Inaugural-Dissertation

zur

Erlangung der Doktorwürde

der

Gesamtfakultät für Mathematik, Ingenieur- und Naturwissenschaften

der

Ruprecht-Karls-Universität

Heidelberg

vorgelegt von

Qin Yu, MSc DIC

aus Volksrepublik China[⋆]

Tag der mündlichen Prüfung: 4. Juli 2025

[⋆] Amtlich eingetragener Geburtsort: Hubei; Tatsächlicher Geburtsort: Guangzhou, Guangdong.

Robust and Accessible Segmentation of Cells and Nuclei in 3D Microscopy

Gutachter:

Prof. Dr. Alexis Maizel

Dr. Hanh Vu

Robust and Accessible Segmentation of Cells and Nuclei in 3D Microscopy

余沁 Qin Yu

DEFENSE COMMITTEE:
Prof. Dr. Alexis Maizel
Dr. Hanh Vu
Prof. Dr. Thomas Greb
Dr. Simone Köhler

SUPERVISOR:
Dr. Anna Kreshuk

Heidelberg, Germany

献给父母

恩重如山

Altstadt vom Heidelberger Schloss | Jan 2024 | Heidelberg, Deutschland.

Contents

List of Tables

List of Figures

List of Abbreviations

CNN	convolutional neural network
GASP	Generalised Algorithm for Signed graph Partitioning
PCA	principal component analysis
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
SAM	Segment Anything Model
SBD	Symmetric Best Dice
RAM	random access memory
VRAM	video random access memory
CPU	central processing unit
GPU	graphics processing unit
CUDA	Compute Unified Device Architecture
OOM	out-of-memory
API	application programming interface
MLOps	Machine Learning Operations
LM	light microscopy
EM	electron microscopy
SPIM	Selective Plane Illumination Microscopy
TIFF	Tagged Image File Format
HDF5	Hierarchical Data Format version 5
OME	Open Microscopy Environment
FAIR	Findable, Accessible, Interoperable and Reusable
SNR	signal-to-noise ratio
HITL	human-in-the-loop
IF	Immunofluorescence
GUI	graphical user interface
CLI	command-line interface

Zusammenfassung

Die volumetrische Mikroskopie liefert bislang ungeahnte Einblicke in Zellen und subzelluläre Strukturen unterschiedlichster biologischer Gewebe. Die Analyse dieser komplexen Datensätze erfordert jedoch robuste Segmentierungsmethoden, die Mehrkanal-Informationen optimal ausnutzen und gleichzeitig den Bedarf an Expertenannotationen reduzieren. In dieser Dissertation stelle ich skalierbare Workflows vor, die den Umfang aufwändiger manueller Annotationen reduzieren und zugleich die Segmentierungsgenauigkeit unter realistischen Bildgebungsbedingungen verbessern.

Zunächst präsentiere ich **GoNuclear**, eine vielseitige Toolbox zur dreidimensionalen (3D) Segmentierung von Zellkernen in Pflanzengeweben, die mit dem kostengünstigen und breit einsetzbaren DNA-bindenden Farbstoff TO-PRO-3 markiert wurden. Anders als genetisch kodierte Marker, die aufwändige genetische Transformationen erfordern, lässt sich TO-PRO-3 direkt auf fixierte und geklärte Gewebeproben anwenden, was insbesondere die Kernsegmentierung in Nicht-Modellorganismen stark vereinfacht. Durch Integration von menschlich unterstützten Annotationen und sorgfältig kuratierten Datensätzen ermöglicht GoNuclear eine präzise Segmentierung auch bei schwachen und verrauschten Signalen sowie eine gute Generalisierbarkeit auf verschiedene Gewebe und Färbetechniken. Dadurch werden nachfolgende Analysen wie Kontrolle der Kerngröße, Kern-zu-Zell-Volumenverhältnisse und räumliche Genexpressionsanalysen erleichtert.

Im Folgenden beschreibe ich wesentliche Verbesserungen an **PlantSeg**, einer auf Deep Learning basierenden Toolbox zur 3D-Gewebesegmentierung. Version 2.0 bietet eine interaktive, *napari*-basierte Benutzeroberfläche, Integration in den BioImage Model Zoo, Unterstützung bei spärliche Instanzsegmentierung, automatische Optimierung von Patch- und Halo-Größen sowie leistungsfähige Korrekturwerkzeuge. Diese Erweiterungen ermöglichen eine präzise Segmentierung sowohl von Zellen als auch Zellkernen in komplexen Mikroskopiedaten und machen fortgeschrittene rechnergestützte Methoden für die wissenschaftliche Gemeinschaft leichter zugänglich.

Danach stelle ich **SPOCO** vor, eine einbettungsbasierte Methode zur Instanzsegmentierung, die nur minimale Annotationen benötigt. Durch gezieltes Transferlernen überträgt SPOCO Modelle, die mit wenigen annotierten Datensätzen trainiert wurden, effizient auf neue Bildgebungsbereiche. Dadurch reduziert sich der Annotationsaufwand erheblich, während die Segmentierungsqualität erhalten bleibt. Zusätzlich zeige ich Verbesserungen der Recheneffizienz durch Dimensionsreduktion und neuartige Clustering-Strategien auf und etabliere SPOCO damit als praktikable Lösung für großangelegte Analysen.

Schließlich zeige ich, wie **mehrkanalige Bildanalysen von Mikroskopiedaten** die Segmentierungs- und Tracking-Genauigkeit deutlich verbessern können. Durch die In-

tegration komplementärer chemischer Färbungen, multipler Bildgebungsmodalitäten, zeitlicher Sequenzen und unterschiedlicher biologischer Strukturen in einem einzigen Datensatz erhöhen diese Strategien sowohl die Genauigkeit als auch die Interpretierbarkeit von Segmentierungsergebnissen, insbesondere bei herausfordernden Bildgebungsbedingungen.

Insgesamt tragen diese Beiträge zur Weiterentwicklung der 3D-Segmentierung von Mikroskopiedaten bei, indem sie den Annotationsaufwand verringern und zugleich die Segmentierungsleistung unter realistischen Bedingungen verbessern. Diese Arbeit liefert praxisnahe und benutzerfreundliche Lösungen für präzise, großskalige biologische Analysen und eröffnet neue Möglichkeiten zur Untersuchung entwicklungsbiologischer Prozesse, zellulärer Architektur und Morphogenese in einer Vielzahl pflanzlicher und tierischer Systeme.

Abstract in German kindly proofread by Jonas Hellgoth.

Abstract

Volumetric microscopy reveals unprecedented details of cells and subcellular structures across diverse biological tissues. However, harnessing these complex datasets requires robust segmentation methods that maximise the use of multi-channel information while minimising the need for expert annotation. In this thesis, I introduce scalable workflows that reduce reliance on extensive expert labelling while enhancing segmentation accuracy under realistic imaging conditions.

First, I present **GoNuclear**, a versatile toolkit for three-dimensional (3D) nuclear segmentation of plant tissues stained with the affordable, broadly applicable DNA-binding dye TO-PRO-3. Unlike genetically encoded markers, which require laborious transformations, TO-PRO-3 can be directly applied to fixed and cleared tissues, greatly simplifying nuclear segmentation in non-model species. By leveraging human-in-the-loop annotations and carefully curated datasets, GoNuclear provides accurate segmentation from weak, noisy signals and generalises effectively across diverse tissues and staining modalities. This enables downstream analyses such as nuclear size control, nuclear-to-cell volume ratios, and spatial gene expression.

Next, I describe substantial enhancements to **PlantSeg**, a deep-learning-based toolkit for 3D tissue segmentation. Version 2.0 features an interactive *napari*-based interface, integration with the BioImage Model Zoo, sparse instance segmentation support, automatic optimisation of patch and halo sizes, and powerful proofreading tools. These improvements enable the accurate segmentation of both cells and nuclei in complex microscopy volumes, making advanced computational methods more accessible to the scientific community.

I then introduce **SPOCO**, an embedding-based instance segmentation method requiring minimal annotations. Through targeted transfer learning, SPOCO adapts models trained on limited annotated datasets to new imaging domains, significantly reducing annotation requirements while preserving segmentation quality. I also demonstrate computational efficiency gains through dimensionality reduction and novel clustering strategies, establishing SPOCO as a practical solution for large-scale analyses.

Finally, I show how **multi-channel bioimage analysis** can substantially improve segmentation and tracking accuracy. By integrating complementary chemical stains, multiple imaging modalities, temporal sequences, and various biological structures into a single dataset, these strategies enhance both the accuracy and interpretability of segmentation outcomes, particularly under challenging imaging conditions.

Together, these contributions advance 3D bioimage segmentation by minimising annotation burdens and improving segmentation performance under realistic conditions. This work provides practical, user-friendly solutions for precise, large-scale biological analyses, opening new avenues for the investigation of developmental biology, cellular architecture, and mor-

phogenesis in a wide range of plant and animal systems.

Acknowledgements

Thank you, dear reader, for taking the time to read my thesis. I hope you find it useful.

My deepest gratitude belongs to my parents, Yu Ruoyu and Wang Xianzhi, whose unconditional love, dedication, and trust have shaped who I am today. Their unwavering support is a debt I can never repay. Growing up in such a nurturing and enlightened family has been my greatest blessing. I am also eternally grateful to my grandparents, Yu Engao, Jiang Shiying, Wang Huanshang, and Liu Zhixiu, whose enduring legacy and love have filled my life with courage and warmth. To Yu Xingchen, I wish boundless curiosity, passion, joy, and freedom in your journey ahead.[1]

My heartfelt thanks go to my supervisor, Dr. Anna Kreshuk, for welcoming me into her academic family of *machine learning for bioimage analysis*, and for providing both intellectual freedom and a nurturing research environment. I am deeply indebted to my senior colleagues, Dr. Adrian Wolny and Dr. Lorenzo Cerrone, from the broader *Research Unit FOR2581 Plant Morphodynamics*, for their meticulous guidance and trust in my capabilities. Special appreciation also extends to Dr. Alba Diz-Muñoz for warmly welcoming me as a bridging postdoc contributing to *mechanobiology at the cell surface*.

I gratefully acknowledge my biological collaborators at EMBL Heidelberg, Universität Heidelberg, TU München, MPIPZ Köln, and RIKEN Kobe, whose imaging data and discussions greatly enriched my research. I particularly thank my co-authors Dr. Athul Vijayan and Dr. Tejasvinee Mody from FOR2581 for the valuable *Arabidopsis thaliana* dataset, and Ruben Tesoro Moreno for his mouse fibroblast dataset.

I am thankful to my Thesis Advisory Committee—Anna Erzberger, Aissam Ikmi, and Steffen Lemke—and to my defence committee—Prof. Dr. Alexis Maizel, Dr. Hahn Vu, Prof. Dr. Thomas Greb, and Dr. Simone Köhler—for their valuable time and input.

My sincere gratitude extends to the wider bioimage analysis community, where I have had the privilege to contribute to and benefit from impactful open-source projects. Special thanks go to Dr. Adrian Wolny for `pytorch-3dunet`, Dr. Lorenzo Cerrone for PlantSeg, Dr. Fynn Beuttenmüller for BioImage.IO, Dr. Carsen Stringer for Cellpose, Dr. Uwe Schmidt for StarDist, Jordão Bragantini for Ultrack, and Dr. Talley Lambert for Napari. Without their tools, dedication, and collaborative spirit, my research and contributions would not have been possible. I am also grateful for discussions with Dr. Christian Tischer from the EMBL Imaging Centre and Jurij Pecar from EMBL IT Services.

My time at EMBL has been immensely enriched by the camaraderie of my academic

[1] In this paragraph, names are written with family names first, as they are spoken in my family.

siblings and the ilastik team: Constantin Pape, Adrian Wolny, Fynn Beuttenmüller, Valentyna Zinchenko, Johannes Hugger, Elena Buglakova, Jonas Hellgoth, Hyoungjun Park, Joshua Talks, Konstantinos Almpanakis, Tomaz Fogaca Vieira, Dominik Kutra, Ricardo Miguel Sanchez Loayza, Benedikt Best, and Emil Melnikov. I truly enjoyed the time we shared—whether working in the office, having meals at the canteen, hiking in nature, attending conferences, relaxing after work, or bonding during retreats. These experiences made my EMBL journey personally fulfilling and deeply memorable.

Beyond the lab, I have enjoyed wonderful times with my predoctoral peers and EMBL colleagues in Heidelberg, Praha, and Cambridge, especially Valeriy Pak, Frosina Stojanovska, Shuting Xu, Minglu Wang, Marina Makharova, Magdalena Schindler, Brian Lai, Ashwin Samudre, Edgar Kaziakhmedov, Maedeh Zarvandi, Shourya Verma, Lukas Dubois, and Theodoros Katzalis. The friendships and experiences we built together, both in and beyond research, are invaluable and will always hold a special place in my heart.

I sincerely thank the EMBL Chinese community for fostering a warm sense of home abroad. Special thanks to Ling Wang, Jinhao Li, Yuchen Xiang, Ziqiang Huang, Shimin Shuai, Zhengyi Yang, Junyan Lu, Liangfu Xie, Linhua Tai, Shengdi Li, Shuchang Hu, Min Zhang, Jing Zhou, Xiaojuan Li, Xueying Li, Yuyao Song, Xiaohan Zhao, Yidan Jiang, Zhe Zhao, Wanlu Zhang, and Xiaojie Zhang for their companionship. I also warmly thank my tandem partner, Leon Buntz, for our engaging and thoughtful cultural exchange.

I thank all my teachers throughout my education, especially Prof. James Brotherston, Prof. John Shawe-Taylor, and Prof. Michael Sternberg, whose encouragement and recommendation made it possible for me to pursue research at EMBL. I am also grateful to the Cognitive Science Network at Heidelberg University for fostering interdisciplinary discussions that broadened my intellectual horizons. My sincere thanks to the EMBL Graduate Office, canteen, cafeteria, photolab, Human Resources, IT Services, Health and Safety, and Facility Management teams for their steadfast support throughout my time at EMBL.

As a seeker of beauty—in moments, places, and flavours—I will forever treasure our beautiful shared memories: COVID quarantine in Shanghai, moving house and the graduation ceremony in London, city walks in Milano, hiking in the Alpen, formal halls in Cambridge, fireworks and concerts in Paris, tulips in Amsterdam, a thunderstorm in Praha, a wedding in Düsseldorf, skiing in Beijing, brunch in Jona, the sea in Barcelona, safari in Maasai Mara, music in Salzburg, adventures in Amazonia, Oktoberfest in München, jazz in Berlin, an unforgettable encounter in Birmingham, cruise on the Thunersee, and magical days in Disneyland and Rulantica.

Finally, I extend my deepest gratitude to the European Molecular Biology Laboratory (EMBL) International PhD Programme (EIPP) for selecting and nurturing me, and to the German Research Foundation (DFG) for supporting the Research Unit FOR2581 Plant Morphodynamics, coordinated by Prof. Dr. Alexis Maizel and Prof. Dr. Kay Schneitz, which has been instrumental in my research. I profoundly appreciate the 29 EMBL member states, including Germany and the United Kingdom, for granting me privileges and immunities. My heartfelt appreciation also goes to China for its enduring care and support for international students and overseas Chinese communities. I completed my PhD peacefully, despite the challenges posed by Brexit, the COVID-19 pandemic, and global conflicts.

In the end, I thank myself for staying true to the path I chose—its course shaped by all of you, and its completion made possible by your presence.

致谢[★]

感谢你翻到此处，愿我的论文能为你带来启发或帮助。

衷心感谢我的父亲余若愚、母亲王先智[♡]，无条件的支持与信任成就了今天的我。恩重如山，无以为报。在一个淳朴、积极、开明的家庭中成长，三生有幸。感谢我的祖父母——余恩高、姜世英、王焕上、刘志秀，家族的传承与厚爱，为我的人生注入了勇气与温暖。愿我的弟弟余星辰永葆对世界的好奇心和对生活的热爱，潇洒自在地走好自己的路。

感谢我的导师安娜·克列舒克博士 (Anna Kreshuk)，给予我自由探索的空间与温暖友善的学术环境。感谢三位师兄阿德里安·沃尔尼博士 (Adrian Wolny)、洛伦佐·切罗内博士 (Lorenzo Cerrone) 和康斯坦丁·帕佩教授 (Constantin Pape) 的悉心指导与信任。感谢阿尔巴·迪兹-穆尼奥斯博士 (Alba Diz-Muñoz) 对我的认可与支持。

感谢合作者生物学家们慷慨提供珍贵的生物图像数据，也感谢论文指导委员会与答辩委员会的各位导师。感谢生物图像分析领域的同行，感谢你们的支持与接纳。感谢一路并肩的师兄师姐、师弟师妹、同事同学以及华人同胞，感谢你们的陪伴与支持。感谢我求学之路上每一位老师，尤其是给予我指导与推荐的恩师们。感谢那些与我一起走遍世界、记录瞬间、分享美食的朋友们。

感谢欧洲分子生物学实验室 (EMBL) 国际博士项目 (EIPP) 的培养，感谢德国研究基金会 (DFG) 支持植物形态动力学研究单元 (FOR2581)，为我的科研工作提供资助。感谢德国、英国等 EMBL 成员国所赋予的便利与保障；也感谢祖国对留学生与海外华人的深切关怀与支持。

感谢一路同行、给予我帮助和陪伴的每一个人；
也感谢那个始终坚持、乐观前行的自己，
走完了自己选择的路。

余沁

[♡] 王先智亦作王先芝。

[★] This page contains a short version of the acknowledgements in Chinese.

Preface

Thesis Structure: For readers new to bioimage analysis, it is recommended to start with chapter 1, which provides an overview of the field and contextualises my contributions. The remaining chapters can be read independently, though cross-references between chapters occasionally appear and are explicitly indicated.

References and Citations: Most web-based references, such as forum discussions and GitHub Pull Requests, are provided exclusively as footnotes. Peer-reviewed scientific literature with assigned DOIs is compiled in the main References section. Software tools cited in this thesis are listed separately in the Software References section. My own publications and software contributions are also collected in My Work for completeness and ease of navigation, although they appear elsewhere in their respective categories.

About the Author: At the time of writing, the author is a predoctoral fellow in the Kreshuk Group, Cell Biology and Biophysics Unit, EMBL Heidelberg, and a joint PhD student at Heidelberg University. He completed his undergraduate studies at University College London (UCL), where he pursued an interdisciplinary degree spanning computer science, mathematics, and statistics. His undergraduate thesis, supervised by Professor John Shawe-Taylor, was entitled *GPU Decomposition Support Vector Machines: Grid-synchronisations and Predicated Block-synchronisation in JuliaGPU*. He subsequently earned a master's degree from the Centre for Integrative Systems Biology and Bioinformatics (CISBIO) at Imperial College London, where his research, under the supervision of Professor Aubrey Cunnington, focused on *Analysing Neutrophil Gene Expression Changes Induced by Malaria Parasites and Other Stimuli*.

He was physically born in Guangzhou, Guangdong Province, China. However, due to an administrative oversight during initial registration, his legal birthplace is officially recorded as Hubei Province—a discrepancy that remains in governmental records to this day.

Photography License: All photographs included in this thesis were taken by the author with care and personal significance. They are licensed under the CC BY-NC-ND 4.0 license, which allows sharing with attribution while prohibiting commercial use and modifications. The images reflect personal moments, professional events, and quiet tributes from the author's time at EMBL.

Errata: Following the defence and prior to final submission to heiDOK (Der Heidelberger Dokumentenserver), two minor corrections were made: the page numbering in the front matter was corrected, and missing footnotes in Figure 3.4 were restored.

The Day I Joined EMBL | Oct 2020 | EMBL Heidelberg.

Chapter 1

Introduction to Bioimage Segmentation

Bioimage segmentation—the task of delineating cellular or subcellular structures in microscopy data—is central to advancing modern biological and biomedical research. Rapid progress in imaging modalities, ranging from confocal and light-sheet microscopy to electron and synchrotron-based X-ray imaging, has unlocked the ability to generate complex, high-dimensional datasets at increasingly high throughput. Yet, extracting meaningful biological insights from these data hinges on the development of robust and scalable computational methods.

At EMBL, my research has focused on addressing core challenges in bioimage segmentation, spanning three interconnected dimensions:

1. **Algorithm Development.** I contributed to the design and optimisation of segmentation algorithms, bridging classical image processing techniques and modern deep learning. Notably, I worked on *Sparse Object-Level Supervision for Instance Segmentation* (SPOCO) [1], which reduces annotation demands in instance segmentation tasks, and advanced multi-channel segmentation strategies across several projects, including the *GoNuclear* toolkit for 3D nuclei segmentation [2]. These contributions are detailed in chapter 4, section 2.3, and chapter 5.
2. **Software Engineering.** I co-developed and improved several open-source bioimage analysis tools, including *GoNuclear*, *PlantSeg 2.0*, and *Cellpose*. My work focused on ensuring that sophisticated segmentation algorithms are accessible, reproducible, and deployable in real-world biological workflows. Further information is provided in section 2.2, section 3.2, and subsection 5.2.2.
3. **Applied Bioimage Segmentation.** I applied segmentation techniques to a wide range of biological datasets and imaging modalities. Examples include planarian regeneration videos recorded with commercial cameras (subsection 5.2.1); 3D confocal microscopy of *Arabidopsis* ovules capturing cell walls and nuclei (subsection 2.3.1, section 5.3); 3D confocal volumes of mouse fibroblasts labelled for actin and synthetic linkers (subsection 5.1.1); mitochondria in mouse and human cells imaged via electron microscopy (section 4.2); and dynamic segmentation of cell membranes and nuclei in *Drosophila* embryo gastrulation with two-photon microscopy (subsection 5.2.2).

In what follows, I outline the broader field of bioimage analysis, highlighting its unique challenges and the need for specialised computational approaches. I then situate my own contri-

butions within the existing landscape of segmentation algorithms and software. It has been a privilege to collaborate on these projects, and I hope that the resulting tools and methods will continue to advance the field of bioimage analysis.

1.1 Bioimaging and Bioimage Analysis

1.1.1 Bioimaging Data

Modern biological imaging spans diverse scales, contrast mechanisms, and spatiotemporal resolutions. Researchers routinely customise microscopes—or devise new imaging protocols—to observe a vast array of samples: from entire organs to ultrastructures within single cells. Confocal, two-photon, and light-sheet microscopes generate volumetric and time-lapse data with optical sectioning; electron microscopy reaches nanometre-scale resolution; synchrotron-based X-ray imaging offers unique contrast for large samples; and even common commercial cameras capture morphological details of model organisms in motion.

This variety in instrumentation and experimental conditions produces *highly heterogeneous* datasets. Imaging runs can essentially become custom protocols, often collecting data in multiple dimensions (2D, 3D, multi-channel, time-lapse). Such diversity poses a core obstacle in bioimage analysis: resolution, contrast, and signal-to-noise ratio differ widely across laboratories and experiments. Any automated method must contend with heterogeneity in both imaging modality and biological content.

One might wonder how imaging modalities affect bioimage analysis. In the era of deep learning, diversity can paradoxically lead to scarcity of data for each experiment, and yet the bioimaging data itself is often treated simply as tensors. In practice, trade-offs also exist among imaging quality, photo-toxicity, financial constraints, and experimental design. A bioimage analyst must master the art of bootstrapping analyses under varying constraints: data quality, biological requirements, computational resources, and human effort. Sometimes one can help shape the imaging experiment; other times, data arrive as a given. Consequently, every bioimage project is a unique interplay between the biological question, the imaging modality, and computational methods. As a growing expert in the field, I continue to adapt to these challenges and opportunities.

