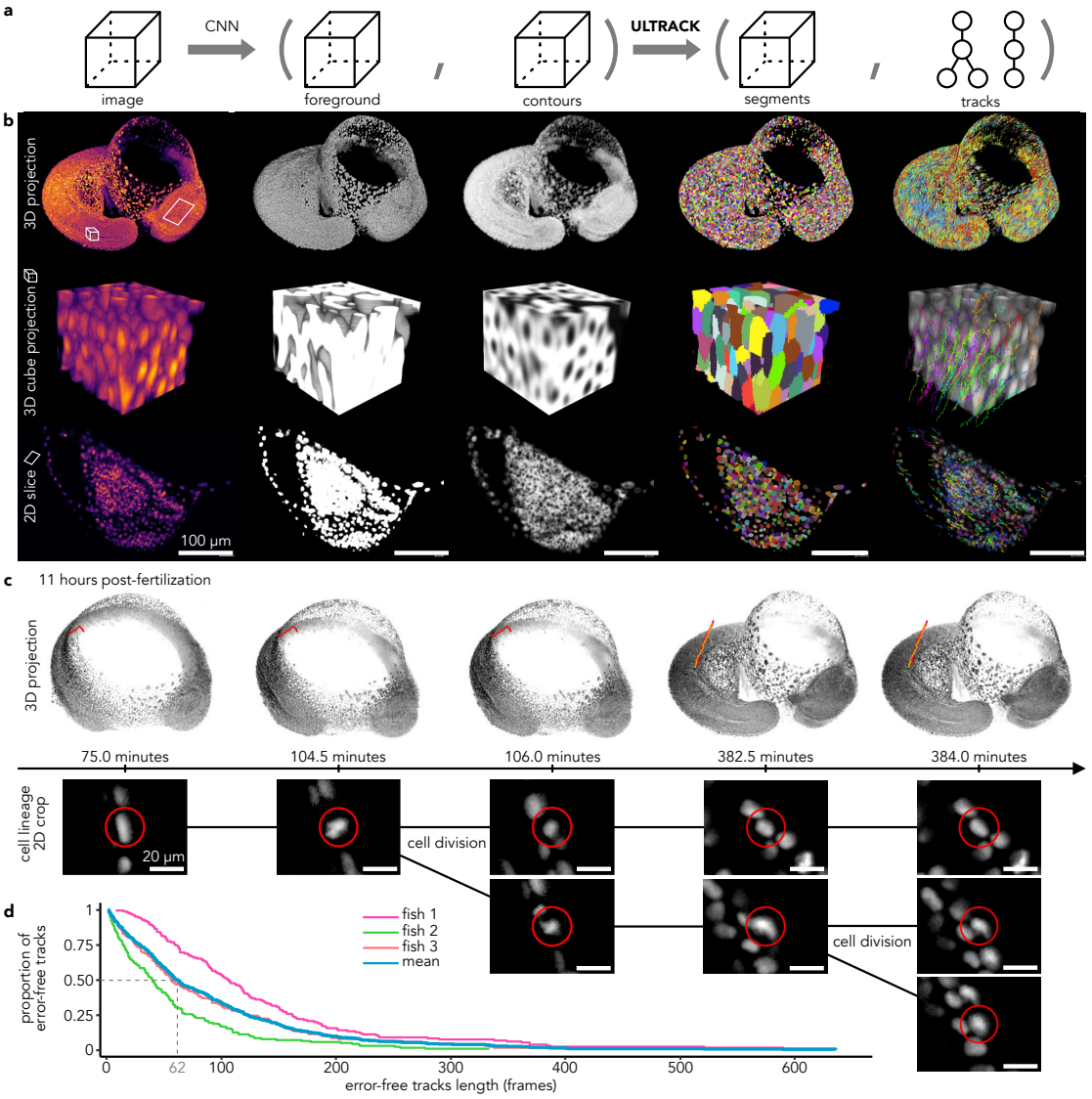


Figure 4.11: Multi-terabyte cell tracking of zebrafish embryo. **a**, pipeline for joint segmentation and tracking using a convolutional neural network and Ultrack. **b**, intermediate results at each stage: original image (first column); predicted foreground (second column) and contour probabilities (third column); Ultrack segmentation (fourth column) and tracking results (fifth column). **c**, detailed views of fully automated lineage reconstructions, showcasing two rounds of cell divisions and their corresponding 2D slices. **d**, distribution of the proportion of error-free tracks for varying lengths across three embryos, indicating that, on average, 50% of the lineages are error-free for at least 62 frames.

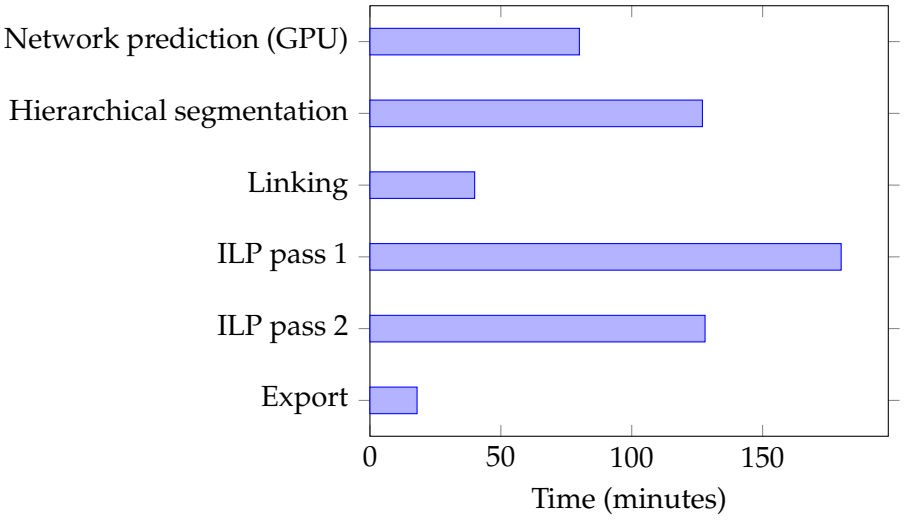


Figure 4.12: Runtime distribution across major pipeline stages for the 3.7 TB zebrafish embryo dataset. The network prediction step used GPUs, while all subsequent steps ran on CPU cores in a distributed environment.

duration of error-free tracks. A stratified sampling strategy was applied to distinct regions, including the head, body, tail bud, and presomitic mesoderm, across multiple time points and varying image qualities. Approximately 140 lineages per embryo (n=3) were randomly sampled. As illustrated in Figure 4.11d, 50% of the lineages remained error-free for at least 62 frames, representing roughly one hour of biological development. Reconstructions in Figure 4.11c demonstrate the framework's ability to maintain identity through consecutive cell generations.

By comparison, previous state-of-the-art methods tracked earlier-stage mouse embryo datasets (which contain fewer and less densely distributed nuclei) in 44 hours with similar computational resources to segment approximately 7 million instances [45]. Although dense ground-truth annotations were unavailable for quantitative benchmarking on this specific zebrafish dataset, results suggest accuracy levels comparable to the embryonic benchmarks reported in the Cell Tracking Challenge, Table 4.1.

These results emphasize the scalability and robustness of Ultrack for multi-terabyte, high-resolution datasets. The method facilitates the analysis of extended developmental periods across diverse hardware configurations, from HPC clusters to personal workstations, establishing it as a versatile tool for joint segmentation and tracking of large-scale microscopy data.

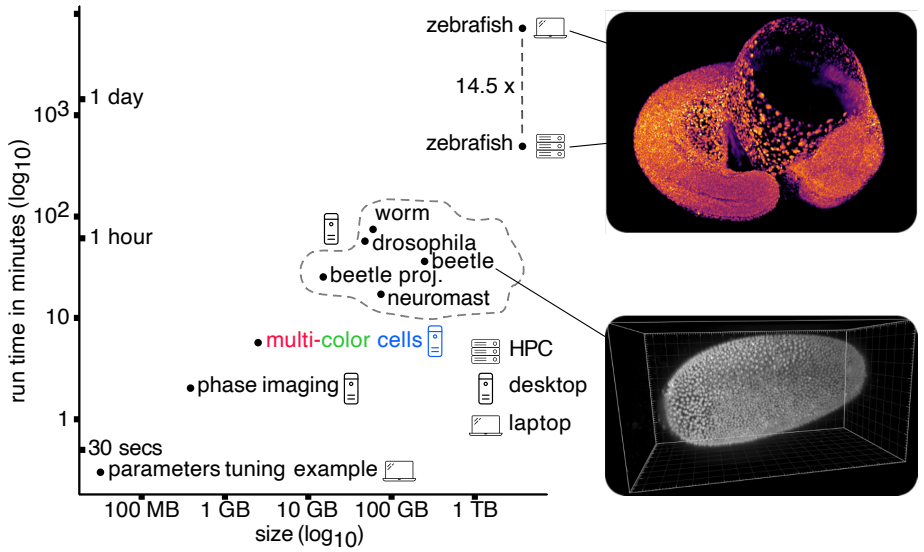


Figure 4.13: Runtime scalability of Ultrack across datasets and hardware configurations. Each point represents the total processing time (y-axis, log scale) as a function of dataset size (x-axis, log scale) for different acquisition types and computing environments (legend). The results span diverse biological datasets, including worm, beetle, neuromast, and zebrafish volumes, ranging from hundreds of megabytes to terabytes. While certain stages of the pipeline, such as segmentation hypotheses generation and integer linear programming (ILP) optimization, scale superlinearly with data size, Ultrack maintains practical runtimes through parallel and distributed computation. Examples of zebrafish and beetle datasets are shown in insets on the right.

4.10 Near-Perfect Cell Tracking of Organ-Scale 3D Datasets

Cell tracking is often pivotal in scenarios with fewer cells, where accurate lineage reconstruction is required to uncover precise details of cell behavior such as division, migration, and death, [170, 171],. To evaluate Ultrack's performance in such a context, we chose the zebrafish neuromast as a model system due to its well-defined structure and its significance in studying sensory organ development. Previous efforts in segmenting and tracking cells within this organ involved manual cell tracing, [172], or semi-automated segmentation restricted to single frames, [173],. To our knowledge, no other study has reported fully automated segmentation and tracking of cells in this complex 3D organ over extended temporal periods.

In this study, a zebrafish neuromast was imaged for 42 hours using membrane and nuclear markers. The nuclear marker, Tg(she:H2B-EGFP), specifically labels neuromast

cells—including hair, support, and mantle cells—while the membrane marker, Tg(cldnb:lyn-mScarlet), labels the neuromast, ionocytes, and surrounding skin cells. A 3D Cellpose model, [58], was fine-tuned to detect only the cells expressing both markers.

Imaging was conducted using an Olympus IX83 microscope equipped with a microlens-based, super-resolution spinning disk confocal system (VT iSIM, VisiTech International) and a 60x 1.3 NA silicone-oil objective. This setup produced a dataset of 500 frames in two color channels, with volumes of 73 × 1024 × 1024 voxels at a resolution of 250 × 76.6 × 76.6 microns per voxel. Frames were acquired at 5-minute intervals.

The resulting Cellpose predictions were converted into foreground and contour maps to serve as input for the Ultrack algorithm, Figure 4.14a,. This approach enabled the reconstruction of lineages spanning the entire time-lapse, capturing diverse cellular events including intra-organ migration, mitosis, and apoptosis, Figure 4.14b-c,.

To assess accuracy, the generated lineages were visually inspected and manually corrected to create ground-truth reference lineages. Performance was quantified using the CTC evaluation routines, Figure 4.14d. Ultrack achieved a tracking accuracy (TRA) score of 0.9989, demonstrating high precision in this scenario, segmentation and tracking results are provided in Video 9 .

The framework was benchmarked against TrackMate, [151], the most widely used tracking software, using identical Cellpose segmentations as input. To ensure a fair comparison, parameters for both Ultrack and TrackMate were optimized using a separate 50-frame dataset before application to the full 500-frame dataset.

In this evaluation, Ultrack exhibited an F1-score of 0.9951 compared to 0.9895 for TrackMate, representing a more than two-fold reduction in errors (279 vs. 721 total mistakes), Figure 4.14d. The primary source of error in the TrackMate results was attributed to imperfect initial segmentations, despite available filtering options, the tool struggled to resolve over-segmentation and false merges. Conversely, Ultrack naturally incorporates these imperfect segments as hypotheses, enabling more accurate merging and temporal association, Figure 4.14e. These results indicate that the joint optimization framework can maintain high accuracy in long-term imaging scenarios, supporting its utility for detailed studies of tissue dynamics.

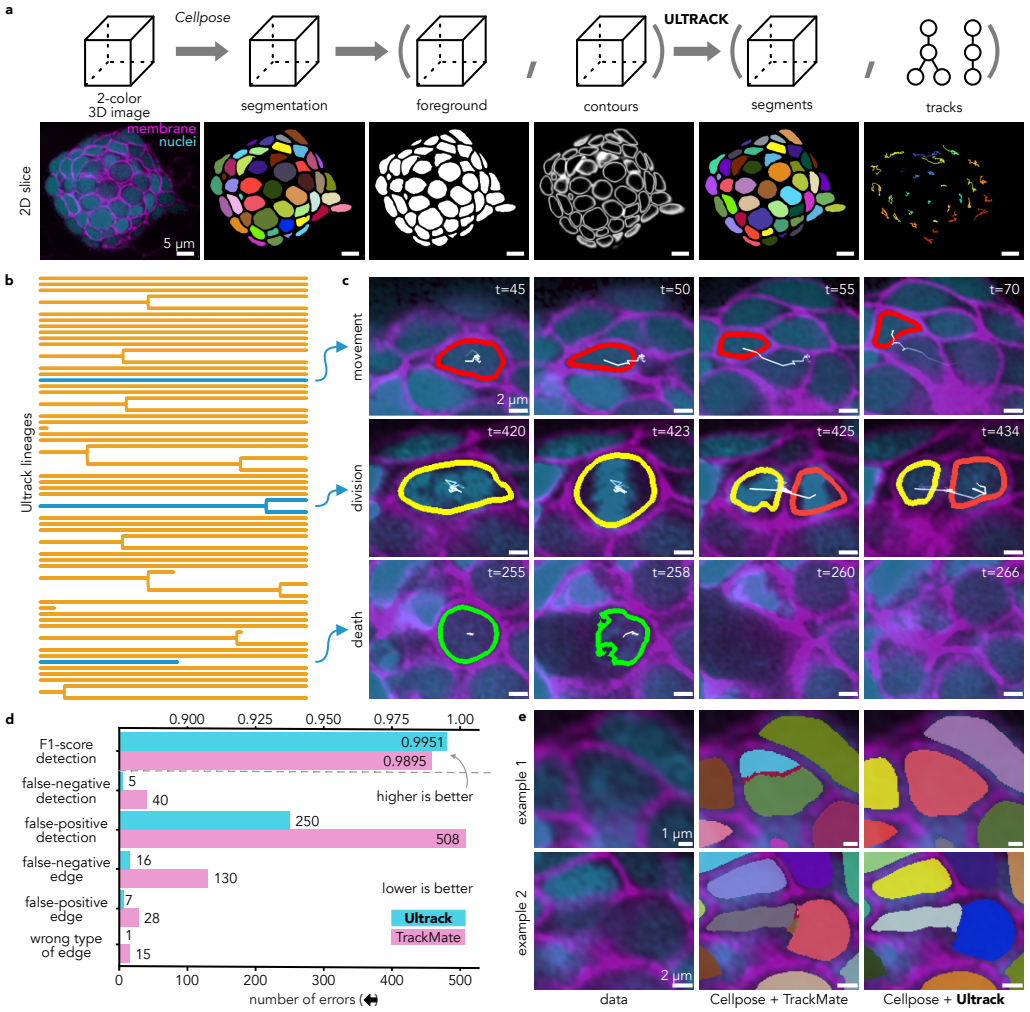


Figure 4.14: Near-perfect nuclear- and membrane-based 3D tracking of zebrafish neuromast cells. a, Pipeline for joint segmentation and tracking using Cellpose and Ultrack, showing 3D schematics (top) and 2D slice examples (bottom) at each stage: original 2-color image, Cellpose segmentation, foreground and contour probabilities, Ultrack segmentation, and tracking results. **b**, Dendrogram recovered by Ultrack for 71 cells over 500 frames (41.7 hours), with orange lines representing cell lineages and blue highlights indicating examples detailed next: **c**, Time-lapse images illustrating cell movement (red outlines), division (yellow/orange outlines), and death (green outlines). **d**, Quantitative comparison between Ultrack (blue) and TrackMate (pink). The bar chart shows error counts for various metrics, with Ultrack obtaining 442 fewer mistakes than TrackMate. **e**, Two examples demonstrating Ultrack's robustness to over-segmentation, showing the original image (left), fragmented TrackMate results (middle), and accurate Ultrack segmentation and tracking (right) despite imperfect Cellpose input.

Conclusions

This chapter has provided an empirical assessment of the Ultrack joint segmentation and tracking framework across a variety of biological and computational contexts. By evaluating performance on standardized benchmarks and massive developmental datasets, these results define the practical utility and technical boundaries of the proposed approach.

Experimental results from the Cell Tracking Challenge (CTC), Section 4.6, and the Epithelial Cell Benchmark (ECB), Section 4.7, demonstrate that the method achieves competitive performance on both sparsely distributed nuclear signals and densely packed membrane-labeled tissues. A notable feature of the framework is its ability to produce accurate lineages using classical image-processing inputs, effectively bypassing the need for extensive 3D manual annotations or model fine-tuning in scenarios where training data is unavailable. Furthermore, the integration of multiple segmentation hypotheses derived from parameter sweeps, Section 4.2, or diverse algorithms was shown to mitigate common tracking errors caused by over- or under-segmentation.

The flexibility of the multilevel contour representation was further validated through:

- **Intensity-based Tracking:** Direct application of the algorithm to processed image intensities allowed for successful tracking in label-free quantitative phase imaging, leveraging virtual staining signals, Section 4.3.
- **Multi-channel Integration:** Combining single-channel labels with color-feature association improved lineage reconstruction in complex multi-colored cell populations, Section 4.4.
- **Motion Compensation:** The use of GPU-accelerated flow-field registration maintained temporal consistency even in datasets with high motility or low temporal resolution, Section 4.5.

The framework successfully scaled to multi-terabyte datasets, maintaining practical runtimes on both high-performance computing clusters and personal workstations

through distributed processing, Section 4.9. Additionally, the dual-channel sparse labeling protocol introduced in Section 4.8 provided a higher-fidelity ground truth for validating long-term tracking in crowded 3D environments, revealing a robust maintenance of cell identities over extended developmental periods.

Despite these results, the performance remains intrinsically linked to the quality of the underlying foreground and contour evidence. Persistent systematic errors in boundary detection can lead the algorithm toward erroneous solutions. Additionally, the current model assumes a degree of segmentation consistency that can be challenged by extreme displacements if temporal registration is not used.

Future refinements may incorporate transformer-based sequence prediction to enhance long-term temporal context or interactive correction tools that leverage the existing pool of hierarchical hypotheses to streamline manual curation. Collectively, the findings in this chapter establish joint optimization as a scalable and adaptable solution for large-scale biological microscopy analysis.

Chapter 5

Stability Clustering

The previous chapters, Chapter 3 and Chapter 4, introduced and assessed the performance of our new method for identifying cells' segments (*i.e.* pixel clusters) from a set of segmentation hypotheses by finding the most temporal consistent segmentations across frames.

In this chapter, we extend this selection process to the general problem of clustering of arbitrary data. We assume the data is provided as a precomputed hierarchical clustering, and our goal is to identify a flat clustering that captures the underlying structure of the data.

In this scenario, the standard approach is to choose an arbitrary threshold on some attribute of the hierarchy's nodes, known as a *horizontal cut*, (*e.g.* linkage distance or dendrogram height) and select the maximal clusters below that threshold. However, this naïve strategy can fail in several situations, such as when the chosen attribute distribution varies across clusters, Figure 5.1. Therefore, to obtain a successful partitioning of the data, the user requires prior knowledge of the data distribution process, which is not always available, for example, the number of clusters or the variance of each group.

To circumvent this, we propose a new method that generalizes the previously introduced joint segmentation and tracking by converting it into a problem of finding *non-horizontal* cuts in an arbitrary hierarchy even when temporal information is unavailable.