1.1.2 Bioimage Analysis

Bioimage analysis comprises a broad set of computational techniques to extract meaningful biological information from microscopy data. Common tasks include:

- **Image Enhancement and Restoration** (e.g. denoising, deconvolution, resolution enhancement),
- **Segmentation** (delineating cytoplasm, nuclei, membranes, organelles),
- **Tracking** (following cells or subcellular compartments over time),
- **Morphological and Intensity Quantification** (measuring shape descriptors, intensity distributions, spatial organisation),

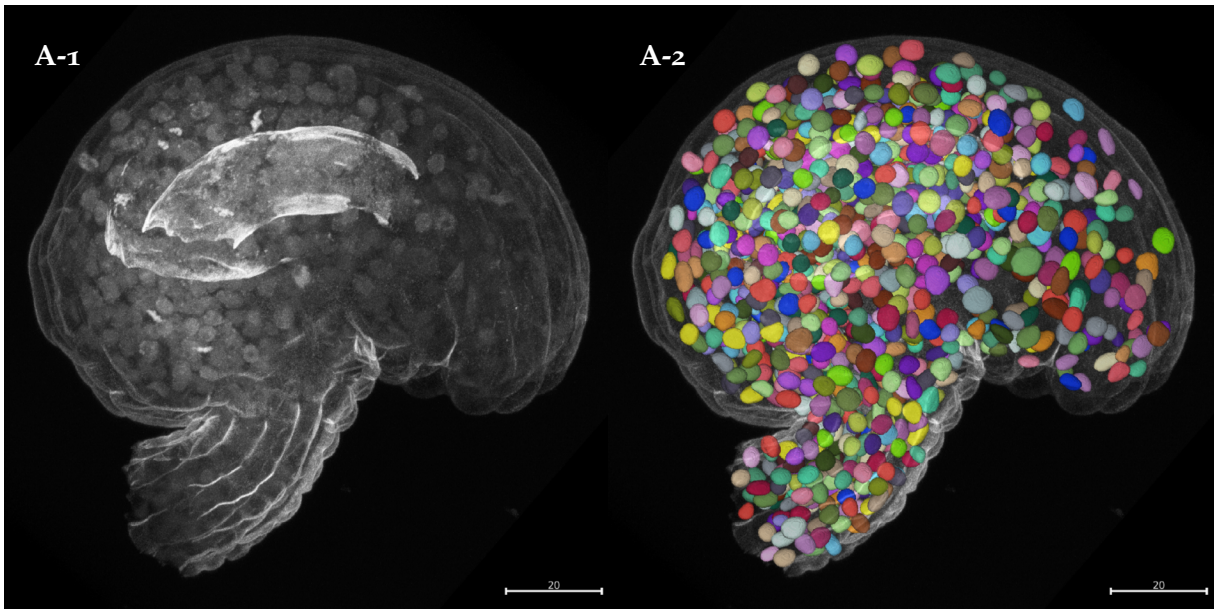


Figure 1.1: Bioimages across Scales 1/4: Confocal Microscopy of a Plant Organ. *Arabidopsis* ovules serve as an excellent model for constructing 3D digital organs, thanks to their immobile cells and well-defined architecture. Accurate segmentation of both nuclei and surrounding cells enables in-depth analyses of cellular and subcellular organization throughout development. (A-1) shows weakly stained TO-PRO-3 nuclei, underscoring the need for precise 3D instance segmentation to measure morphological features and nuclear-to-cell volume ratios (N/C), which vary across ovule tissues and developmental stages. (A-2) illustrates segmentation overlays from StarDist and U-Net; tools such as *GoNuclear* and MorphoGraphX further facilitate cell type-specific insights and quantitative mapping [2]. This figure is from chapter 2.

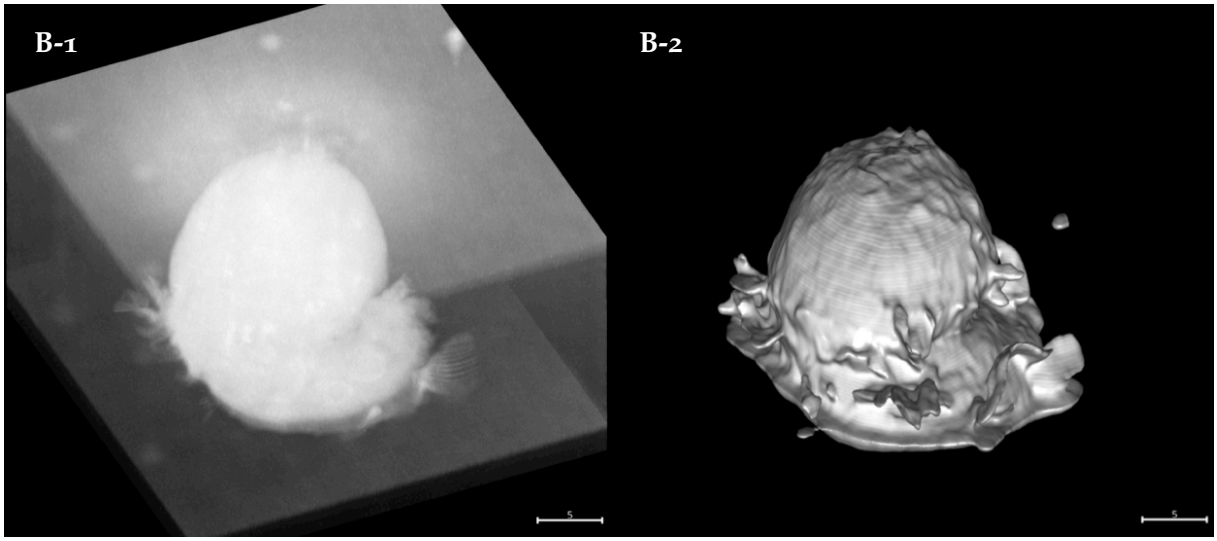


Figure 1.2: Bioimages across Scales 2/4: Confocal Microscopy of a Mammalian Cell. High-fidelity segmentation of the cell surface is critical for investigating membrane-cortex attachment, which governs processes such as shape regulation, migration, and division. Shown here are 3T3 mouse fibroblasts imaged with genetically engineered linker proteins that tether the actin cortex to the plasma membrane, enabling the study of linker length and membrane viscosity effects on actin organization. Due to substantial noise in the actin channel, conventional methods often fail to produce accurate segmentations. (B-1) depicts the negative (non-cell) channel, which provides a complementary noise profile and guards against overfitting during training. The resulting segmentation (B-2), generated by the "initial plus" deep-learning model, enhances surface delineation and underpins subsequent morphological and biophysical analyses of membrane-cortex coupling. This figure is from subsection 5.1.1.

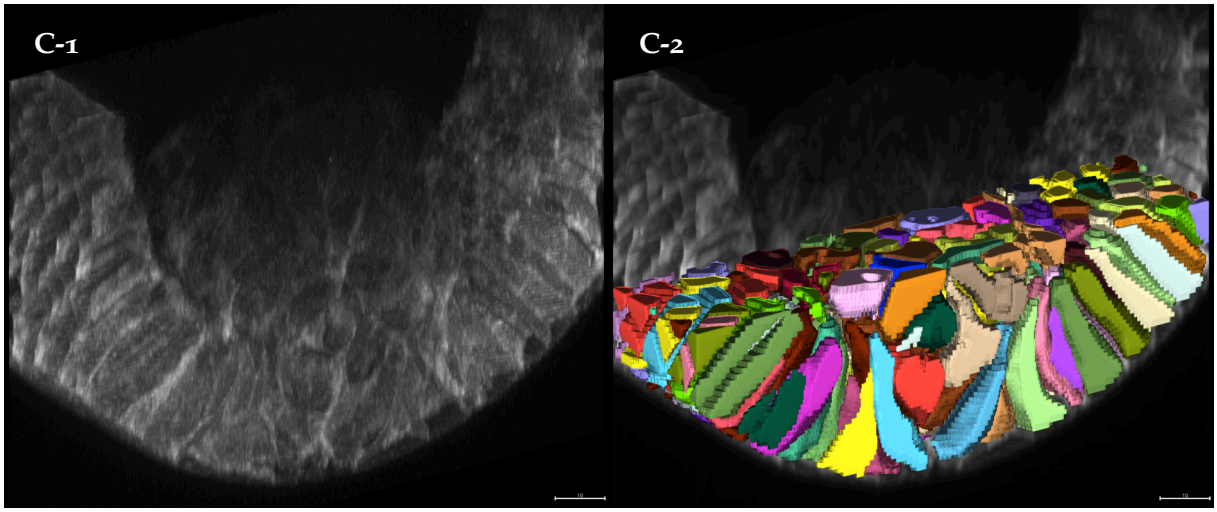


Figure 1.3: Bioimages across Scales 3/4: Two-Photon Microscopy Video of a Gastrulating Insect Embryo. Live imaging of *Drosophila* gastrulation reveals a rapidly changing cellular environment in which both membranes and nuclei must be segmented and tracked in 3D. (C-1) shows a two-photon microscopy slice of the membrane channel, illustrating pronounced noise and signal loss; (C-2) presents the final segmentation, achieved via a pipeline that integrates Noise2Void denoising, boundary prediction in PlantSeg, and Ultrack for tracking. This approach ensures temporally consistent, high-fidelity cell instance segmentation, aiding inquiries into whether nuclei supply mechanical energy to power tissue morphogenesis. This figure is from subsection 5.2.2.

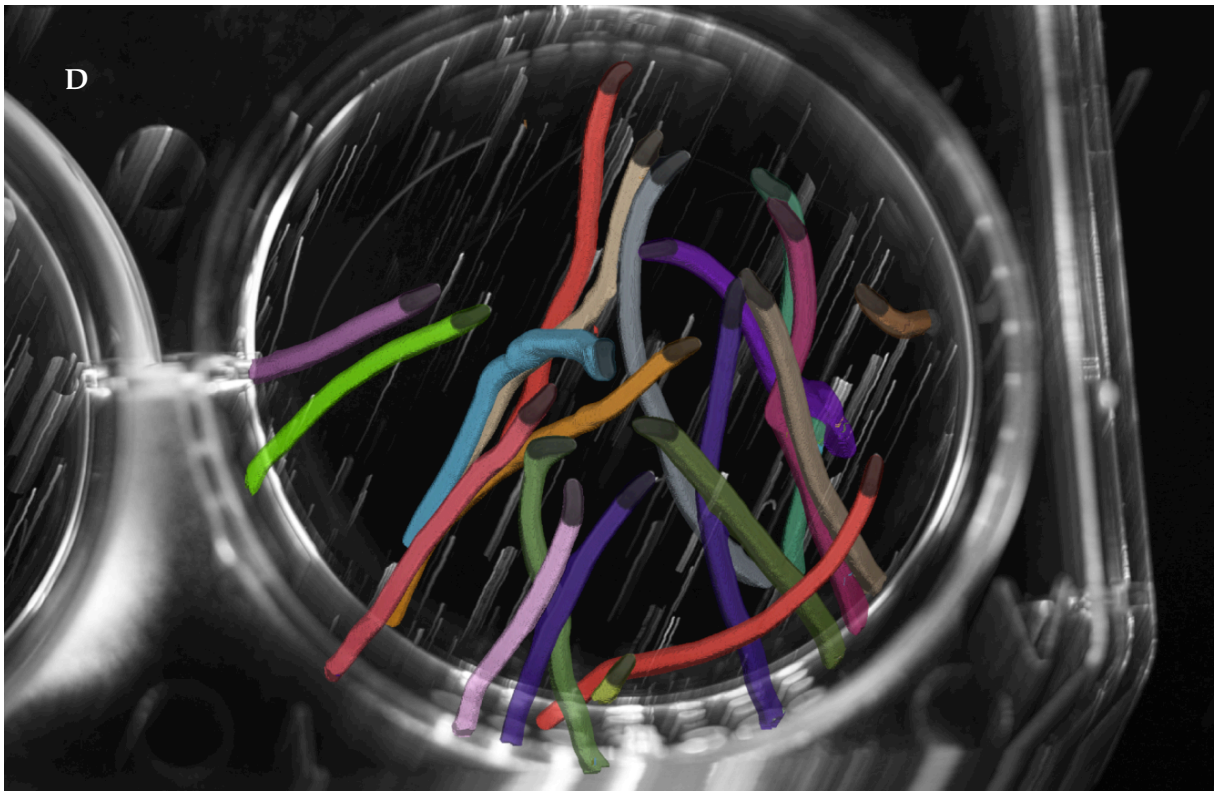


Figure 1.4: Bioimages across Scales 4/4: Camera Video of Multiple Regenerating Planarian Organisms. Time-lapse recordings of regenerating *Phagocata velata* fragments document distinct movement patterns linked to different regenerative strategies. In (D), the 3D visualization of the stacked frames reveals individually segmented fragments using a boundary-based volumetric approach. Transforming the temporal sequence into a 3D volume and applying standard segmentation methods (e.g. watershed and GASP) effectively reframes tracking as 3D instance segmentation. This technique enhances temporal coherence, simplifies fragment identity assignment, and supports quantitative evaluations of morphological changes and motion during regeneration. This figure is from subsection 5.2.1.

- **Registration and Fusion** (aligning images from different modalities, time points, or views),
- **Classification** (assigning biological labels to objects or image regions).

Among these, *segmentation* is often the pivotal first step, determining the spatial extent of biological structures for all subsequent analyses. Accuracy here underlies countless downstream applications, from quantifying cell morphology and tissue architecture to studying nuclear mechanics and multimodal data fusion (e.g. aligning fluorescence and electron microscopy volumes). Despite many advances, segmentation remains challenging due to noise, blur, heterogeneous backgrounds, and the diversity in shape and intensity profiles of biological structures.

1.2 Historical Perspective: From Handcrafted Techniques to Deep Learning

Segmentation in bioimaging has evolved substantially over the past decades. Early approaches relied on *handcrafted* methods such as thresholding, edge detection, and region-growing. Although sometimes effective, these heuristics often required extensive domain expertise and dataset-specific tuning. The advent of *classical machine learning* methods—notably Random Forest classifiers [3]—offered improvements over fully handcrafted pipelines by replacing manual rule design with trainable classifiers. Tools like *ilastik* [4] popularised this paradigm: users annotate a few pixels, and the system classifies the entire image based on engineered features (e.g. intensity, texture, edges). While such approaches still rely on feature engineering, they enable rapid annotation and interactive semantic segmentation.

The real turning point began around 2014–2015, when GPUs and open-source frameworks (e.g. TensorFlow, PyTorch) enabled training of *deep neural networks*. A defining moment for the bioimage community was the introduction of the U-Net architecture [5], which delivered end-to-end learning of hierarchical representations from relatively small datasets. This spurred an explosion of deep learning-based methods, rapidly outperforming traditional segmentation in many microscopy contexts.

Even so, the widespread heterogeneity of imaging protocols, along with limited availability of high-quality annotations, has compelled the community to develop domain-specific adaptations of deep learning, novel training strategies, and software frameworks specifically geared toward bioimaging.

1.3 Machine Learning for Bioimage Segmentation

1.3.1 Key Neural Network Architectures in Bioimage Analysis

In modern bioimage segmentation, two main classes of deep architectures predominate: *Convolutional Neural Networks (CNNs)* and *Vision Transformers (ViTs)*. These are frequently wrapped in an **autoencoder-style** encoder–decoder structure (e.g. U-Net) to handle high-dimensional input data.

Convolutional Neural Networks (CNNs). CNNs use learnable filters to capture local spatial relationships in images. For bioimage segmentation, models often adopt an encoder–decoder *U-Net* design [5], in which skip connections preserve high-resolution details. Variants such as *3D U-Net* handle volumetric data, and *residual* blocks can further refine performance. Many tools, including *PlantSeg* [6], *StarDist* [7], and *Cellpose* [8], employ CNN-based backbones, often augmented with specialised post-processing to convert semantic predictions into instance masks.

Vision Transformers (ViTs). First popularised in natural language processing, Transformers have gained traction in image tasks by replacing local convolutional operations with *global attention* [9]. ViTs require larger datasets to learn meaningful representations, as they lack the inherent inductive biases (such as locality and translation invariance) characteristic of CNNs. Within bioimage analysis, domain-specific ViT models—like µSAM [10]—have begun to appear, but the scarcity of large-scale 3D or multi-channel bioimage datasets, coupled with the computational cost of training ViTs on volumetric data, has limited their use primarily to 2D contexts.

In practice, many modern approaches combine features from both CNNs and Transformers, typically within an autoencoder-based framework that encodes data into a latent space and reconstructs at full resolution. This overarching structure continues to dominate bioimage segmentation pipelines. In my work, I primarily rely on CNN-based architectures for three reasons: first, the inherent inductive bias of convolutions aligns with the spatial nature of biological data and the goal of analysis tasks; second, the scarcity of large-scale, high-quality training datasets hampers the effective use of ViTs (section 1.6); and third, the computational demands of ViTs are often prohibitive for high-dimensional bioimaging tasks (section 5.2).

1.3.2 Learning Paradigms in Deep Bioimage Analysis

Deep learning methods can be categorised based on how they obtain supervision from data:

Supervised learning. Models are trained on *annotated* datasets with pixel- or voxel-level ground truth. A canonical example is U-Net, which requires manually segmented images for training. Tools such as ilastik, PlantSeg, StarDist, Cellpose, and µSAM all rely on supervised learning paradigms [4, 6–8, 10].

Self-supervised learning (SSL). Models extract training signals directly from unlabelled data. In bioimaging, popular SSL-based denoising methods include *Noise2Noise* [11], *Noise2Self* [12], and *Noise2Void* [13]. SSL has also gained traction in representation learning for morphological profiling, with methods such as *Cytoself* [14], *SubCell* [15], cellular MAEs [16], and *Cell-DINO* [17].

Unsupervised learning. These models uncover latent structures in unlabelled data. While purely unsupervised segmentation rarely achieves high-precision results, SSL methods are technically unsupervised but often evaluated against annotated ground truth. For example, MorphoFeatures [18] extracts rich morphological descriptors in an unsupervised manner. Some approaches like *Cell-DINO* [17] learn embeddings solely for exploratory analysis, but are still framed as self-supervised in practice.

Semi-supervised learning. A hybrid approach leveraging both labelled and unlabelled data, where a small annotated subset steers the model while unlabelled data refine its representations and act as regularisation.

Weakly supervised learning. This approach mitigates annotation bottlenecks by relying on coarse or partial labels—such as bounding boxes, scribbles, or object-level tags. *SPOCO* [1] exemplifies instance segmentation guided by sparse object-level annotations (chapter 4).

Active learning. A human-in-the-loop (HITL) approach where the model queries the most informative or uncertain samples for annotation. Although *GoNuclear* [2] does not implement active querying, it does incorporate HITL workflows: segmentation errors are reviewed externally by myself and put into auxiliary images that guide expert corrections (section 2.3). This pragmatic setup reduces annotation effort while incrementally improving performance.

Transfer learning and domain adaptation. Pretrained models from large-scale datasets (e.g. ImageNet [19]) are fine-tuned with a small number of labelled bioimage samples. This is especially beneficial when annotated data are scarce. For instance, Cellpose 2.0 [8] supports user corrections for fine-tuning, and PlantSeg 2.0 plans to adopt a similar feature. In SPOCO, I systematically explored transfer learning strategies for sparse instance segmentation (section 4.2).

Reinforcement learning (RL). Though rare in segmentation, RL has been used for tasks like landmark detection and interactive refinement. It operates by maximising rewards without explicit labels, but can be slow and unstable. Given these limitations, RL remains niche in bioimage analysis [20].

Over the course of my PhD, I mainly relied on *supervised* methods—further enhanced by transfer learning to accommodate heterogeneous biological experiments—and integrated weakly supervised strategies to reduce manual labelling. While *unsupervised* learning does appear in some contexts, it often needs even larger datasets to discern biologically meaningful structures without annotations. Moreover, without ground truth, it is difficult to confirm whether a model is capturing relevant biological features or simply reflecting technical artifacts. Thus, if a reliable set of annotations is already available, supervised methods generally offer a more straightforward and precise path to actionable results.

1.4 Semantic, Instance, and Panoptic Segmentation

Segmentation tasks can be categorised by the granularity of object delineation:

Semantic segmentation assigns each pixel (or voxel) to a predefined class (e.g. "cell," "background," etc.) but does not distinguish between individual objects of the same class. Figure 1.2 illustrates semantic segmentation of the cell surface in 3T3 mouse fibroblasts.

Instance segmentation identifies and labels each object individually (e.g. each cell or nucleus receives a unique ID), making it essential for object counting, morphological analysis, and cell tracking. Figure 1.1 and Figure 1.3 show instance segmentation examples in *Arabidopsis* ovules and *Drosophila* embryos.

Panoptic segmentation unifies semantic and instance segmentation by assigning every pixel both a class label and, if applicable, an instance identifier. In my experience, panoptic segmentation is uncommon in bioimaging. Biologists typically focus on specific structures of interest in light microscopy images, rather than exhaustively chemically labelling or digitally annotating all object classes. Including more classes increases annotation burden, often requiring significant expert effort. Moreover, boundaries between objects are frequently scale-dependent and context-specific, complicating consistent labeling across datasets.

Most of my research has targeted *instance segmentation*, where each cell, nucleus, or organelle must be distinctly identified. While semantic models are generally simpler, many biological questions do need to be investigated with instance-level knowledge. For example, understanding the morphodynamics of *Arabidopsis* organs or the interplay between cells and nuclei during *Drosophila* embryo gastrulation.

The Challenge of Instantiation in Deep Learning. A subtle but important challenge in deep learning-based instance segmentation is that *instantiating*, or explicitly separating, individual objects is not differentiable. In purely semantic segmentation, each pixel is assigned a class label, which integrates naturally into gradient-based optimisation. However, identifying discrete object instances typically requires non-differentiable post-processing.

SPOCO [1] addresses this by learning pixel embeddings that implicitly encode object boundaries, enabling a "soft" notion of objecthood and fully end-to-end training, but requiring clustering of embeddings to get instances. *Cellpose* [8] predicts spatial vector fields that guide each pixel toward the centre of its corresponding object, grouping pixels via flow integration. *StarDist* [7] instead predicts radial distances along fixed rays and reconstructs star-convex polygons or polyhedra around high-confidence object centres. *PlantSeg* [6] predicts boundary probability maps and applies graph-based partitioning algorithms to segment individual objects.

Each of these methods sidesteps the non-differentiability of instance enumeration through carefully designed representations and post-processing pipelines.

1.5 Challenges and Data Scarcity

Deep neural networks thrive on abundant labelled data, yet bioimaging commonly suffers from *data scarcity* in crucial ways:

- **Expensive Expert Input.** Generating pixel- or voxel-level ground truth in 2D, 3D, or even 4D requires substantial manual effort from domain experts—typically biologists involved in bioimage analysis. Reliable annotation often demands specialised knowledge, such as interpreting electron microscopy stacks, understanding the effects of specific chemical stains, or being familiar with cellular biology at microscopy scale, which significantly increases both the time and cost of dataset curation.
- **Inter-Lab Variability.** Differences in sample preparation protocols, microscope configurations, biological specimens, and research objectives hinder the standardisation needed

for large-scale generalisation. While repositories like the BioImage Archive [21] host vast collections of imaging data, the underlying heterogeneity remains a major challenge. Even relatively large datasets—such as LiveCell [22], TissueNet [23], and DeepBacs [24]—often fail to produce models that generalise to custom lab conditions. This is evident throughout this thesis: methods like Cellpose [25], Mesmer [23], and µSAM [10], despite being trained on diverse datasets, struggle with the real-world variability present in the bioimages I analyse.

In my case specifically, the mismatch is even more pronounced: I primarily work with multi-channel, volumetric data, whereas nearly all publicly available large-scale datasets are 2D. Although tools like Cellpose and µSAM are technically capable of processing 3D inputs, their architectures and training procedures are fundamentally based on 2D representations. Without genuine 3D training data, their performance on true volumetric segmentation remains inherently limited.

Researchers have responded with creative training paradigms to reduce reliance on dense labels:

- **Self-Supervised Learning.** Denoising methods (Noise2Noise [11], Noise2Self [12]) obviate the need for "clean" ground truth. Morphological feature learning (Cytoself [14]) extracts useful embeddings from unlabelled data.
- **Large Pretrained Models.** CNN-based *Cellpose* [8] and *Mesmer* [23] train on sizable, annotated datasets. Transformer-based frameworks like µSAM [10] employ prompt-based workflows for broad generalisation.
- **Weak and Sparse Labels.** *SPOCO* [1] achieves instance segmentation with partial or sparse annotations, reducing labeling burden.
- **Transfer Learning.** Models trained on one dataset are fine-tuned on another, with fewer annotated data. This is particularly useful when data are scarce or when the target domain differs from the source domain. I explored transfer learning strategies in SPOCO to improve instance segmentation performance (section 4.2).
- **Multi-Channel Exploitation.** Combining multiple imaging channels can improve segmentation accuracy. As shown in chapter 2 and chapter 5, cross-channel cues may reduce annotation overhead by providing complementary signals.

Given the variety of imaging contexts, data scarcity will persist. A guiding theme of this thesis is to leverage whatever data and annotations are available in the most effective way possible.

1.6 The Field of Bioimage Analysis

Although deep learning has become pervasive in biomedical research, *bioimage analysis* still occupies a relatively niche position compared to more standardised subfields like medical imaging (Figure A.6). For instance, Figure 1.5 shows that only 146 papers using the specific term "bioimage" (or "bio-image") appeared in 2024 on PubMed, whereas broader keywords (e.g. "biomedical image" or "bioimaging") yielded thousands of hits. This discrepancy arises

partly because microscopy-based research often appears under varied terminologies depending on the biological context, and partly because the core challenges of bioimage analysis differ significantly from those in clinical imaging.

In *bioimage analysis*, questions and datasets revolve around experimental, often custom, microscopy techniques applied to highly heterogeneous biological samples, from single proteins to entire organisms. By contrast, *biomedical image analysis* typically addresses standardised, regulated clinical modalities such as CT, MRI, and ultrasound. While methods can sometimes transfer between the two domains, they rarely do so without substantial adaptations. The heterogeneity of imaging protocols at institutes like EMBL, where each project has its own labelling strategy or microscopy modality, further complicates the development of universally applicable solutions.

This inherent variety also helps explain why there is no single, dominating approach to bioimage segmentation analogous to DeepMind's AlphaFold in protein structure prediction. During a visit to EMBL Heidelberg in 2022, Sir Demis Hassabis noted that problems ripe for large-scale deep learning typically satisfy three conditions:

1. A vast combinatorial search space,
2. A clearly defined objective function, and
3. Either an enormous annotated dataset or a high-fidelity simulator.

Although biological images are abundant, consistent *pixel-level* annotations are not, and the diversity of sample preparation and imaging protocols defies a single unifying framework. There simply is no large-scale universal simulator or database for arbitrary bioimage segmentation, and the number of unique imaging contexts is enormous.

In light of these challenges, I have focused on creating robust and flexible software tools and algorithms that can be adapted to various subdomains. This includes methods for sparse labelling and transfer learning to reduce annotation requirements (chapter 4), multi-channel and interactive segmentation for zero-shot scenarios (chapter 5), support for a wide range of staining and imaging conditions (chapter 2), and extending dense segmentation approaches toward more universal solutions (chapter 3). Such strategies embrace rather than avoid the inherent diversity of bioimaging tasks, facilitating broader applicability across different experimental setups.

1.7 Existing Segmentation Software and Limitations

1.7.1 A Spectrum of Generalisation

A range of open-source tools implement segmentation algorithms at different points on the spectrum from specialised to general-purpose:

ilastik [4] employs Random Forests in a user-friendly, interactive setup, but it does not include large pre-trained models. It excels at rapid annotation for simpler tasks but may

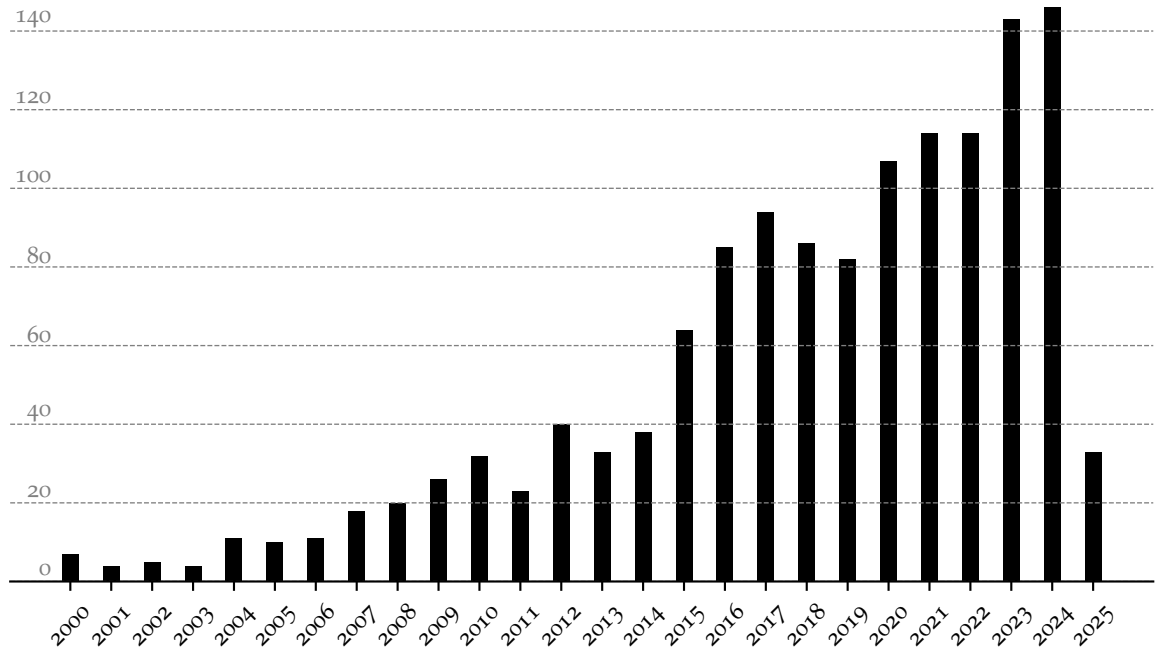


Figure 1.5: PubMed Bioimage Publication Counts (2000–Present). Counts include publications with the keywords "bioimage" and "bio-image." Statistics as of 23 March 2025.

struggle with complex data. Due to its "shallow" capacity, it is highly robust; however, it saturates quickly compared to "deep" methods.

Cellpose [8] relies on a U-Net-like CNN architecture that predicts both cell probability maps and directional flow fields from cell centers to boundaries, enabling robust instance segmentation. It comes with pre-trained models (e.g., for cells and nuclei) and allows user fine-tuning on new data.

StarDist [7] approximates objects with star-convex polygons (2D) or polyhedra (3D), excelling with roundish or potato-shaped structures such as nuclei. Its strong shape constraints, however, may limit performance on samples with irregular shapes or variable sizes.

PlantSeg [6] couples CNN-based boundary predictions with graph partitioning, making it well-suited for 3D volumes without explicit shape constraints. It provides pre-trained 3D models and can be fine-tuned or extended as needed.

µSAM [26] adapts a prompt-based "Segment Anything" approach for microscopy images. Users provide prompts (e.g., points, boxes, or polygons) to guide the segmentation. While it aspires to generalise across microscopy modalities, its performance often necessitates modality-specific fine-tuning, and user interaction remains crucial in many cases.

No universal panacea has emerged. Tools typically perform best on data resembling their training domain. The quest for broad generalisation is ongoing, often requiring complex post-processing or user feedback.

1.7.2 Open-Source Ecosystem and Ongoing Challenges

A notable strength of the bioimage analysis community is its culture of *open-source* development. Most methods are released with code, often in Python, and pre-trained models are increasingly common. However, many real-world issues—undocumented dependencies, label export bugs, channel misalignments—remain absent from polished publications. Applying these methods often requires "tinkering under the hood" to make them work in practice.

Throughout my PhD, I have:

- Led the development of PlantSeg 2.0, transforming it into a user-focused, interactive segmentation tool (section 3.2, section A.2).
- Created the *GoNuclear* project, releasing robust nuclear segmentation models for StarDist and PlantSeg (chapter 2).
- Integrated BioImage.IO support into Cellpose, PlantSeg, and StarDist (section 2.2), enabling FAIR model sharing and deployment.
- Identified and fixed critical bugs in *Cellpose*, *Ultrack*, and *pytorch-3dunet* (section 2.2, subsection 5.2.2).

Each of these tools targets slightly different user bases and application domains, but none works perfectly out of the box. The reliability and reproducibility of segmentation methods often depend less on the algorithms themselves than on how well they are implemented, maintained, and deployed. Iterative debugging and community-driven improvements remain essential.

It is not always as grand or sophisticated as it appears. *Cellpose* would mistakenly normalise cell signals using the nuclear channel and invert the wrong one; *StarDist*, despite its popularity for nuclear segmentation, lacked a publicly available 3D model; *Ultrack* export corrupted tracking results; *pytorch-3dunet*, one of the most widely used 3D U-Net implementations, introduced border artefacts due to incorrect input padding; *PlantSeg*'s proofreading tool allowed correcting AI errors but lacked an undo button for human mistakes; *ilastik*, the widely used interactive labelling tool, would lose annotations upon reopening; and *Segment Anything for Microscopy* (µSAM), a prompt-based tool, initially failed to segment anything from microscopy images when using point prompts. This was largely because the original *Segment Anything* codebase hides its training scripts. However, µSAM is now capable of segmenting with any box prompts from nothing. To make matters worse, some bugs are not even reproducible and require manual recovery.

It often feels chaotic: no software works exactly as advertised, and even their names can mislead. Nevertheless, I genuinely appreciate the efforts behind these tools and have enjoyed the years spent digging into their code. I personally identified and resolved most of the issues mentioned above, and I hope my contributions have helped make these tools more robust and usable for the community.

Beyond developing *GoNuclear* (chapter 2) and *PlantSeg 2.0* (chapter 3), I have contributed to several widely used open-source tools. My work on *Cellpose*, *StarDist*, and *BioImage.IO* is

detailed in section 2.2; my contributions to *PlantSeg* and *magicgui* in section 3.2 and section A.2; and my fixes for *Ultrack* in subsection 5.2.2.

Closing Remarks. Bioimage analysis stands at the intersection of computational innovation and biological discovery. As this thesis will demonstrate, methodological advances—grounded in domain knowledge and robust software—can push the limits of what is measurable and knowable from biological images. The following chapters will explore these developments in depth, addressing both the practical and conceptual challenges of modern bioimage analysis.

FOR2581 Young Scientists Retreat | Jun 2022 | Eibsee, Germany.

Chapter 2

GoNuclear[1]: A deep learning-based toolkit for 3D nuclei segmentation

TO-PRO-3 is a commercially available DNA-binding dye that can be directly applied to fixed and cleared plant tissues without the need for genetic transformation. This makes it especially valuable for 3D nuclear segmentation in both model and non-model species, including plants for which no transformation protocols exist. In contrast, genetically encoded nuclear markers such as H2B:tdTomato offer high signal clarity but require stable transformation and the generation of transgenic lines, limiting their applicability to a small number of species with well-established protocols. This practical distinction has significant implications for generalisability, motivating the development of segmentation models that can handle weak, noisy signals such as those from TO-PRO-3 staining.

Summarised in Figure 2.1, this study addresses key challenges in 3D nuclear segmentation by integrating biological insights with technical innovations. My contributions include:

1. **A high-quality 3D nuclear dataset with reliable annotations:** I leveraged algorithmic bias to guide the semi-automated annotation of a multi-channel dataset using a human-in-the-loop (HITL) strategy. This dataset provides high-quality training data for 3D nuclear segmentation in weakly stained tissues.

2. **Open-source software tools for training and inference:** I developed `run-stardist` [27], a scalable and accessible software for training StarDist [7] models and performing inference, enabling accessible workflows for researchers. Additionally, I co-developed *PlantSeg* 2.0 (chapter 3, [28]) and enhanced CellPose [29].

3. **Optimisation and benchmarking of 3D nuclear segmentation models:** I systematically explored the impact of key training configurations—including object size variations, 2D/3D rotation augmentation, voxel (an)isotropy, backbone architecture changes

[1]The GoNuclear study published in *Development* [2] was co-first-authored by me (Qin Yu) and two biology-focused collaborators, Athul Vijayan and Tejasvinee Mody. This chapter, while overlapping with that work, includes substantial unpublished content, omitting my co-authors' biological analyses and focusing on my methodological contributions. I did not perform wet-lab work, imaging, or morphological/biological analyses. However, I led the human-in-the-loop (HITL) strategy and was solely responsible for developing the software infrastructure, training deep-learning models, and conducting model evaluations. All model predictions, segmentations, and evaluation scores presented in the figures and tables—whether made in LaTeX or adapted from the paper—are my work. The text has been revised to accurately reflect my contributions.

in StarDist, and different 3D information merging strategies in CellPose—to refine training protocols. The resulting models were quantitatively assessed under these conditions and qualitatively benchmarked across diverse datasets and imaging modalities to identify their strengths and limitations in 3D nuclear segmentation.

4. **Two widely applicable deep learning models for 3D nuclear segmentation:** I provided robust, high-accuracy models that generalise well across diverse datasets, imaging techniques, and staining protocols from both plant and animal tissues. These models address key limitations in existing tools and offer reliable solutions for 3D nuclear segmentation.

Collectively, these contributions address fundamental challenges in data availability, software development, and model generalisation, establishing a scalable framework for accurate 3D segmentation and bioimage analysis. Moreover, this work enables downstream analyses of cellular and nuclear features in complex tissues, supporting advanced biological investigations such as nuclear-to-cell volume ratio quantification, spatial gene expression dynamics, and their roles in tissue morphogenesis and organ development using platforms like *MorphoGraphX* [30].

2.1 Introduction

2.1.1 Biological Motivation

An important practical motivation for this study stems from the need to perform nuclear segmentation in plant tissues without relying on transgenic lines. TO-PRO-3 is a commercially available DNA stain that is compatible with cleared plant samples and can be applied directly to fixed tissues, making it accessible across a broad range of species, including those without established transformation protocols. In contrast, genetically encoded nuclear markers such as H2B:tdTomato provide strong, uniform nuclear labelling but require stable genetic transformation and the generation of transgenic lines. These constraints limit their use to a few model species and restrict the generalisability of segmentation tools trained exclusively on transgenic datasets. Thus, models capable of handling weak, noisy signals—such as those from TO-PRO-3—are essential for expanding the applicability of 3D nuclear segmentation across diverse plant systems.

Tissue morphogenesis is a highly intricate, multi-scale process culminating in the formation of organs or tissues with specific sizes, shapes, and characteristic 3D cellular architectures. Recent advances in imaging and image processing technologies have enabled the development of 3D digital organ models at cellular resolution, particularly from fixed and cleared tissues. These models have proven invaluable in elucidating the feedback mechanisms between molecular regulatory circuits and cellular architecture during tissue and organ development. Plants serve as ideal systems for constructing 3D digital organs due to their immobile cells and clearly defined cellular structures, which are readily observable using advanced microscopy techniques.

Digital 3D models of plant organs have been instrumental in advancing our understanding of developmental processes such as embryo, root, and ovule development [31–41]. These models allow single-cell analyses in 3D and have yielded fundamental insights into plant

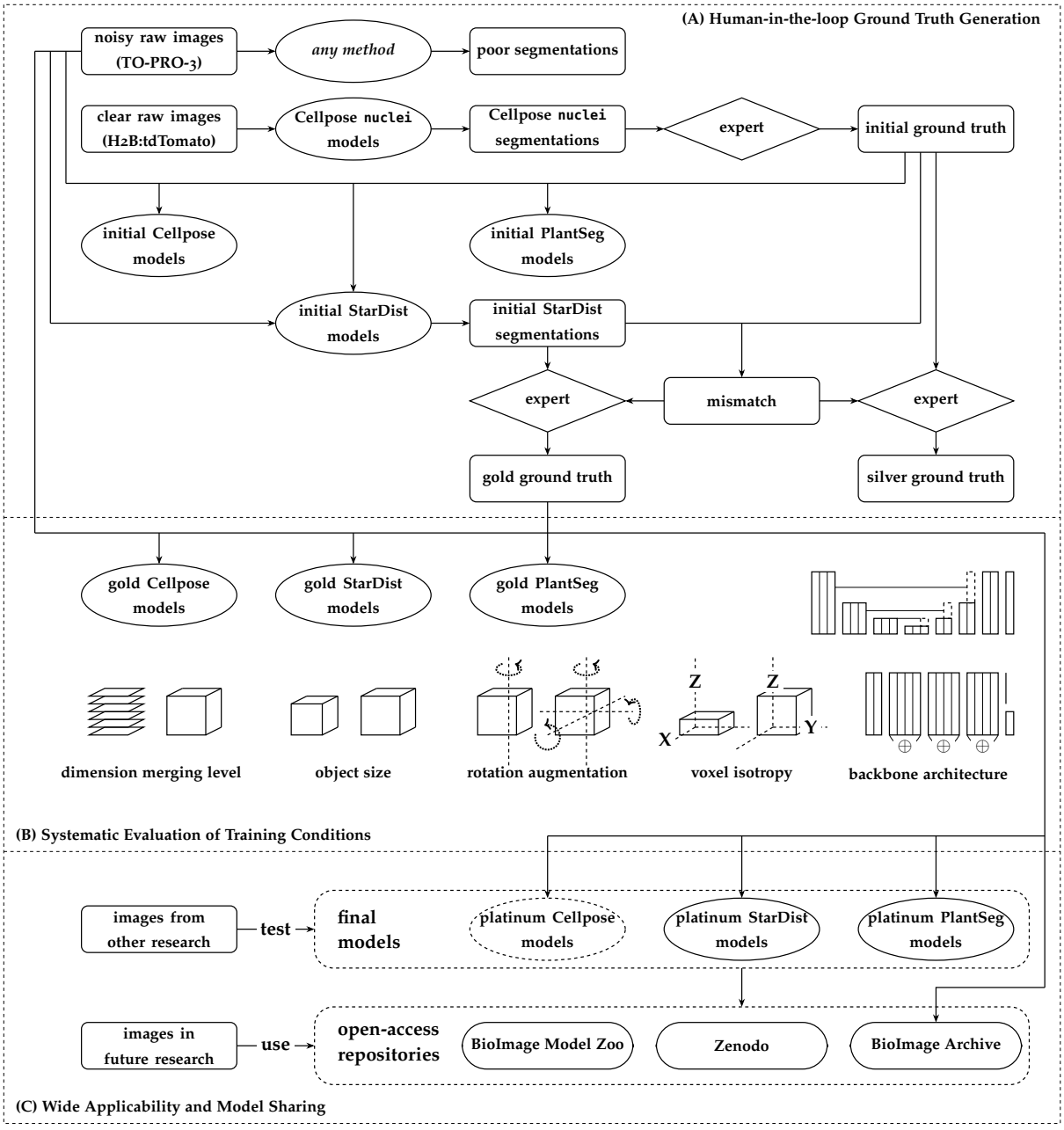


Figure 2.1: Overview of the GoNuclear Project. (A) human-in-the-loop (HITL) Ground Truth Generation section 2.3. The *noisy raw image* is initially unsegmented, while the *clear raw image* can be approximated using *Cellpose's* `nuclei` *model*. Biology experts proofread this segmentation to create an *initial ground truth* for training Cellpose, StarDist, and PlantSeg. *Initial StarDist models* improved upon errors in the ground truth but remained inadequate. I identified *mismatches* between predictions and targets, enabling experts to refine segmentations into *gold standard ground truth*, while *silver ground truth* was discarded, as only false positives and over-segmentation could be corrected. **(B)** Systematic Evaluation of Training Conditions. I trained Cellpose and fine-tuned its `nuclei` and `cyto2` models under two configurations: (i) 2D XY slice training with 3D stitching and (ii) isotropic XY, YZ, ZX training with flow prediction and merging. StarDist ResNet and PlantSeg were trained using (i) anisotropic input, (ii) isotropic input with Z-axis rotation augmentation, and (iii) isotropic input with 3D rotation augmentation. I explored multi-scale StarDist training to optimise *objectness*, while PlantSeg, being boundary-based, was not trained across scales. A final comparison between StarDist ResNet and U-Net was conducted using optimal configurations. All methods were rigorously evaluated using N-fold cross-validation. **(C)** Model Generalisation and Dissemination. I trained *platinum models* using the best configurations and full dataset, demonstrating robustness across diverse datasets, imaging modalities, and staining techniques in plant and animal tissues. Final models and gold ground truth were published on Findable, Accessible, Interoperable and Reusable (FAIR) platforms. I enhanced and extended all the aforementioned software to enable this project. Figure created by Q. Yu.

biology. However, a critical limitation of current 3D digital models is the absence of integrated nuclear size and shape information within the cellular framework. Such integration is crucial for studying biological processes like nuclear size control [42] and spatial gene expression regulation. For instance, analyzing the nuclear-to-cell volume (N/C) ratio provides important clues about cellular differentiation and functionality. Additionally, spatial gene expression patterns can be assessed with cellular resolution using ratiometric nuclear reporters driven by gene-specific promoters [43]. These insights underscore the need for robust 3D nuclear segmentation and the ability to link nuclear architecture with surrounding cells in tissue-specific contexts.

2.1.2 Technical Motivation

In recent years, significant advancements have been made in 3D cell segmentation using machine learning-based software such as CellPose [29], PlantSeg [6], StarDist [7], and others [39, 44–47]. These tools, combined with platforms like MorphoGraphX [30, 48], have enabled the creation of 3D digital organs with cellular resolution, offering powerful capabilities for tissue-level analyses. However, these computational pipelines face limitations when applied to challenging datasets, such as weakly stained or noisy nuclear signals commonly encountered in plant tissues.

Protocols based on the tissue-clearing reagent ClearSee have greatly facilitated tissue clearing, enabling the visualization of cell walls and nuclei using cytological dyes without requiring transgenic plants [49–52]. ClearSee is also compatible with fluorescent protein-based reporters, making it an indispensable tool for imaging deeper tissues. A notable application of ClearSee-based protocols is the development of a 3D digital reference atlas of *Arabidopsis* ovule development [40], where cell walls were stained with SCRI Renaissance (SR2200) [50, 53] and nuclei were labelled with TO-PRO-3 [54, 55]. Although this atlas provided detailed insights into the ovule's cellular architecture, it lacked integrated nuclear size and shape information. Furthermore, nuclear segmentation using TO-PRO-3 poses challenges due to variable staining intensity, scatter, and photobleaching in deeper tissue layers.

Existing machine learning-based segmentation tools are not readily applicable to weakly stained plant nuclei. Pre-trained models like those in PlantSeg [6] and CellPose [29] fail to generalise to such datasets, while 3D StarDist [7] requires training and lacks publicly available pre-trained models. The absence of high-quality 3D ground truth datasets, essential for training robust models, exacerbates the challenge. Annotation of these datasets is labor-intensive, even for high signal-to-noise ratio (SNR) images, creating a significant bottleneck in the workflow.

2.2 Developing Tools for Nuclear Segmentation

Instance segmentation of cell nuclei is a long-standing challenge in both biological and computer vision research. Advances in fluorescence, microscopy, and computer science have made imaging more accessible and creative. Yet, the ever-growing volumes and diversity of data still require robust computational tools for processing and analysis. During my PhD, deep learning methods have increasingly dominated the field of bioimage analysis, offering a wide range of software choices. The choice of tool depends on imaging modality, data quality, re-

quired analysis, and computational resources. In this section, I explain why I focused on three specific software frameworks—StarDist, PlantSeg, and Cellpose—outline my contributions to each, and describe the additional components I built to enable 3D nuclear segmentation in the GoNuclear study.

Despite the range of existing packages for bioimage analysis, at the time of this study (and likely still), no other 3D nuclear instance segmentation tools were available. Even recent large models, such as microSAM, remain limited to 2D segmentation. The computer vision and deep learning communities typically target natural images, which are high in signal-to-noise ratio (SNR), multi-channel (e.g. RGB), and two-dimensional, leaving a gap for specialised 3D instance segmentation in low-SNR biological data.

To address this shortfall, I assessed three prominent frameworks: StarDist, PlantSeg, and Cellpose. Their respective methods, along with other relevant software, are described in section 1.7. A core principle in scientific and engineering disciplines is to leverage existing solutions rather than reinventing them. Therefore, I opted to adapt these tools instead of creating entirely new methods. Each tool employs a deep learning approach based on U-Net-like architectures but differs in its segmentation strategy:

- **StarDist** provides a semantic foreground probability map and models each instance as a star-convex, "potato-shaped" object.
- **Cellpose** approaches segmentation as a vector field estimation problem, predicting the direction to the nearest instance center for each pixel. Objects of a predefined size are delineated from these vectors.
- **PlantSeg** focuses on enhancing instance boundaries, using them to partition the image into segments.