While in the tracking setting, temporal consistency across image sequences provides a natural source to evaluate the quality of the segmentation hypotheses, in the general

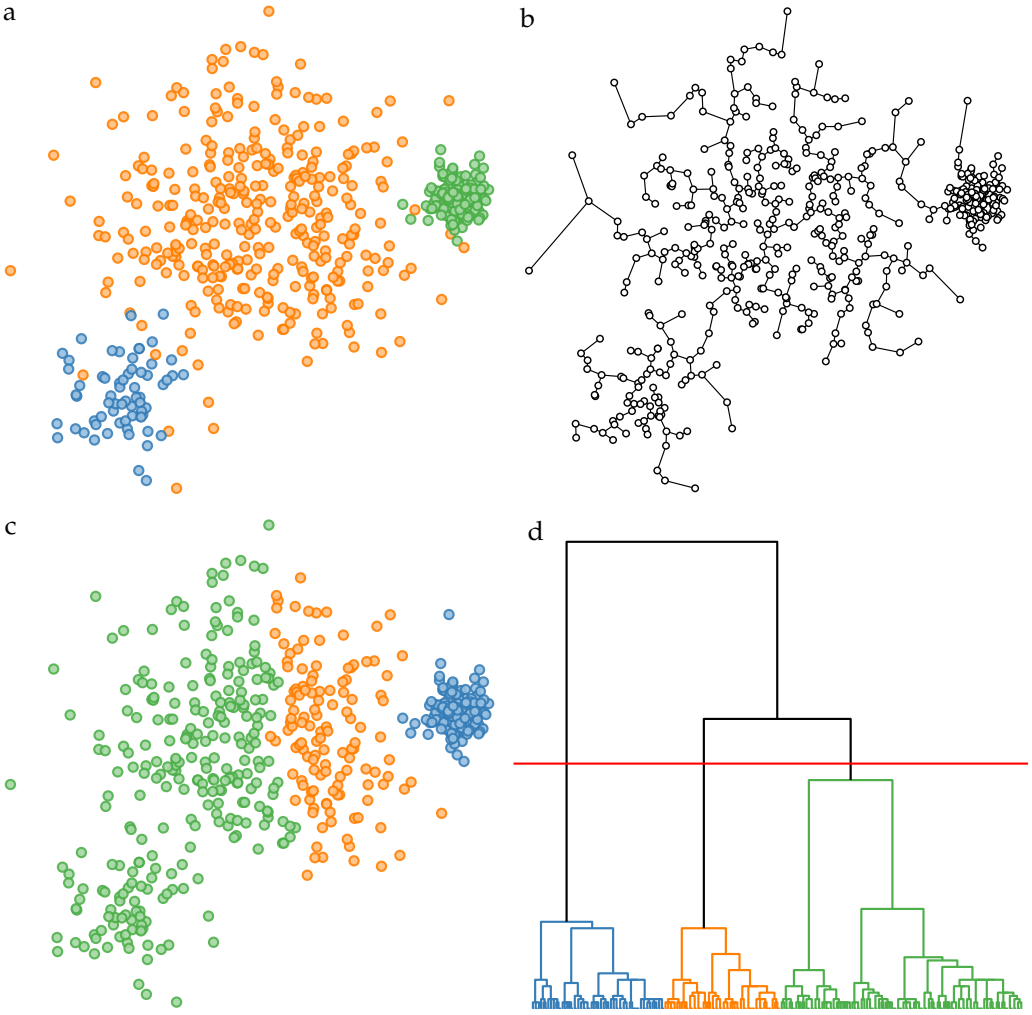


Figure 5.1: Naïve horizontal cut of a hierarchy. **a,** Ground truth clustering, **b,** Hierarchical clustering edge layout representation, **c,** Predicted clustering, **d,** Dendrogram with horizontal cut as red line. Each cluster has a distinct spread and number of elements.

clustering problem most datasets are not time-series. To compensate, we introduce perturbations to the data, generating variations of the original dataset. The key assumption is that, under moderate perturbations, the hierarchy will only change in non-significant regions, while the true clusters will remain stable. This idea builds on prior work in *stability clustering* [174, 175] and *stability model selection* [176, 177].

Stability clustering rests on the principle that meaningful clusters should be reproducible when data is resampled from the underlying distribution. Typically, the clustering algorithm is fixed, and stability-based methods are used for model selection, most

notably, to estimate the number of clusters [178]. Where a clustering algorithm defines the optimal solution given the provided parameters. For example, in k-means, the cluster assignment minimizes within-cluster variance. Our approach differs in that the hierarchy itself defines the space of possible cluster structures, and our proposed method identifies the *globally most stable* (optimal) flat clustering while evaluating combinations of multiple possible partitions. To our knowledge, this the first method to find the most stable clustering over a combinatorial set of partitions rather than searching through individual flat clustering solutions.

Compared to an existing approach for non-horizontal hierarchical cuts [96], which is restricted to a small family of monotonic increasing energies (cut costs), our formulation is more general. However, because the number of possible partitions in the hierarchy increases exponentially relatively to the number of leaves, the ILP does not scale to large datasets. To address this, we develop an approximate dynamic programming algorithm that achieves similar solution quality with polynomial-time complexity.

Defining the perturbations is a non-trivial task. For groups with distinct variances, too little perturbation can over-partition the large-variance groups, while too much can under-partition the small-variance groups. To circumvent this, we introduce a distribution-free noise model for hierarchical clustering. Avoiding restrictive assumptions about the data distribution or the need for explicit resampling [179] — an essential advantage for datasets containing small clusters.

We evaluate our method on synthetic datasets and real-world applications, particularly single-cell RNA sequencing (scRNAseq) data. Interest in clustering quality assessment has recently resurged in this field, as clustering is a central step in scRNAseq analysis pipelines. Recent studies have explored measures such as cluster stability [180] or cell-type classification accuracy [181]. Despite this renewed attention, earlier theoretical work questioned the utility of stability-based approaches [182, 183], highlighting that while instability implies poor clustering, multiple distinct stable solutions can exist, especially in symmetric data, Figure 5.2. Indicating that non-stability indicates poor clustering, but stability does not guarantee uniqueness. We acknowledge and confirm this theoretical limitation and focus on identifying highly stable partitions that capture biologically meaningful structure in scRNAseq data.

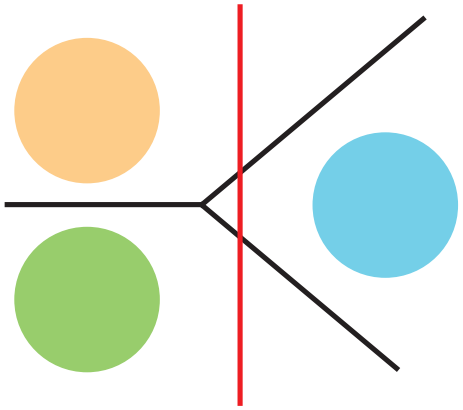


Figure 5.2: Dataset with multiple stable clustering solutions. Individual clusters are indicated by circles with different colors. The red line indicates a stable partition into two clusters. Black lines indicate a stable partition into three clusters. Note that the orange and green groups are closer together than the blue group.

5.1 Sequential Stability Non-Horizontal Cut

This first section introduces the new definitions for stability clustering and establishes its connection to the tracking problem. We begin with a formulation that uses sequential alignment of perturbations. This setup is not ideal as it depends on the arbitrary ordering of the perturbed views of the data. However, it is closely related to tracking and illustrates the connection between the two problems. In the following section, Section 5.2, we will present a more robust *one-vs-rest* formulation that removes this dependency.

The algorithm proceeds as follows. Given an edge-weighted graph G and a hierarchical clustering algorithm $\Phi : G \mapsto H$, it repeatedly applies Φ to cluster perturbed versions of G, obtaining a set of hierarchies $\mathcal{H} = \{H_0, H_1, \ldots, H_m\}$, where m is the number of perturbations and $H_0 = \Phi(G)$ is the unperturbed hierarchy. By ordering the hierarchies in $\mathcal{H}$ and defining a similarity function between the nodes of adjacent hierarchies, the algorithm identifies the most stable, non-overlapping nodes across all hierarchies, as illustrated in Figure 5.3. The clustering algorithm is analogous to the tracking problem introduced in Chapter 3 without divisions, entries, or exits from the field of view.

For simplicity, the final flat clustering is derived from the selected nodes of the reference hierarchy, H_0. Extensions that recover fuzzy cluster memberships or quantify uncertainty at cluster boundaries are possible but beyond the scope of this work.

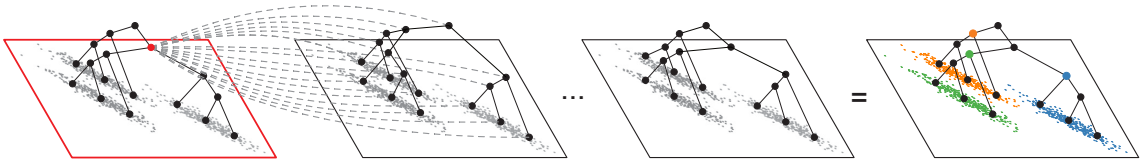


Figure 5.3: Graphical example of sequential stability non-horizontal cut. Data points are shown in gray on the 2D planes with their respective hierarchies represented as a dendrogram. Starting from a reference hierarchy H_0 (red box), multiple perturbed hierarchies $H_1, \ldots, H_m$ are generated. Edges connect nodes across adjacent hierarchies, weighted by the similarity function A. To avoid clutter, only the connections from the node highlighted in red are shown. From these similarity scores, the proposed algorithm identifies the most stable, non-overlapping nodes across all hierarchies, resulting in the flat clustering shown in the right-most figure.

Formally, let X be a sample of n elements from an unknown population $\mathcal{X}$, and let $w^G : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}^+$ denote an edge-weight function, such as the inverse distance or a similarity measure. The graph representation is then $G = (V^G, E^G, w^G)$, where the vertices are $V^G = \{x_1, \ldots, x_n\}$ and the edges $E^G \subseteq \{(i,j) \mid i,j \in V^G \times V^G\}$, often a k-nearest neighbor graph. A hierarchical clustering $H = (V^H, E^H)$ is a tree whose leaves correspond to the elements of X, and whose internal nodes represent clusters of X. This is often represented as a binary tree, such that each internal node has exactly two children — a merge of two clusters. In this scenario, H has $|V^H| = 2|V^G| - 1$ nodes. This structure is equivalent to the nested segmentation framework used for joint segmentation and tracking in Chapter 3.

Let $\eta : G \times \Phi \mapsto H$ be a perturbation function, to be specified later, such that $\eta(G, \Phi) = H'$ produces a perturbed hierarchy H' that differs from $\Phi(G)$ almost surely. The leaves of H' are identical to those of H, while the internal nodes (except for the root) can differ.

To measure stability, we define a similarity function between nodes of different hierarchies, $A : V^H \times V^{H'} \mapsto \mathbb{R}^+$. Because each node in H corresponds to a (sub)tree whose leaves represent the clustered elements, the similarity between two nodes can be expressed in terms of the overlap of their leaf sets. Let $\Lambda(p)$ denote the set of leaves of a subtree $p \in V^H$. Then the similarity is defined as

$$A(p,q) = |\Lambda(p) \cap \Lambda(q)|.$$

Thus, we formalize below a variant of the optimal assignment problem, where the

objective is to maximize node similarity across hierarchies while satisfying non-overlap constraints, Equation 5.4, such that nodes of G (leaves of H) are selected at most once.

$$\arg\max_{\mathbf{x}} \quad \frac{1}{|\mathcal{H}|-1} \sum_{h\in\{1,\dots,|\mathcal{H}|-1\}} \sum_{p\in V_{h-1}} \sum_{q\in V_h} A(p,q)\, x_{hpq} \tag{5.1}$$

$$\text{s.t.} \quad y_{(h-1)p} = \sum_{q\in V_h} x_{hpq} \quad \forall\, h,p \in \mathcal{H}\times V_{h-1} \mid h>1 \tag{5.2}$$

$$y_{(h+1)q} = \sum_{p\in V_h} x_{hpq} \quad \forall\, h,q \in \mathcal{H}\times V_{h+1} \mid h<|\mathcal{H}|-1 \tag{5.3}$$

$$1 \geq \sum_{p\in \text{anc}_h(q)\cup\{q\}} y_{hp} \quad \forall\, h,q \in \mathcal{H}\times V^G \tag{5.4}$$

$$y_{hp} \in [0,1] \quad \forall\, h,p \in \mathcal{H}\times V_h \tag{5.5}$$

$$x_{hpq} \in \{0,1\} \quad \forall\, h,p,q \in \mathcal{H}\times V_h \times V_{h-1} \mid h>1 \tag{5.6}$$

Here, x_{hpq} is a binary indicator variable for the correspondence between nodes p and q in consecutive hierarchies H_{h-1} and H_h, respectively, and y_{hp} indicates whether node p in H_h is selected. Equation 5.2 and Equation 5.3 enforce that each cluster (node) in a hierarchy can match with at most one cluster in the adjacent hierarchy. Equation 5.4 ensures that each leaf in H (node in G) belongs to at most a single flat clustering.

In the perfectly stable case, where the selected clusters have perfect overlap with their correspondents, the ILP score is the number of leaves of H_0 ($|V_G|$). By dividing it by $|V_G|$ it becomes a normalized score upper bounded by 1. This score provides an assessment of the cluster quality, evaluating the fitness of the selected clustering to the data without any assumption about the data distribution.

As mentioned above, this formulation is equivalent to the tracking problem introduced in Chapter 3, differing in the similarity function and the absence of auxiliary variables that enables cell divisions, entries, or exits from the field of view.

Because the ordering of hierarchies in $\mathcal{H}$ is arbitrary and not inherent to the data, this formulation is sensitive to the chosen sequence of perturbations. In the next section, we introduce a new formulation that eliminates this dependency.

5.2 One vs Rest Stability Non-Horizontal Cut

To eliminate the dependency on the ordering of the hierarchies, we modify the formulation to match all perturbed hierarchies to the reference hierarchy simultaneously, as shown in Figure 5.4.

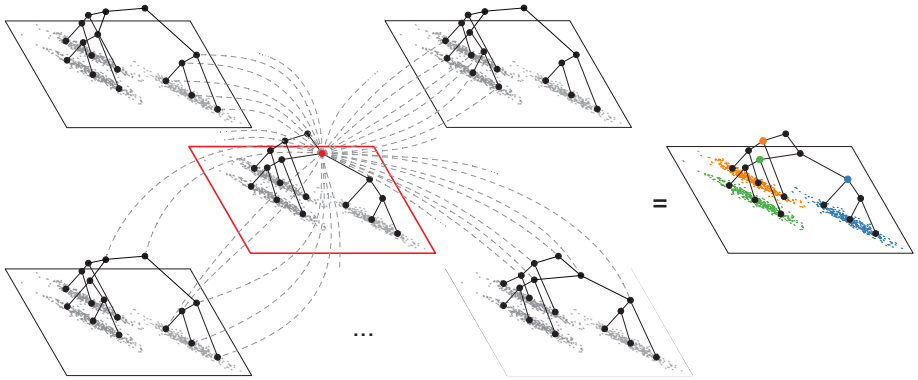


Figure 5.4: Graphical example of one vs rest stability non-horizontal cut. Starting from a reference hierarchy H_0 (red box), multiple perturbed hierarchies $H_1, \ldots, H_m$ are generated. The nodes between the reference hierarchy and each perturbed hierarchy are connected by edges, weighted by the similarity function A. A subset of connection are shown for the node highlighted in red. From these similarity score, the proposed algorithm identifies the most stable, non-overlapping nodes across all hierarchies, resulting in the flat clustering shown in the right-most figure.

The objective function remains the same as in the previous section, but the constraints differ:

$$\arg\max_{\mathbf{x}} = \frac{1}{|\mathcal{H}|-1} \sum_{h\in\{1,\ldots,|\mathcal{H}|-1\}} \sum_{p\in V_0} \sum_{q\in V_h} A(p,q)x_{hpq} \tag{5.7}$$

$$\text{s.t.} \quad y_{0p} = \sum_{q\in V_h} x_{hpq} \quad \forall\, h,p \in \mathcal{H}\times V_0 \mid h>1 \tag{5.8}$$

$$y_{hq} = \sum_{p\in V_0} x_{hpq} \quad \forall\, h,q \in \mathcal{H}\times V_h \mid h>1 \tag{5.9}$$

$$1 \geq \sum_{q\in \text{anc}_h(p)\cup\{p\}} y_{hq} \quad \forall\, h,p \in \mathcal{H}\times V^G \tag{5.10}$$

$$y_{hp} \in [0,1] \quad \forall\, h,p \ \in \mathcal{H}\times V_h \tag{5.11}$$

$$x_{hpq} \in \{0,1\} \quad \forall\, h,p,q \in \mathcal{H}\times V_0 \times V_h \mid h>1 \tag{5.12}$$

Equation 5.8 enforces that every selected node p in the reference hierarchy H_0 must be linked to exactly one node q in each perturbed hierarchy H_h. Symmetrically, Equation 5.9 requires that every selected node q in a perturbed hierarchy must be linked to exactly one node p in the reference hierarchy H_0.

As in the previous formulation, the number of elements in $\mathbf{x}$ (array of x_{hpq} variables) is $O(|\mathcal{H}||V^H||V^H|) = O(|\mathcal{H}|(2|V^G|-1)(2|V^G|-1)) = O(|\mathcal{H}||V^G|^2)$, and the number of $\mathbf{y}$ variables is $O(|\mathcal{H}||V^H|) = O(|\mathcal{H}||V^G|)$. Equation 5.8 and Equation 5.9 contribute to $O(|\mathcal{H}||V^G|)$ constraints. Equation 5.10 yields $O(|\mathcal{H}||V^G|)$ constraints. Given that solving binary ILPs is NP-hard, this formulation is intractable for datasets with a few thousands of nodes.

In the next section, we introduce an approximate algorithm that achieves similar solution quality with polynomial-time complexity.

5.3 An Approximate Algorithm for Clustering Stability

The approximate algorithm is motivated by two main observations:

1. The *one-vs-rest* formulation can, in principle, be decomposed into separate subproblems, rather than solved as a single ILP in which all perturbed trees are simultaneously matched to the reference. The optimal global objective would then be the sum of the optimal objectives of these subproblems. However, all trees must agree with respect to the selection of nodes on the reference tree, which if enforced simultaneously brings us back to the original global ILP formulation.

2. Unlike tracking, the explicit association between hierarchies is not of primary interest. Only the node selection variables $\mathbf{y}$ matter. Moreover, $\mathbf{y}$ is $O(|V^G|)$ times smaller than $\mathbf{x}$. Thus, if we could eliminate $\mathbf{x}$ and work only with $\mathbf{y}$, the problem would be substantially simplified.

These observations lead to the following question: *Can the overlap and single-assignment constraints be relaxed into individual ILPs, and the resulting variables be projected onto a valid global solution afterward?*

Because the variables $\mathbf{y}_0$ (selected internal nodes of the reference hierarchy H_0) must respect their respective tree structure, we were inclined to believe that an efficient projection algorithm exists.

Starting from the original *one-vs-rest* ILP objective, Equation 5.13, and its constraints C (from Equation 5.8 to Equation 5.12), we can conceptually decompose it into subproblems as follows:

$$\max_{\mathbf{x}} \frac{1}{|\mathcal{H}|-1} \sum_{h\in\{1,\dots,|\mathcal{H}|-1\}} \sum_{p\in V_0} \sum_{q\in V_h} A(p,q)\, x_{hpq} \quad \text{s.t. } C \tag{5.13}$$

$$\approx \frac{1}{|\mathcal{H}|-1} \sum_{h\in\{1,\dots,|\mathcal{H}|-1\}} \underbrace{\max_{\mathbf{x}_h} \sum_{p\in V_0} \sum_{q\in V_h} A(p,q)\, x_{hpq}}_{\text{individual ILP s.t. } C'} \tag{5.14}$$

Here, C' denotes the constraints of the individual ILPs for each perturbed hierarchy. In Equation 5.14, the original optimization problem is approximated by m (number of perturbations, $m = |\mathcal{H}| - 1$) independent ILPs. Ideally, the combined solution would still satisfy the global constraints C, which is not guaranteed a priori but will be enforced by a subsequent projection step. In practice, the constraint set C' is further relaxed and the non-overlap constraints are ignored within the individual ILPs because the projection step will enforce them later. Thus, each subproblem is an standard optimal assignment problem and can then be solved in polynomial time, $O(|V^G|^3)$, using the Hungarian matching algorithm [116].

What remains is to modify the objective function so it only depends on the variables $\mathbf{y}_0$ and to define the projection step.

Because each individual ILP (or matching problem) guarantees a single assignment q per perturbed hierarchy for each node p in H_0, we can assign to each node p a *association score* $z_h(p)$ for each perturbed hierarchy $H_{h|h>1}$, defined as

$$z_h(p) = \sum_{q\in V_h} A(p,q)\, x_{hpq} \quad \forall\, h,p \in \mathcal{H}\times V_0 \mid h > 1. \tag{5.15}$$

Using these scores, the objective function (Equation 5.14) can be rewritten in terms of the reference-tree selection variables $\mathbf{y}_0$:

$$\max_{\mathbf{y}_0} \frac{1}{|\mathcal{H}|-1} \sum_{h\in\{1,\ldots,|\mathcal{H}|-1\}} \sum_{p\in V_0} y_{0p}\, z_h(p) \tag{5.16}$$

As the number of perturbed trees increases, this objective can be interpreted as the expected score of the reference hierarchy H_0, where the perturbed hierarchies H_h are sampled from the perturbation function η:

$$\max_{\mathbf{y}_0} \sum_{p\in V_0} y_{0p} \mathop{\mathbb{E}}_{H_h\sim\eta(G,\Phi)}\big[z_h(p)\big] \quad \text{s.t. } C. \tag{5.17}$$

For simplicity, we will call $z_0(p) = \mathbb{E}_{H_h\sim\eta(G,\Phi)}\big[z_h(p)\big]$ the *expected association score* of node p of the reference hierarchy H_0.

Therefore, we have converted the edge-weighted objective function between multiple trees, Equation 5.13, into a node-weighted function of a single hierarchy, Equation 5.17.

We propose a bottom-up dynamic programming algorithm to obtain a valid assignment (*i.e.* a cut in the reference hierarchy H_0) from the expected association scores $z_0(p)$. Under the assumption that $z_0(p)$ is non-negative for all $p \in V_0$.

Algorithm 2 works as follows. We interpret $z_0(p)$ as the weight of selecting node p as a cluster representative. For each node p in H_0, the algorithm computes two quantities: (i) $T(p)$, the maximum total weight obtainable from the subtree rooted at p under the non-overlap (ancestor) constraint, and (ii) $P(p)$ a partition (*i.e.* cut), representing the respective partition (cluster) that resulted from the optimal solution for the subtree rooted at p. The procedure starts by initializing all nodes in the tree as their own cluster and their respective stability score, $P(p) = \{p\}$ and $T(p) = z_0(p)$.