Each framework also has both theoretical and practical limitations:

- **StarDist** struggles with irregularly shaped or heterogeneous objects of varying sizes.
- **Cellpose** relies on a size-based object detection mechanism and does not offer a true 3D architecture, due to high computational demands for volumetric data.
- **PlantSeg** is tuned for dense segmentation, which complicates sparse nuclear segmentation. It tends to segment all tissue components, including cytoplasm and empty space.

Furthermore, none of these tools had been trained on data using the staining protocols adopted in this study, rendering their existing pre-trained models unsuitable for my datasets. However, deep learning-based methods can be retrained when annotated data are available. Thus, my collaborators and I took on the intertwined tasks of generating adequate training datasets and adapting or extending these software tools to meet the requirements.

section 2.3 covers how the initial training data were produced by my collaborators and how both data and software were optimised by me for the experimental setup, while section 2.5 details the final model architecture, training strategies, and associated experiments. The remainder of this section concentrates on my software engineering contributions that facilitated the GoNuclear study and improved accessibility for the broader research community.

2.2.1 Extending PlantSeg to Sparse Instance Segmentation

(All PlantSeg-related features discussed here are summarised in section 3.2.)

PlantSeg is a powerful platform for 3D instance segmentation, but its usability has historically been a bottleneck, especially for researchers with limited programming experience. Even users comfortable with coding often found it challenging to exploit all its features. My contributions primarily targeted improving accessibility and user-friendliness by: developing a graphical user interface (GUI) in Napari for interactive, real-time feedback (subsection 3.2.1); automating inference parameter selection (subsection 3.2.3); extending proofreading tools for segmentation corrections (subsection 3.2.4); creating modern documentation to guide new users through the software functionalities (subsection 3.2.6).

For GoNuclear, my most critical PlantSeg extension was implementing a nuclei-guided segmentation strategy (subsection 3.2.5). This involved enabling multi-channel output to support sparse instance segmentation, a feature not previously available in PlantSeg. I also integrated the BioImage Model Zoo into PlantSeg (subsection 3.2.2), allowing users to run models from different deep learning frameworks without leaving a single environment. Moreover, I experimented with combining predictions from different models; for instance, using StarDist's semantic foreground probability to refine PlantSeg-based segmentations. In future iterations, BioImage.IO may provide modular post-processing capabilities, with PlantSeg acting as an integration hub that unifies multiple segmentation pipelines and workflows.

2.2.2 Simplifying StarDist Training, Inference and Sharing

Introducing `run-stardist`. A key component of GoNuclear is the `run-stardist` package, a configuration-based command-line interface (CLI) for training and applying StarDist models. Although StarDist is often integrated into interactive environments like Jupyter Notebook or graphical user interfaces (e.g. Napari and ImageJ/Fiji), it initially lacked a built-in CLI.

Other tools, such as PlantSeg and Cellpose, include CLI support for training and batch processing, which is crucial for large-scale experiments on computational clusters without a graphical interface. To fill this gap, I developed `run-stardist`, allowing StarDist to be trained and applied via the command line, and making it straightforward to deploy on high-performance computing clusters.

Notably, StarDist (without this enhancement) has fewer than one thousand GitHub stars at the time of writing, whereas `pytorch-3dunet`, which supports batch processing, has more than twice as many. The popularity of command-line tools underscores the demand for automated workflows and large-scale data processing. For GoNuclear, `run-stardist` enabled rigorous model comparisons and systematic batch processing.

I designed `run-stardist` as a minimal wrapper around StarDist, taking inspiration from `pytorch-3dunet` but focusing on usability. Installation requires only one `conda` command, and both training and inference share a unified configuration file:

```
train-stardist --config CONFIG_PATH
predict-stardist --config CONFIG_PATH
```

It also integrates with Weights & Biases [56] for Machine Learning Operations (MLOps), facilitating scalable training and seamless collaboration. Supported input formats include Tagged Image File Format (TIFF) and Hierarchical Data Format version 5 (HDF5), covering most common uses; OME-Zarr [57] support is planned. Documentation can be found in [27].

Enabling FAIR Model Sharing. After successfully training and deploying models, it is important to share them. A principal outcome of my work in [2] was providing the first FAIR [58] 3D StarDist model to the broader bioimage community, offering a pre-trained solution that circumvents the need for new staining and imaging.

BioImage.IO is emerging as a standard for sharing models in bioimage analysis, but it remains technical for newcomers. To facilitate model sharing, I contributed to both StarDist and BioImage.IO, addressing two issues in StarDist and one in BioImage.IO.[2]

2.2.3 Integrating Cellpose with BioImage.IO

Cellpose is among the most popular tools for bioimage instance segmentation. Although its core models are 2D, they can be extended to 3D datasets by averaging predictions across multiple dimensions (X, Y, and Z). While general-purpose Cellpose models struggled with the challenging nuclear stain, the fine-tuned models I developed offer improved performance for specific scenarios. These models serve as a convenient off-the-shelf option for researchers who work in 2D or cannot take full advantage of the 3D "platinum" models from StarDist and PlantSeg.

Despite Cellpose's wide adoption, many scientists remain unaware of its existence. BioImage.IO aims to function as a central repository for discovering such tools, yet Cellpose was never included. I spearheaded the effort to integrate Cellpose into BioImage.IO, resolving technical issues that had previously deterred similar attempts. Although BioImage.IO has significant funding and an ambitious roadmap, its rapid development has led to multiple bugs and evolving specifications, complicating model integration.

The main problems I identified in BioImage.IO, which affected Cellpose support, included:

1. The neural network specification did not allow nested iterables, whereas Cellpose outputs a tuple containing tensors and a list of tensors.[3]

2. Ongoing changes to the specification resulted in inconsistent or misleading records.[4]

3. The verification process was cumbersome and prone to bugs, hindering reliable model validation across multiple formats.[5]

4. The upload workflow was flawed, preventing new model contributions.[6]

[2] GitHub – stardist/stardist/pull/254
[3] GitHub – bioimage-io/spec-bioimage-io/issues/615
[4] GitHub – bioimage-io/spec-bioimage-io/issues/598
[5] GitHub – bioimage-io/core-bioimage-io-python/issues/396
[6] GitHub – bioimage-io/bioimageio-uploader/issues/92

Additional complications arose from the interplay between Cellpose networks and the hardware.[7] Nonetheless, by collaborating with the BioImage.IO team, I resolved critical bugs and successfully released the first Cellpose model through BioImage.IO.

Beyond BioImage.IO integration, I made important modifications to Cellpose itself:

1. The `CPnet` class (Cellpose net) did not comply with BioImage.IO specifications; I extended its core class to standardise its output and refactored its weight-loading mechanism.[8]
2. Cellpose lacked export tools for BioImage.IO; I developed and documented this feature, positioning Cellpose as a long-term partner of BioImage.IO.[8,9]
3. I fixed intensity inversion and channel normalization bugs within Cellpose.[10]
4. I fixed the flow fields saving and improved storage efficiency for segmentation results by 99.5%.[11,12]

Through these developments, Cellpose has become more accessible and better aligned with emerging community-driven standards for bioimage analysis, providing researchers with a viable tool for a diverse range of segmentation tasks.

2.3 From Initial Data to High-Quality Ground Truth

This section corresponds to (A) in Figure 2.1. subsection 2.3.1 describes the starting point of this project and how my collaborators generated the initial dataset. subsection 2.3.2 details the iterative refinement process that I led, applying a HITL approach to improve segmentation accuracy.

2.3.1 Initial Data

As discussed in the technical motivation of this study, a previous effort to create a digital 3D reference atlas of *Arabidopsis* ovules [40] did not achieve instance-level segmentation of TO-PRO-3-labelled nuclei. This limitation arose from challenges such as inconsistent staining intensity, signal scatter, and photobleaching in deeper tissue layers, which hindered accurate nuclear segmentation. Initial attempts using PlantSeg's pre-trained models (`confocal_3D_unet_ovules_nuclei_ds1x` and `lightsheet_3D_unet_root_nuclei_ds1x`) failed to produce high-quality nuclear segmentations suitable for ground truth data. Consequently, Cellpose [8, 29] was employed for 3D nuclear segmentation due to its existing nuclei model. While Cellpose generated segmentation results, it exhibited significant errors in detecting and delineating nuclear boundaries, particularly for TO-PRO-3-stained nuclei (Figure 2.2 C). These issues were likely caused by the weak, diffuse nature of TO-PRO-3 staining and its variability,

[7] GitHub – bioimage-io/collection/issues/79, bioimage-io/collection/issues/84
[8] GitHub – MouseLand/cellpose/pull/988
[9] GitHub – MouseLand/cellpose/pull/1011
[10] GitHub – MouseLand/cellpose/pull/981
[11] GitHub – MouseLand/cellpose/pull/1103
[12] GitHub – MouseLand/cellpose/pull/1131

especially in deeper tissue layers. Moreover, the absence of signal in nucleoli created artifacts, with nuclei appearing as though their surfaces were extruded into holes (Figure 2.2 A).

To address these challenges, my collaborators employed advanced staining strategies and optimised sample preparation to establish a more reliable 3D ground truth dataset for model training. A transgenic *Arabidopsis* line was developed expressing a translational fusion of the fluorescent protein tdTomato to histone H2B, driven by the UBIQUITIN10 (UBQ) promoter (pUBQ::H2B:tdTomato). This transgenic line provided strong, uniform nuclear labeling. Ovules from the line were fixed, cleared, and triple-stained with TO-PRO-3 for nuclei, SR2200 for cell walls, and H2B:tdTomato for robust nuclear signals, producing three distinct imaging channels (Figure 2.2 A, B).

The strong H2B:tdTomato signal enabled segmentation using the Cellpose `nuclei` model. However, while the segmentation captured general nuclear features, it required expert proofreading to correct segmentation errors such as over- and under-segmentation and missed nuclei. These errors highlighted the inherent limitations of Cellpose in generalising to weakly stained, diffuse datasets (Figure 2.2 C). In contrast, the SR2200-labelled cell wall channel was segmented with high accuracy using PlantSeg's `generic_confocal_3D_unet` model (Figure 2.2 E), as this PlantSeg model was specifically designed for such tasks.

Although exhaustive corrections were impractical, the human-proofread nuclear segmentations from the strong H2B:tdTomato channel provided a robust basis for developing an initial ground truth. This dataset addressed challenges associated with weak TO-PRO-3 signals and variable nuclear features, forming the foundation for subsequent advancements in nuclear segmentation workflows. Details of the datasets are summarised in Table 2.1 and Table 2.2 and visually depicted in Figure 2.2 A, D. For detailed information on sample preparation and imaging, please refer to my collaborators' sections in [2]: *Plant work and transformation*, *Recombinant DNA work*, *Clearing and staining of ovules*, and *Microscopy and data acquisition*.

Dataset	Ovule ID	Stage	Cell/Nucleus Count	Voxel Size (*xyz* µm)
N1	1135	3-V	1118	$0.126 \times 0.126 \times 0.284$
N2	1136	3-IV	1487	$0.127 \times 0.127 \times 0.284$
N3	1137	3-V	1849	$0.126 \times 0.126 \times 0.284$
N4	1139	3-III	1536	$0.126 \times 0.126 \times 0.279$
N5	1170	2-II	3961	$0.126 \times 0.126 \times 0.279$

Table 2.1: GoNuclear Training and Testing Datasets. Datasets represent confocal 3D *z*-stacks of *Arabidopsis* ovules at various developmental stages. Each dataset is assigned an ID and a dataset number for reference in model training workflows, as outlined in Table 2.3. Each dataset contains three raw 3D volumes, referred to as 'channels,' detailed in Table 2.2.

2.3.2 Iterative Ground Truth Generation

Using the initial ground truth—human-proofread nuclear segmentations from the strong H2B:tdTomato reporter channel—as the target and the weak, noisy nuclei channel as input, I trained three sets of 3D segmentation models for Cellpose, StarDist, and PlantSeg. The goal was to identify the best-performing model for segmenting the weak TO-PRO-3-stained nuclei and ideally conclude the project without further iterations. However, as discussed in section 2.7, a more creative or at least strategic approach to training data selection could have

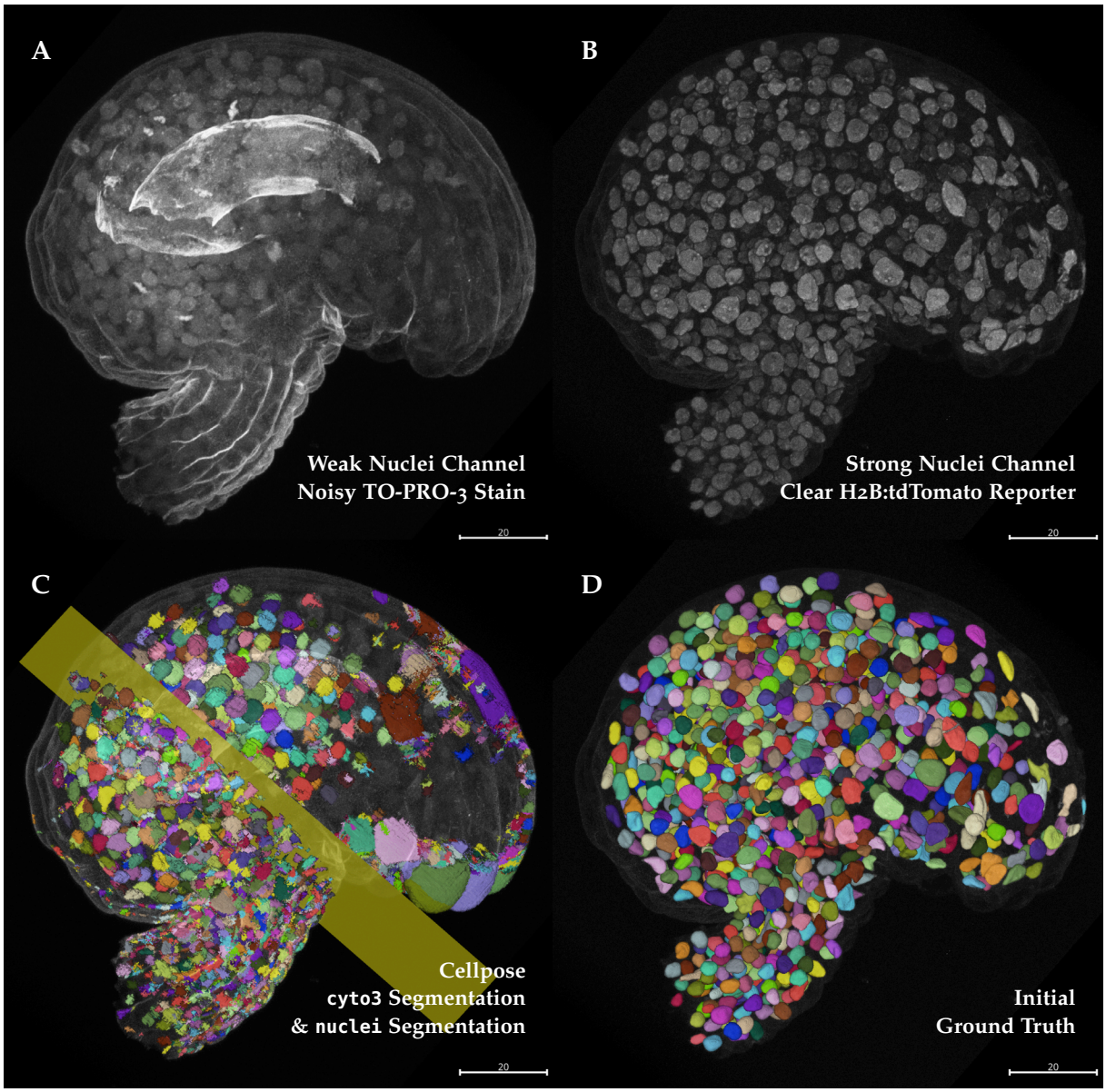


Figure 2.2: The GoNuclear Dataset 1/2: Initial Data. (A) A weak TO-PRO-3 channel with noisy nuclear staining. (B) A strong H2B:tdTomato channel providing clear nuclear signals. (C) Pretrained Cellpose models, `nuclei` (bottom) and `cyto3` (top), fail to segment nuclei accurately due to weak staining and variable signal intensity. (D) Initial ground truth segmentations generated using Cellpose `nuclei` applied to the strong H2B:tdTomato channel. Scale bars: 20 µm. Figure continues.

Raw Channel	Stain	Modality	Voxel Size (xyz µm)
`noisy` nuclei	TO-PRO-3	Confocal	$0.126 \times 0.126 \times 0.28$
`clear` nuclei	H2B:tdTomato	Confocal	$0.126 \times 0.126 \times 0.28$
`wall`	SR2200	Confocal	$0.126 \times 0.126 \times 0.28$

Label Channel	Software	Model	Voxel Size (xyz µm)
`initial` nuclei	Cellpose	`nuclei` on `clear`	$0.126 \times 0.126 \times 0.28$
`plantseg` cells	PlantSeg	`generic_confocal_3D_unet` on `wall`	$0.126 \times 0.126 \times 0.28$

Table 2.2: GoNuclear Initial Channels in *Arabidopsis* Ovule Datasets. The five datasets (N1-N5) listed in Table 2.1 each contain three raw channels: a `noisy` nuclei channel with weak signals from TO-PRO-3, a `clear` nuclei channel with strong signals from H2B:tdTomato, and a cell `wall` channel stained with SR2200. These channels are illustrated in Figure 2.2 A, B. Nuclei and cells are segmented into instances using respective software models: the `clear` nuclei channel is segmented with the `nuclei` model from Cellpose, resulting in `initial` ground truth (Figure 2.2 D), while the cell `wall` channel is segmented using the `generic_confocal_3D_unet` model from PlantSeg, producing highly accurate cell instances (Figure 2.2 E).

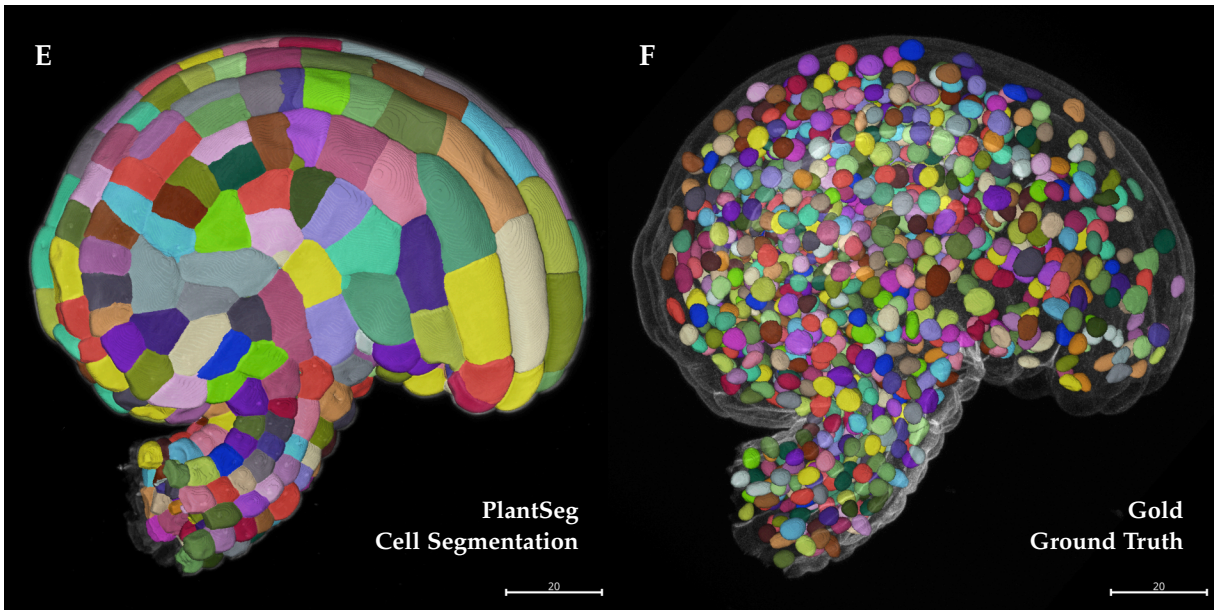


Figure 2.2: The GoNuclear Dataset 2/2: Final Ground Truth. (E) PlantSeg segmentation of the SR2200-labelled cell wall channel. (F) The final refined ground truth, derived from an initial StarDist ResNet segmentation of the noisy TO-PRO-3 channel. Scale bars: 20 µm.

been beneficial if further investigation had been anticipated.

Quantitative benchmarking against the initial ground truth was not meaningful, as it only measured discrepancies between models without indicating whether they represented improvements. Although I conducted an evaluation, it was not used for model selection. But the substantial mismatch between most models and the initial ground truth are obvious and suggested that the latter was likely insufficiently accurate. Therefore, I initiated an iterative HITL cycle to further refine the ground truth.

The next step involved selecting a model to generate segmentations for manual proofreading. Possible improvements could be applied either to the segmentation itself or to the initial

ground truth. The initial Cellpose models included fine-tuned `nuclei` and `cyto2` models, as well as models trained from scratch. The StarDist models were tested with both ResNet and U-Net backbones, while PlantSeg and StarDist models were compared under isotropic and anisotropic input conditions to ensure fairness with Cellpose. Notably, only Cellpose benefited from isotropic voxel sizes in this initial training phase, as explored in section 2.5.

Recognising that my StarDist models introduced segmentation improvements, I generated segmentations and provided them—along with auxiliary images and quantitative mismatch reports—to my biologist collaborators for proofreading. To streamline corrections, I identified segmentation instances that lacked 50% overlap with any instance in the initial ground truth, organising these discrepancies into Excel files and highlighting differences visually in image files.

Proofreading required balancing effort and accuracy, a trade-off that is difficult to quantify. While the manually corrected StarDist segmentations were not perfect, they reduced over-segmentation errors. StarDist, which converts semantic predictions into instance segmentations of smooth, star-convex objects, demonstrated robustness to errors in the initial ground truth. In cases where correcting the initial StarDist segmentation required less effort than fixing the original ground truth, I prioritised the corrected StarDist segmentations.

As a result, I discarded the re-proofread initial ground truth (the "silver ground truth") in favour of the manually corrected StarDist segmentations, which I designated as the "gold ground truth." This dataset served as the foundation for training final segmentation models under various conditions.

In section 2.5, I outline the exact training process, including dataset preparation, parameter selection, and model optimisation. Using the "gold ground truth" and the weak TO-PRO-3-stained nuclei, I trained multiple 3D models—including those implemented in PlantSeg, Cellpose, and StarDist—to identify the best-performing configuration.

Finally, two robust and widely applicable *platinum models* are proposed where all five datasets (N1-N5) were used for training final robust models: `PlantSeg_3Dnuc_platinum` and `StarDist-ResNet_3Dnuc_platinum`. I provide these two platinum models through the BioImage Model Zoo for FAIR use within different client tools of our community. For reproducibility, I also provide the full bundle of models I trained: *initial*, *gold*, and *platinum*, to be downloaded from Biostudies repository S-BIAD1026 (Table 2.4).

2.4 Quantitative Evaluation Strategy and Metrics

2.4.1 Evaluation Strategy: N-Fold Cross-Validation

n-fold cross-validation is a widely used technique in machine learning and statistics for evaluating model performance while making efficient use of limited data. It involves partitioning the dataset into *n* equally sized subsets (or "folds"), iteratively training the model on *n*-1 folds while testing on the remaining fold. This process is repeated *n* times, each time using a different fold for testing. By averaging performance across all *n* iterations, this method provides a more reliable estimate of a model's generalisation ability and reduces the risk of overfitting to a specific train-test split. The approach originates from classical statistical resampling

techniques, where it was introduced to improve model robustness by mitigating variance introduced by data partitioning. Compared to simple train-test splits, *n*-fold cross-validation ensures that every data point is used for both training and validation, thereby maximising the efficiency of limited datasets.

To systematically evaluate the segmentation performance of PlantSeg, StarDist, and Cellpose models, I employed a fivefold cross-validation strategy for both *initial* and *gold* training phases. Each iteration involved training on three images, validating on a fourth, and testing on the fifth. This configuration facilitated a robust assessment of segmentation quality across varying conditions and training regimes (Table 2.3).

5-fold scoring	Testing	Training 1	Training 2	Validation	Training 3
Model 1	N1	N2	N3	N4	N5
Model 2	N2	N3	N4	N5	N1
Model 3	N3	N4	N5	N1	N2
Model 4	N4	N5	N1	N2	N3
Model 5	N5	N1	N2	N3	N4

Final training	Validation	Training 1	Training 2	Training 3	Training 4
Platinum model	N1	N2	N3	N4	N5

Table 2.3: Fivefold Cross-Validation Training Strategy. N1-N5 represent image datasets used for training, validation, and testing. Each fold ensures a comprehensive assessment by rotating the testing dataset while training on the remaining samples. The final "Platinum model" is trained using all datasets except for one held-out validation set.

This evaluation strategy ensures a rigorous assessment of model performance, mitigating bias from any single train-test split. The final "Platinum model" was trained on all datasets except for a dedicated validation set, serving as the ultimate benchmark for segmentation quality. By adopting this systematic approach, I ensured the reliability and robustness of model comparisons across different training regimes.

2.4.2 Evaluation Metrics: Average Precision and Intersection over Union

To quantitatively assess and compare model performance, I use mean Average Precision (mAP) as the primary evaluation metric, following [59]. Given the diverse AP definitions in the literature [60], I ensure transparency by making the evaluation code publicly available in [27].

Intersection over Union (IoU), also known as the Jaccard index, quantifies the overlap between a predicted mask and the corresponding ground-truth mask. It is computed as the ratio of the intersection to the union of these masks and is expressed on a scale from 0 to 1. A value of 1 indicates a perfect match at the pixel level, while 0.5 signifies that the correctly matched pixels are equal in number to the sum of false positives and false negatives.

The precision of a segmentation result at a given IoU threshold t is defined as:

$$\text{precision}(t) = \frac{TP(t)}{TP(t) + FP(t) + FN(t)},$$

where:

- $TP(t)$ is the number of predicted objects correctly matching a ground-truth object with an IoU above t,
- $FP(t)$ is the number of predicted objects that do not correspond to any ground-truth object,
- $FN(t)$ is the number of ground-truth objects missing from the segmentation.

To obtain a comprehensive measure of segmentation quality, I compute the Average Precision (AP) over a range of IoU thresholds:

$$AP^{t_1:\Delta t:t_M} = \frac{1}{M} \sum_{i=1}^{M} \text{precision}(t_i),$$

where M is the number of IoU thresholds, spanning from t_1 to t_M in increments of Δt.

Unlike conventional AP calculations, which evaluate a single model over multiple images, my approach evaluates five models, each tested on one image, and averages their scores. Consequently, the mean AP (mAP) is computed as:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP^{t_1:\Delta t:t_M},$$

where N is the total number of images and models.

For evaluation, I use two primary AP metrics:

- **Detection Score**: The fivefold average precision at 50% IoU, denoted as mAP^{50}.
- **Instance Segmentation Score**: The fivefold average precision computed over the range $\{50\%, 55\%, \ldots, 95\%\}$ IoU, denoted as $mAP^{50:5:95}$ or simply mAP.

This approach ensures a rigorous assessment of model performance across varying levels of segmentation quality, encompassing both object detection and instance segmentation accuracy.

2.4.3 Evaluating Model Performance Gains from HITL

To assess the impact of HITL on segmentation performance, I compared models trained with *initial* versus *gold* datasets.

Model performance was quantified using fivefold AP scores (Table 2.5). With the experts' decision that the gold ground truth is indeed the better ground truth, the mAP measured big difference indicate that all segmentation methods greatly benefited from the HITL-enhanced gold training. PlantSeg and StarDist-ResNet outperformed Cellpose-Finetune-nuclei against the gold ground truth, demonstrating superior segmentation precision compared to their initial counterparts.

In total, the evaluation included 15 (3×5) initial models and 30 (6×5) gold models (Table 2.4). Performance metrics were computed for each, with mAP scores reported alongside standard deviations. The final results (Table 2.5 and Table 2.6) provide a comparative analysis of different segmentation strategies, illustrating the performance gains achieved through HITL effort.

For comprehensive performance quantification, detailed AP score breakdowns are available in Table A.4.

	Model name	Software	Algorithm	Version	Count
1	PlantSeg_3Dnuc_initial	PlantSeg	UNet GASP	Initial	5
2	StarDist-ResNet_3Dnuc_initial	StarDist	ResNet	Initial	5
3	Cellpose-Finetune-nuclei_3Dnuc_initial	Cellpose	Finetune nuclei	Initial	5
4	PlantSeg_3Dnuc_gold	PlantSeg	UNet GASP	Gold	5
5	StarDist-ResNet_3Dnuc_gold	StarDist	ResNet	Gold	5
6	StarDist-UNet_3Dnuc_gold	StarDist	UNet	Gold	5
7	Cellpose-Finetune-nuclei_3Dnuc_gold	Cellpose	Finetune nuclei	Gold	5
8	Cellpose-Cyto2_3Dnuc_gold	Cellpose	Finetune cyto2	Gold	5
9	Cellpose-Scratch_3Dnuc_gold	Cellpose	Train from scratch	Gold	5
10	PlantSeg_3Dnuc_platinum	PlantSeg	UNet GASP	Platinum	1
11	StarDist-ResNet_3Dnuc_platinum	StarDist	ResNet	Platinum	1

Table 2.4: List of Key Models. Model from each category—initial, gold, and platinum—are available for download from the BioStudies repository (S-BIAD1026).

Tool	Model	Version	mAP ± STD (5-fold)
PlantSeg	UNet GASP	Initial	57.40 ± 7.70
PlantSeg	UNet GASP	Gold	78.80 ± 1.98
StarDist	ResNet	Initial	67.61 ± 6.50
StarDist	ResNet	Gold	78.33 ± 1.73
Cellpose	Finetune nuclei	Initial	43.64 ± 12.88
Cellpose	Finetune nuclei	Gold	51.96 ± 12.51

Table 2.5: Performance comparison between initial and gold models. Mean AP values indicate improvements in segmentation quality after HITL refinement.

Tool	Model	mAP ± STD (5-fold)
PlantSeg	UNet GASP	78.80 ± 1.98
StarDist	ResNet	78.33 ± 1.73
StarDist	UNet	78.25 ± 1.84
Cellpose	Finetune nuclei	51.96 ± 12.51
Cellpose	Finetune cyto2	51.05 ± 12.93
Cellpose	Trained from scratch	51.26 ± 13.75

Table 2.6: Performance Comparison of Gold Model Training. The test dataset was segmented using each listed method, and the mean average precision (mAP) was recorded. All models were evaluated against the same 'gold ground truth' for consistency. Training configuration files are available alongside the model releases.

2.5 An Empirical Study of Training Conditions

This section corresponds to (B) in Figure 2.1. To identify optimal configurations for 3D segmentation of weakly stained nuclei, I systematically varied training inputs and parameters for PlantSeg, StarDist, and Cellpose. Specifically, I examined how (i) anisotropic vs. isotropic image resampling, (ii) rotation augmentations, (iii) dimensional "merging" in Cellpose, (iv) StarDist object-size considerations, and (v) StarDist backbone architectures affected segmentation outcomes.

Tables 2.7 and 2.8 summarise these experiments and highlight how specific training choices influence final model performance. Through this empirical study, I identified "sweet spots" for PlantSeg and StarDist in challenging 3D conditions, culminating in two final platinum models that offer broad applicability. My overarching goal was to clarify each tool's behaviour under these circumstances, pinpoint optimal configurations for segmenting weakly stained *Arabidopsis thaliana* ovule nuclei, and provide general guidelines for training robust 3D nuclear segmentation models.

2.5.1 Isotropy, Anisotropy, and 3D Augmentation

A key question in 3D nuclear segmentation is whether resampling images to isotropic voxel sizes or applying 3D rotation augmentation improves model performance. In this study, confocal stacks were acquired with anisotropic voxel spacing, where the z-axis is typically coarser than x and y. Some methods, such as Cellpose, recommend isotropic input for stable performance, whereas others, like PlantSeg and StarDist, can directly process anisotropic 3D data but may still be influenced by voxel anisotropy. Moreover, while 3D rotation augmentation might theoretically enhance orientation-invariant feature learning, it can introduce padding artifacts. To systematically investigate these factors, I trained and evaluated models under different training–inference pairings (e.g., original–original vs. isotropic–isotropic) and examined how each tool responded to voxel resampling and 3D rotation.

PlantSeg and StarDist: Isotropy and 3D Rotation. Because PlantSeg performs semantic boundary segmentation, it is relatively insensitive to the size or shape of individual nuclei. Training with the original anisotropic data consistently gave the highest segmentation accuracy (mAP $\approx$ 79%), whereas resampling to isotropic voxel sizes reduced performance by about 10% (Table 2.7). Similarly, StarDist obtained its best results ($\approx$ 78% mAP) when trained and tested on anisotropic data. StarDist requires carefully setting its `grid` parameter to match the typical object size to the network's receptive field; forcing isotropic resampling adds artifacts and disrupts this balance, lowering accuracy.

I also tested whether 3D rotation augmentation could improve generalisation in PlantSeg and StarDist. As shown in Table 2.7, there was no performance benefit. My interpretation is twofold: (i) at the object level, the training data already encompassed natural variability in nuclear orientation, making explicit 3D rotation redundant; and (ii) at the voxel level, applying 3D rotation after isotropic resampling requires padded boundaries, artificially distorting local features rather than improving model robustness. Consequently, I did not use 3D rotation augmentation in my final gold or platinum models.

Cellpose: Training, Isotropy, and Dimensional Merging Strategies. Unlike PlantSeg or StarDist, Cellpose is inherently a 2D model, even when applied across multiple orthogonal planes for 3D segmentation. Owing to its size-sensitive object detection mechanism, Cellpose requires isotropic input so that nuclei appear at a consistent scale across slices. I explored several strategies to determine the best way to train and apply Cellpose:

- **2D anisotropic slicing and stitching:** Training on *xy* slices in their original anisotropy and then stitching the 2D outputs into 3D; this yielded moderate performance.
- **2D isotropic slicing along all three axes:** After resampling the volumes to isotropic voxels, I prepared 2D slices from the *xy*, *yz*, and *zx* planes, which gave the best Cellpose results.

To study these setups in detail, I trained three classes of Cellpose models: (i) fine-tuned `nuclei`, (ii) fine-tuned `cyto2`, and (iii) models trained from scratch. Training inputs included original anisotropic *xy* slices or isotropic *xy*, *yz*, and *zx* slices, and inference was carried out by (i) segmenting 2D *xy* planes and stitching them, (ii) segmenting the raw 3D volume while specifying anisotropy, or (iii) segmenting manually resampled isotropic 3D volumes. Preprocessing volumes to isotropic voxel sizes for both training and inference raised Cellpose's mAP from 43.6% to 52% (Table 2.7), but accuracy still lagged behind PlantSeg and StarDist.

Notably, models fine-tuned from pre-trained weights and those trained from scratch achieved similar performance. Although only five volumes (four for gold models) were available, the thousands of 2D slices provided enough examples for effective training regardless of model initialization. Nevertheless, Cellpose could not match PlantSeg and StarDist performance, even though Cellpose was key in my initial HITL process.

Summary. These experiments show that resampling 3D data to isotropic voxel sizes is unnecessary—and can even be detrimental—for 3D-native methods like PlantSeg and StarDist. In contrast, Cellpose benefited from isotropic resampling but still fell short of PlantSeg and StarDist in final accuracy. Additionally, slicing along all three axes emerged as the best practice for Cellpose training, yet the method remains fundamentally 2D. Similarly, 3D rotation augmentation offered no improvements for PlantSeg or StarDist, reinforcing the advantage of training with native anisotropic volumes for segmenting weakly stained nuclei. Overall, these findings illustrate the importance of tailoring both preprocessing and augmentation strategies to each segmentation framework.

2.5.2 Optimising Scale, Backbone, and Patch Size

StarDist and Cellpose are both sensitive to object size due to their architectural and algorithmic constraints. Cellpose estimates a dynamics flow field, which depends on object size, while StarDist represents each nucleus as a star-convex shape of relatively fixed dimensions. In contrast, PlantSeg segments a semantic boundary class, which theoretically should be less sensitive to object size. However, my investigation beyond the scope of this chapter suggest that voxel size during inference can still influence PlantSeg's performance. Unlike Cellpose, which estimates object size dynamically during inference, StarDist does not include an explicit mechanism to adjust for varying object sizes in new inputs. Therefore, I systematically evaluated how object size affects StarDist performance.

Tool + Inference Condition	Training Isotropy	Inference Isotropy	mAP ± STD (%)
PlantSeg + GASP	Original	Original	78.80 ± 1.98
PlantSeg + GASP	Isotropic	Isotropic	68.25 ± 1.85
PlantSeg + GASP	Isotropic 3D Rotation	Isotropic	68.42 ± 2.03
StarDist ResNet + default	Original	Original	78.33 ± 1.73
StarDist ResNet + default	Isotropic	Isotropic	65.79 ± 0.93
StarDist ResNet + default	Isotropic 3D Rotation	Isotropic	62.78 ± 1.61
Cellpose `nuclei` + 2D stitched	Original XY Planes	Original XY Planes	26.80 ± 4.94
Cellpose `nuclei` + 3D isotropy	Original XY Planes	Original 3D	43.64 ± 12.18
Cellpose `nuclei` + 3D isotropy	Isotropic XY, YZ, ZX Planes	Original 3D	42.27 ± 10.56
Cellpose `nuclei` + 3D	Isotropic XY, YZ, ZX Planes	Isotropic 3D	51.96 ± 12.51

Table 2.7: Model performances under isotropic vs. original voxel dimensions. Segmentation accuracy for PlantSeg, StarDist, and Cellpose when trained and tested on data in either native anisotropic or resampled isotropic form. Mean AP values are reported against human-proofread ground truth.

Impact of Object Size on StarDist. StarDist segmentation accuracy is closely tied to how well the nuclei fit within the network's receptive field. If the nuclei are too large, segmentation accuracy decreases due to insufficient context within the field of view. As shown in Table 2.8, StarDist ResNet models trained and tested with input downsampled by $0.5 \times 0.25 \times 0.25$ achieved higher mAP than those using $1 \times 0.5 \times 0.5$. This aligns with expectations, as objects must be smaller than the field of view of StarDist's backbone to be segmented accurately. The U-Net backbone proved more robust when object sizes were mismatched, though the best absolute results came from ResNet with optimal input scaling.

Backbone Selection for StarDist. StarDist offers a choice of U-Net or ResNet backbones. My findings suggest that:

- **ResNet** often achieves higher mAP when nuclei size is tuned to the receptive field.
- **U-Net** is more robust to moderate object-size mismatches, yet it typically yields a slightly lower ceiling on mAP.

Since I could control the approximate scale of nuclei during training and inference, I opted for the ResNet backbone in the final "platinum" StarDist model (section 2.6).

Patch Size Considerations. Patch-based training is crucial for large volumes under limited GPU memory. My experiments showed that so long as each patch includes entire nuclei and enough context, StarDist and PlantSeg perform robustly. Patch size alone had less impact than ensuring consistent object size within the network's receptive field.

2.5.3 Performance of Gold Models

I performed both quantitative and qualitative evaluations of the resulting "gold" models. Table 2.6, Figure 2.3, and Figure 2.4 highlight their accuracy and depict representative instance masks.

Tool + Inference Condition	Patch Size	Scale Factor	mAP ± STD (%)
PlantSeg + GASP	192 × 368 × 368	Original	76.96 ± 2.29
PlantSeg + GASP	96 × 96 × 96	Original	78.80 ± 1.98
StarDist U-Net + default	96 × 96 × 96	0.5 × 0.25 × 0.25	78.25 ± 1.84
StarDist U-Net + default	96 × 96 × 96	1 × 0.5 × 0.5	75.79 ± 0.97
StarDist ResNet + default	192 × 368 × 368	0.5 × 0.25 × 0.25	77.63 ± 1.69
StarDist ResNet + default	96 × 96 × 96	0.5 × 0.25 × 0.25	78.33 ± 1.73
StarDist ResNet + default	192 × 368 × 368	1 × 0.5 × 0.5	67.32 ± 1.95
StarDist ResNet + default	96 × 96 × 96	1 × 0.5 × 0.5	69.70 ± 2.99

Table 2.8: Influence of patch size and voxel scaling on model performance. Comparing PlantSeg and StarDist with both U-Net and ResNet backbones under various patch sizes and scale factors. mAP scores and standard deviations are shown.

PlantSeg. PlantSeg produces a foreground prediction (a nuclear center map) and a boundary prediction (a nuclear envelope map), both visuallised in Figure 2.3 C. Although PlantSeg was designed for dense tissue segmentation, it can easily be adapted to sparsely labelled nuclei: The boundary prediction is segmented into superpixels by watershed algorithm and the superpixels are grouped by Generalised Algorithm for Signed graph Partitioning (GASP) [61] into an instance segmentation. The foreground prediction can be used in between watershed and agglomeration steps to filter out false positives in superpixels as well as to filter out false positives in the final instance segmentation. For further details, see subsection 3.2.5.

StarDist. StarDist-ResNet and StarDist-UNet predict a foreground probability (Figure 2.3 E,G) which is post-processed into an instance segmentation (Figure 2.3 F,H). By fitting star-convex shapes, these models produce smooth, uniform masks with excellent separation between nuclei. Elongated nuclei may shrink slightly in the probability map and can be under-segmented if the star representation underestimates their extent. In this project, I found StarDist-ResNet (after adjusting `grid` size) provided the cleanest segmentations and minimal merges. Minor over-segmentation or under-segmentation is still possible but easily resolved by manual split/merge corrections.

Cellpose. Because Cellpose is fundamentally 2D, it relies on isotropic preprocessing and a diameter parameter (default 30 for `cyto2` models, 17 for `nuclei`). Even when fine-tuned or trained from scratch, Cellpose performance remained lower than StarDist or PlantSeg, as shown by Figure 2.3 B and Table 2.6. Nevertheless, it proved invaluable in jump-starting the initial ground-truth annotations when no human-proofread dataset was available.

Summary of Gold Model Evaluations. With the exception of Cellpose, all gold models performed exceptionally well on weakly stained nuclei. They avoided artifacts such as extruded "nucleolar holes" and yielded higher mAP scores. Figure 2.4 also shows relatively small variance across the StarDist and PlantSeg cross-validations, confirming their robustness. In more challenging data, a simple combination of smoothing and well-chosen voxel scaling often sufficed to match the networks' expected size distribution. This gave a solid basis for producing the final, broader-coverage "platinum" models (see section 2.6).

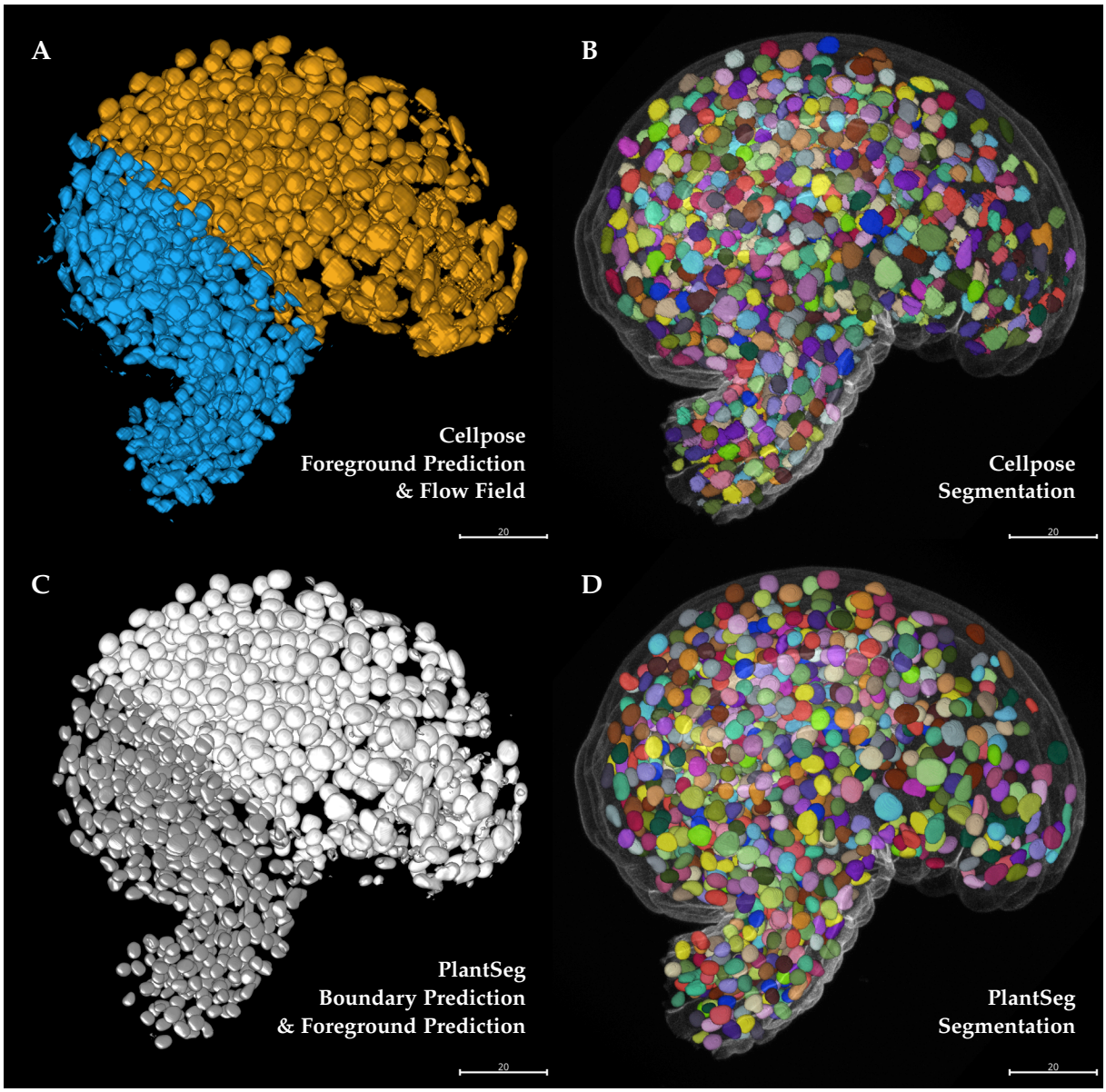


Figure 2.3: Qualitative Comparison of Gold Model Outputs (1/2). (A-B) gold Cellpose model: predicted flow field (blue, lower panel), semantic foreground (yellow, upper panel), and final instance segmentation. (C-D) gold PlantSeg model: boundary probability (white) and nuclear foreground probability (gray), plus final instance segmentation. Scale bars: 20 µm.

2.6 Widely Applicable 3D Nuclear Segmentation Models

Because `PlantSeg_3Dnuc_gold` and `StarDist-ResNet_3Dnuc_gold` emerged as the top-performing approaches, I used all five datasets (N1-N5) to train two final *platinum* models: `PlantSeg_3Dnuc_platinum` and `StarDist-ResNet_3Dnuc_platinum`. These 3D platinum models are released via the BioImage Model Zoo, and I provide the GoNuclear repository[13] for batch processing using either pipeline.