We then process the tree in a bottom-up manner through a queue, Q, that tracks the nodes to be processed next. Whenever a node q is dequeued, we aggregate the best solutions from its children and compare the resulting total weight with the score of selecting

q itself. Where the children's score is the sum of their current best solution weights,

$$w_{\text{children}} = \sum_{s \in \texttt{children}(q|H_0)} T(s),$$

and compare it to $z_0(q)$. If $w_{\text{children}} > z_0(q)$, the optimal choice for the subtree rooted at q is to keep the union of the children's cuts, *i.e.* $P(q) = \bigcup_s P(s)$ and $T(q) = w_{\text{children}}$. Otherwise, it is better to keep q as the single cluster, in which case $P(q) = \{q\}$ and $T(q) = z_0(q)$. After processing q, we decrement a counter that tracks how many children of its parent remain to be processed; once all children of a parent p are done, p is added to the queue and handled in the same way.

This process continues until the root of H_0 is processed. The final solution $P(\texttt{root}(H_0))$ is a set of nodes with disjoint leaves that induces a valid flat clustering on H_0 while maximizing the total expected association score $\sum_{p \in P(\texttt{root}(H_0))} z_0(p)$.

This projection step is closely related to the Framework for Optimal Selection of Clustering (FOSC) [184]. FOSC showed that, for a restricted class of node-weighted hierarchy (often called quality scores), an optimal flat clustering can be obtained efficiently, provided the weight function satisfies two conditions: (i) *locality*, meaning that the score of each node can be computed independently of the final clustering; and (ii) *additivity*, meaning that aggregating scores along subtrees is meaningful with respect to the global quality measure. Under these assumptions, the optimal flat clustering can be recovered by a single bottom-up traversal of the tree, as done in Algorithm 2. In which each internal node compares its own quality score against the sum of the scores of its children and replaces them whenever its own score is larger. Our dynamic programming projection on H_0 follows exactly this principle: the expected association scores $z_0(p)$ play the role of local quality scores, and the quantities $T(p)$ aggregate them additively over subtrees. Thus, our procedure can be seen as an instance of the FOSC framework applied to the stability-based weights induced by perturbations.

Input: $z_0 : V_0 \to \mathbb{R}^+$: expected association score of each node $p \in V_0$
$H_0 = (V_0, E_0)$: reference hierarchy (tree)
Output: $P(\texttt{root}(H_0))$: selected nodes of H_0 forming the flat clustering
$T(\texttt{root}(H_0))$: total weight of the selected nodes

```
// Initializing all nodes as its own cluster
foreach p ∈ V_0 do
    P(p) ← {p}
    T(p) ← z_0(p)
end
z_0(root(H_0)) ← 0 // To avoid trivial solution
// Track how many children of each node remain to be processed
foreach p ∈ V_0 do
    remaining_children(p) ← |children(p | H_0)|
end
// Initialize queue Q with all leaves of H_0
Q ← {ℓ | ℓ ∈ Λ(root(H_0))}
while Q ≠ ∅ do
    q ← Q.pop()
    // If q is not a leaf, we evaluate its children
    if q ∉ Λ(root(H_0)) then
        w_children ← Σ_{s∈children(q|H_0)} T(s)
        if w_children > z_0(q) then
            T(q) ← w_children
            P(q) ← ⋃_{s∈children(q|H_0)} P(s)
    end
    if q ≠ root(H_0) then
        p ← parent(q | H_0)
        remaining_children(p) ← remaining_children(p) − 1
        if remaining_children(p) = 0 then
            Q.append(p)
        end
    end
end
return P(root(H_0)), T(root(H_0))
```

Algorithm 2: Optimal flat clustering of the reference hierarchy H_0 with maximum expected association score

5.4 Perturbation model

The perturbation or sampling function η is a key component of any stability procedure. Non-parametric bootstrapping is one of the most common approaches to generate clustered variants of the data [179]. It corresponds to sampling a subset of the data with replacement and recomputing the clustering. This strategy can be detrimental when small clusters are present, as they may be underrepresented or even disappear in the bootstrap samples. Additionally, our approach requires a fixed set of leaves (*i.e.* input samples) across all hierarchies, thus, not being compatible with bootstrap resampling.

When the data-generating process is known, as in sequencing or imaging, a more recent alternative is *data thinning* [185, 186]. Data thinning samples the observations at the measurement level rather than at the sample level. For example, in sequencing, thinning is applied to the read counts rather than the cells, either through non-parametric bootstrapping or sampling from a Poisson or Negative Binomial distribution. Therefore, the dimensions (length and width) of the cell-by-gene data matrix is preserved, but the total sum of counts (*i.e.* sum of the matrix) is reduced. A similar idea applies to imaging, where each pixel records a photon count. In the scRNA-seq applications in Section 5.5 we will compare against this sampling strategy as a perturbation.

Another alternative that avoids resampling at the sample level is to replace a fraction of the data by noise, or to add noise directly to the observations (*i.e. jittering*) [187]. This work showed that bootstrapping or sub-sampling often outperforms jittering, and that replacing part of the data by noise can improve boostrapping strategies in some scenarios. In practice, however, choosing an appropriate noise level is non-trivial and typically requires assumptions about the data-generating distribution. Our approach avoids these assumptions by perturbing the hierarchy directly.

In contrast to the approaches above, our final goal is to perturb the *clustering hierarchy* rather than the data itself. We take advantage of the fact that most hierarchical clustering algorithms used in practice are constructive (*e.g.* agglomerative clustering) and operate in a bottom-up manner: at each step, selecting an edge of the input graph to merge two clusters, where the strongly connected regions are merged first. We propose subsampling the data at the *edge level*, because strongly connected regions (*i.e.* most likely true clusters)

contain many internal edges and are therefore robust to moderate edge subsampling, remaining unchanged. In contrast subgraphs which are connected by fewer edges (*i.e.* weakly connected), are more likely to change by subsampling, indicating that they are unstable and should not be selected as part of the final clustering. An example of this perturbation mechanism is shown in Figure 5.5.

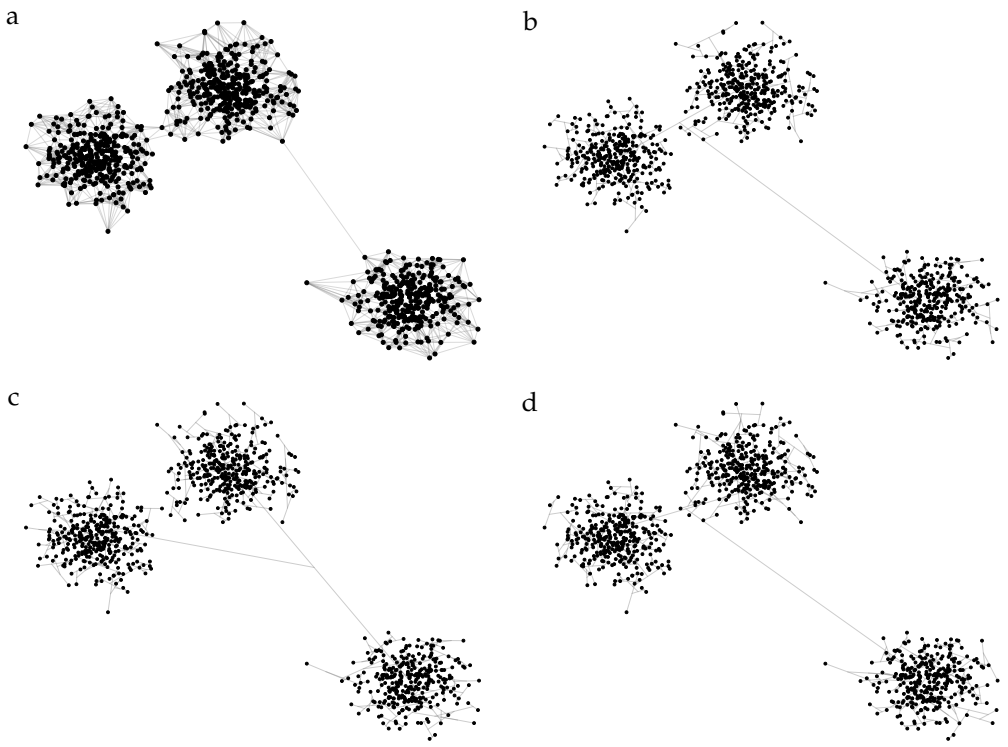


Figure 5.5: Hierarchies from edge perturbation. Three gaussians distribution in a 2D plane, with their respected edges. **a,** Input combined minimum spanning tree and 10-nearest neighbors graph, **b,** Original hierarchy H_0 represented such that edge end-points are the average location of leaves of the two merged clusters, **c,d** Perturbed hierarchies H_1 and H_2 with $pd = 0.5$ perturbation displacement proportion.

Because edge sub-sampling may produce disconnected components in the graph, in practice we rearrange the edge ordering induced by the agglomerative procedure, Algorithm 3. The motivation is that, if we were to remove a single edge, and this specific edge creates a disconnected graph, it could be added last, once the aglomeration is complete to guarantee a connected graph.

Let $R_0 \in \{1, 2, \ldots, |E| \mid R_{0i} \neq R_{0j} \; \forall \; i, j \in |E| \wedge i \neq j\}$ be the edge ranks computed when creating the reference hierarchy H_0, note that R_0 is a permutation of the indices of

the edges E_0.

The perturbation algorithm simply adds a random displacement to the edge ranks defined by the agglomerative procedure ordering, where the displacement is bounded by an user defined parameter, the perturbation displacement proportion, pd. The perturbed edge ranks R' are then used to compute the perturbed hierarchy H' by applying single-linkage to the graph weighted by the perturbed edge ranks. The time complexity of this procedure is $O(|R'| \log |R'|)$, the time complexity of sorting the reranked edges in the Kruskal's algorithm [136] with Tarjan's union-find takes $O(|R'|\alpha(|R'|))$ as described in [132], where α is the inverse Ackermann function [133], which is practically constant.

Input: G: input graph
R_0: edge ranks computed when creating the reference hierarchy H_0
$pd \in [0, 1]$: perturbation displacement proportion
Output: H': perturbed hierarchy
1 $max\_displacement \leftarrow \lfloor pd \cdot |R_0| \rfloor$
2 **foreach** $i \in \{1, \ldots, |E_0|\}$ **do**
3 $\quad R'_i \leftarrow R_{0i} + \lfloor \mathcal{U}(0, max\_displacement) \rfloor$
4 **end**
5 $H' \leftarrow$ single-linkage(G, R') `// Time complexity:` $O(|R'| \log |R'|)$
6 **return** H'

Algorithm 3: Perturbation of a hierarchy H_0, resulting from G and R_0, by edge reranking.

5.5 Experiments

We evaluate our clustering algorithm on both synthetic and real datasets. The synthetic experiments are based on two-dimensional (2D) data, which, although less challenging than real data, are carefully designed to exhibit different geometric and density structures. This allows us to visualize the behaviour of our method, to highlight its failure modes, and to objectively compare it against alternative clustering algorithms.

We then turn to single-cell RNA sequencing (scRNA-seq) data, where clustering is a central step in downstream analysis pipelines. In this setting, we compare our approach against commonly used clustering methods and assess its performance in terms of cluster stability and agreement with known biological annotations.

5.5.1 Synthetic Data Algorithmic Comparison

To generate the synthetic datasets, we use the scikit-learn library [188], each containing 1000 samples with distinct geometric and density structures, Figure 5.6. We compare against popular clustering algorithms spanning diverse algorithmic families: spherical methods (K-means, Gaussian Mixture Model), density-based approaches (DBSCAN, HDBSCAN, MeanShift), graph-based methods (Spectral Clustering, Leiden), and hierarchical linkage methods (Single-linkage, Average-linkage, Ward).

Experimental setup. To simulate realistic scenarios in which users cannot visually inspect the data structure or easily tune parameters, we use fixed parameter configurations for all methods. The only exception is for algorithms requiring a predefined number of clusters (*e.g.* K-means, Gaussian Mixture Model, Spectral Clustering), where we set $k = 2$ for the first two datasets based on the known ground truth and $k = 3$ for everything else. For hierarchical methods (Single-linkage, Average-linkage), flat clusterings are obtained by selecting dendrogram thresholds that yield the expected number of clusters. HDBSCAN [184] used its default cluster selection based on persistence (originally called Excess of Mass). We note that while Leiden [189] and DBSCAN [190] can achieve perfect results with careful per-dataset parameter tuning, such manual intervention is impractical for real-world applications where cluster structure is unknown. See Appendix A.3 for each algorithm's specific parameters.

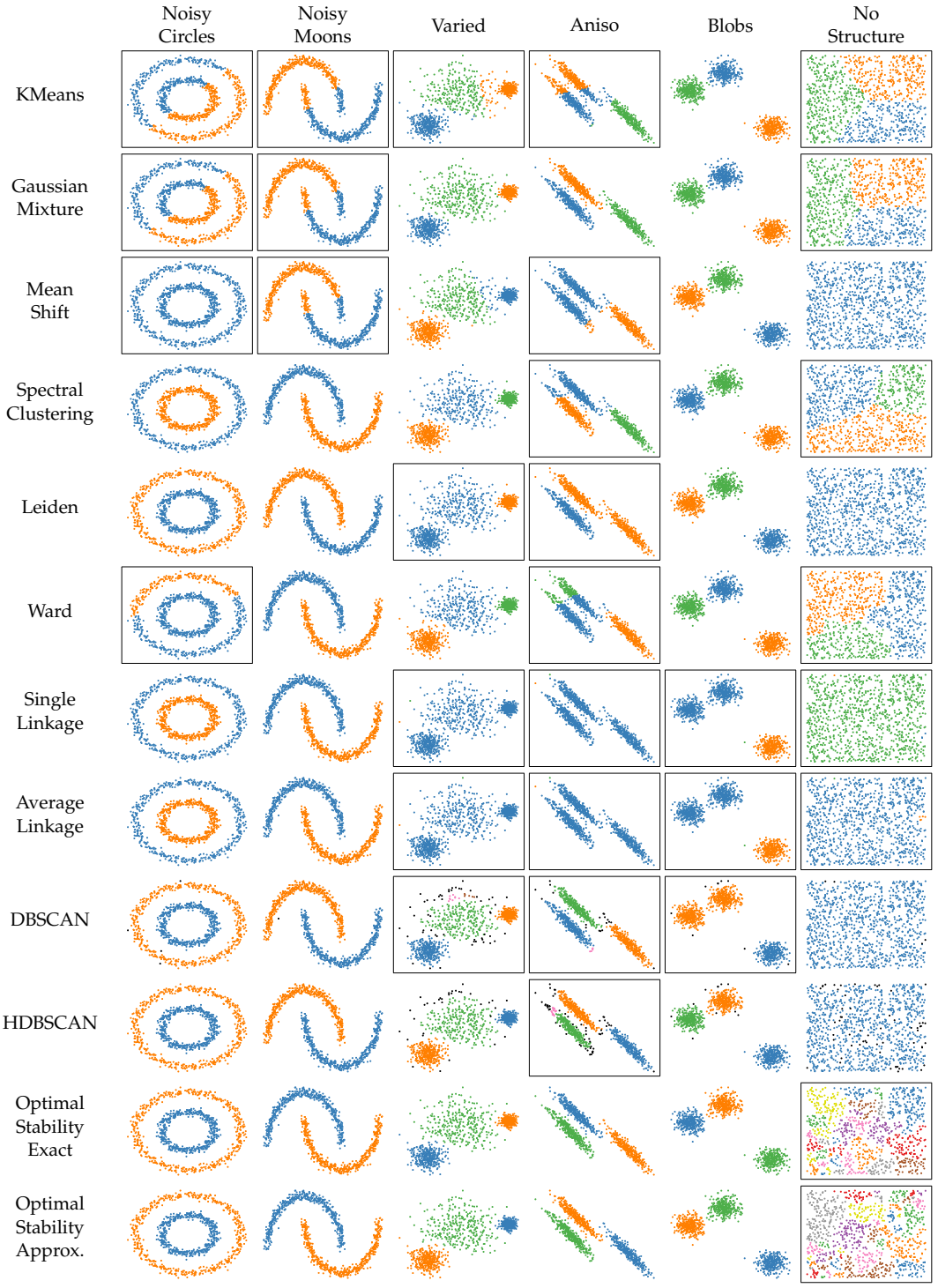


Figure 5.6: Synthetic data clustering comparison. Algorithms arranged in rows, datasets in columns, clustering results are indicated by the color of the nodes. All algorithms were executed with a fixed parameter setting. Bottom two rows show the results of our optimal stability exact and approximate algorithms. Instances where the clustering did not achieve the expected results are highlighted with a black border.

Failure modes across algorithm families. The experiments failure patterns that correlate with specific algorithmic assumptions. Methods relying on spherical or isotropic distance metrics (K-means, GMM, MeanShift, Ward) consistently fail on datasets with non-convex geometries (Noisy Circles, Noisy Moons) and anisotropic variance structures (Aniso dataset). These failures stem from their fundamental inability to capture manifold structure or adapt to directional variance in the data.

The Varied dataset, containing isotropic Gaussians with distinct variances, exposes a different challenge. Here, algorithms must handle groups with dramatically different spreads. We observe three distinct failure modes: (i) K-means correctly identifies the cluster centers but misplace boundaries due to its assumption of equal cluster variance; (ii) Leiden under-partition the high-variance cluster; (iii) DBSCAN fragments the data by creating a small spurious cluster.

The Aniso dataset proves most challenging overall. Beyond the expected failures of isotropic methods (K-means, MeanShift, Ward), we observe additional failures in methods typically robust to complex geometries. Both linkage-based algorithms select outliers as their own clusters, failing to capture the global structure. Leiden and MeanShift fused clusters, while DBSCAN and HDBSCAN exhibit the opposite problem, fragmenting the data by creating small clusters in lower-density regions.

The Blobs dataset, containing three isotropic Gaussians with two positioned close together, reveals how algorithms handle cluster ambiguity. The challenge lies in determining whether to preserve the separation between nearby clusters or merge them. Different algorithms make different trade-offs, with no clear consensus on the correct resolution without additional domain knowledge.

The rightmost column (No Structure dataset) exposes a fundamental limitation shared by nearly all clustering algorithms: they impose structure even when none exists. Our method inherits this limitation, as we must exclude the root node to avoid a trivial single-cluster solution, which prevents us from naturally representing the absence of meaningful clusters. However, this failure mode can be partially mitigated in practice, because clustering of unstructured data tend to produce low stability scores, which can be used as a diagnostic measure to assess if clustering may be inappropriate for the given dataset.

It is worth highlighting that the *exact* and *approximate* algorithms results are equivalent for all datasets, only showing differences in No Structure dataset where no meaningful clusters exist.

Performance of optimal stability clustering. Both exact and approximate variants of our proposed method successfully recover the expected cluster structure across five of the six test scenarios, failing only on the No Structure dataset where no meaningful clusters exist. This success is achieved without per-dataset parameter tuning, demonstrating robustness to diverse geometric patterns (convex, non-convex, elongated), varying density structures (uniform, varied variance), and different scales. The consistent performance across these challenges suggests that stability-based optimization via perturbations effectively identifies meaningful cluster boundaries when they exist in the underlying hierarchy, while remaining agnostic to the specific geometric or density assumptions that limit other approaches.

5.5.2 Synthetic Data Ablation Study

Next, we assessed how sensitive our approach is to the choice of parameters by comparing the results when varying the amount of perturbation, the graph degree, and hierarchy construction algorithm and the number of replicates. We vary a single parameter at a time while keeping the others fixed, considering as baseline the configuration used in the main experiment with a combined minimum spanning tree and 10-nearest neighbors graph, a watershed hierarchy ordered by the number of non-leaf nodes [130], 0.5 proportion of perturbed edges and 10 perturbed replicates.

We first start by studying the two parameters intrinsic to our approach, the number of perturbed replicates, m, and the percent displacement in the rank stability calculation, pd. Figure 5.7 shows the effect of varying the number of replicates does not affect the results, indicating that the stability score estimation does not oscillate with few perturbation samples.

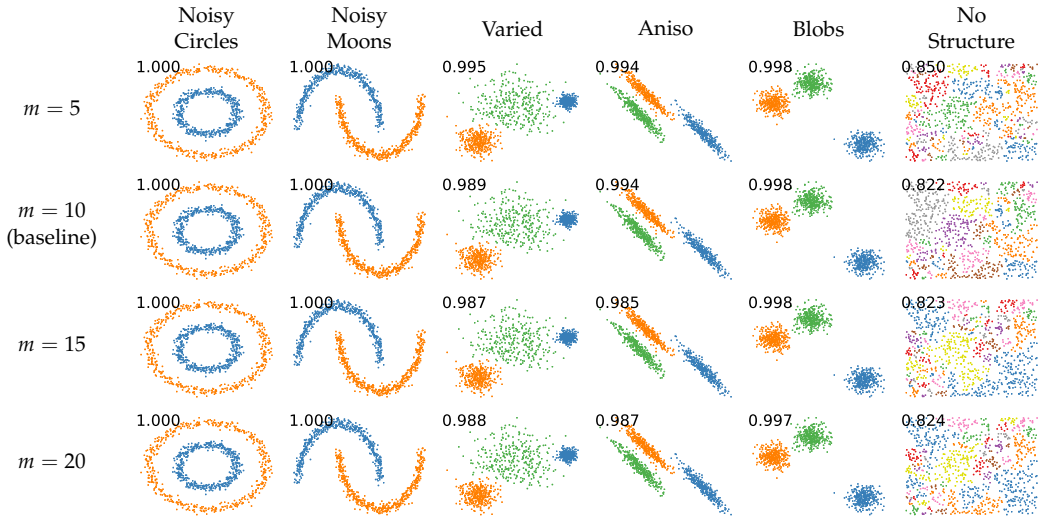


Figure 5.7: Effect of varying the number of replicates (m) on the approximate optimal stability clustering algorithm. Other parameters fixed at baseline: $k = 10$ neighbors, $pd = 0.5$ percent displacement, $hm =$ parents hierarchy mode. The stability score is shown in the top-left corner of each plot.