To confirm the broad applicability of these platinum models, I tested them on diverse, often challenging 3D nuclear datasets (different tissues, species, and imaging modalities). These

[13] `https://github.com/kreshuklab/go-nuclear`

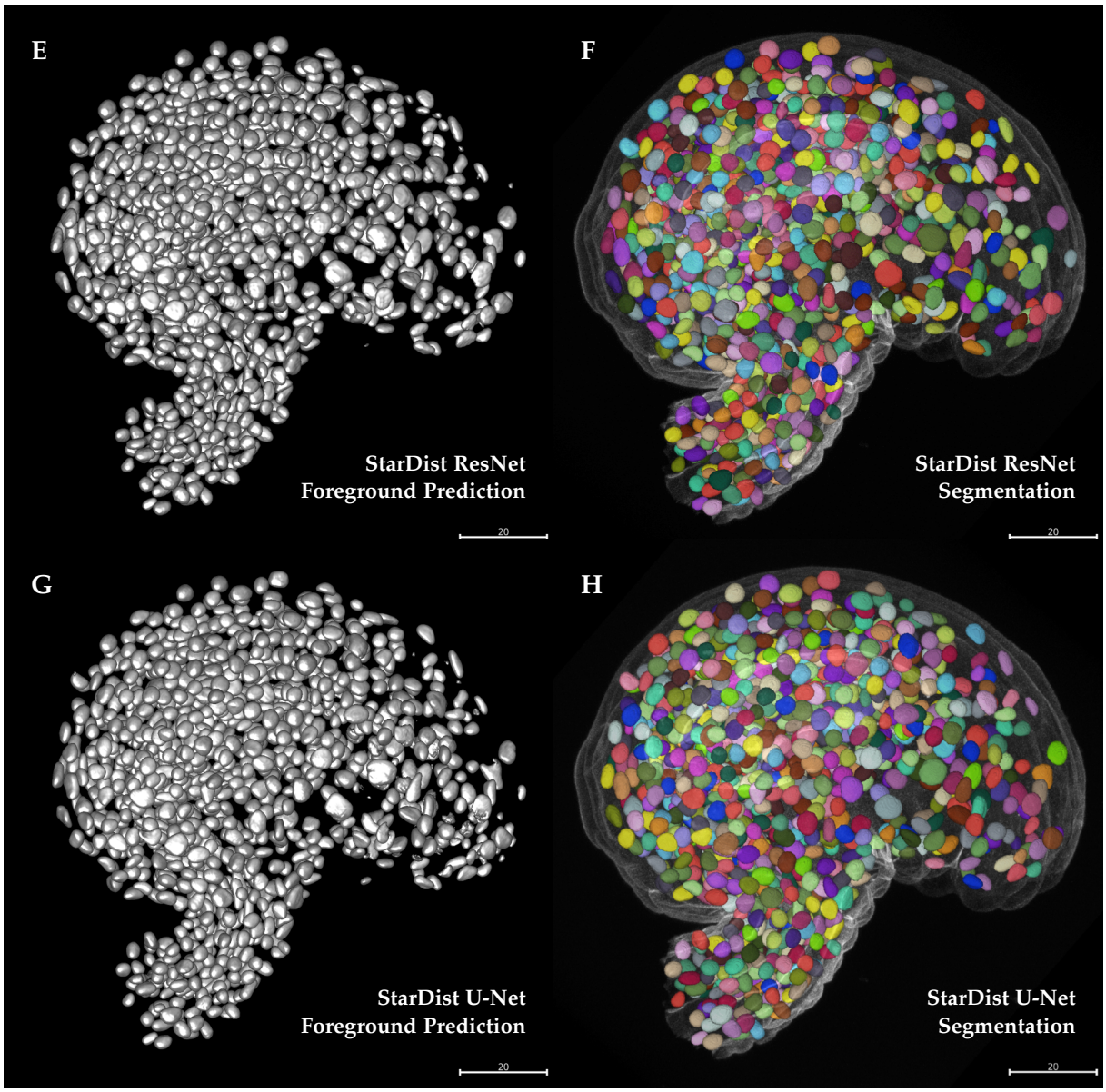


Figure 2.3: Qualitative Comparison of Gold Model Outputs (2/2). (E-F) gold StarDist ResNet: foreground probability and instance segmentation. (G-H) gold StarDist U-Net: foreground probability and instance segmentation. U-Net prediction is slightly noisier. Scale bars: 20 µm.

included:

- *Antirrhinum majus* ovule nuclei (TO-PRO-3-stained, cleared),
- *Arabidopsis thaliana* ovule nuclei (DAPI-stained, cleared),
- Live *Arabidopsis* sepal nuclei expressing pATML1::mCitrine-ATML1 [62],
- Live *Cardamine hirsuta* leaf nuclei expressing pChCUC2g::Venus [63],
- Fixed, cleared *Arabidopsis* shoot apical meristem (SAM) expressing pFD:3xHA-mCHERRY-FD [64, 65],
- A BlastoSPIM dataset of early mouse embryos (H2B-miRFP720-labelled, SPIM-imaged) [66].

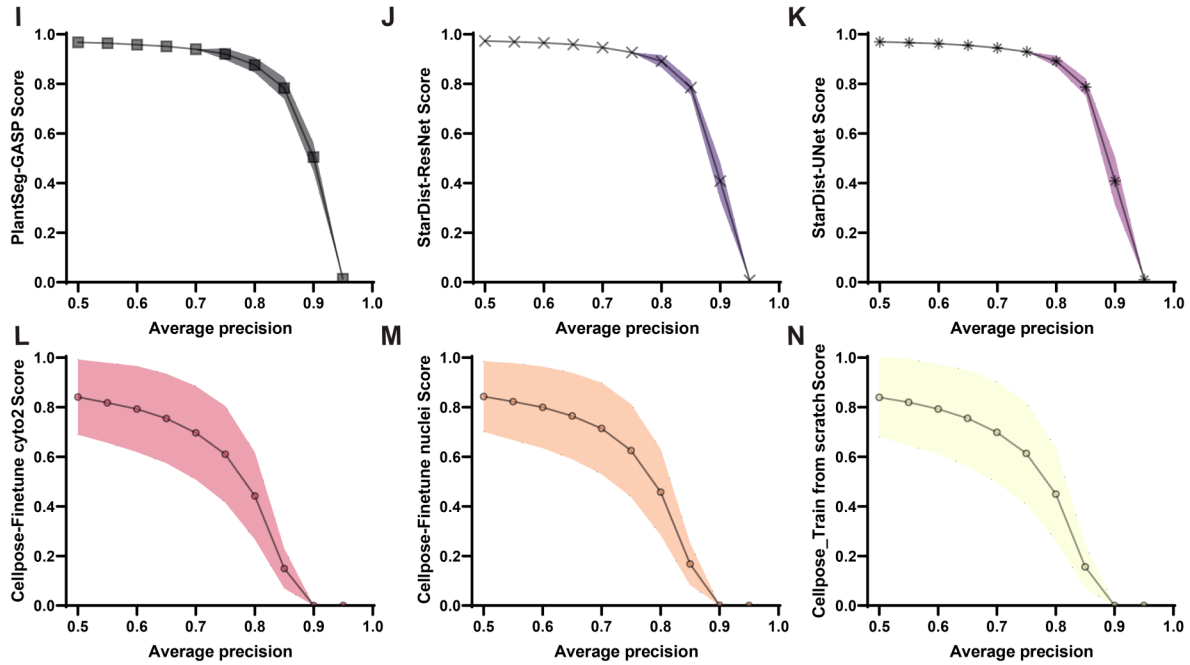


Figure 2.4: Cellpose comparison and average precision scores for all trained models. Fivefold mAP curves for PlantSeg (I), StarDist ResNet (J), StarDist U-Net (K), Cellpose `cyto2` (L), Cellpose `nuclei` (M), and Cellpose scratch (N). Shade indicates standard deviation. *Figure adapted from [2] Fig. S1. Values by Q. Yu. Layout by A. Vijayan.*

Both platinum models yielded comparable, high-quality results (Figure 2.5), even though these samples varied greatly in staining intensity, anisotropy, and resolution. Light preprocessing (e.g. mild smoothing, scaling) was often sufficient to bring the data into a domain compatible with the trained networks (Table 2.9). The nuclear segmentations provided an excellent foundation for subsequent morphological or fluorescence quantifications.

I segmented the above-mentioned datasets using the StarDist-ResNet and PlantSeg platinum models after image preprocessing (Table 2.9). The preprocessing was required to ensure the datasets to be segmented matched the training datasets in nuclear size and quality. I observed that the nuclei of all mentioned datasets could be properly 3D segmented using the proposed models (Figure 2.5). In addition, I could 3D segment nuclei of noticeable size differences in mature *Arabidopsis* leaf tissue with guard cell nuclei exhibiting a volume of $33.7 \pm 5.87\,\mu m^3$ (mean ± s.d.) and palisade mesophyll cell nuclei a volume of $92.7 \pm 13.92\,\mu m^3$. Further, even though the models were trained on cleared, high-resolution datasets, they are capable of segmenting nuclei from low resolution datasets as well, for example the *Cardamine* leaf nuclei and mouse embryo nuclei from live samples. A precise segmentation of the pChCUC2g::Venus nuclear signal further allows for quantification of the number of pChCUC2g::Venus-expressing nuclei along with signal quantification if required. The StarDist-ResNet platinum model could also segment more taxing datasets with high variation in intensities after applying some preprocessing. The results demonstrate the broad applicability of the platinum models in 3D segmentation of nuclei of different tissues and species.

Overall, the PlantSeg_3Dnuc and StarDist-ResNet platinum models show consistently high performance for 3D nuclear segmentation in diverse plant and animal datasets. By combining improved sample preparation, a carefully assembled training set, and advanced deep learning

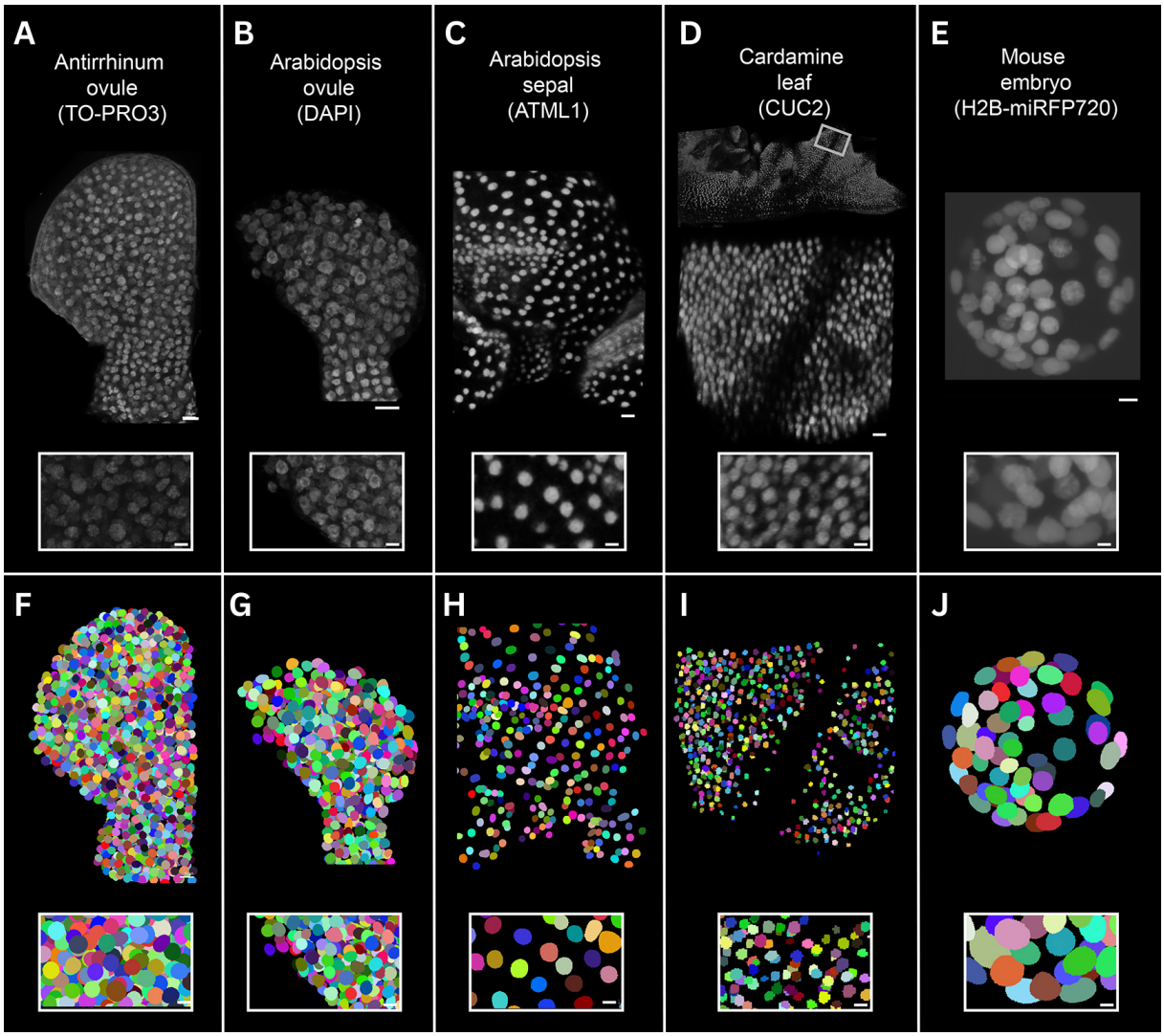


Figure 2.5: Wide applicability of the StarDist-ResNet platinum model across diverse plant and animal tissues. (A, F) *Antirrhinum majus* ovule nuclei stained with TO-PRO-3. (B, G) *Arabidopsis thaliana* ovule nuclei stained with DAPI. (C, H) *Arabidopsis* sepal nuclei expressing the pATML1::mCitrine-ATML1 reporter. (D, I) *Cardamine hirsuta* leaf nuclei expressing the pChCUC2g::Venus reporter. (E, J) Mouse blastocyst nuclei expressing the H2B-miRFP720 reporter. (A-E): raw images, (F-J): segmentation results. Insets highlight nuclear details. Scale bars: 10 µm (full organ views); 5 µm (insets). *Figure adapted from [2] Fig. 3. The sources of (A-E) are explicitly cited in section 2.6; (F-J) were contributed by Q. Yu. Figure layout by A. Vijayan and Q. Yu.*

frameworks, GoNuclear overcomes the challenges of weak or noisy staining and paves the way for comprehensive, high-throughput analysis of nuclear structure and function.

2.7 Outlook

The GoNuclear models have demonstrated strong performance in 3D nuclear segmentation across various biological datasets and imaging modalities. The GoNuclear GitHub repository, including the `run-stardist` package, has been fully documented and has attracted community contributions. The GoNuclear dataset will continue to be a valuable 3D training resource for future research, and the ground truth generation approach demonstrated in this chapter serves

Organism	Organ	Nuclear Stain/Fluorescence	Microscopy	Raw Data Voxel Size (µm)	Post-processing
Arabidopsis thaliana	Shoot apical meristem	pFD:3xHA-mCHERRY-FD	CLSM, 40x 1.25NA Gly objective, cleared sample	$0.242 \times 0.242 \times 0.4$	Median filtering
Cardamine hirsuta	Leaf	pChCUC2g::Venus	CLSM, 16x 0.6NA water dipping objective, live sample	$0.498 \times 0.498 \times 0.5$	Upsampled to $0.125 \times 0.125 \times 0.25$
Antirrhinum majus	Ovule	TO-PRO-3 iodide	CLSM, 63x 1.3NA Gly objective, cleared sample	$0.126 \times 0.126 \times 0.33$	Downsampled to $0.25 \times 0.25 \times 0.33$; smooth $2\times$
Arabidopsis thaliana	Ovule	DAPI	CLSM, 63x 1.3NA Gly objective, cleared sample	$0.063 \times 0.063 \times 0.27$	Downsampled to $0.25 \times 0.25 \times 0.28$; smooth $2\times$
Arabidopsis thaliana	Sepal	pATML1::mCitrine-ATML1	CLSM, 20x 1.0NA Water objective	$0.276 \times 0.276 \times 0.8$	Autobright; smooth $3\times$
Mouse	Early embryo	H2B-miRFP720	SPIM	$0.208 \times 0.208 \times 2$	Downsampled in x, y to $0.6 \times 0.6 \times 2$
Arabidopsis thaliana	Mature leaf 2	pUBQ::H2B:tdTomato	CLSM, 40x 1.25NA Gly objective, cleared sample	$0.15 \times 0.15 \times 0.25$	Median filtering ($1 \times 1 \times 1$)

(table continues)

Table 2.9: Datasets used to demonstrate broad applicability of the proposed 3D nuclear segmentation approach. Summaries of tissue/organ, label type, microscopic method, raw voxel size, and any preprocessing needed before applying the platinum models.

as a successful example of initiating image analysis from challenging conditions. However, several avenues for further research and development can enhance its applicability, robustness, and usability. This section outlines key directions for future improvements and extensions.

Expanding Benchmarking and Validation The original goal of providing the GoNuclear models was to enable morphodynamics studies for my plant biology collaborators. Fortunately, these models have proven to be useful across a wide range of biological datasets and have been published following FAIR principles. This ensures that future challenging 3D nuclear segmentation tasks worldwide can start from an established baseline rather than from scratch. However, the GoNuclear models have not yet been tested on public datasets in a quantitative manner. While they have already been validated within the plant biology community, additional benchmarking against publicly available nuclear segmentation datasets could help convince a broader audience of their utility. For instance, evaluating GoNuclear models on the Cell Tracking Challenge datasets [67] could provide a direct comparison against state-of-the-art nuclear segmentation tools across diverse imaging conditions and cell types.

Improving Training Strategies Since expert proofreading was required to refine segmentation results, each training iteration was conducted as if it were the final round. Ideally, if the initial ground truth had been sufficiently accurate—an unknown factor until model convergence and testing—the project could have concluded earlier. The GoNuclear models were always trained with the noisy, challenging nuclear channel as the sole input to ensure direct applicability to such data. However, had I anticipated the need for an iterative HITL strategy as illustrated in Figure 2.1 (A), I would have explored alternative input-output combinations to enhance the gold ground truth, even if those models were not directly applicable to the original nuclear channel.

Future improvements could explore:

- Training new nuclear segmentation models using a broader set of nuclear datasets, in-

cluding both plant and animal samples, to enhance cross-domain generalisation.

- Experimenting with different input combinations, such as the pixel-wise sum of multiple nuclear imaging channels alongside the gold ground truth, to improve signal robustness.
- Adopting a multi-channel training approach by incorporating all available imaging and ground truth segmentation channels to leverage additional information.
- Conducting hyperparameter tuning and architectural search to optimise network depth, receptive field size, and augmentation strategies.

Optimising StarDist for Large-Scale Volumes StarDist-ResNet has proven to be highly effective for 3D nuclear segmentation, but it faces challenges when applied to very large volumes due to hardware (random access memory (RAM)) limitations. A potential enhancement involves implementing an additional layer of stitching for extra-large volumes. Initially, during the development of `run-stardist`, I implemented a custom stitching approach with the assistance of Constantin Pape. However, this was later removed in favor of StarDist's built-in stitching, which was deemed sufficient for most cases. Revisiting this aspect could further improve scalability for large datasets.

2.8 Conclusion

This study presents GoNuclear, a deep learning-based toolkit for 3D nuclear segmentation, addressing key challenges in nuclear segmentation by integrating biological insights with technical innovations. My contributions include the development of a high-quality, annotated 3D nuclear dataset using a HITL approach, open-source software tools for model training and inference, optimisation of segmentation models through systematic benchmarking, and the provision of robust deep-learning models applicable to diverse imaging datasets.

By leveraging existing deep-learning frameworks such as StarDist, PlantSeg, and Cellpose, I developed workflow enhancements that extend the capabilities of these tools for weakly stained, noisy nuclear signals. In particular, the introduction of `run-stardist` facilitates scalable training and inference of StarDist models, while modifications to PlantSeg enable its application to sparse nuclear segmentation tasks. Additionally, integrating Cellpose with BioImage.IO improves model accessibility and deployment. These software contributions ensure that GoNuclear remains a widely applicable and easily deployable toolkit for researchers working on 3D nuclear segmentation.

A systematic evaluation of training configurations demonstrated that model performance is highly sensitive to voxel anisotropy, training augmentation strategies, and object-scale considerations. Notably, PlantSeg and StarDist-ResNet models performed optimally when trained with anisotropic voxel sizes, without requiring isotropic resampling or 3D rotation augmentation. In contrast, Cellpose benefited from isotropic input but remained constrained by its 2D nature, leading to lower segmentation accuracy compared to the other methods. My findings underscore the importance of tailoring preprocessing and training strategies to specific segmentation frameworks to achieve the best performance.

Through iterative refinement of training data and segmentation models, I generated two widely applicable 3D nuclear segmentation models: `PlantSeg_3Dnuc_platinum` and

`StarDist-ResNet_3Dnuc_platinum`. These models demonstrate high generalisation capabilities across various plant and animal datasets, encompassing different imaging modalities, tissue types, and staining protocols. Their robust performance, validated through extensive benchmarking and application to diverse biological datasets, highlights their potential for broader use in nuclear morphometrics, spatial transcriptomics, and tissue morphogenesis studies.

Looking ahead, future improvements could include expanding benchmarking efforts on publicly available datasets, refining training strategies by integrating additional imaging modalities, and optimising segmentation pipelines for large-scale volumetric imaging. Moreover, implementing adaptive model selection and dynamic inference strategies could further enhance the usability and scalability of GoNuclear for high-throughput biological applications. By providing accessible, reproducible, and scalable deep-learning models, this work lays a foundation for advancing 3D nuclear segmentation and facilitating quantitative analyses in complex biological systems.

Chapter 3

PlantSeg 2.0[1]: Powerful, User-Friendly Tissue Segmentation

3.1 Introduction

Quantitative analysis of plant morphogenesis demands precise 3D segmentation of cellular structures within complex volumetric microscopy datasets. Despite advances in imaging technologies, accurately delineating individual cells in densely packed tissues remains a formidable challenge. Traditional segmentation pipelines often struggle with balancing computational efficiency, algorithmic robustness, and usability. These limitations hinder the study of dynamic processes such as tissue growth, cell division, and mechanical interactions, where accurate geometric and topological representations are essential.

PlantSeg 1.0 emerged as a pivotal solution, combining convolutional neural network (CNN) with graph-based partitioning to achieve high-quality segmentations across diverse plant tissues and imaging modalities. Its modular architecture allowed flexibility in boundary prediction and agglomeration workflows, supported by a suite of algorithms including Multicut and GASP. However, reliance on configuration files and limited interactivity imposed technical barriers, particularly for researchers without computational expertise. Furthermore, manual parameter tuning for patch-based inference and the absence of integrated proofreading tools constrained reproducibility and scalability.

PlantSeg 2.0 addresses these challenges through a comprehensive redesign focused on usability, performance, and extensibility. Central to this update is the integration of Napari, an interactive visualization platform, which enables real-time parameter adjustment, multi-layer visualization, and direct proofreading within a unified interface. The software now supports multi-channel data processing, leveraging nuclear markers via GoNuclear to enhance cell segmentation accuracy through guided agglomeration. Integration with the BioImage.IO Model Zoo grants access to a growing repository of pre-trained models, ensuring compatibility with community standards and cutting-edge architectures. Underlying technical advancements—including automatic patch and halo size optimization, robust metadata handling, and

[1] "PlantSeg 1.0" refers to the first official release, version 1.0.1, which was published alongside the paper [6]. "PlantSeg 2.0" refers to the upcoming official release of version 2.0.0, which had not yet occurred at the time of thesis submission. "PlantSeg v1" encompasses all versions based on the configuration-file-based PlantSeg 1.0 framework, ranging from 1.0.1 to 1.4.3. "PlantSeg v2" includes all versions based on the Napari-based interactive framework introduced in PlantSeg 1.5.0, covering versions 1.5.0 through the pre-release versions 2.0.0a (alpha) and 2.0.0b (beta), up to the final 2.0.0 release. Versioning follows PEP 440 - Version Identification and Dependency Specification [68].