Figure 5.8 shows the effect of varying the percent displacement in the rank stability calculation, pd. Multiple datasets are affected by the percent displacement. The most notable effect is on the Blobs dataset, where the two top clusters are merged into a single cluster, while yielding the highest possible stability score of 1.0. These clusters are closer together than the bottom cluster, therefore, grouping them together yield a solution with as clear separation as the original one with three clusters. This is also reflected in the Aniso result for $pd = 0.2$'s. This is the case described in [182], where multiple solutions are stable, highlighting if a flat-clustering should really be preferred over a hierarchical one. The configuration $pd = 0.1$ also failed in correctly indentifying clusters in the Varied dataset.

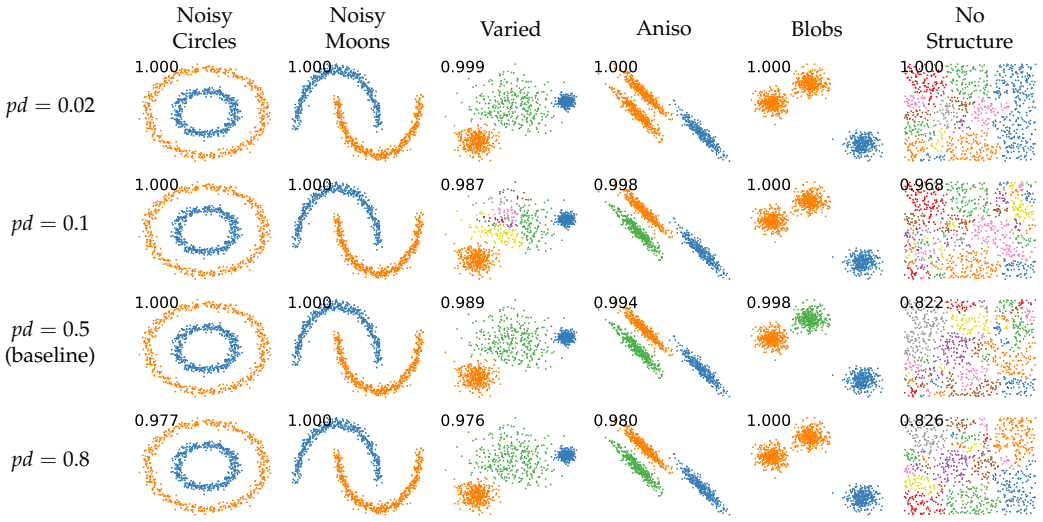


Figure 5.8: Effect of varying the percent displacement (pd) on the approximate optimal stability clustering algorithm. The percent displacement controls the maximum allowed displacement in the rank stability calculation. Other parameters fixed at baseline: $k = 10$ neighbors, $m = 10$ perturbed replicates, $hm =$ parents hierarchy mode. The stability score is shown in the top-left corner of each plot.

Now we study the effects of varying stability clustering inputs, by varying the graph construction by changing the number of neighbors, k and the algorithm used to construct the hierarchy, hm, which defines the possible partitions of our data.

Figure 5.9 shows the effect of varying the number of neighbors, k. There are minor changes in the results, but a trend is noticeable, fewer neighbors yield more partitions as shown in the Noisy Moons dataset for $k = 5$. Where, a greater number of neighbors, $k = 20$ and $k = 100$, results in fewer partitions. This is a consequence of how connectivity strength varies as the number of neighbors increases. With more neighbors, the connectivity strength is stronger, therefore, the our algorithm is less likely to partition the data into more clusters. The opposite effect is observed when the connection between components is weaker, where more clusters are identified.

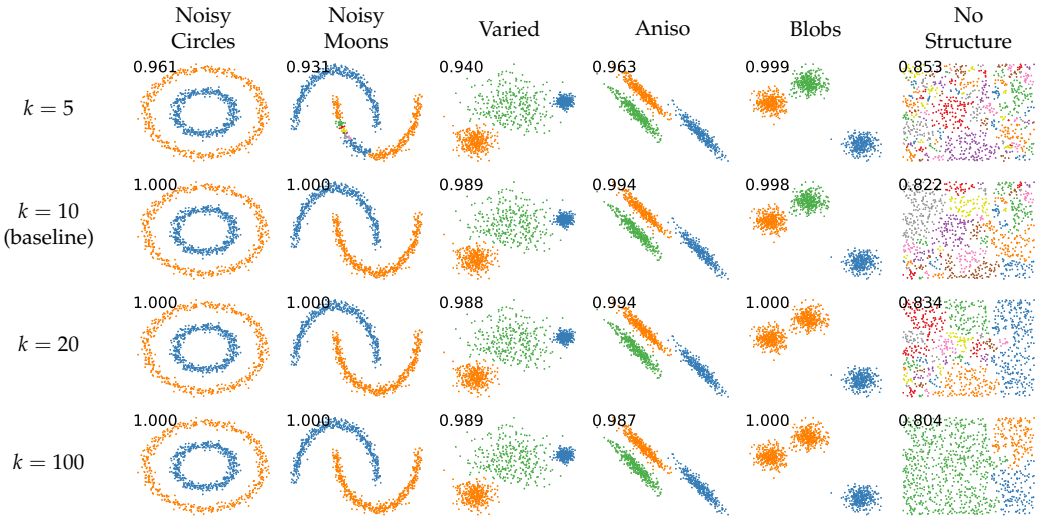


Figure 5.9: Effect of varying the number of neighbors (k) in the kNN graph construction on the approximate optimal stability clustering algorithm. Other parameters fixed at baseline: $m = 10$ perturbed replicates, $pd = 0.5$ percent displacement, $hm =$ parents hierarchy mode. The stability score is shown in the top-left corner of each plot.

Figure 5.10 shows the most critical consideration when using our approach. How the hierarchy is constructed. Afterall, the only possible set of partitions are the ones that exist in the reference hierarchy. The minimum-spanning tree (single-linkage), $hm = mst$, is heavily susceptible to noise, because a single-edge potentially noisy edge can merge two clusters, often, merging clusters prematurely[1]. While other approaches as a watershed hierarchy by area or number of non-leaf nodes, regularize this behavior, yielding better partitions and therefore better stability clustering results.

[1]Often referred as leakage in the context of image segmentation

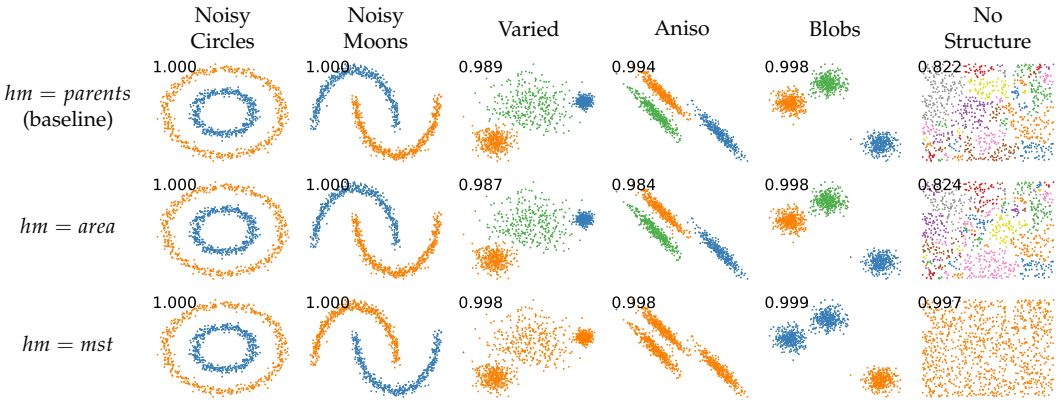


Figure 5.10: Effect of varying the hierarchy mode (*hm*) on the approximate optimal stability clustering algorithm. The hierarchy mode determines how the hierarchical clustering tree is constructed. Other parameters fixed at baseline: $k = 10$ neighbors, $m = 10$ perturbed replicates, $pd = 0.5$ percent displacement. The stability score is shown in the top-left corner of each plot.

5.5.3 Runtime Analysis

As described in Chapter 5.5.1, the exact and approximate algorithms results are equivalent for all datasets, only showing differences when there are no meaningful clusters and the partition is arbitrary.

Figure 5.11 displays the runtime comparison between the exact and approximate algorithms with the same experimental setup, except we varied the number of samples. The approximate algorithms is always at least 50 times faster, and this relationship holds as the number of samples increases. The exception is the No Structure dataset, where the speed-up is in more than 300 times, this is because the lack of meaningful structure makes the ILP much harder to solve, thus slowing down the exact algorithm, also an optimality gap of 0.1% was used when running the ILP because finding a globally optimal solution was taking considerably more time.

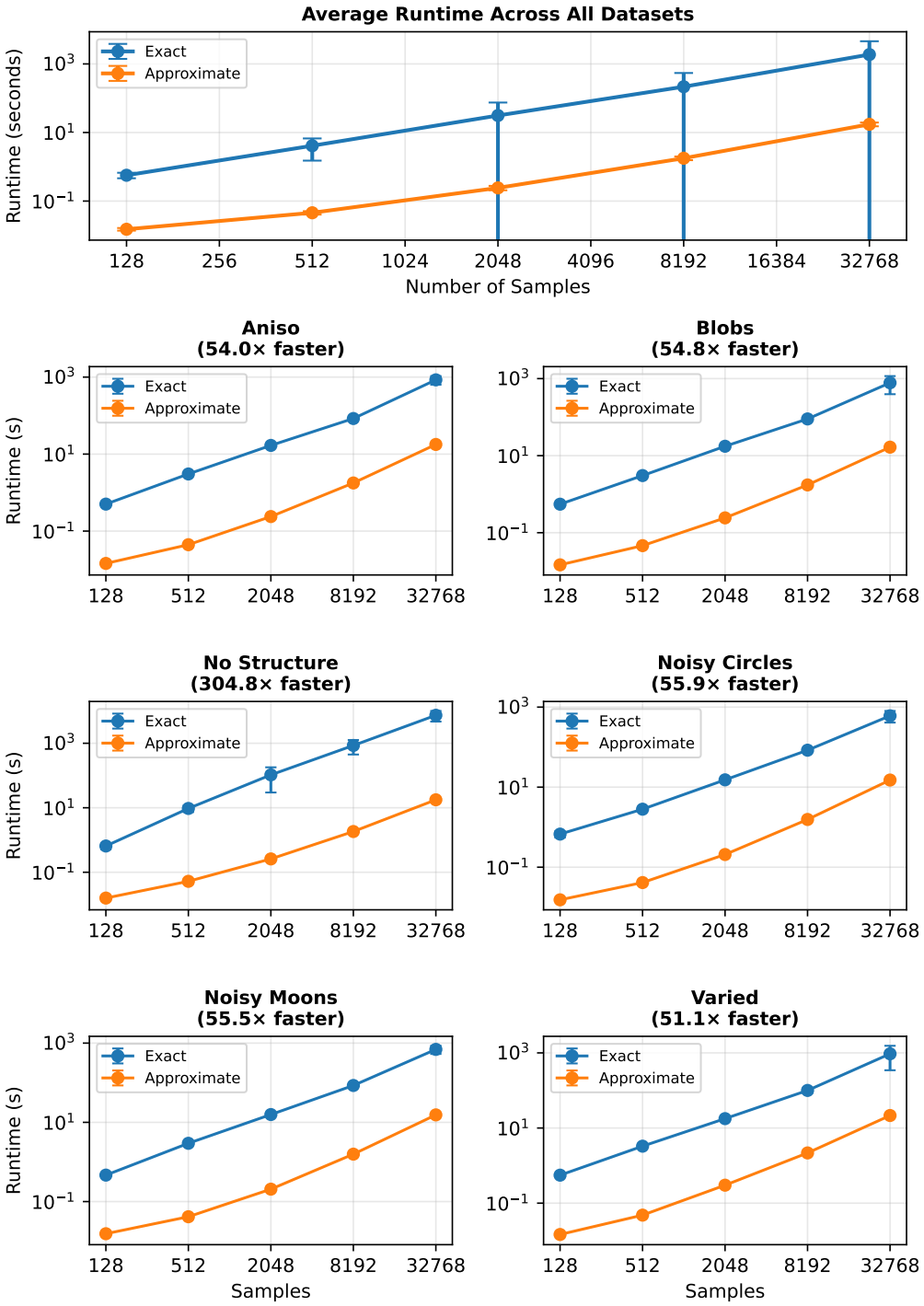


Figure 5.11: Exact and approximate clustering runtime analysis. Runtime comparison of the exact and approximate optimal stability clustering algorithms. Top and larger plot shows the average runtime, and smaller plots show the runtime for each dataset with five replicates each. See text for more details.

5.5.4 Evaluating Stability-Based Clustering on Pancreatic Development

To evaluate the performance of our method on real-world biological manifolds, we applied it to a single-cell RNA sequencing dataset capturing the development of mouse pancreatic endocrine cells across multiple embryonic stages [191]. Developmental datasets present a significant challenge for traditional clustering algorithms due to the presence of continuous transitions and branching manifolds. As cells differentiate and specialize, they traverse a phenotypic manifold where boundaries between states are often blurred or transient.

Pancreatic endocrinogenesis is a highly regulated process in which multipotent progenitor cells differentiate into specialized hormone-secreting cells within the Islets of Langerhans. The primary terminal states include: **Alpha** cells (secreting glucagon), **Beta** cells (secreting insulin), **Delta** cells (secreting somatostatin), and **Epsilon** cells (secreting ghrelin). This process can be modeled as a continuous transition of cellular states using RNA velocity, which infers the directionality of differentiation based on the ratio of unspliced to spliced mRNA [192]. Our reproduction of this RNA velocity trajectory is shown in Figure 5.12. The results delineate a clear trajectory starting from ductal progenitors ($Ngn3^{low}$), through an endocrine-primed state ($Ngn3^{high}$), and finally bifurcating into the differentiated islet populations.

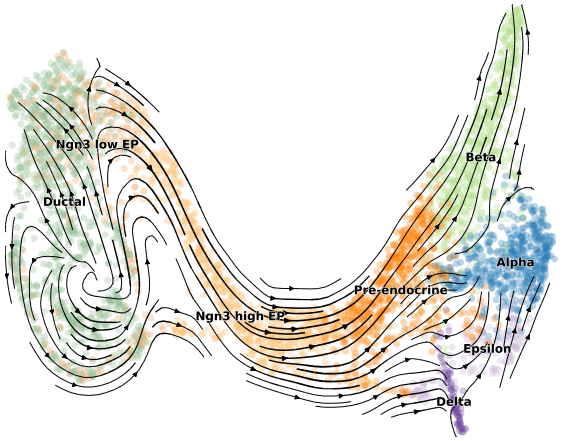


Figure 5.12: RNA velocity stream of pancreatic endocrine cells. UMAP projection illustrating the developmental manifold. Arrows indicate inferred transition probabilities between cell states, transitioning from progenitors (left) to differentiated islet cells (right). Annotations derived from [192].

We compared our stability-based clustering method against the Leiden algorithm, which was originally used to assign the cell types with manual curation, we applied it across a range of resolutions – its main parameter, Figure 5.13. Our method identified five stable clusters, providing a solution that captures more granular detail than Leiden at resolution 0.2 but remains more robust than the ten clusters observed at resolution 0.5. It's worth to highlight that Leiden clustering is non-hierarchical, increasing the resolution does not necessarily refine existing clusters but can lead to merges of clusters before the re-partitioning of the manifold.

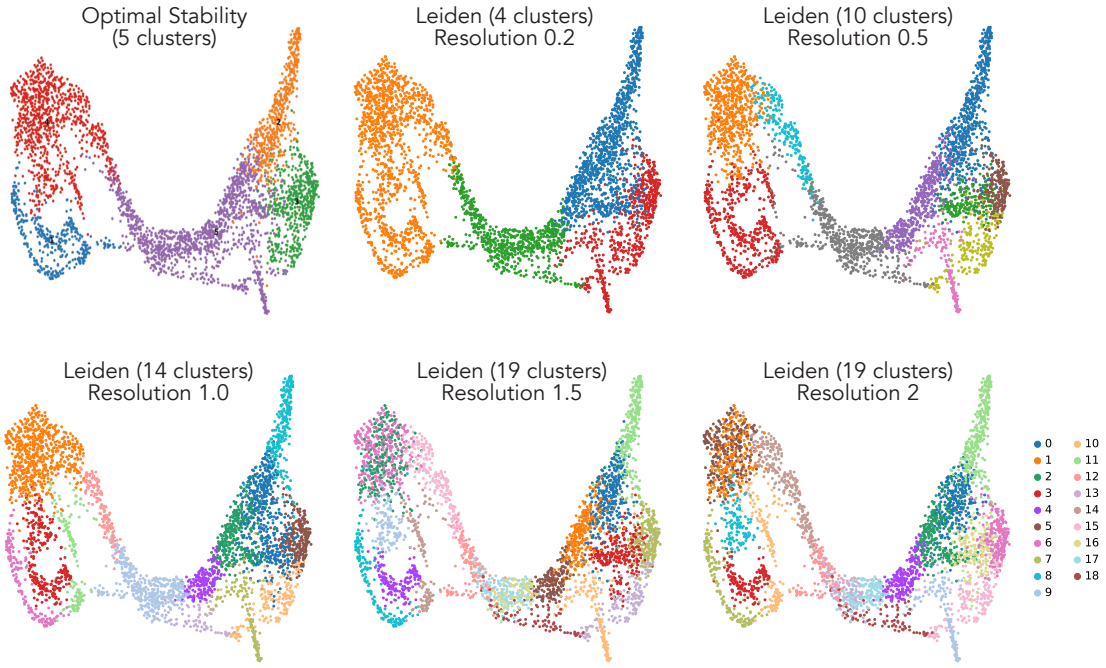


Figure 5.13: Comparison of Stability-based and Leiden Clustering. UMAP projections comparing our optimal stability solution (5 clusters) against multiple Leiden resolutions.

Further investigation into the stability hierarchy, Figure 5.14, revealed other nodes with high individual stability scores (> 0.70). Figure 5.14a illustrates the simplified lineage, where node size corresponds to cell count and color represents stability. Figure 5.14b maps these high-stability nodes back to the UMAP and assigning a simplified subtree rooted at each optimal stability cluster.

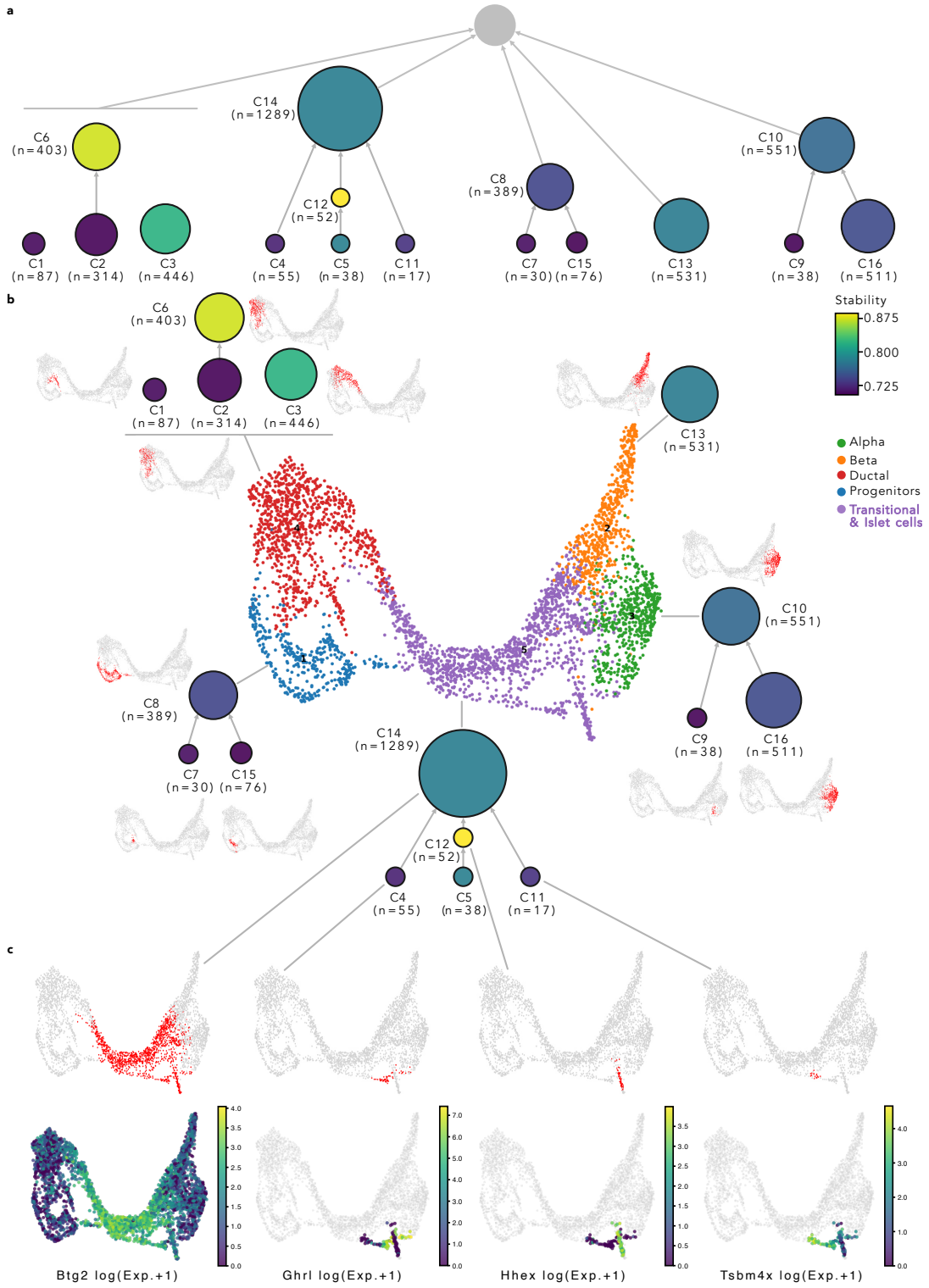


Figure 5.14: Lineage and Marker Gene Analysis of Pancreatic Development. a, Simplified stability hierarchy. **b,** UMAP and the assignment of individual nodes of the hierarchy. **c,** Gene expression analysis of subtree rooted at C14. See text for further discussion.

In Figure 5.14c, we performed a targeted analysis of one of the subtrees rooted at the C14 cluster (violet-highlighted cluster). This region is of particular interest due to its transitional nature. The stability score filtering identified the terminal differentiated states as high-density "sinks," while removing the continuous transitions present in the coarser resolved clustering. Differential expression analysis identified this pool through markers such as *Btg2* (associated with mitotic exit and maturation) and *Tmsb4x* (associated with cytoskeletal remodeling during migration). We also observed strong expression of lineage-defining genes *Ghrl* and *Hhex*, marking the Epsilon and Delta lineages.

Interestingly, our method identified a specific sub-cluster (C11) characterized by high *Tmsb4x* expression and a signature of ribosomal proteins (*Rps*9, *Rpl*32, *Rps*19, *Rps*4*x*) involved in active protein biogenesis. This population was not emphasized in original descriptions of this dataset, but aligns with recent findings from the *moscot* framework [193]. Where Klein et al. utilized paired scRNA-seq and chromatin accessibility (ATAC-seq) to identify an "Active Endocrine Progenitor" pool regulated by *Neurod2*.

The high ribosomal expression we observed corresponds to the translational protein synthesis demand required for the transition from a proliferative progenitor to a specialized endocrine cell. While the *moscot* study relied on multimodal data to resolve these relationships, our stability-based algorithm successfully isolated this transitional state using transcriptomics alone. This demonstrates the potential of stability-based clustering to identify persistent cell states within challenging transient and continuous development manifolds.

Conclusion

In this chapter, we presented a novel framework for stability-based hierarchical clustering that generalizes the principles of temporal consistency from cell tracking to arbitrary data partitioning. By reframing the selection of clusters as an optimization problem over a combinatorial set of non-horizontal cuts, our method effectively identifies stable cellular states without the need for manual parameter tuning or restrictive assumptions regarding cluster geometry and variance.

The development of a polynomial-time dynamic programming approximation addressed the scalability limitations of the initial Integer Linear Programming (ILP) formulation. Our runtime analysis demonstrated that this approximate algorithm achieves solutions equivalent to the exact formulation while providing at least a 50-fold speedup, enabling the analysis of large-scale datasets containing tens of thousands of samples.

Through evaluation on synthetic datasets, we showed that stability-based optimization is robust across diverse geometric structures and varying density patterns where traditional spherical or density-based methods frequently fail. In the context of real-world biological manifolds, the application to pancreatic endocrinogenesis demonstrated the algorithm's ability to resolve complex differentiation terminal states. Notably, our method identified a distinct "Active Endocrine Progenitor" pool characterized by high expression of *Tmsb4x* and ribosomal proteins. This finding is significant as it aligns with recent state-of-the-art results from the *moscot* framework, which required multimodal RNA and ATAC sequencing to resolve the same population.

The success of this approach in isolating biologically relevant cell states from continuous manifolds using transcriptomics underscores its efficiency and potential as a hypothesis-generation tool for biological research.

It is important to acknowledge the theoretical limitations identified by Ben-David et al. [182, 183], stability does not guarantee uniqueness of clustering solutions. While our method identifies highly stable partitions that align with known biological annotations, alternative stable existed and were observed to be important when analyzing the subtrees of the hierarchy. Nonetheless, the stability score itself provides a measure of partition robustness, and in practice, the terminal cell types showed consistently high stability across perturbations, while transitional states were more ambiguous, a reflection of the underlying biological reality rather than a methodological limitation.

This observation suggests an important future direction, rather than seeking a single "optimal" flat clustering, methods for hierarchy simplification that preserve high-stability nodes while explicitly representing regions of ambiguity may capture more closely the true structure of biological data. Such approaches would acknowledge that some biological states are discrete and stable in different degrees, while others exist along continuous

manifolds, providing a more nuanced representation than forcing all data into a flat partition.

Chapter 6

Conclusion and Outlook

This thesis was motivated by the necessity for improved methods in terabyte-scale cell tracking to enable the investigation of dynamic patterns in large-scale live microscopy samples. In investigating existing approaches, we posed two fundamental questions: (i) *Is association or segmentation the bottleneck of cell tracking?* and (ii) *Can temporal information across frames be leveraged to inform the optimal selection of spatial segments from a hierarchy?*

Our initial hypothesis was that segmentation represented the primary bottleneck and that temporal information could provide the necessary constraints to resolve it.

In answering these questions, we developed Ultrack, Chapter 3, a new joint cell segmentation and tracking method that offers a robust, scalable solution for analyzing complex cellular dynamics across various scales and dimensions. Through a series of diverse experiments, Chapter 4, we demonstrated that Ultrack is capable of tracking cells in a wide range of scenarios, from sparse annotations to dense, multi-channel data, regardless of whether deep-learning segmentation models are utilized as input while not neglecting the possibility of using them to boost performance.

Additionally, we contributed to the assessment of cell tracking methods by creating the largest tracking dataset to date, Section 4.8, encompassing over five terabytes of imaging data. This was achieved through a novel sparse labeling approach that significantly simplifies the curation of tracking results.

Based on our findings, we can conclude that *segmentation is the primary bottleneck* up to a certain quality threshold. This is particularly evident in Table 4.1, where in the

3D whole Fly and Beetle embryo datasets, where segmentation is notably challenging, Ultrack achieved the largest performance margins over competing methods (0.708 vs. 0.617 [1] for the Fly and 0.841 vs. 0.804 for the Beetle). These results were achieved without a deep-learning segmentation model due to limited training data, highlighting that our specialized joint optimization algorithm was fundamental to this success. This provides an affirmative answer to our second research question, *temporal information can indeed improve segmentation performance*. The tracking of membrane-labeled cells in Drosophila (Section 4.7) further supports this conclusion, as our method performed on par with or better than state-of-the-art deep-learning and graph-based segmentation approaches.

Broadening the scope, we observed that the proposed tracking methodology functions as a non-horizontal cut within any hierarchical clustering. Therefore, we extended this approach to the general problem of clustering arbitrary data by translating the temporal consistency problem into a stability clustering problem, Chapter 5. We proposed an approximate dynamic programming algorithm capable of scaling to tens of thousands of samples. The algorithm perturbs the data to create different views and identifying the most stable disjoint clusters across all variants, this method could be further extended to multi-view clustering, a field of growing importance as multi-omics and multi-modal imaging become more prevalent in the biological sciences.

Looking forward, while Ultrack achieves high performance, the problem of cell tracking is not fully resolved. The performance ceiling of Ultrack is inherently less flexible than that of end-to-end deep learning approaches, which can be fed increasing amounts of data for training. Although such massive 3D annotated datasets are currently unavailable, the field is rapidly moving toward higher scales of data and compute. In this evolving landscape, methods like Ultrack provide essential initial solutions that can produce the high-quality lineages for manual curation and refinement, to be used for training of the next generation of deep learning models for joint segmentation and tracking [107, 194].

[1] CTB score, the higher the better

A

Appendix

A.1 Videos

The items below and their respective QR codes redirect to the videos resulting from this thesis.

1. Ultrack intuition and overview;

2. Using multiple segmentation hypotheses with temporal consistency helps reduce the challenges of parameter tuning;

3. Improved cell tracking in label-free imaging by leveraging virtual staining;

4. Enhanced multi-color cell tracking by integrating segmentation results from multiple algorithms;

5. Enhancing tracking accuracy through temporal registration;

6. Visualizing Ultrack's Cell Tracking Challenge benchmark results;

7. Sparse fluorescence labeling enables high-fidelity tracking validation over extended time-lapses;

8. Multi-terabyte cell tracking of zebrafish embryo;

9. Near-perfect nuclear- and membrane-based 3D tracking of zebrafish neuromast cells;

10. Ultrack's guided user interfaces across multiple platforms facilitate cell tracking, catering to diverse user needs and computational environments.

11. Epithelial cell tracking benchmark Peripodial cells results;

12. Epithelial cell tracking benchmark Proper discs cells results;

A.2 Cell Tracking Challenge Submission Details

This section describes the experimental details of our Cell Tracking Challenge submission, Section 4.6.

All datasets were linearly interpolated on the z-axis to make them isotropic. While isotropy is not necessary, it benefits the algorithm's accuracy. They were normalized by their lower and upper quantiles (lq and uq in Table A.1), the DRO dataset had corrupted voxels which were set to 0.

In the datasets MDA231 and CE, where 3D segmentation labels are available, we trained a U-Net to predict the foreground and the cell boundaries, see Section 3.3. In the DRO, TRIF and TRIC datasets, where 3D labels are not available, we tried two different approaches:

- IM: where the cells and their contours were estimated using traditional image processing operators;
- PL: where we trained the U-Nets on labels generated by an algorithm and not manually annotated.

Both PL and IM processing started the same way. The cells were detected using a difference of Gaussians with σ_1 and σ_2 thresholding at 0.5, 0.5, and 0.75 times their Otsu

for DRO, TRIF and TRIC, respectively. The cell contour maps were simply the inverse of the Gaussian blurred image with σ_1 and normalized to be between 0 and 1. Through visual inspection, we selected $\sigma_1 = 2$ and $\sigma_2 = 8$ for the DRO dataset, $\sigma_1 = (0, 1, 1)$ and $\sigma_2 = (1, 4, 4)$ for the TRIF dataset ((z, y, x) order), and $\sigma_1 = 1$ and $\sigma_2 = 6$ for the TRIC dataset. For the pseudo-labels, we used these contour maps and detections to run the hierarchical watershed [132] and performed a horizontal cut in the hierarchy to obtain the flat segments used for training the network. In IM, they were used as the tracking input directly.

For the training set, the normalized time-lapses were used as input to fit the network weights and find the optimal hyper-parameters using grid search and cross-validation between videos. Such that, we trained a network in a single time-lapse and computed the results on the other. For example, we trained on time-lapse 01 and predicted the network outputs, tracked, and measured the accuracy on time-lapse 02 for each dataset.

The U-Nets were trained for 20 epochs using a learning rate of 10^{-4} with exponential weight decay of 0.95 at every epoch. The networks used a kernel size of 5, with simple convolution blocks without residual connections or batch normalization. The number of planes used was 32, 64, 128, and 256. The network features were linearly interpolated in the up-sampling step of the U-Net decoder. The best parameters found through the grid search are reported in Table A.1.

The images were processed in overlapping tiles to avoid boundary artifacts of the CNNs' inference. Once a whole frame was computed, the foreground detection channels were binarized with a *Threshold* parameter, and the contours were Gaussian filtered with σ, values at Table A.1. The PL and IM approaches shared the same parameters.

The DRO and TRIC datasets, where cells move coherently during embryo development, had their hierarchies registered with a local movement estimation between adjacent frames, as in Section 4.5. The local movement is pre-computed as a low-resolution time-lapse and then applied to translate each candidate segment, influencing their location when accessing their neighbors in adjacent frames and their IoU. The TRIF datasets were registered with a global translation between adjacent frames because some frames are shifted, probably due to acquisition issues.

For the final submission, both training datasets were used for learning the weights, and the segmentation and tracking were done using the hyper-parameters found on the training sets as described above.

	normalization		CNN output		hierarchy construction				tracking			
Dataset	lq	uq	σ	Threshold	Min. Area	Max. Area	Contour Strength	Noise	Link Radius	w_α	w_β	w_δ
MDA231	0.001	0.9999	0.5	0.5	2500	25000	0.2	0	75	-0.01	-0.01	-0.001
CE	0.001	0.9999	1	0.1	5000	1000000	0	0.4	100	-0.005	-0.005	0
DRO	0.001	0.9999	1	0.5	1500	50000	0.1	0.1	50	-0.001	-0.001	-0.001
TRIF	0.5	0.99999	$1^\dagger$	0.5	1000	15000	0	0	50	-0.001	-0.001	0
TRIC	0.05	0.999	2	0.5	250	15000	0	0	25	-0.01	-0.001	-0.001

Table A.1: Table of parameters used for our Cell Tracking Challenge submission. † Gaussian filter applied on yx axes only.

Table A.2 reports our cross-validation results on the training set from Section 4.6.

The image processing solution showed superior results to the pseudo-label alternatives in the DRO, TRIF, and TRIC datasets. Hence, it was the regime used in the final submission. Notably, the gap in evaluation metrics is the largest in the extremely challenging DRO dataset, where the leaderboard scores are smaller than other datasets. This is one of the datasets in which we achieved the top score in the hidden test set, Table 4.1.

Dataset	Regime	File	TRA	SEG	CTB
MDA231	Sup	1	0.888	0.613	0.751
		2	0.923	0.626	0.774
		AVG	0.905	0.620	0.763
CE	Sup	1	0.970	0.705	0.838
		2	0.955	0.649	0.802
		AVG	0.963	0.677	0.820
DRO	PL	1	0.693	0.425	0.559
		2	0.923	0.528	0.726
		AVG	0.808	0.476	0.642
	IM	1	0.812	0.512	0.662
		2	0.934	0.556	0.745
		AVG	0.873	0.534	0.704
TRIF	PL	1	0.813	0.607	0.710
		2	0.811	0.623	0.717
		AVG	0.812	0.615	0.714
	IM	1	0.870	0.687	0.779
		2	0.918	0.679	0.799
		AVG	0.894	0.683	0.789
TRIC	PL	1	0.946	0.552	0.749
		2	0.880	0.690	0.785
		AVG	0.913	0.621	0.767
	IM	1	0.980	0.527	0.754
		2	0.853	0.733	0.793
		AVG	0.917	0.630	0.773

Table A.2: Results on the training set of the Cell Tracking Challenge; Fully supervised (Sup) and pseudo-labels (PL) regimes used cross-validation; Image processing (IM) does not require training and achieved better results, especially on the DRO dataset.

A.3 Synthetic Clustering Experiments Parameters

This section details the parameter configurations used for the clustering algorithm comparison presented in Section 5.5.1. All datasets contained 1000 samples and were normalized using `StandardScaler` from scikit-learn prior to clustering. The parameters were kept fixed across all datasets to simulate realistic scenarios where manual per-dataset tuning is impractical, with the exception of algorithms requiring a predefined number of clusters.

A.3.1 Common parameters

The following parameters were shared across multiple algorithms or used for preprocessing:

- **Number of neighbors** (k): For graph-based methods (Leiden, Ward, Single-linkage, Average-linkage), a k-nearest neighbor connectivity matrix with $k = 10$ was constructed and made symmetric via element-wise maximum, $\max(C, C^T)$.

A.3.2 Algorithm-specific parameters

- **K-means.**
  - Number of clusters: 3 (2 for Noisy Circles and Noisy Moons datasets)
- **Gaussian Mixture Model (GMM).**
  - Number of components: 3 (2 for Noisy Circles and Noisy Moons datasets)
  - Full covariance matrix
- **MeanShift.**
  - Bandwidth: estimated using `estimate_bandwidth` with quantile=0.3
- **Spectral Clustering.**
  - Number of clusters: 3 (2 for Noisy Circles and Noisy Moons datasets)
  - Constant affinity
- **Leiden.**
  - Resolution: 0.01
  - Modularity objective function
  - Constant affinity

- **Ward Hierarchical Clustering.**
  - Number of clusters: 3 (2 for Noisy Circles and Noisy Moons datasets)
  - Linkage: ward
- **Single-linkage Hierarchical Clustering.**
  - Number of clusters: 3 (2 for Noisy Circles and Noisy Moons datasets)
  - Linkage: single
- **Average-linkage Hierarchical Clustering.**
  - Number of clusters: 3 (2 for Noisy Circles and Noisy Moons datasets)
  - Linkage: average
- **DBSCAN.**
  - Epsilon (ε): 0.18
  - Minimum samples: default (5)
- **HDBSCAN.**
  - Minimum samples: 3
  - Minimum cluster size: 15
- **Optimal Stability Clustering (OSC) — Exact and Approximate.**
  - Number of replicates: 10
  - Maximum displacement percentage: 0.5
  - Hierarchy: watershed hierarchy ordered by the number of non-leaf nodes (*i.e.* parents [130])

A.4 Ultrack's User Interfaces

Cell tracking software often represents one of the most complex components in image analysis pipelines, primarily due to the necessity of handling the time dimension — a factor frequently overlooked in segmentation and other image processing operations. Moreover, it operates on non-matrix data (*i.e.* track lineages), where efficient data processing routines are less prevalent. This complexity has historically limited accessibility to a broad user base, with only a few longstanding solutions like TrackMate [151] and others [22, 42, 43], alongside specialized solutions for handling larger datasets [45]. The development of the interfaces were led by Ilan Theodoro.

Ultrack addresses these challenges by offering a versatile, user-friendly solution that caters to a wide range of users, from biologists with minimal programming experience to machine learning experts requiring customizable inputs. By providing multiple interfaces and scalable processing capabilities, Ultrack democratizes access to advanced cell tracking technology, enabling efficient analysis of datasets of various sizes across different computational resources, as shown in Figure 4.13 in Section 4.9.

Ultrack offers two options for users who prefer a graphical interface: napari and Fiji.

The *napari* plugin, Figure A.1a, integrates transparently with the Python ecosystem [195], providing a comprehensive set of features for non-coding users. Napari is a Python library for n-dimensional image visualization, annotation, and analysis with a GUI built with Qt [196], which is extensible with Qt primitives in Python. Our custom plugin scales with large datasets and offers graphics-hardware-accelerated preprocessing operations (including Gaussian filtering, difference of Gaussians, and vector-field flow computation) that can benefit from GPU acceleration when available. All Ultrack parameters can be tuned through the GUI interface. The processing runs in the background in a separate thread, enabling real-time visualization of intermediate results using the native Napari viewer. Upon completion, results can be exported to multiple formats, including TrackMate [151], NetworkX graph [197], and Zarr array.

The Fiji plugin [198], Figure A.1b,c, exposes Ultrack's algorithm to Fiji's mature ecosystem and broad user base, allowing seamless integration with existing Fiji workflows and ensuring compatibility with popular tools like TrackMate [151] and Mastodon [199]. Since Fiji is a Java-based software that can be extended through Java plugins, we developed a RESTful API [200] using the FastAPI framework [201] to facilitate communication between Java and Python. This API employs websockets [202] for persistent communication, enabling real-time event logging and output streaming from Ultrack to Fiji. The graphical user interface, implemented in Java using the Bootstrap 5 [203] front-end framework, serves as a Fiji plugin that leverages the API provided with the Ultrack Python package. To promote broader interoperability, Ultrack adheres to API specifications detailed in our documentation (`https://royerlab.github.io/ultrack/rest_api.html`), allowing integration with various programming languages beyond Java and Python.

For users comfortable with programming, Ultrack provides a user-friendly Python API (Figure A.1d) that exposes all of its functionalities, allowing for easy customization and reproducible workflows. This API accepts various array formats (numpy, zarr, dask, etc.) and can export data into multiple file formats, including networkx [197], tables (*i.e.*, data frames) [204], Cell Tracking Challenge [30], and TrackMate [151] formats, enhancing its interoperability with other tools and workflows. Additionally, Ultrack offers a command line interface (CLI, see Figure A.1e) designed for batch processing and streamlined workflows, which supports distributed processing in high-performance computing environments. This CLI accommodates any file format supported by existing *napari* reader plugins, eliminating the need for data conversion — a crucial feature for handling large datasets.

A small tour of the user interfaces is shown in Video 10 .

A.5 Additional Softwares

This work would not have been possible without the open-source community. An non exhaustive list of the software used in this work is listed below: `Blosc`, `CuPy`, `FastAPI`, `imagecodecs`, `iohub`, `inTRACKtive`, `httpx`, `ggplot2`, `napari`, `networkx`, `Numba`, `NumPy`, `Pandas`, `PyTorch`, `scikit-image`, `SciPy`, `SLURM`, `tifffile`, `traccuracy`, `uvicorn`, `websockets`, and `zarr` [205, 206, 4, 201, 207, 204, 141, 139, 208, 209, 195, 197, 169].

A big part of my time was spent developing also our own softwares during this work and making them publicly available for the community, the most important ones are described below.

A.5.1 tracksdata

`tracksdata`[1] is a unified Python API for multi-object tracking. Each tracking algorithm typically implements its own custom data structures to represent tracking data (*i.e.* graphs), which complicates the integration and comparison of different approaches. `tracksdata` provides a common interface to address this limitation.