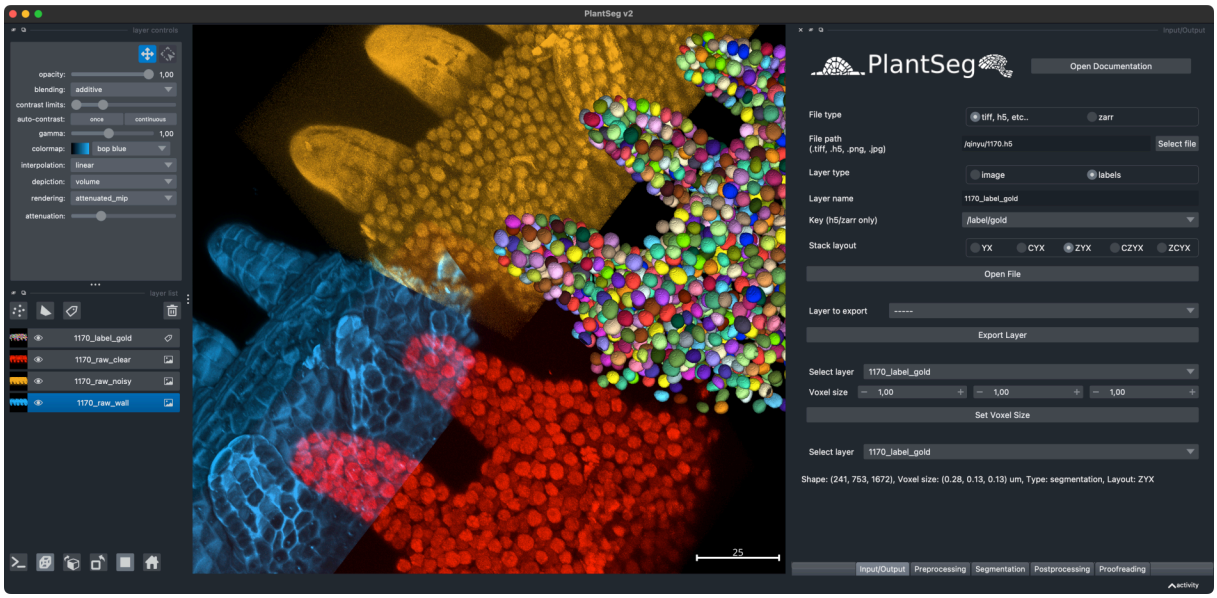


Figure 3.1: PlantSeg 2.0 - Interactive Napari Interface. Upon opening a hierarchical file format (e.g. HDF5 or Zarr), PlantSeg automatically detects available groups and datasets. Metadata such as voxel size is extracted and displayed for biologically scale-aware processing. The image appears in the Napari viewer for interactive exploration in 2D and 3D. **Left:** Layer list and controls. **Right:** PlantSeg widgets for on-the-fly configuration; results are shown in the central viewer. Nearly all figures in this thesis are screenshots from the PlantSeg 2.0 GUI running in Napari.

a refactored codebase—eliminate manual configuration bottlenecks while improving segmentation fidelity. Enhanced documentation and tutorials further lower entry barriers, empowering researchers to adopt advanced computational workflows with minimal overhead.

By bridging the gap between algorithmic sophistication and user-centric design, PlantSeg 2.0 establishes a new benchmark for accessible, scalable bioimage analysis. This chapter details the methodological and technological innovations driving these improvements, demonstrating their collective impact on developmental biology and broader applications in volumetric tissue segmentation.

3.1.1 Overview of PlantSeg 2.0

This section provides a walkthrough of a typical PlantSeg 2.0 workflow using the GUI, which can subsequently be exported into a configuration file for batch processing on distributed systems.

Launching the Interactive PlantSeg GUI: PlantSeg's GUI offers an interactive platform to monitor segmentation processes, providing immediate feedback and opportunities for trial and error. Users can visualize the output at each step, adjust parameters, and proceed accordingly. However, this interactive mode requires user presence to monitor and manually execute steps. Once the desired output is achieved, the GUI allows users to export a configuration file that records all steps and parameters, enabling headless mode execution for subsequent runs. To launch the GUI, execute the following command in the terminal: `plantseg --napari`.

Opening an Image File: Within the GUI, users can open image files using the "Open File" button. Supported file formats include TIFF, HDF5, and Zarr, with options for 2D or 3D (*volumetric*) images containing multiple channels. Upon opening, the image is displayed in the Napari viewer. The voxel size is extracted from the metadata of the file and stored in the PlantSegImage object to ensure scale-aware processing. Figure 3.1 shows the initial step of loading a dataset into PlantSeg.

Pre-processing Images for Prediction: Before running prediction models, users can enhance image quality through pre-processing. Available operations include Gaussian smoothing, rescaling, cropping, and image-pair manipulation. Parameters for these steps are adjustable via GUI widgets, with results displayed in the Napari viewer for inspection.

- **Rescaling** addresses voxel size differences between user images and model training data.
- **Cropping** removes irrelevant background regions to focus on areas of interest.
- **Image-pair operations** combine multi-channel images creatively, depending on the task.
- **Gaussian smoothing** enhances boundary and foreground prediction accuracy for certain neural networks.

For instance, smoothing may refine membrane boundaries, while rescaling adapts spatial resolutions to the model's requirements. Experienced users can just know what to do, while less experienced users can rely on the interactive graphical interface to explore the effects of different preprocessing and parameters on the images. Then they can choose the best parameters for their dataset and batch process the rest of the images once the workflow is finalised and exported as a configuration file.

Predicting Boundaries from Images: At the core of PlantSeg lies the boundary prediction step, where deep learning models delineate object boundaries. These boundaries serve as input for subsequent image partitioning. While raw images can be directly used as boundary maps if they exhibit sufficient contrast, the prediction step enhances boundary visibility or generates auxiliary representations such as foreground probability maps.

PlantSeg supports a range of pre-trained models, including 2D- and 3D-UNets, and integrates seamlessly with external frameworks like PyTorch and TensorFlow via compatibility with the BioImage Model Zoo. Additionally, PlantSeg extends its support to specialised models such as StarDist and Cellpose, enabling diverse outputs like foreground probability maps (as in StarDist) or flow maps (as in Cellpose). Users can select models via the prediction widget, which automates model loading and determines optimal halo and patch shapes. Inference is executed on user-specified hardware, and the prediction results are visualised in the Napari viewer, allowing flexible downstream usage such as generating binary masks or instance-specific outputs. This modular design empowers users to tailor prediction outputs to their segmentation objectives, ensuring adaptability across various imaging modalities and sample types.

Segmenting Images Using Boundaries: Predicted boundaries, or any high-contrast boundary images, can be transformed into instance segmentations through PlantSeg's two-stage process, comprising the "Boundary to Superpixels" watershed step and the "Superpixels to Instance Segmentation" agglomeration step. The watershed algorithm [69], leveraging the distance transform of the boundary predictions, first segments the image into superpixels. Subsequently, an agglomeration method, such as Generalised Agglomerative Clustering of Signed Graphs (GASP) [61], Mutex Watershed [70], or Multicut [71], merges these superpixels into final instances. The selection of the appropriate segmentation strategy depends on dataset characteristics, with Multicut providing globally optimal partitions and GASP offering computational efficiency. By combining robust boundary detection with graph-based partitioning, PlantSeg ensures accurate cell segmentation across diverse datasets and imaging modalities.

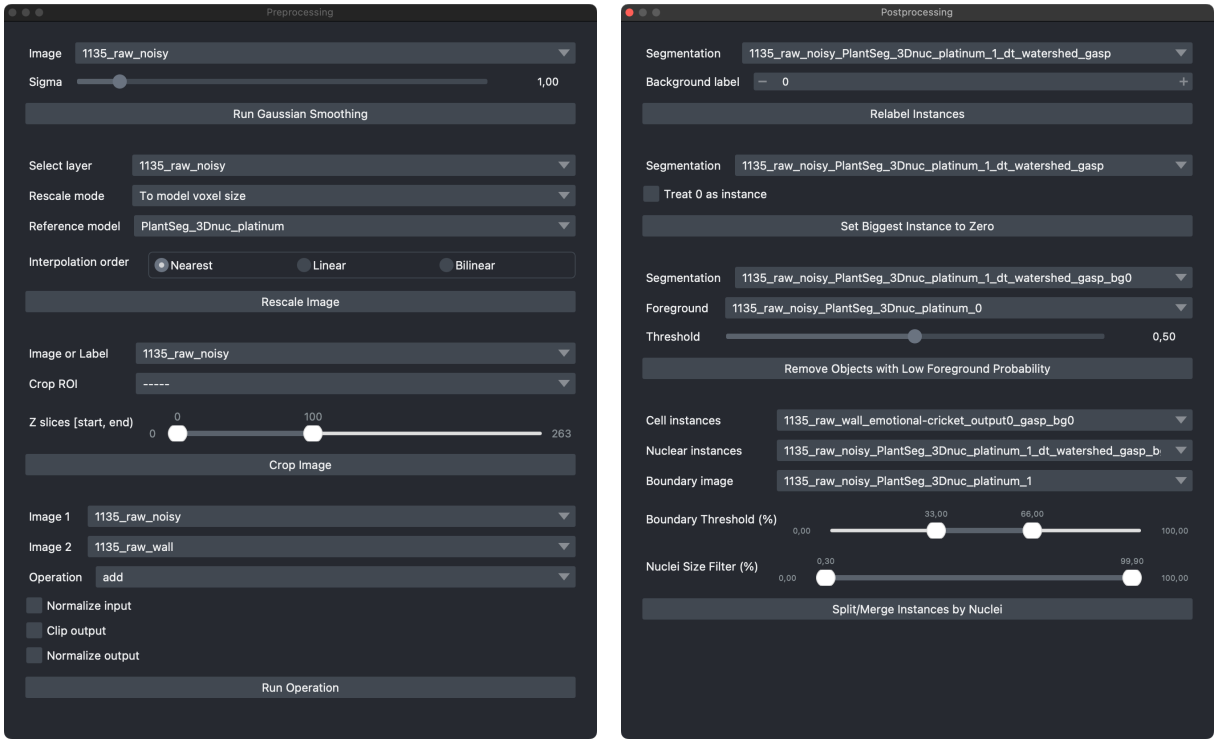


Figure 3.2: PlantSeg 2.0 - Image Pre-processing and Post-processing. Left: Pre-processing widgets allow users to interactively apply operations such as denoising (Gaussian blur), cropping, rescaling, and multi-channel image combination. All operations are non-destructive and visualised in real time. Auxiliary functions help automatically determine optimal parameters, reducing user effort. The widgets are highly customisable to suit different datasets and tasks. **Right:** Post-processing widgets enable segmentation refinement. Users can remove the largest instance (typically background), eliminate false positives by filtering objects with low foreground probability, and improve semantic accuracy using multi-channel input. These tools extend PlantSeg from dense tissue segmentation to general-purpose instance segmentation. For example, combining nuclear and cell channels allows automatic splitting or merging of cells based on the assumption that each contains exactly one nucleus. Manual corrections are also supported via the proofreading tool (Figure 3.6).

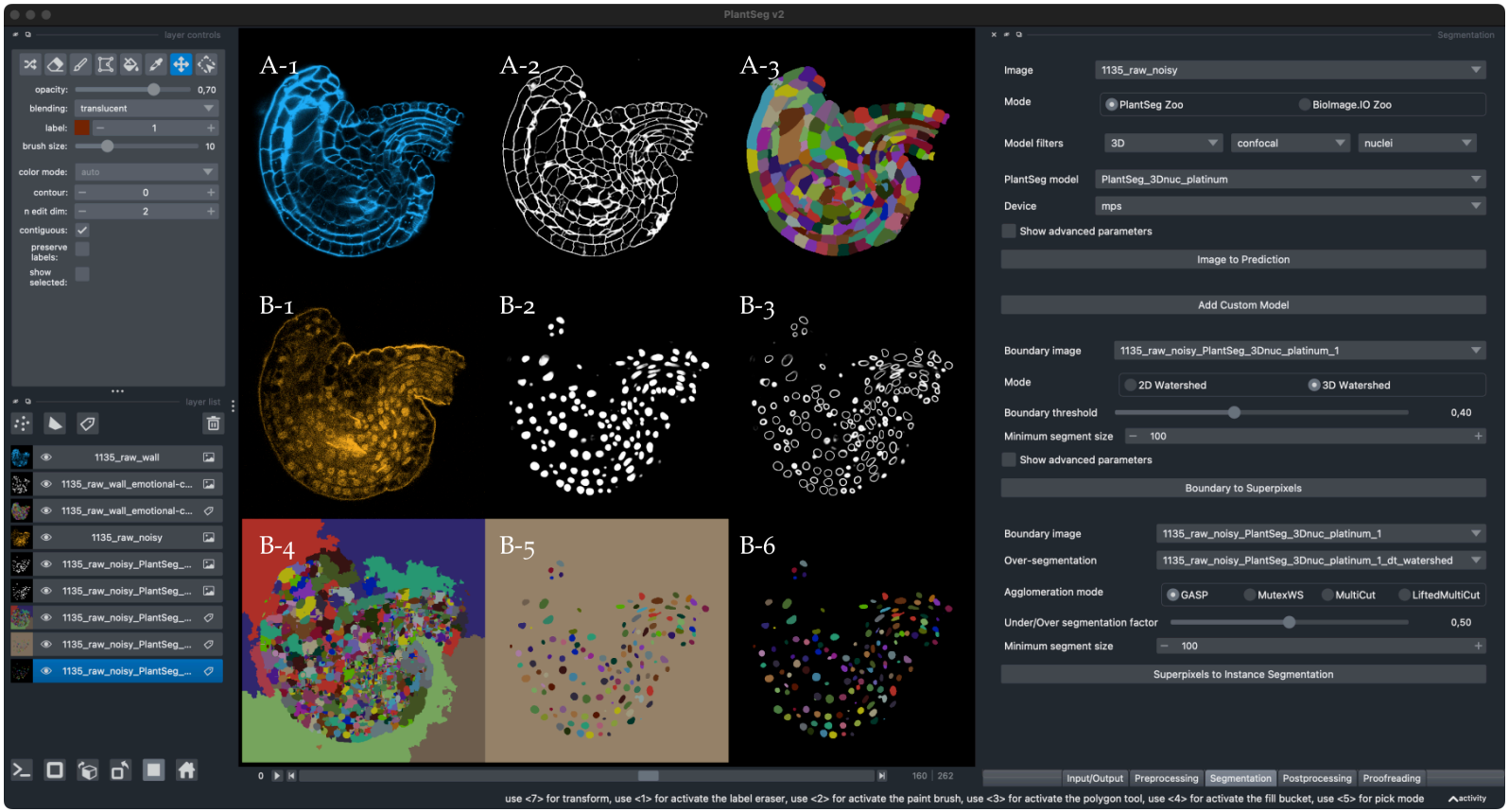


Figure 3.3: PlantSeg 2.0 - Main Prediction-Segmentation Workflow. The PlantSeg 2.0 workflow consists of two main stages managed by three widgets: **1. Semantic Prediction:** Users select an input image and a model from either the PlantSeg Model Zoo [6] or the BioImage Model Zoo [72]. For PlantSeg models, patch and halo sizes are automatically computed using my algorithms (Algorithm 1, Algorithm 2), based on the input image, model architecture, and available hardware. BioImage.IO models are executed via my integration with BioImage.IO Core [73]. This step supports any compatible model; boundary predictions can then be used for instance segmentation. **2. Instance Segmentation:** Boundaries are first transformed into superpixels (via watershed), then into instances using agglomeration algorithms such as GASP or Multicut. Separating these two steps improves user understanding and allows efficient exploration of segmentation parameters, especially after a costly prediction step. I also integrated Lifted Multicut, which uses nuclear segmentation to constrain cell segmentation such that each instance contains exactly one nucleus. **Figure panels:** Two 3D segmentation pipelines are shown using a representative z-slice. **(A)** Cell segmentation of ovule ID 1135 from the GoNuclear dataset (subsection 2.3.1): (1) raw cell wall image; (2) boundary prediction from the `emotional-cricket` model (BioImage Model Zoo); (3) Watershed-GASP result with background removed. **(B)** Nuclear segmentation of its weakly stained TO-PRO-3 channel: (1) raw image; (2/3) foreground and boundary predictions from the `PlantSeg_3Dnuc_platinum` model (PlantSeg Model Zoo); (4) watershed (superpixels); (5) GASP result; (6) background removed using the post-processing widget shown in Figure 3.2.

Post-processing Segmentations: After generating instance segmentations, users can refine results through post-processing steps:

- **Remove background:** The "Set Biggest Instance to Zero" widget removes the largest instance, typically representing background.
- **Eliminate false positives:** The "Remove Object with Low Foreground Probability" widget excludes regions with low mean foreground probability.
- **Split/merge cells using nuclei:** This widget uses nuclei segmentation or probability maps to refine cell instances, assuming each cell contains exactly one nucleus.

Post-processed segmentations are displayed in the Napari viewer for visual verification.

Proofreading Segmentations: For manual refinement of imperfect automated results, users can employ the "Proofreading" widget. This tool supports splitting and merging instances in 2D and 3D, with undo/redo functionality and the ability to save/load proofreading states. Note that manual proofreading steps are not recorded in the configuration file and are specific to individual sessions.

Exporting Results and Workflow: Once segmentation results meet user expectations, they can be exported along with the workflow as a configuration file. This file documents all steps and parameters used in the GUI, allowing replication of the process in headless mode for batch processing or distributed systems.

3.2 Key Features in PlantSeg 2.0

PlantSeg 2.0 represents a substantial update to the platform, designed to improve usability, extend functionality, and enhance compatibility with community standards. This major release features a redesigned graphical user interface for seamless interactive image processing and introduces several important tools and improvements. I contributed to this release by implementing the following key features:

- **Interactive Napari Interface**: I co-developed a Napari-based GUI to provide an interactive platform for real-time visualization and proofreading. This integration significantly enhances usability and user experience, enabling researchers to explore and interact with their data efficiently.
- **Integration with BioImage.IO Model Zoo and Core**: I enabled the use of all models from the BioImage.IO Model Zoo, ensuring broad compatibility with community standards and future expansion.
- **Proofreading Tools**: I co-developed built-in tools for efficient human correction of 3D segmentations, enabling faster, more accessible, and accurate automated corrections.

- **GoNuclear Support**: I integrated GoNuclear to utilise multi-channel data, combining cell and nuclear information for improved segmentation results. For nuclear segmentation itself, I developed a strategy to extend PlantSeg's capabilities from densely packed 3D volumetric images to 3D images with sparse nuclei.

- **Automatic Patch and Halo Size Finders**: I implemented algorithms to automatically determine a good pair patch and halo sizes, reducing user effort and optimising computational resource usage.

- **Documentation Website**: I developed a comprehensive online resource to assist users in understanding and effectively applying PlantSeg's features.

3.2.1 Interactive Napari Interface

PlantSeg 1.0 was widely recognised for its powerful segmentation capabilities, but its user interface was limited to two options: a Python API supporting only prediction and segmentation functions, and a configuration file-based interface requiring users to manually edit YAML files in a text editor or PlantSeg's classic GUI. The Tk-based GUI merely served as a configuration editor without real-time image visualization, forcing users to check outputs in external software like Fiji or Napari. This cumbersome workflow leads to slower iteration cycles and reduced productivity. Given the trial-and-error nature of bioimage analysis, the lack of interactivity in PlantSeg 1.0 significantly hampered usability.

To address these limitations, PlantSeg 2.0 integrates Napari as its primary graphical interface, providing a powerful, interactive platform for image visualization, annotation, and analysis. I developed several widgets[2] and co-developed the rest, enabling real-time user interaction with processed results.

Napari is a Python-based, open-source image viewer designed for multi-dimensional bioimage analysis. Positioned as a potential successor to Fiji, it benefits from Python's dominance in scientific computing, enabling seamless integration with libraries such as NumPy, PyTorch, and BioImage.IO. Built on the Qt framework, Napari offers a responsive, user-friendly graphical interface with GPU-accelerated rendering and native support for n-dimensional and 3D data. Its extensible plugin system fosters innovation by allowing developers to expand its functionality, making it a dynamic tool for the bioimage community. As a community-driven project with financial backing from the Chan Zuckerberg Initiative, Napari ensures long-term sustainability through active development and user-driven improvements.

Integrating Napari with PlantSeg presented challenges, as both Napari and its backend, Qt, are powerful yet complex tools. PlantSeg uses `magicgui` to create widgets that bridge Python functionality with the Qt-Napari interface. However, not all essential Qt features are covered by `magicgui`. For instance, I collaborated with the founder of Napari to contribute a separator element[3] to `magicgui`, improving dropdown menu organization in the Napari interface—an enhancement that benefits all `magicgui` users and potentially all Napari-based app developers.

[2] A Napari widget is an interactive graphical component that extends Napari's functionality by providing a user interface for specific tasks, such as image processing, segmentation, annotation, or parameter tuning. Widgets allow users to interact with their data through intuitive controls (e.g., sliders, buttons, dropdowns) rather than writing code.

[3] GitHub `pyapp-kit/magicgui` - feature: Add Separator to ComboBox PR #638

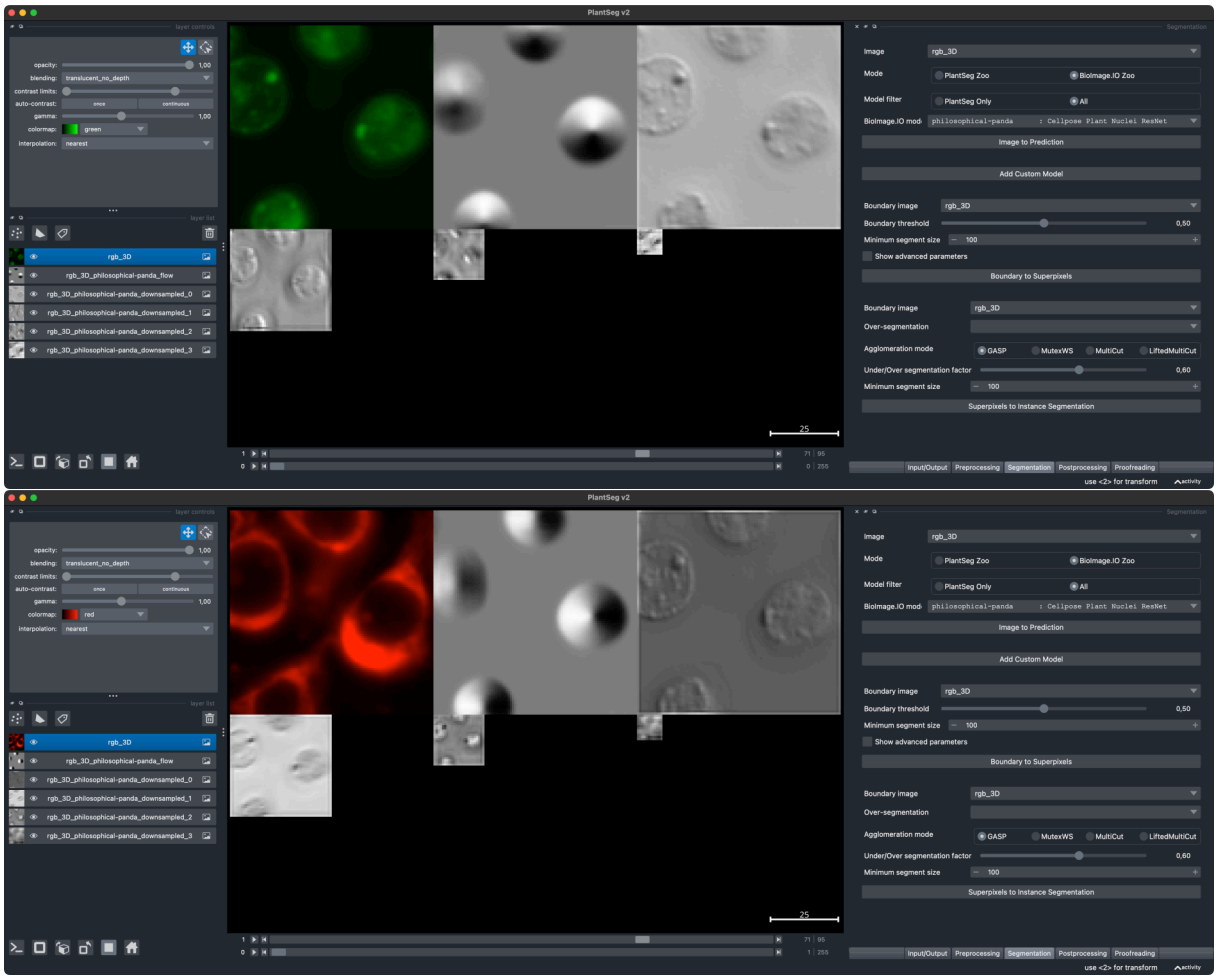


Figure 3.4: PlantSeg 2.0 - BioImage Model Zoo Integration. Demonstration of the first Cellpose model in the BioImage Model Zoo on official Cellpose test data. I integrated Cellpose and BioImage.IO[4,5] and uploaded the `philosophical-panda` model to the zoo[6] (see section 2.2). **Top:** first channel of the 2-channel raw image, followed by the 3D flow field and higher-order feature channels. PlantSeg 2.0 can execute any model following the BioImage.IO specification and visualise its outputs. Future BioImage.IO Core updates will provide direct postprocessing for Cellpose and StarDist; in the meantime, these predictions can be used for semantic segmentation or as references for manual proofreading. An additional example of BioImage.IO usage in PlantSeg appears in Figure 3.3 (A).