[1] `https://github.com/royerlab/tracksdata`

The package generalizes the multi-processing and out-of-core processing capabilities developed for Ultrack into a tracking-agnostic framework. It models tracking as a directed graph where nodes represent detections at specific time points and edges connect objects across consecutive frames. The package provides three graph backend implementations: RustworkX [210] for efficient in-memory operations, SQL-based backends for larger-than-memory datasets, and GraphView for lightweight subgraph operations that maintain connections to the original graph, also called the root graph. All backends are accessible through the same API.

Tracking data comprises multiple interconnected representations: segmentation masks defining cell shapes, centroids indicating positions, and lineage graphs (*i.e.* tracks) connecting objects across time. Manual curation of tracking results requires maintaining consistency across these representations, as changes to one must propagate correctly to the others to prevent information loss or inconsistencies.

`tracksdata` addresses this through a `BaseGraph` abstraction that serves as a single source of truth. Each representation can be lazily rendered on demand. For instance, converting a `BaseGraph` instance to a numpy array when indexing a subset of the graph. This approach eliminates manual synchronization by ensuring all representations derive from the same underlying data structure. The package uses an R-tree spatial index [2] for spatial queries, enabling efficient bounding box operations and graph slicing.

This architecture is currently being used to implement `TrackEdit`[3] by Teun Huijben, our curation tool built on napari [195] and `motile` [4], which uses `tracksdata`'s unified representation to provide manual correction and quality assessment of tracking results.

The package employs a modular operator-based design with three categories: node operators to create objects and define their attributes, edge operators to establish connections between objects and compute edge attributes, and solver operators to compute optimal tracking solutions. Built-in solvers include nearest-neighbor tracking and ILP-based optimization. To support different tracking methodologies, `tracksdata` can parse mathematical expressions for edge and node attributes, composing them dynamically at the ILP construction stage. Users can also incorporate manual annotations by setting

[2] `https://github.com/funkelab/spatial_graph`
[3] `https://github.com/royerlab/trackedit`
[4] `https://github.com/funkelab/motile`

specific nodes or edges to ∞ or $-\infty$ to forbid or enforce their inclusion in the solution (*e.g.* from user annotations). The package maintains compatibility with the Cell Tracking Challenge format and integrates with cell tracking evaluation metrics.

The Ultrack algorithm was reimplemented as node operator in C++ to be compatible with the `tracksdata` framework [5].

Below is a basic example of how to use `tracksdata` to solve the Cell Tracking Challenge (CTC) problem.

[5] `https://github.com/royerlab/ultrack-td`

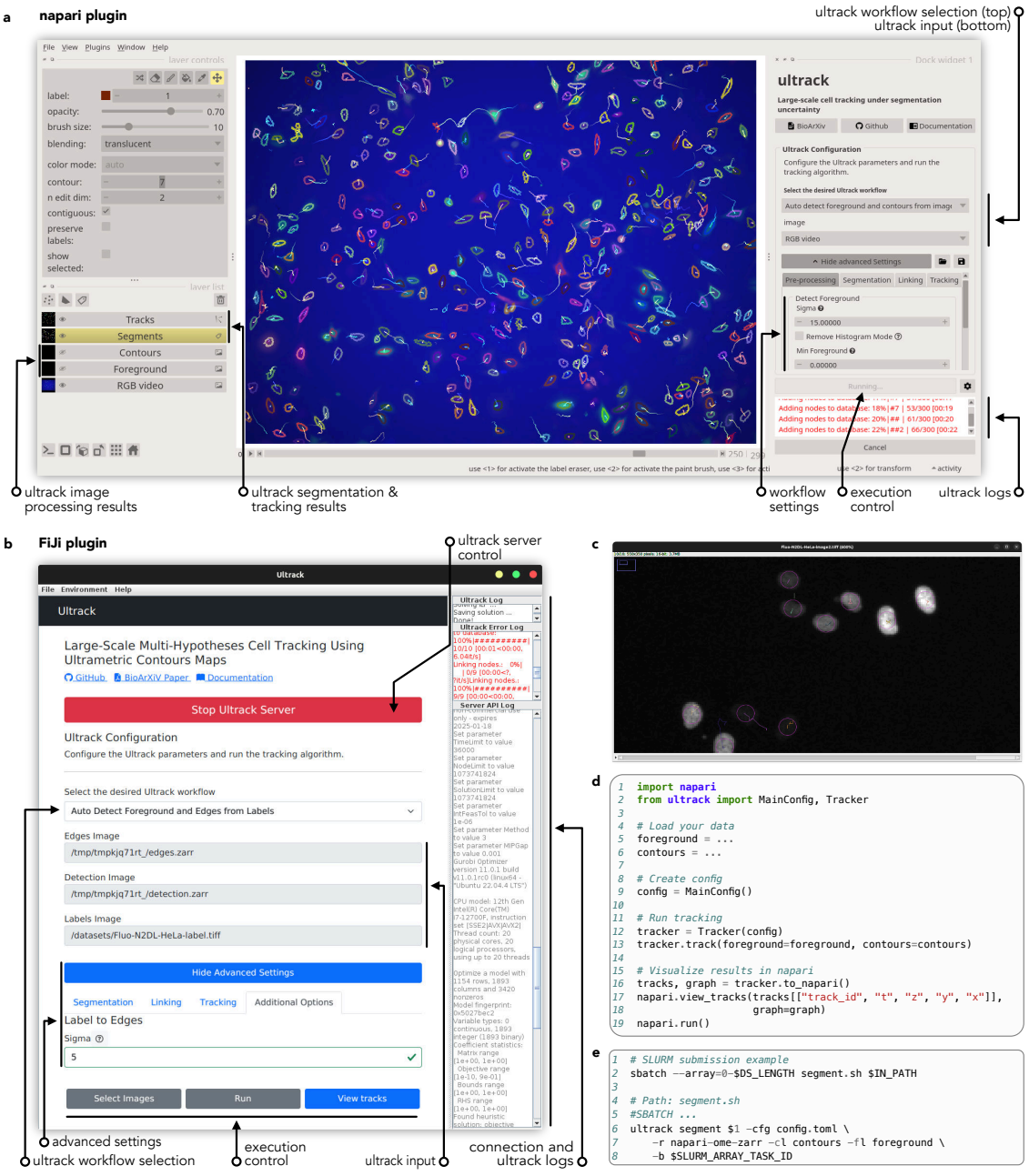


Figure A.1: Ultrack's user interfaces. **a**, Napari plugin interface showcasing multi-color automatic cell tracking results. The interface includes workflow selection, input configuration, image processing results, segmentation and tracking visualization, workflow settings, and execution control. **b**, Fiji plugin interface guiding users through workflow selection, input specification, and advanced settings. It also provides execution control and displays Ultrack server logs. **c**, Example of image data overlaid with cell detections and tracks as viewed in the Fiji interface. **d**, A concise Python API example (18 lines) demonstrating data loading, configuration, tracking execution, and result visualization in *napari*. **e**, Sample SLURM submission script for Ultrack's CLI, illustrating its integration with high-performance computing environments for distributed processing.

```
# Load example data from Cell Tracking Challenge format
data_dir = Path(os.environ["CTC_DIR"]) / "training/Fluo-N2DL-HeLa/01_GT/TRA"
assert data_dir.exists(), f"Data directory {data_dir} does not exist."
# Load all timepoints as a 3D array: (time, height, width)
labels = np.stack(
    [imread(p) for p in sorted(data_dir.glob("*.tif"))],
)
graph = td.graph.InMemoryGraph() # Initialize empty graph
# Extract object features using region properties
# This creates one node per object per timeframe
td.nodes.RegionPropsNodes().add_nodes(graph, labels=labels)
# Add distance-based edges between objects in consecutive timeframes
# Only connects objects within distance_threshold and limits to n_neighbors
td.edges.DistanceEdges(distance_threshold=30.0, n_neighbors=5).add_edges(graph)
# Compute IoU between connected objects to measure shape similarity
iou_operator = td.edges.IoUEdgeAttr(output_key="iou").add_edge_attrs(graph)
# Create edge weights combining distance and IoU information
dist_weight = 1 / dist_operator.distance_threshold
# Edge weights aredefined as - IoU(e_ij) * exp(-distance(e_ij) /
↪  dist_threshold)
solution = td.solvers.ILPSolver(
    edge_weight=-td.EdgeAttr("iou") * (td.EdgeAttr("weight") *
    ↪  dist_weight).exp(),
    node_weight=0.0,
    appearance_weight=1.0,
    disappearance_weight=1.0,
    division_weight=1.0,
).solve()
# Create napari-compatible views of the tracking results
tracks_df, track_graph, track_labels = td.functional.to_napari_format(solution,
↪  labels.shape, mask_key="mask")
viewer = napari.Viewer()
viewer.add_labels(track_labels, name="Labels")
viewer.add_tracks(tracks_df, graph=track_graph, name="Tracks")
napari.run()
```

A.5.2 Image Processing Pipelines

The large-scale nature of the datasets processed in this work required robust and scalable computational infrastructure. To address this, we utilized and developed several open-source software libraries for high-performance image processing:

- `dexp`[6]: A specialized library for light-sheet Dataset EXploration and Processing started by Loïc Royer. It provides a comprehensive suite of GPU-accelerated functions for light-sheet microscopy, including equalization, denoising, dehazing, registration, stabilization, deskewing, deconvolution, and multi-view fusion. Built on CuPy, Dask, and Zarr, it features both a functional API and a command-line interface (CLI) to pipeline multi-terabyte processing jobs from raw data to rendered video.
- `dexpv2`[7]: A simplified iteration of the original `dexp` library that focuses on modular image processing components for massive datasets. It utilizes the Array API standard to ensure high-performance execution across both CPU and GPU environments when `cupy` and `cucim` are available. This library is the backbone for our distributed HPC pipelines.
- `slurmkit`[8]: A minimal Python toolkit designed to streamline the deployment of Python-based workflows on clusters managed by the SLURM workload manager. It provides a simple API for converting Python functions into distributed SLURM jobs with dependency management, facilitating the orchestration of the computationally intensive imaging pipelines required for this thesis.

Together, these repositories formed the computational backbone for the automated analysis of the whole embryo time-lapses presented in this thesis.

A.5.3 In-Silico Fate Mapping

The initial application of Ultrack was the Zebrahub project [3], which investigated zebrafish embryonic development through the integration of high-resolution imaging and

[6] `https://github.com/royerlab/dexp`
[7] `https://github.com/royerlab/dexpv2`
[8] `https://github.com/royerlab/slurmkit`

single-cell RNA sequencing [3]. Prior to the advent of accurate, large-scale lineage tracing, a standard approach for investigating cell fates and migration patterns involved staining cells at early developmental stages and imaging them subsequently to determine their final spatial distribution.

To modernize this workflow, we developed the `in-silico-fate-mapping` application. This tool performs in-silico fate mapping by leveraging tracking results even in the absence of complete lineage information, thereby reducing the necessity for exhaustive and repeated in-vivo experiments. The software allows users to select specific regions of interest and estimate their spatial coordinates across different time points, enabling the reconstruction of developmental trajectories.

The underlying method constructs radial regression models at each time point to estimate cell trajectories. Users define regions of interest via the napari Points layer interface; the algorithm then propagates these coordinates forward or backward in time based on local spatial correlations derived from the tracking data. The plugin accommodates tracking data in both napari Track layers and CSV formats (containing `track_id, t, z, y, x` coordinates). Key parameters for the radial regression model include the spatial radius for local regression, the number of samples for prediction, and smoothing factors, as illustrated in Figure A.2.

This software was used by Merlin Lange to analyze zebrafish embryo datasets from Zebrahub [3] for predicting cell fates computationally before performing in-vivo validation experiments, Figure A.3. **a**, Early in-silico photoconversion at 11 hours post-fertilization (hpf) showing the marked neuromesodermal progenitor (NMP) territory and tracked cell distributions at 11 and 19 hpf, with cyan indicating cells tracked from the marked region. The boundary between presumptive mesoderm and neural tissue is shown with a dashed line. **b**, Quantification of fate distribution from early in-silico photoconversion showing a mixed distribution of neural and mesodermal fates ($n = 5$). **c**, Late in-silico photoconversion at 14 hpf showing the marked NMP territory and tracked cell distributions at 15 and 19 hpf, with red indicating cells tracked from the marked region. **d**, Quantification showing predominantly mesodermal fate from late in-silico photoconversion ($n = 4$). **e**, Early in-vivo photoconversion experiment at 10 hpf using H2B-Dendra2,

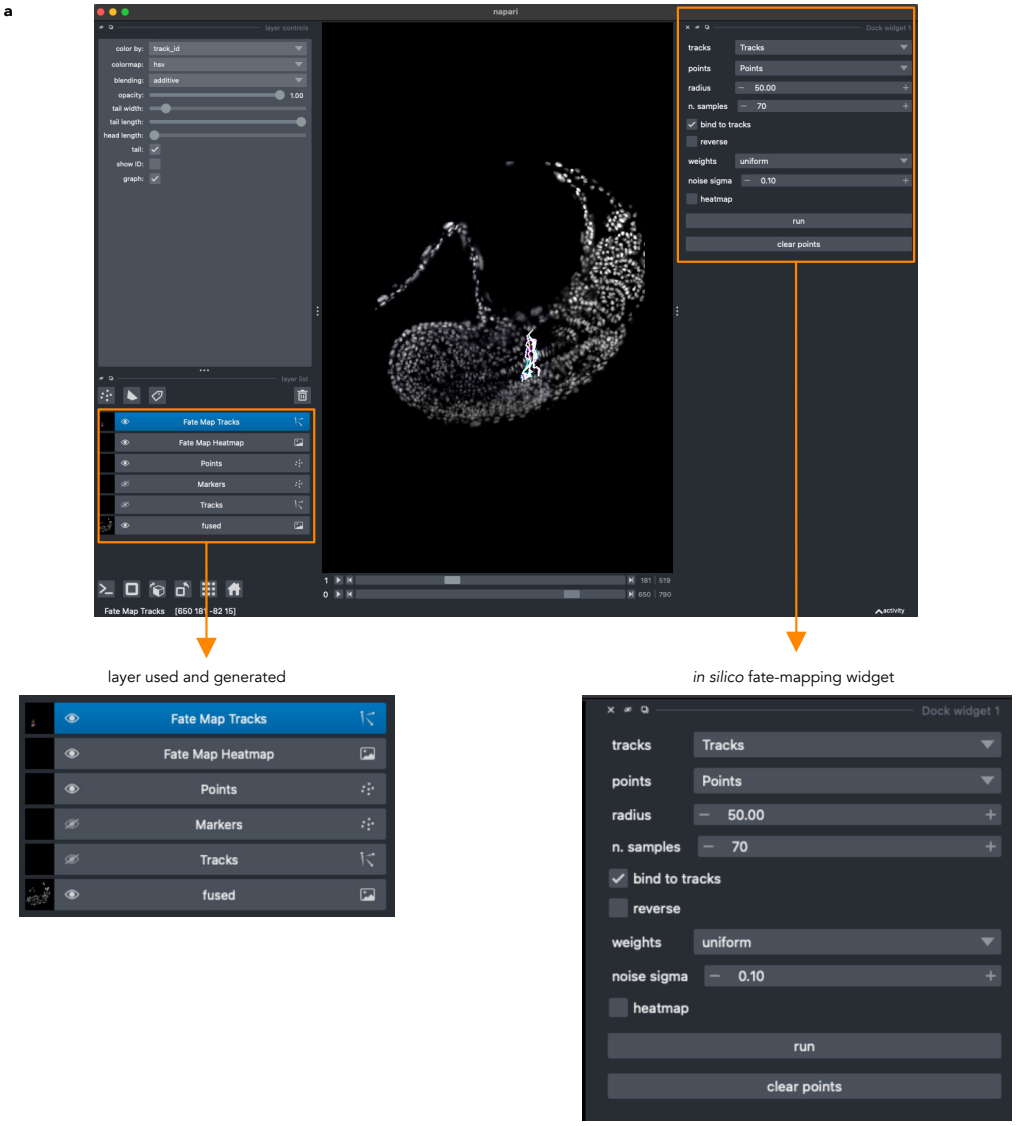


Figure A.2: In-silico fate mapping plugin interface. **a**, napari viewer showing the main interface with cell tracks and the plugin widget. Left inset: napari layers panel showing input data (Points and Tracks), auxiliary data (fused image), and output layers (Fate Map Tracks and Fate Map Heatmap). Right inset: plugin parameters including track and point layer selection, spatial radius (50.00), number of samples (70), noise sigma (0.10), and execution controls. Modified from [3] supplementary materials.

showing photoconverted cells (red) and non-photoconverted cells (blue). **f**, Cell identification and fate quantification from early photoconversion, showing approximately equal distribution between mesodermal and neural tissue ($n = 5$). **g**, Late in-vivo photoconversion at 14 hpf showing the spatial distribution of photoconverted cells at 14 and 17

hpf. **h**, Quantification from late photoconversion showing predominantly mesodermal fate with minimal neural contribution ($n = 6$). The in-silico predictions enabled hypothesis generation for the in-vivo photoconversion experiments, demonstrating the utility of tracking-based fate mapping for experimental design. See the Zebrahub paper [3] for experimental details.

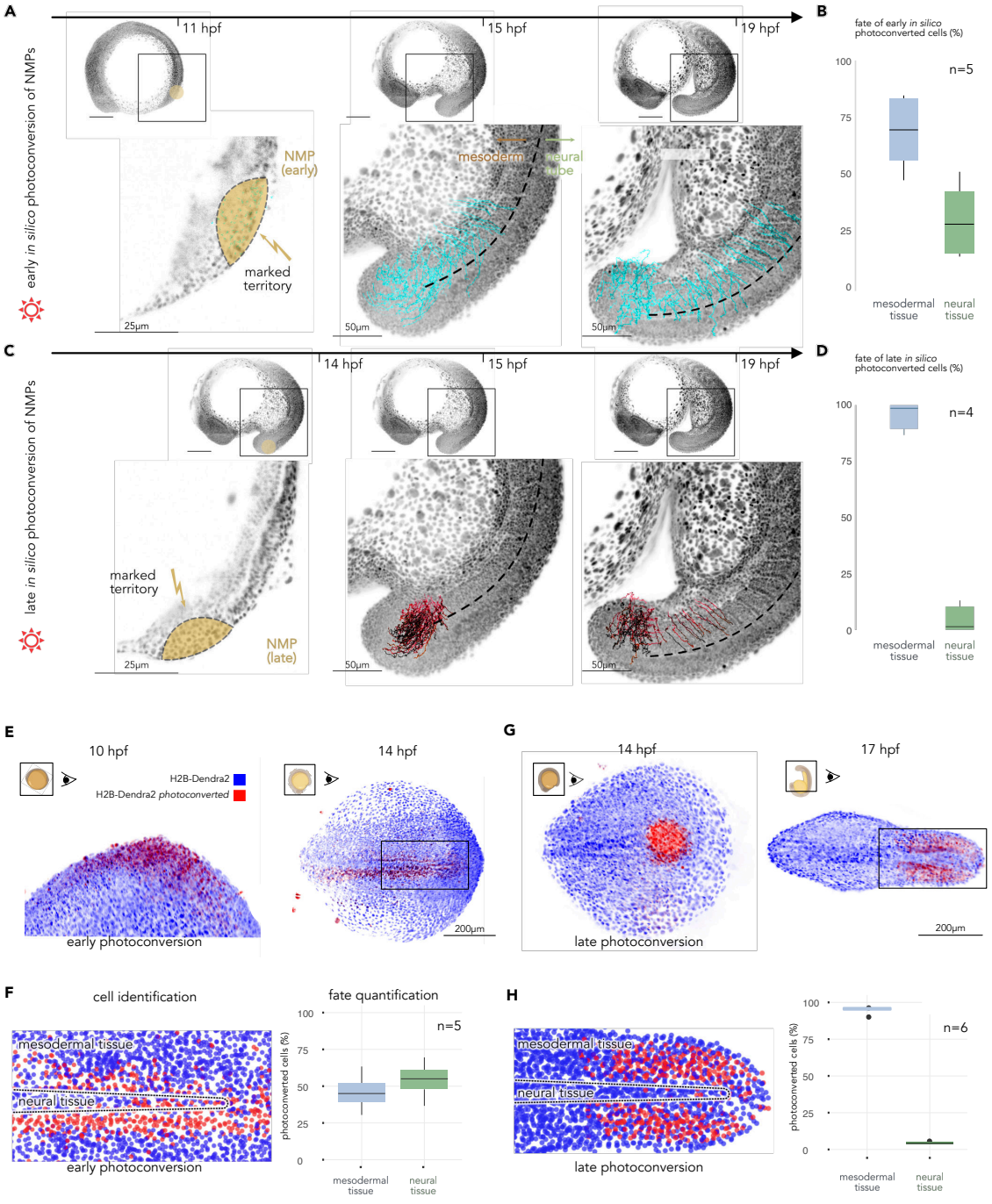


Figure A.3: In-silico fate mapping validation with photoconversion experiments. Early and late stage in-silico fate mapping predictions **a-d** and corresponding in-vivo photoconversion validation experiments **e-h** for neuromesodermal progenitor (NMP) cells in zebrafish embryos. See text for detailed description. Modified from [3].

Bibliography

[1] J. Bragantini, I. Theodoro, X. Zhao, T. A. Huijben, E. Hirata-Miyasaki, S. VijayKumar, A. Balasubramanian, T. Lao, R. Agrawal, S. Xiao, *et al.*, "Ultrack: pushing the limits of cell tracking across biological scales," *Nature Methods*, pp. 1–14, 2025.

[2] J. Bragantini, M. Lange, and L. Royer, "Large-scale multi-hypotheses cell tracking using ultrametric contours maps," in *European Conference on Computer Vision*, 2024.

[3] M. Lange, A. Granados, S. VijayKumar, J. Bragantini, S. Ancheta, Y.-J. Kim, S. Santhosh, M. Borja, H. Kobayashi, E. McGeever, *et al.*, "A multimodal zebrafish developmental atlas reveals the state-transition dynamics of late-vertebrate pluripotent axial progenitors," *Cell*, vol. 187, no. 23, pp. 6742–6759, 2024.

[4] T. A. Huijben, A. G. Anderson III, A. Sweet, E. Hoops, C. Larsen, K. Awayan, J. Bragantini, M. Lange, C.-L. Chiu, and L. A. Royer, "inTRACKtive: a web-based tool for interactive cell tracking visualization," *Nature Methods*, pp. 1–3, 2025.

[5] E. Eck, B. Moretti, B. H. Schlomann, J. Bragantini, M. Lange, X. Zhao, S. VijayKumar, G. Valentin, C. Loureiro, D. Soroldoni, *et al.*, "Single-cell transcriptional dynamics in a living vertebrate," *bioRxiv*, 2024.

[6] J. Moore, D. Basurto-Lozada, S. Besson, J. Bogovic, J. Bragantini, E. M. Brown, J.-M. Burel, X. Casas Moreno, G. de Medeiros, E. E. Diel, *et al.*, "Ome-zarr: a cloud-optimized bioimaging file format with international community support," *Histochemistry and Cell Biology*, vol. 160, no. 3, pp. 223–251, 2023.

[7] B. Yang, M. Lange, A. Millett-Sikking, X. Zhao, J. Bragantini, S. Vijay Kumar, M. Kamb, R. Gómez-Sjöberg, A. C. Solak, W. Wang, *et al.*, "Daxi—high-resolution,

large imaging volume and multi-view single-objective light-sheet microscopy," *Nature Methods*, vol. 19, no. 4, pp. 461–469, 2022.

[8] R. Hooke, "Micrographia," in *Literature and Science, 1660-1834, Part II vol 5*, pp. 1–14, Routledge, 1665.

[9] A. H. Coons, H. J. Creech, and R. N. Jones, "Immunological properties of an antibody containing a fluorescent group.," *Proceedings of the society for experimental biology and medicine*, vol. 47, no. 2, pp. 200–202, 1941.

[10] M. Knoll and E. Ruska, "Das elektronenmikroskop," *Zeitschrift für physik*, vol. 78, no. 5, pp. 318–339, 1932.

[11] S. W. Hell and J. Wichmann, "Breaking the diffraction resolution limit by stimulated emission: stimulated-emission-depletion fluorescence microscopy," *Optics letters*, vol. 19, no. 11, pp. 780–782, 1994.

[12] M. J. Rust, M. Bates, and X. Zhuang, "Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (storm)," *Nature Methods*, vol. 3, no. 10, pp. 793–796, 2006.

[13] E. Betzig, G. H. Patterson, R. Sougrat, O. W. Lindwasser, S. Olenych, J. S. Bonifacino, M. W. Davidson, J. Lippincott-Schwartz, and H. F. Hess, "Imaging intracellular fluorescent proteins at nanometer resolution," *Science*, vol. 313, no. 5793, pp. 1642–1645, 2006.

[14] J. Huisken, J. Swoger, F. Del Bene, J. Wittbrodt, and E. H. Stelzer, "Optical sectioning deep inside live embryos by selective plane illumination microscopy," *Science*, vol. 305, no. 5686, pp. 1007–1009, 2004.

[15] P. J. Keller, A. D. Schmidt, J. Wittbrodt, and E. H. Stelzer, "Reconstruction of zebrafish early embryonic development by scanned light sheet microscopy," *Science*, vol. 322, no. 5904, pp. 1065–1069, 2008.

[16] L. A. Royer, W. C. Lemon, R. K. Chhetri, Y. Wan, M. Coleman, E. W. Myers, and P. J. Keller, "Adaptive light-sheet microscopy for long-term, high-resolution imaging in living organisms," *Nature Biotechnology*, vol. 34, no. 12, pp. 1267–1278, 2016.

[17] W. H. Richardson, "Bayesian-based iterative method of image restoration," *Journal of the optical society of America*, vol. 62, no. 1, pp. 55–59, 1972.

[18] L. B. Lucy, "An iterative technique for the rectification of observed distributions," *Astronomical Journal, Vol. 79, p. 745 (1974)*, vol. 79, p. 745, 1974.

[19] M. Weigert, U. Schmidt, T. Boothe, A. Müller, A. Dibrov, A. Jain, B. Wilhelm, D. Schmidt, C. Broaddus, S. Culley, *et al.*, "Content-aware image restoration: pushing the limits of fluorescence microscopy," *Nature Methods*, vol. 15, no. 12, pp. 1090–1097, 2018.

[20] C. Pape, T. Beier, P. Li, V. Jain, D. D. Bock, and A. Kreshuk, "Solving large multicut problems for connectomics via domain decomposition," in *Proceedings of the IEEE International conference on computer vision workshops*, pp. 1–10, 2017.

[21] L. Heinrich, D. Bennett, D. Ackerman, W. Park, J. Bogovic, N. Eckstein, A. Petruncio, J. Clements, S. Pang, C. S. Xu, *et al.*, "Whole-cell organelle segmentation in volume electron microscopy," *Nature*, vol. 599, no. 7883, pp. 141–146, 2021.

[22] F. Amat, W. Lemon, D. P. Mossing, K. McDole, Y. Wan, K. Branson, E. W. Myers, and P. J. Keller, "Fast, accurate reconstruction of cell lineages from large-scale fluorescence microscopy data," *Nature Methods*, vol. 11, no. 9, pp. 951–958, 2014.

[23] G. Shah, K. Thierbach, B. Schmid, J. Waschke, A. Reade, M. Hlawitschka, I. Roeder, N. Scherf, and J. Huisken, "Multi-scale imaging and analysis identify pan-embryo cell dynamics of germlayer formation in zebrafish," *Nature Communications*, vol. 10, no. 1, p. 5753, 2019.

[24] B.-C. Chen, W. R. Legant, K. Wang, L. Shao, D. E. Milkie, M. W. Davidson, C. Janetopoulos, X. S. Wu, J. A. Hammer III, Z. Liu, *et al.*, "Lattice light-sheet microscopy: imaging molecules to embryos at high spatiotemporal resolution," *Science*, vol. 346, no. 6208, p. 1257998, 2014.

[25] B. Chen, B.-J. Chang, P. Roudot, F. Zhou, E. Sapoznik, M. Marlar-Pavey, J. B. Hayes, P. T. Brown, C.-W. Zeng, T. Lambert, *et al.*, "Resolution doubling in light-sheet microscopy via oblique plane structured illumination," *Nature Methods*, vol. 19, no. 11, pp. 1419–1426, 2022.

[26] Y. Shi, J. S. Tabet, D. E. Milkie, T. A. Daugird, C. Q. Yang, A. T. Ritter, A. Giovannucci, and W. R. Legant, "Smart lattice light-sheet microscopy for imaging rare and complex cellular events," *Nature Methods*, pp. 1–10, 2024.

[27] M. B. Bouchard, V. Voleti, C. S. Mendes, C. Lacefield, W. B. Grueber, R. S. Mann, R. M. Bruno, and E. M. Hillman, "Swept confocally-aligned planar excitation (scape) microscopy for high-speed volumetric imaging of behaving organisms," *Nature Photonics*, vol. 9, no. 2, pp. 113–119, 2015.

[28] T.-L. Liu, S. Upadhyayula, D. E. Milkie, V. Singh, K. Wang, I. A. Swinburne, K. R. Mosaliganti, Z. M. Collins, T. W. Hiscock, J. Shea, *et al.*, "Observing the cell in its native state: Imaging subcellular dynamics in multicellular organisms," *Science*, vol. 360, no. 6386, p. eaaq1392, 2018.

[29] F. Amat, B. Höckendorf, Y. Wan, W. C. Lemon, K. McDole, and P. J. Keller, "Efficient processing and analysis of large-scale light-sheet microscopy data," *Nature Protocols*, vol. 10, no. 11, pp. 1679–1696, 2015.

[30] M. Maška, V. Ulman, P. Delgado-Rodriguez, E. Gómez-de Mariscal, T. Nečasová, F. A. Guerrero Peña, T. I. Ren, E. M. Meyerowitz, T. Scherr, K. Löffler, *et al.*, "The cell tracking challenge: 10 years of objective benchmarking," *Nature Methods*, pp. 1–11, 2023.

[31] D. Feldman, A. Singh, J. L. Schmid-Burgk, R. J. Carlson, A. Mezger, A. J. Garrity, F. Zhang, and P. C. Blainey, "Optical pooled screens in human cells," *Cell*, vol. 179, no. 3, pp. 787–799, 2019.

[32] I. E. Ivanov, E. Hirata-Miyasaki, T. Chandler, R. C. Kovilakam, Z. Liu, C. Liu, M. D. Leonetti, B. Huang, and S. B. Mehta, "Mantis: high-throughput 4d imaging and analysis of the molecular and physical architecture of cells," *bioRxiv*, pp. 2023–12, 2023.

[33] S. Kim, M. Pochitaloff, G. A. Stooke-Vaughan, and O. Campàs, "Embryonic tissues as active foams," *Nature Physics*, vol. 17, no. 7, pp. 859–866, 2021.

[34] K. McDole, L. Guignard, F. Amat, A. Berger, G. Malandain, L. A. Royer, S. C. Turaga, K. Branson, and P. J. Keller, "In toto imaging and reconstruction of post-implantation mouse development at the single-cell level," *Cell*, vol. 175, no. 3, pp. 859–876, 2018.

[35] Y. Wan, Z. Wei, L. L. Looger, M. Koyama, S. Druckmann, and P. J. Keller, "Single-cell reconstruction of emerging population activity in an entire developing circuit," *Cell*, vol. 179, no. 2, pp. 355–372, 2019.

[36] J. D. Currie, A. Kawaguchi, R. M. Traspas, M. Schuez, O. Chara, and E. M. Tanaka, "Live imaging of axolotl digit regeneration reveals spatiotemporal choreography of diverse connective tissue progenitor pools," *Developmental cell*, vol. 39, no. 4, pp. 411–423, 2016.

[37] Ç. Çevrim, B. Laplace-Builhé, K. Sugawara, M. L. Rusciano, N. Labert, J. Brocard, A. Almazán, and M. Averof, "Long-term live imaging, cell identification and cell tracking in regenerating crustacean legs," *eLife*, vol. 14, p. RP107534, 2025.

[38] U. Schmidt, M. Weigert, C. Broaddus, and G. Myers, "Cell detection with star-convex polygons," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 265–273, Springer, 2018.

[39] C. Stringer, T. Wang, M. Michaelos, and M. Pachitariu, "Cellpose: a generalist algorithm for cellular segmentation," *Nature Methods*, vol. 18, no. 1, pp. 100–106, 2021.

[40] A. Wolny, L. Cerrone, A. Vijayan, R. Tofanelli, A. V. Barro, M. Louveaux, C. Wenzl, S. Strauss, D. Wilson-Sánchez, R. Lymbouridou, *et al.*, "Accurate and versatile 3d segmentation of plant tissues at cellular resolution," *eLife*, vol. 9, p. e57613, 2020.

[41] A. Archit, L. Freckmann, S. Nair, N. Khalid, P. Hilt, V. Rajashekar, M. Freitag, C. Teuber, G. Buckley, S. von Haaren, *et al.*, "Segment anything for microscopy," *Nature Methods*, pp. 1–13, 2025.

[42] K. Ulicna, G. Vallardi, G. Charras, and A. R. Lowe, "Automated deep lineage tree analysis using a bayesian single cell tracking approach," *Frontiers in Computer Science*, vol. 3, p. 734559, 2021.

[43] K. Sugawara, Ç. Çevrim, and M. Averof, "Tracking cell lineages in 3d by incremental deep learning," *eLife*, vol. 11, p. e69380, 2022.

[44] K. Löffler and R. Mikut, "Embedtrack—simultaneous cell segmentation and tracking through learning offsets and clustering bandwidths," *IEEE Access*, vol. 10, pp. 77147–77157, 2022.

[45] C. Malin-Mayor, P. Hirsch, L. Guignard, K. McDole, Y. Wan, W. C. Lemon, D. Kainmueller, P. J. Keller, S. Preibisch, and J. Funke, "Automated reconstruction of whole-embryo cell lineages by learning from sparse annotations," *Nature Biotechnology*, pp. 1–6, 2022.

[46] Y. T. Fukai and K. Kawaguchi, "Laptrack: linear assignment particle tracking with tunable metrics," *Bioinformatics*, vol. 39, no. 1, p. btac799, 2023.

[47] R. Reme, A. Newson, E. Angelini, J.-C. Olivo-Marin, and T. Lagache, "Particle tracking in biological images with optical-flow enhanced kalman filtering," in *Int. Symposium on Biomedical Imaging (ISBI)*, pp. 1–5, IEEE, 2024.

[48] B. Gallusser and M. Weigert, "Trackastra: Transformer-based cell tracking for live-cell microscopy," in *European Conference on Computer Vision*, 2024.

[49] M. A. Betjes, R. N. Kok, S. J. Tans, and J. S. van Zon, "Cell tracking with accurate error prediction," *Nature Methods*, pp. 1–11, 2025.

[50] V. Ulman, M. Maška, K. E. Magnusson, O. Ronneberger, C. Haubold, N. Harder, P. Matula, P. Matula, D. Svoboda, M. Radojevic, *et al.*, "An objective comparison of cell-tracking algorithms," *Nature Methods*, vol. 14, no. 12, pp. 1141–1152, 2017.

[51] M. Schiegg, P. Hanslovsky, B. X. Kausler, L. Hufnagel, and F. A. Hamprecht, "Conservation tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2928–2935, 2013.

[52] F. Jug, T. Pietzsch, D. Kainmüller, J. Funke, M. Kaiser, E. van Nimwegen, C. Rother, and G. Myers, "Optimal joint segmentation and tracking of escherichia coli in the

mother machine," in *Bayesian and graphical Models for Biomedical Imaging: First International Workshop, BAMBI 2014, Cambridge, MA, USA, September 18, 2014, Revised Selected Papers*, pp. 25–36, Springer, 2014.

[53] E. Türetken, X. Wang, C. J. Becker, C. Haubold, and P. Fua, "Network flow integer programming to track elliptical cells in time-lapse sequences," *IEEE Transactions on Medical Imaging*, vol. 36, no. 4, pp. 942–951, 2016.

[54] F. Jug, E. Levinkov, C. Blasse, E. W. Myers, and B. Andres, "Moral lineage tracing," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5926–5935, 2016.

[55] M. Rempfler, J.-H. Lange, F. Jug, C. Blasse, E. W. Myers, B. H. Menze, and B. Andres, "Efficient algorithms for moral lineage tracing," in *IEEE International Conference on Computer Vision*, pp. 4695–4704, 2017.

[56] T. Ben-Haim and T. R. Raviv, "Graph neural network for cell tracking in microscopy videos," in *European Conference on Computer Vision*, pp. 610–626, Springer, 2022.

[57] H. Zhou, S. Kim, Z. Zhao, J. Fan, W. Huang, X. Sui, L. Shao, H. An, J.-R. Zhang, J. Wu, *et al.*, "CELLECT: contrastive embedding learning for large-scale efficient cell tracking," *Nature Methods*, pp. 1–12, 2025.

[58] M. Pachitariu and C. Stringer, "Cellpose 2.0: how to train your own model," *Nature Methods*, vol. 19, no. 12, pp. 1634–1641, 2022.

[59] A. Vijayan, T. A. Mody, Q. Yu, A. Wolny, L. Cerrone, S. Strauss, M. Tsiantis, R. S. Smith, F. Hamprecht, A. Kreshuk, *et al.*, "A deep learning-based toolkit for 3d nuclei segmentation and quantitative analysis in cellular and tissue context," *bioRxiv*, pp. 2024–02, 2024.

[60] S. Beucher, "Use of watersheds in contour detection," in *Proceedings of the Int. Workshop on Image Processing*, pp. 17–21, 1979.

[61] L. Najman and M. Schmitt, "Geodesic saliency of watershed contours and hierarchical segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 12, pp. 1163–1173, 1996.

[62] A. X. Falcão, J. Stolfi, and R. A. de Lotufo, "The image foresting transform: Theory, algorithms, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 19–29, 2004.

[63] E. W. Dijkstra, "A note on two problems in connection with graphs," 1959.

[64] P. A. V. Miranda and L. A. C. Mansilla, "Oriented image foresting transform segmentation by seed competition," *IEEE Transactions on Image Processing*, vol. 23, no. 1, pp. 389–398, 2014.

[65] J. Bragantini, S. B. Martins, C. Castelo-Fernandez, and A. X. Falcão, "Graph-based image segmentation using dynamic trees," in *Iberoamerican Congress on Pattern Recognition*, pp. 470–478, Springer, 2018.

[66] R. d. A. Lotufo, A. X. Falcão, and F. A. Zampirolli, "IFT-watershed from gray-scale marker," in *Conf. on Graphics, Patterns and Images (SIBGRAPI)*, pp. 146–152, IEEE, 2002.

[67] J. H. Kappes, M. Speth, B. Andres, G. Reinelt, and C. Schn, "Globally optimal image partitioning by multicuts," in *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 31–44, Springer, 2011.

[68] M. Keuper, E. Levinkov, N. Bonneel, G. Lavoué, T. Brox, and B. Andres, "Efficient decomposition of image and mesh graphs by lifted multicuts," in *IEEE International Conference on Computer Vision*, pp. 1751–1759, 2015.

[69] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[70] Y. Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in nd images," in *IEEE International Conference on Computer Vision*, vol. 1, 2001.

[71] C. Couprie, L. Grady, L. Najman, and H. Talbot, "Power watershed: A unifying graph-based optimization framework," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 7, pp. 1384–1399, 2011.

[72] S. Wolf, C. Pape, A. Bailoni, N. Rahaman, A. Kreshuk, U. Kothe, and F. Hamprecht, "The mutex watershed: efficient, parameter-free image partitioning," in *European Conference on Computer Vision*, pp. 546–562, 2018.

[73] S. Wolf, A. Bailoni, C. Pape, N. Rahaman, A. Kreshuk, U. Köthe, and F. A. Hamprecht, "The mutex watershed and its objective: Efficient, parameter-free graph partitioning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3724–3738, 2020.

[74] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

[75] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[76] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[77] C. Stringer and M. Pachitariu, "Cellpose3: one-click image restoration for improved cellular segmentation," *Nature Methods*, vol. 22, no. 3, pp. 592–599, 2025.

[78] M. Pachitariu, M. Rariden, and C. Stringer, "Cellpose-sam: superhuman generalization for cellular segmentation," *bioRxiv*, pp. 2025–04, 2025.

[79] K. Briggman, W. Denk, S. Seung, M. Helmstaedter, and S. C. Turaga, "Maximin affinity learning of image segmentation," in *Advances in Neural Information Processing Systems*, vol. 22, 2009.

[80] J. Funke, F. Tschopp, W. Grisaitis, A. Sheridan, C. Singh, S. Saalfeld, and S. C. Turaga, "Large scale image segmentation with structured loss based deep learning for connectome reconstruction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 7, pp. 1669–1680, 2018.

[81] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

[82] S. Wolf, L. Schott, U. Kothe, and F. Hamprecht, "Learned watershed: End-to-end learning of seeded segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2011–2019, 2017.

[83] L. Cerrone, A. Zeilmann, and F. A. Hamprecht, "End-to-end learned random walker for seeded image segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12559–12568, 2019.

[84] L. Grady, "Random walks for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1768–1783, 2006.

[85] P. Bosilj, E. Kijak, and S. Lefèvre, "Partition and inclusion hierarchies of images: A comprehensive survey," *Journal of Imaging*, vol. 4, no. 2, p. 33, 2018.

[86] C. Berger, T. Géraud, R. Levillain, N. Widynski, A. Baillard, and E. Bertin, "Effective component tree computation with application to pattern recognition in astronomical imaging," in *IEEE International Conference on Image Processing*, vol. 4, pp. IV–41, 2007.

[87] P. Salembier, A. Oliveras, and L. Garrido, "Antiextensive connected operators for image and sequence processing," *IEEE Transactions on Image Processing*, vol. 7, no. 4, pp. 555–570, 1998.

[88] T. Géraud, E. Carlinet, S. Crozet, and L. Najman, "A quasi-linear algorithm to compute the tree of shapes of n d images," in *Mathematical Morphology and Its Applications to Signal and Image Processing*, pp. 98–110, Springer, 2013.

[89] P. Salembier and L. Garrido, "Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval," *IEEE Transactions on Image Processing*, vol. 9, no. 4, pp. 561–576, 2000.

[90] L. Najman, "On the equivalence between hierarchical segmentations and ultrametric watersheds," *Journal of Mathematical Imaging and Vision*, vol. 40, no. 3, pp. 231–247, 2011.

[91] P. Arbelaez, "Boundary extraction in natural images using ultrametric contour maps," in *Conference on Computer Vision and Pattern Recognition Workshop*, pp. 182–182, IEEE, 2006.

[92] J. Pont-Tuset, P. Arbelaez, J. T. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping for image segmentation and object proposal generation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 1, pp. 128–140, 2016.

[93] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[94] K.-K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool, "Convolutional oriented boundaries: From image segmentation to high-level tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 819–833, 2017.

[95] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2010.

[96] B. R. Kiran and J. Serra, "Global–local optimizations by hierarchical cuts and climbing energies," *Pattern Recognition*, vol. 47, no. 1, pp. 12–24, 2014.

[97] A. Bailoni, C. Pape, N. Hütsch, S. Wolf, T. Beier, A. Kreshuk, and F. A. Hamprecht, "GASP, a generalized framework for agglomerative clustering of signed graphs and its application to instance segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11645–11655, 2022.

[98] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[99] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2008.

[100] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *IEEE International Conference on Image Processing*, pp. 3464–3468, IEEE, 2016.

[101] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking without bells and whistles," in *IEEE International Conference on Computer Vision*, pp. 941–951, 2019.

[102] S. Tang, B. Andres, M. Andriluka, and B. Schiele, "Subgraph decomposition for multi-target tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5033–5041, 2015.

[103] S. Tang, M. Andriluka, B. Andres, and B. Schiele, "Multiple people tracking by lifted multicut and person re-identification," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3539–3548, 2017.

[104] M. Keuper, S. Tang, B. Andres, T. Brox, and B. Schiele, "Motion segmentation & multiple object tracking by correlation co-clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 1, pp. 140–153, 2018.

[105] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell system technical journal*, vol. 49, no. 2, pp. 291–307, 1970.

[106] T. Meinhardt, A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer, "Trackformer: Multi-object tracking with transformers," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8844–8854, 2022.

[107] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, *et al.*, "Sam 2: Segment anything in images and videos," in *International Conference on Learning Representations*, 2025.

[108] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision*, pp. 213–229, Springer, 2020.

[109] O. M. O'Connor and M. J. Dunlop, "Cell-TRACTR: A transformer-based model for end-to-end segmentation and tracking of cells," *PLOS Computational Biology*, vol. 21, no. 5, p. e1013071, 2025.

[110] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "Mot16: A benchmark for multi-object tracking," *arXiv preprint arXiv:1603.00831*, 2016.

[111] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé, "Mot20: A benchmark for multi object tracking in crowded scenes," *arXiv preprint arXiv:2003.09003*, 2020.

[112] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, IEEE, 2012.

[113] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International Conference on Machine Learning*, pp. 1597–1607, PMLR, 2020.

[114] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, *et al.*, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[115] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, *et al.*, "Segment anything," in *IEEE International Conference on Computer Vision*, pp. 4015–4026, 2023.

[116] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[117] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, no. 4, pp. 325–340, 1987.

[118] E. Moen, E. Borba, G. Miller, M. Schwartz, D. Bannon, N. Koe, I. Camplisson, D. Kyme, C. Pavelchek, T. Price, *et al.*, "Accurate cell tracking and lineage construction in live-cell imaging experiments with deep learning," *Biorxiv*, p. 803205, 2019.

[119] D. Neven, B. D. Brabandere, M. Proesmans, and L. V. Gool, "Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8837–8845, 2019.

[120] J. Hayashida and R. Bise, "Cell tracking with deep learning for cell detection and motion estimation in low-frame-rate," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 397–405, Springer, 2019.

[121] J. Hayashida, K. Nishimura, and R. Bise, "Mpm: Joint representation of motion and position map for cell tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3823–3832, 2020.

[122] P. Hirsch, C. Malin-Mayor, A. Santella, S. Preibisch, D. Kainmueller, and J. Funke, "Tracking by weakly-supervised learning and graph optimization for whole-embryo c. elegans lineages," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 25–35, Springer, 2022.

[123] X. Lou, M. Schiegg, and F. A. Hamprecht, "Active structured learning for cell tracking: algorithm, framework, and usability," *IEEE Transactions on Medical Imaging*, vol. 33, no. 4, pp. 849–860, 2014.

[124] D. D. Pop, P. Le Bodic, and J. Nunez-Iglesias, "Needles in the haystack-rapid error detection for cell tracking solutions," in *Int. Symposium on Biomedical Imaging (ISBI)*, pp. 1–4, IEEE, 2025.

[125] M. Lalit and J. Funke, "An investigation of unsupervised cell tracking and interactive fine-tuning," in *IEEE International Conference on Computer Vision Workshops*, pp. 5792–5800, 2025.

[126] J. H. Friedman, "Stochastic gradient boosting," *Computational statistics & data analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[127] V. Kolmogorov, Y. Boykov, and C. Rother, "Applications of parametric maxflow in computer vision," in *IEEE International Conference on Computer Vision*, pp. 1–8, IEEE, 2007.

[128] J. Yarkony, A. Ihler, and C. C. Fowlkes, "Fast planar correlation clustering for image segmentation," in *European Conference on Computer Vision*, pp. 568–581, Springer, 2012.

[129] K. E. Magnusson, *Segmentation and tracking of cells and particles in time-lapse microscopy*. PhD thesis, KTH Royal Institute of Technology, 2016.

[130] B. Perret, J. Cousty, S. J. F. Guimaraes, and D. S. Maia, "Evaluation of hierarchical watersheds," *IEEE Transactions on Image Processing*, vol. 27, no. 4, pp. 1676–1688, 2017.

[131] B. Perret, G. Chierchia, J. Cousty, S. J. F. Guimaraes, Y. Kenmochi, and L. Najman, "Higra: Hierarchical graph analysis," *SoftwareX*, vol. 10, p. 100335, 2019.

[132] L. Najman, J. Cousty, and B. Perret, "Playing with kruskal: algorithms for morphological trees in edge-weighted graphs," in *Mathematical Morphology and Its Applications to Signal and Image Processing*, pp. 135–146, Springer, 2013.

[133] R. E. Tarjan, "Efficiency of a good but not linear set union algorithm," *Journal of the ACM (JACM)*, vol. 22, no. 2, pp. 215–225, 1975.

[134] F. Meyer, A. Oliveras Vergés, P. J. Salembier Clairon, and C. Vachier, "Morphological tools for segmentation: connected operators and watersheds," *Annales des télecommunications. Annals of telecommunications*, vol. 52, no. 7-8, pp. 366–379, 1997.

[135] B. Perret, J. Cousty, S. J. F. Guimarães, Y. Kenmochi, and L. Najman, "Removing non-significant regions in hierarchical clustering and segmentation," *Pattern Recognition Letters*, vol. 128, pp. 433–439, 2019.

[136] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical society*, vol. 7, no. 1, pp. 48–50, 1956.

[137] R. Deng, C. Shen, S. Liu, H. Wang, and X. Liu, "Learning to predict crisp boundaries," in *European Conference on Computer Vision*, pp. 562–578, 2018.

[138] S. Xie and Z. Tu, "Holistically-nested edge detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1395–1403, 2015.

[139] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, *et al.*, "Scipy 1.0: fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 2020.

[140] M. Pereyra, A. Drusko, F. Krämer, F. Strobl, E. H. Stelzer, and F. Matthäus, "Quickpiv: Efficient 3d particle image velocimetry software applied to quantifying cellular migration during embryogenesis," *BMC bioinformatics*, vol. 22, pp. 1–20, 2021.

[141] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, pp. 8026–8037, 2019.

[142] T. Gals, "traccuracy."

[143] P. Matula, M. Maška, D. V. Sorokin, P. Matula, C. Ortiz-de Solórzano, and M. Kozubek, "Cell tracking accuracy measurement based on comparison of acyclic oriented graphs," *PloS one*, vol. 10, no. 12, p. e0144959, 2015.

[144] V. Goyal, N. J. Schaub, T. C. Voss, and N. A. Hotaling, "Unbiased image segmentation assessment toolkit for quantitative differentiation of state-of-the-art algorithms and pipelines," *BMC bioinformatics*, vol. 24, no. 1, p. 388, 2023.

[145] A. Ruggieri, E. Dazert, P. Metz, S. Hofmann, J.-P. Bergeest, J. Mazur, P. Bankhead, M.-S. Hiet, S. Kallis, G. Alvisi, *et al.*, "Dynamic oscillation of translation and stress granule formation mark the cellular response to virus infection," *Cell host & microbe*, vol. 12, no. 1, pp. 71–85, 2012.

[146] M. Lab, "recOrder."

[147] Z. Liu, E. Hirata-Miyasaki, S. Pradeep, J. Rahm, C. Foley, T. Chandler, I. Ivanov, H. Woosley, T. Lao, A. Balasubramanian, *et al.*, "Robust virtual staining of landmark organelles," *bioRxiv*, pp. 2024–05, 2024.

[148] D. Cai, K. B. Cohen, T. Luo, J. W. Lichtman, and J. R. Sanes, "Improved tools for the brainbow toolbox," *Nature Methods*, vol. 10, no. 6, pp. 540–547, 2013.

[149] K. Y. Chan, C.-C. S. Yan, H.-Y. Roan, S.-C. Hsu, T.-L. Tseng, C.-D. Hsiao, C.-P. Hsu, and C.-H. Chen, "Skin cells undergo asynthetic fission to expand body surfaces in zebrafish," *Nature*, vol. 605, no. 7908, pp. 119–125, 2022.

[150] K. Weber, M. Thomaschewski, D. Benten, and B. Fehse, "Rgb marking with lentiviral vectors for multicolor clonal cell tracking," *Nature protocols*, vol. 7, no. 5, pp. 839–849, 2012.

[151] D. Ershov, M.-S. Phan, J. W. Pylvänäinen, S. U. Rigaud, L. Le Blanc, A. Charles-Orszag, J. R. Conway, R. F. Laine, N. H. Roy, D. Bonazzi, *et al.*, "Trackmate 7: integrating state-of-the-art segmentation algorithms into tracking pipelines," *Nature Methods*, vol. 19, no. 7, pp. 829–832, 2022.

[152] N. Otsu, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.

[153] J. B. Pawley, "Fundamental limits in confocal microscopy," in *Handbook of biological confocal microscopy*, pp. 20–42, Springer, 2006.

[154] A. Jain, *Molecular, Cellular and Mechanical basis of Epithelial Morphogenesis during Tribolium Embryogenesis*. PhD thesis, Technische Universität Dresden, 2018.

[155] G. Balakrishnan, A. Zhao, M. R. Sabuncu, J. Guttag, and A. V. Dalca, "Voxelmorph: a learning framework for deformable medical image registration," *IEEE Transactions on Medical Imaging*, vol. 38, no. 8, pp. 1788–1800, 2019.

[156] I. Heemskerk and S. J. Streichan, "Tissue cartography: compressing bio-image data by dimensional reduction," *Nature Methods*, vol. 12, no. 12, pp. 1139–1142, 2015.

[157] R. McGill, J. W. Tukey, and W. A. Larsen, "Variations of box plots," *The american statistician*, vol. 32, no. 1, pp. 12–16, 1978.

[158] J. I. Murray, Z. Bao, T. J. Boyle, M. E. Boeck, B. L. Mericle, T. J. Nicholas, Z. Zhao, M. J. Sandel, and R. H. Waterston, "Automated analysis of embryonic gene expression with cellular resolution in c. elegans," *Nature Methods*, vol. 5, no. 8, pp. 703–709, 2008.

[159] C. E. Akbaş, V. Ulman, M. Maška, F. Jug, and M. Kozubek, "Automatic fusion of segmentation and tracking labels," in *Computer Vision–ECCV 2018 Workshops: Munich, Germany, September 8-14, 2018, Proceedings, Part VI 15*, pp. 446–454, Springer, 2019.

[160] D. Kim, C. Kwon, and I. Hwang, "Gaussian mixture probability hypothesis density filter against measurement origin uncertainty," *Signal Processing*, vol. 171, p. 107448, 2020.

[161] O. Dzyubachyk, W. A. Van Cappellen, J. Essers, W. J. Niessen, and E. Meijering, "Advanced level-set-based cell tracking in time-lapse fluorescence microscopy," *IEEE Transactions on Medical Imaging*, vol. 29, no. 3, pp. 852–867, 2010.

[162] F. Meyer, "Morphological segmentation revisited," *Space, Structure and Randomness: Contributions in Honor of Georges Matheron in the Field of Geostatistics, Random Sets and Mathematical Morphology*, pp. 315–347, 2005.

[163] J. Stegmaier, J. C. Otte, A. Kobitski, A. Bartschat, A. Garcia, G. U. Nienhaus, U. Strähle, and R. Mikut, "Fast segmentation of stained nuclei in terabyte-scale, time resolved 3d microscopy image stacks," *PloS one*, vol. 9, no. 2, p. e90036, 2014.

[164] J. Funke, L. Mais, A. Champion, N. Dye, and D. Kainmueller, "A benchmark for epithelial cell tracking," in *Proceedings of The European Conference on Computer Vision (ECCV) Workshops*, pp. 0–0, 2018.

[165] B. Aigouy, D. Umetsu, and S. Eaton, "Segmentation and quantitative analysis of epithelial tissues," *Drosophila: Methods and Protocols*, pp. 227–239, 2016.

[166] M. Januszewski, J. Kornfeld, P. H. Li, A. Pope, T. Blakely, L. Lindsey, J. Maitin-Shepard, M. Tyka, W. Denk, and V. Jain, "High-precision automated reconstruction of neurons with flood-filling networks," *Nature Methods*, vol. 15, no. 8, pp. 605–610, 2018.

[167] S. Berg, D. Kutra, T. Kroeger, C. N. Straehle, B. X. Kausler, C. Haubold, M. Schiegg, J. Ales, T. Beier, M. Rudy, *et al.*, "Ilastik: interactive machine learning for (bio) image analysis," *Nature Methods*, vol. 16, no. 12, pp. 1226–1232, 2019.

[168] X. Zhao, "Created in BioRender," 2025.

[169] A. B. Yoo, M. A. Jette, and M. Grondona, "SLURM: Simple linux utility for resource management," in *Job Scheduling Strategies for Parallel Processing: 9th International Workshop*, pp. 44–60, Springer, 2003.

[170] P. K. Shah, A. Santella, A. Jacobo, K. Siletti, A. Hudspeth, and Z. Bao, "An in toto approach to dissecting cellular interactions in complex tissues," *Developmental cell*, vol. 43, no. 4, pp. 530–540, 2017.

[171] A. Erzberger, A. Jacobo, A. Dasgupta, and A. Hudspeth, "Mechanochemical symmetry breaking during morphogenesis of lateral-line sensory organs," *Nature physics*, vol. 16, no. 9, pp. 949–957, 2020.

[172] O. Viader-Llargués, V. Lupperger, L. Pola-Morell, C. Marr, and H. López-Schier, "Live cell-lineage tracing and machine learning reveal patterns of organ regeneration," *eLife*, vol. 7, p. e30823, Mar. 2018. Publisher: eLife Sciences Publications, Ltd.

[173] M. N. Hewitt, I. A. Cruz, T. H. Linbo, and D. W. Raible, "Spherical harmonics analysis reveals cell shape-fate relationships in zebrafish lateral line neuromasts," *Development*, vol. 151, p. dev202251, Jan. 2024.

[174] A. Ben-Hur, A. Elisseeff, and I. Guyon, "A stability based method for discovering structure in clustered data," *Biocomputing 2002*, pp. 6–17, 2001.

[175] U. Von Luxburg *et al.*, "Clustering stability: an overview," *Foundations and Trends® in Machine Learning*, vol. 2, no. 3, pp. 235–274, 2010.

[176] N. Meinshausen and P. Bühlmann, "Stability selection," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 72, no. 4, pp. 417–473, 2010.

[177] H. Liu, K. Roeder, and L. Wasserman, "Stability approach to regularization selection (stars) for high dimensional graphical models," in *Advances in Neural Information Processing Systems*, vol. 23, 2010.

[178] A. Rakhlin and A. Caponnetto, "Stability of k-means clustering," in *Advances in Neural Information Processing Systems*, vol. 19, 2006.

[179] H. Yu, B. Chapman, A. Di Florio, E. Eischen, D. Gotz, M. Jacob, and R. H. Blair, "Bootstrapping estimates of stability for clusters, observations and model selection," *Computational Statistics*, vol. 34, no. 1, pp. 349–372, 2019.

[180] M. Tang, Y. Kaymaz, B. L. Logeman, S. Eichhorn, Z. S. Liang, C. Dulac, and T. B. Sackton, "Evaluating single-cell cluster stability using the jaccard similarity index," *Bioinformatics*, vol. 37, no. 15, pp. 2212–2214, 2021.

[181] C. Sant, L. Mucke, and M. R. Corces, "CHOIR improves significance-based detection of cell types and states from single-cell data," *Nature Genetics*, pp. 1–11, 2025.

[182] S. Ben-David, U. Von Luxburg, and D. Pál, "A sober look at clustering stability," in *International Conference on Computational Learning Theory*, pp. 5–19, Springer, 2006.

[183] S. Ben-David and U. Von Luxburg, "Relating clustering stability to properties of cluster boundaries," in *International Conference on Computational Learning Theory*, pp. 379–390, Omnipress, 2008.

[184] R. J. Campello, D. Moulavi, A. Zimek, and J. Sander, "A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies," *Data Mining and Knowledge Discovery*, vol. 27, pp. 344–371, 2013.

[185] A. Neufeld, A. Dharamshi, L. L. Gao, and D. Witten, "Data thinning for convolution-closed distributions," *Journal of Machine Learning Research*, vol. 25, no. 57, pp. 1–35, 2024.

[186] A. Dharamshi, A. Neufeld, K. Motwani, L. L. Gao, D. Witten, and J. Bien, "Generalized data thinning using sufficient statistics," *Journal of the American Statistical Association*, vol. 120, no. 549, pp. 511–523, 2025.

[187] C. Hennig, "Cluster-wise assessment of cluster stability," *Computational Statistics & Data Analysis*, vol. 52, no. 1, pp. 258–271, 2007.

[188] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[189] V. A. Traag, L. Waltman, and N. J. Van Eck, "From louvain to leiden: guaranteeing well-connected communities," *Scientific reports*, vol. 9, no. 1, pp. 1–12, 2019.

[190] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, p. 226–231, AAAI Press, 1996.

[191] A. Bastidas-Ponce, S. Tritschler, L. Dony, K. Scheibner, M. Tarquis-Medina, C. Salinno, S. Schirge, I. Burtscher, A. Böttcher, F. J. Theis, *et al.*, "Comprehensive single cell mrna profiling reveals a detailed roadmap for pancreatic endocrinogenesis," *Development*, vol. 146, no. 12, p. dev173849, 2019.

[192] V. Bergen, M. Lange, S. Peidli, F. A. Wolf, and F. J. Theis, "Generalizing rna velocity to transient cell states through dynamical modeling," *Nature Biotechnology*, vol. 38, pp. 1408–1414, Aug. 2020.

[193] D. Klein, G. Palla, M. Lange, M. Klein, Z. Piran, M. Gander, L. Meng-Papaxanthos, M. Sterr, L. Saber, C. Jing, *et al.*, "Mapping cells through time and space with moscot," *Nature*, vol. 638, no. 8052, pp. 1065–1075, 2025.

[194] N. Carion, L. Gustafson, Y.-T. Hu, S. Debnath, R. Hu, D. Suris, C. Ryali, K. V. Alwala, H. Khedr, A. Huang, *et al.*, "Sam 3: Segment anything with concepts," *arXiv preprint arXiv:2511.16719*, 2025.

[195] N. Sofroniew, T. Lambert, K. Evans, J. Nunez-Iglesias, G. Bokota, P. Winston, G. Peña-Castellanos, K. Yamauchi, M. Bussonnier, D. Doncila Pop, A. Can Solak, Z. Liu, P. Wadhwa, A. Burt, G. Buckley, A. Sweet, L. Migas, V. Hilsenstein, L. Gaifas, J. Bragantini, J. Rodríguez-Guerra, H. Muñoz, J. Freeman, P. Boone, A. R Lowe, C. Gohlke, L. Royer, A. Pierré, H. Har-Gil, and A. McGovern, "napari: a multi-dimensional image viewer for Python."

[196] The Qt Company, "Qt - cross-platform software libraries and apis." `https://www.qt.io/`, 2024. Accessed: 2024-07-19.

[197] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[198] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, *et al.*, "Fiji: an open-source platform for biological-image analysis," *Nature Methods*, vol. 9, no. 7, pp. 676–682, 2012.

[199] T. Pietzsch and J.-Y. Tinevez, "Mastodon."

[200] R. T. Fielding, *Architectural styles and the design of network-based software architectures*. University of California, Irvine, 2000.

[201] S. Ramírez, "FastAPI."

[202] A. Melnikov and I. Fette, "The WebSocket Protocol." RFC 6455, Dec. 2011.

[203] M. Otto and J. Thornton, "Bootstrap 5."

[204] W. McKinney and P. Team, "Pandas-powerful python data analysis toolkit," 2015.

[205] R. Nishino and S. H. C. Loomis, "Cupy: A numpy-compatible library for nvidia GPU calculations," *Advances in Neural Information Processing Systems Workshop*, vol. 151, no. 7, 2017.

[206] Z. Liu, I. Ivanov, J. Bragantini, T. Chandler, A. C. Solak, E. Hirata-Miyasaki, C. Foltz, L.-H. Yeh, and S. Mehta, "iohub v0.1.0," February 2024.

[207] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, *et al.*, "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[208] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, "scikit-image: image processing in python," *PeerJ*, vol. 2, p. e453, 2014.

[209] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A llvm-based python jit compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pp. 1–6, 2015.

[210] M. Treinish, I. Carvalho, G. Tsilimigkounakis, and N. Sá, "rustworkx: A high-performance graph library for python," *Journal of Open Source Software*, vol. 7, no. 79, p. 3968, 2022.