By leveraging Napari, PlantSeg 2.0 significantly improves interactivity and usability. Researchers can explore multi-dimensional images, adjust contrast, and interact with segmentation results in real time. Napari's layering capabilities allow users to overlay raw data, probability maps, and segmentations, while built-in editing tools, such as brushing and region selection, enable direct manual corrections. These features streamline segmentation workflows, improving both efficiency and research outcomes.

Integrating Napari provides PlantSeg with a modern, high-performance visualization platform that lowers technical barriers and enhances bioimage segmentation workflows. With its interactive capabilities and growing adoption within the bioimage analysis community, Napari positions PlantSeg as a more accessible and effective tool for scientific discovery.

[4]GitHub – MouseLand/cellpose/pull/988
[5]GitHub – MouseLand/cellpose/pull/1011
[6]GitHub – bioimage-io/spec-bioimage-io/pull/604

3.2.2 Integration of BioImage.IO

PlantSeg 1.0 was released with a range of pre-trained models, including 2D and 3D U-Nets, to support diverse segmentation tasks. However, the field of bioimage analysis is rapidly evolving, with new models and architectures emerging regularly. To ensure PlantSeg remains at the forefront of innovation and maintains compatibility with the latest advancements, I integrated the BioImage.IO Core and BioImage Model Zoo into PlantSeg 2.0.

BioImage.IO is an open-source initiative that standardises the exchange of deep learning models for bioimage analysis. It provides a unified format for packaging, sharing, and executing models across different frameworks and platforms, bridging the gap between model developers and end-users. By adopting BioImage.IO, PlantSeg gains access to a vast repository of pre-trained models, enabling users to leverage cutting-edge architectures and workflows seamlessly. This integration broadens the range of available segmentation models and allows users to experiment with state-of-the-art approaches tailored to their specific datasets.

To integrate BioImage.IO into PlantSeg, I implemented a streamlined interface that allows users to directly access and import models from the repository. This required adapting PlantSeg's backend to accommodate BioImage.IO's model specification standards, including support for ONNX and TensorFlow SavedModel formats. Additionally, I added functionality for automatically downloading models and their metadata, verifying compatibility, and configuring input/output parameters.

As a result, PlantSeg 2.0 offers a user-friendly pipeline where researchers can browse BioImage.IO for relevant models and apply them to their data effortlessly. This integration enhances the flexibility and power of PlantSeg, providing users with an expanding library of models while ensuring ease of use.

3.2.3 Automatic Patch and Halo Shape Finders

In neural network-based bioimage segmentation, *inference* refers to applying a trained model to new data in order to generate segmentation predictions. For large 3D biological samples—such as plant tissues imaged at cellular resolution—processing the entire volume in a single pass is often impractical due to hardware memory constraints, typically limited by the amount of *VRAM* available on GPUs. Although modern GPUs can have several gigabytes of memory, volumetric microscopy datasets can exceed these capacities by tens or even hundreds of gigabytes. Moreover, as input data propagates through the network during inference, additional intermediate representations (e.g., feature maps in CNNs) are generated, further increasing the overall memory footprint. Consequently, the volume must be divided into smaller *patches*, each processed independently. After segmenting each patch, the resulting outputs are combined (*stitched*) to form the final segmentation mask for the entire volume.

A key challenge in patch-based processing arises near patch boundaries. Convolutional neural networks typically require contextual information around each voxel for accurate predictions. Near edges, that context is truncated, leading to boundary artifacts such as abrupt transitions in the predicted intensities or labels. To mitigate these artifacts, it is common to add a *halo*—an overlapping margin around the patch boundaries. This additional region provides extra context to the network, producing smoother and more accurate predictions along patch edges.

Choosing appropriate patch and halo dimensions is critical. Larger patches offer more contextual information but increase memory usage, potentially slowing down inference or requiring specialised high-memory GPUs. Conversely, smaller patches reduce memory load but require more patches to cover the volume, increasing overhead and possibly missing broader spatial structures. Likewise, a halo that is too small fails to eliminate boundary artifacts, while an excessively large halo duplicates computations unnecessarily.

An *automatic patch and halo shape finder* addresses these trade-offs by analyzing both the 3D volume and the available hardware resources—particularly GPU memory capacity—and determining shapes that maximise segmentation quality while staying within memory limits. This automation spares users from manual parameter tuning, ensuring final segmentation results are both accurate (via sufficient context) and practical (fitting within hardware constraints).

Previous Implementation in PlantSeg 1.0. Running 2D and 3D U-Nets to predict boundary probabilities has always been at the core of PlantSeg. In PlantSeg 1.0, users were required to specify the model, device, patch shape, and stride size for the network prediction step. The software would then split the input volume into patches of the chosen shape, apply mirrored padding to these patches, perform inference, and stitch the results together to produce a full-volume segmentation.

Through my investigation, I identified two major issues with this workflow:

1. **Complex Configuration:** Users needed a thorough understanding of both the model architecture and their computational environment to select suitable patch shapes and stride sizes.

2. **Flawed Padding Strategy:** PlantSeg 1.0 employed a mirroring strategy at patch borders instead of providing genuine neighboring context, which introduced tiling artifacts and exacerbated boarder-related issues. This approach misled developers into undervaluing the necessity of a proper halo region, leading them to omit it from the configuration in favor of a expedient but suboptimal solution using stride parameters. Consequently, this decision inadvertently perpetuated boundary artifacts in the resulting segmentations.

During my exploration of these artifacts, I found that the incorrect border mirroring was the primary cause. Prior to this bug, PlantSeg 1.0 had included a halo-based design with a fixed, albeit suboptimal, parameter determined by empirical trial and error on a particular image set. Over time, as PlantSeg grew in popularity—its core inference package `pytorch-3dunet` amassed around 1800 GitHub stars[7] and over 9500 Anaconda downloads[8,9], and PlantSeg itself was cited by roughly 200 publications[10], the subtlety of this bug allowed it to go unnoticed. Although such a suboptimal or technically incorrect implementation may not have immediately undermined scientific findings, it did degrade segmentation fidelity and limited the tool's broader applicability. Once I fixed this bug, PlantSeg's segmentation results became more accurate, reliable, and suitable for a wider range of imaging data and tasks.

[7] GitHub Star History - `wolny/pytorch-3dunet`
[8] Anaconda.org - `conda-forge/packages/pytorch-3dunet`
[9] Anaconda.org - `awolny/packages/pytorch-3dunet`
[10] Google Scholar - Citation of [6] by 2024

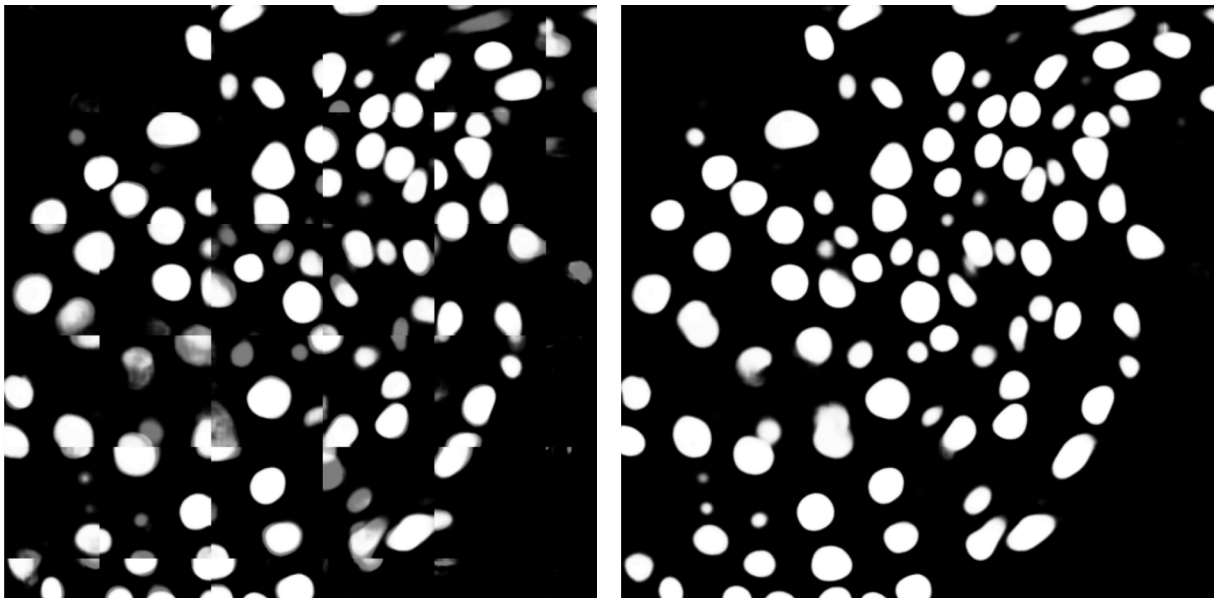


Figure 3.5: PlantSeg 2.0 - Fixing Tiling Artefacts. Both images are semantic foreground prediction of nuclei output from the same network, but with different input padding strategies. Patch shape, stride, and halo sizes are the same in both cases. **Left:** Tiling artefact caused by mirror-padding in PlantSeg v1 and `pytorch-3dunet`, historically mitigated by enlarging the patch size as much as hardware permitted. **Right:** Halo-based padding in PlantSeg 2.0 eliminates these artefacts. The required halo (at least half the model's receptive field) and optimal patch size are automatically determined from the model architecture, image dimensions, and hardware limits, ensuring accurate predictions for 2D and 3D datasets.

Automatic Halo Shape Finder. A major new feature I introduced in PlantSeg 2.0 is the *automatic halo shape finder*, which builds on the theoretical framework of Majurski et al. [74]. To guarantee artifact-free pixel classification, the halo must be at least half the network's receptive field. My implementation aggregates spatial contributions from each convolutional kernel along the longest path through the network, accounting for the kernel's stride and the amount of downsampling and upsampling at each layer.

Let U-Net be described by a sequence of convolutional layers $c = 0, \ldots, N-1$, each associated with a level l_c, and let each convolution use a square kernel $(k_c \times k_c)$. The level l_c is the difference between the number of max-pooling layers and the number of up-convolution layers encountered from the input to layer c. The required halo size for such a network can be computed by Equation 3.1:

$$\text{Halo} = \sum_{c=0}^{N-1} 2^{l_c} \left\lfloor \frac{k_c}{2} \right\rfloor \tag{3.1}$$

While this equation aligns with the receptive field framework of [75], it provides a U-Net-specific decomposition, enabling analysis of how each layer contributes to the receptive field. In PlantSeg's default models of classes `UNet3D` and `UNet2D`, which use a fixed kernel size of 3 for all convolutional layers[11], the formula further simplifies to Equation 3.2:

[11] PlantSeg's UNet models are customiable in terms of kernel size, but such customisation is not exposed in any user interface.

$$\text{Halo} = \sum_{c=0}^{N-1} 2^{l_c} \tag{3.2}$$

This summation effectively captures the expansions and contractions (downsampling and upsampling) that occur along the network's encoder-decoder pathway. Algorithm 1 demonstrates how PlantSeg implements this halo calculation.

Algorithm 1 Compute Halo Size for U-Net

Require: *unet*: a U-Net model (`UNet2D` or `UNet3D`)
Ensure: *halo*: computed halo size (integer)

function ComputeHalo(unet)
  $level \leftarrow 0$; $halo \leftarrow 0$
  for $layer$ textbfin $unet$ **do**
    if $layer$ is *MaxPool* **then**
      $level \leftarrow level + 1$ ▷ Encoder: increase depth
    else if $layer$ is *Upsampling* **then**
      $level \leftarrow level - 1$ ▷ Decoder: decrease depth
    else if $layer$ is *Conv* **then**
      $halo \leftarrow halo + 2^{level} \times \left\lfloor \frac{\text{kernel size}(layer)}{2} \right\rfloor$ ▷ Kernel size 3 in default models
    end if
  end for
  return $halo$
end function

Automatic Patch Shape Finder. Another critical enhancement I made in PlantSeg 2.0 is the *automatic patch shape finder*, which determines an optimal patch shape for inference based on (1) the model architecture and the theoretical minimum halo it requires, (2) the available GPU memory, and (3) the dimensions of the input volume. Previously, users manually configured the patch shape, which required a nuanced understanding of the interaction between GPU memory, network architecture, and image size. By automating this step, PlantSeg removes both guesswork and the risk of out-of-memory (OOM) errors.

Determining memory usage for a particular input shape is nontrivial. Packages like `torchinfo` [76], `pytorch_memlab` [77], or `pytorch_modelsize` [78] attempt to estimate memory usage by executing inference with a test input. If the test input is too large, the process triggers an OOM error. Because frameworks like PyTorch create dynamic computation graphs (whose intermediate activation sizes depend on runtime data), this trial-and-error approach is often the only way to gauge a model's memory footprint without explicit analytic formulas.

PlantSeg's solution is to probe for a maximal feasible patch shape via a series of forward passes, stopping when the GPU memory limit is exceeded. Specifically:

- **3D U-Net models:** A binary search is performed over a range of isotropic patch sizes of the form $(16 \times n, 16 \times n, 16 \times n)$, with n spanning from a lower bound (e.g. 2) up to an upper bound (e.g. 50).

- **2D U-Net models:** A linear, decrementing search is used for 2D patch shapes $(16 \times n, 16 \times n)$, starting from a maximum (e.g. 200) with a step size (e.g. 20) and halting once an acceptable size that fits the GPU is found.

After finding the approximate maximum feasible patch shape, PlantSeg compares it to the actual volume dimensions and adjusts if necessary. The function `find_patch_and_halo_shapes` ensures that if the volume is smaller than the largest feasible patch size in one or more dimensions, it can stretch the patch to match the volume in those dimensions. Halo margins are only applied along dimensions that truly require tiling (i.e. when the maximum possible input size is smaller than the volume). This strategy makes full use of GPU memory without unnecessary overhead.

Algorithm 2 outlines the logic for reconciling the maximum feasible patch size $\mathbf{b}$, the full volume size $\mathbf{a}$, and the minimum required halo $\mathbf{h}$ into an optimal patch-halo configuration $(\mathbf{b}', \mathbf{h}')$.

Algorithm 2 Compute Optimal Patch and Halo Shape Pair.

Require: $\mathbf{a} = (a_1, a_2, a_3)^T$: Shape of the full 3D sample volume.
Require: $\mathbf{b} = (b_1, b_2, b_3)^T$: Maximum feasible patch shape for the hardware.
Require: $\mathbf{h} = (h_1, h_2, h_3)^T$: Minimum required halo size per side.
Ensure: $\mathbf{b}' = (b'_1, b'_2, b'_3)^T$: Optimal patch shape for the sample and hardware.
Ensure: $\mathbf{h}' = (h'_1, h'_2, h'_3)^T$: Corresponding halo size per side.

function ComputePatchHaloPair($\mathbf{a}, \mathbf{b}, \mathbf{h}$)
  $d_i \leftarrow \mathbf{1}\{b_i > a_i\}$ ▷ indicator $\mathbf{1}\{P\}$ is 1 if proposition P is true, else 0.
  $h'_i \leftarrow (1 - d_i) \cdot h_i$ ▷ add halo only if $b_i < a_i$.
  $N_a \leftarrow \mathbf{1}^T\mathbf{a}, \quad N_b \leftarrow \mathbf{1}^T\mathbf{b}$ ▷ $\mathbf{1}^T = (1, 1, 1)$.
  $\mathbf{b}' \leftarrow \min(\mathbf{a}, \mathbf{b})$ ▷ $\forall i.\ [\min(\mathbf{a}, \mathbf{b})]_i = \min(a_i, b_i)$.
  if $N_b \geqslant N_a$ **then** ▷ $N_b \geqslant N_a \implies \mathbf{1}^T\mathbf{d} = 3$
    $\mathbf{b}' \leftarrow \mathbf{a}, \quad \mathbf{h}' \leftarrow \mathbf{0}$.
  else if $\mathbf{1}^T\mathbf{d} = 2$ **then** ▷ Exactly two dimensions of $\mathbf{b}$ exceed $\mathbf{a}$.
    let (i, j, k) be a permutation of $(1, 2, 3)$
    let $b_i > a_i$ and $b_j > a_j$
    $(b_i, b_j, b_k) \leftarrow (a_i, a_j, \lfloor N_b/(a_i a_j) \rfloor)$
  else if $\mathbf{1}^T\mathbf{d} = 1$ **then** ▷ Exactly one dimension of $\mathbf{b}$ exceeds $\mathbf{a}$.
    let (i, j, k) be a permutation of $(1, 2, 3)$
    let $b_i > a_i$
    $(b_i, b_j, b_k) \leftarrow (a_i, \lfloor \sqrt{N_b/a_i} \rfloor, \lfloor N_b/a_i \rfloor)$
  else if $\mathbf{1}^T\mathbf{d} = 0$ **then** ▷ $\forall i.\ b_i < a_i; N_b < N_a$.
    $\mathbf{b}' \leftarrow \mathbf{b}$
  end if
  $\mathbf{b}' \leftarrow \mathbf{b}' - \mathbf{h}'$ ▷ subtract halo from the effective patch.
  return $\mathbf{b}', \mathbf{h}'$
end function

By automating both *patch* and *halo* shape selection, PlantSeg 2.0 significantly simplifies network inference, removing the guesswork traditionally involved in tiling configurations. The result is a more robust, efficient, and user-friendly segmentation pipeline that accommodates a wide range of hardware setups and imaging datasets.

3.2.4 Human Proofreading

Proofreading is a critical step in the segmentation workflow, enabling users to correct errors and refine results to achieve higher accuracy. Once a segmentation is generated in PlantSeg, it can be further improved either automatically, using post-processing steps (as detailed in subsection 3.2.5), or manually through proofreading. Manual proofreading allows users to directly interact with the segmentation, ensuring more precise outcomes.

PlantSeg 2.0 introduces a dedicated proofreading widget that empowers users to interactively adjust and refine segmentations. This tool significantly enhances the quality of the final output and, once integrated with the training/finetuning widget, will allow users to generate new training data and fine-tune PlantSeg models using their corrections.

The proofreading widget builds upon the original tool developed for PlantSeg Tools [79]. In this workflow, users can make corrections by drawing strokes with the brush tool in Napari in 2D. PlantSeg interprets these strokes as seeds and uses them to perform splitting and merging operations on 2D and 3D segmentations. These adjustments are made relative to a reference probability map or a raw image with well-defined boundaries. Specifically:

Merging: If the seeds share the same color, the corresponding objects will be merged.

Splitting: If the seeds have different colors, the corresponding objects will be split.

While the original proofreading tool was effective, users reported two key limitations: the lack of an undo/redo function and the inability to save or load proofreading states. To address these issues, I enhanced the widget by introducing undo/redo functionality, allowing users to easily revert or repeat their edits. Additionally, I implemented support for saving and loading proofreading states, enabling users to pause their work and resume later without losing progress.

3.2.5 Nuclear Segmentation and Nuclei-Guided Cell Segmentation

The integration of cell and nucleus segmentation represents a major advancement in PlantSeg's capabilities, driven by the growing availability of datasets where both cells and nuclei are stained and imaged. Accurate nuclear instance segmentation, combined with semantic boundary segmentation from the cell channel, enables new workflows in PlantSeg. For example, the lifted multi-cut agglomeration algorithm (accessible via the "LiftedMultiCut" option in the "Superpixels to Instance Segmentation" widget) can improve cell instance segmentation. Additionally, the "Split/Merge Instances by Nuclei" widget in PlantSeg's post-processing steps allows users to refine existing cell instance segmentation by leveraging nuclear instance segmentation as a reference. During proofreading, users can also cross-reference either segmentation to correct the other, enhancing accuracy.

Light microscopic images present three common scenarios for instance segmentation:

Non-touching instances separated by empty spaces: These can be handled straightforwardly by thresholding the foreground prediction from neural network during post-prediction steps.

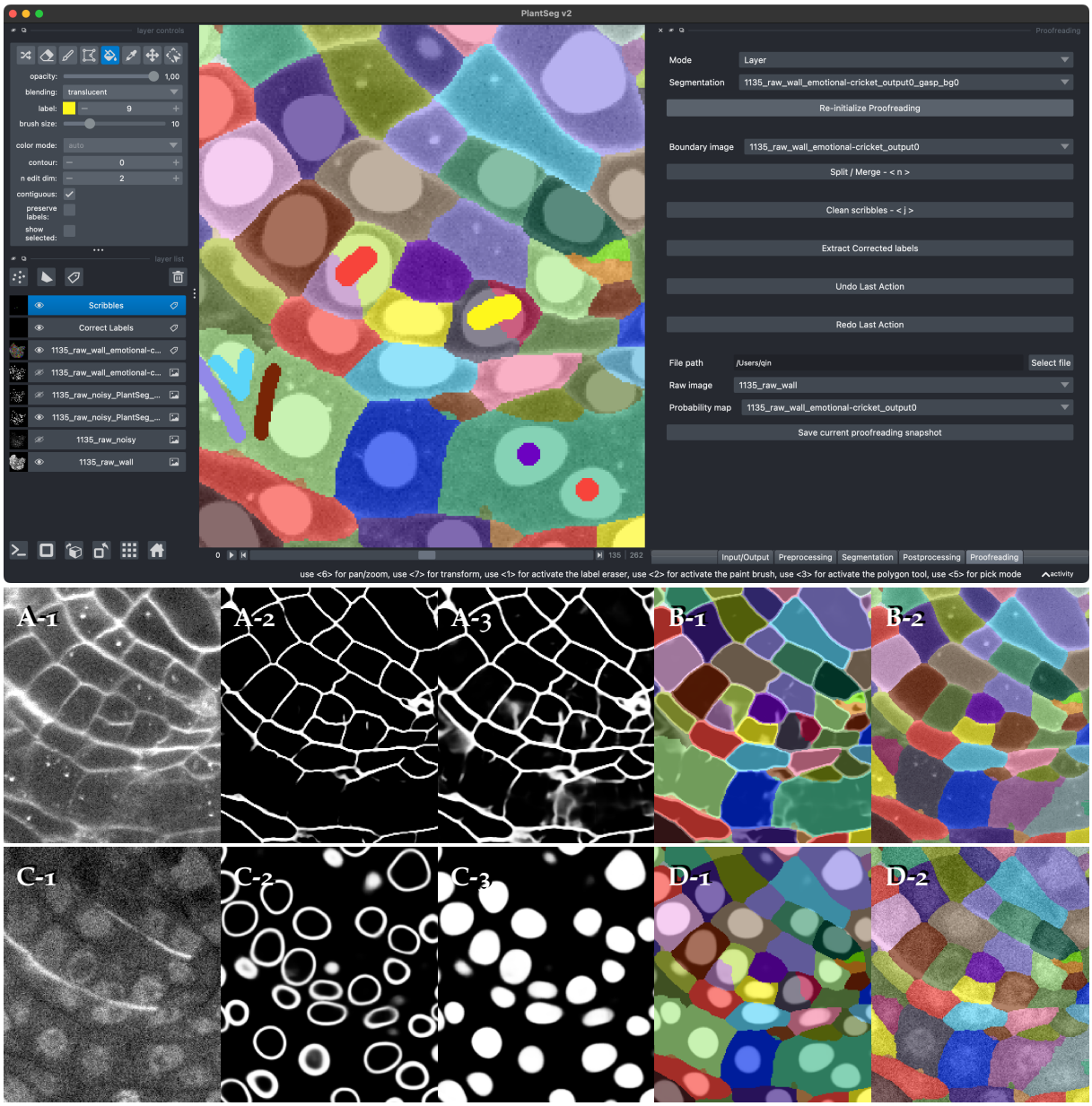


Figure 3.6: PlantSeg 2.0 - Interactive Proofreading Tool. An example of imperfect neural network outputs demonstrates how PlantSeg's proofreading widget fixes under- and over-segmentation. **(A-1)** shows a raw cell-wall image, while **(A-2)** is its probability map with faint boundaries and low-confidence edges that become more visible in **(A-3)** using different contrast settings. These uncertain boundaries can cause both over-segmentation (imaginary boundaries splitting single cells) and under-segmentation (merged cells). Instead of tuning or switching models, users can employ the interactive proofreading tool (shown at the top), painting a single ID to merge instances or multiple IDs to split them; **(B-2)** shows the result. **(C-1)**-**(C-3)** depict a noisy nuclear channel with boundary and foreground predictions. The foreground map clarifies the true signal (**C-3**), aiding manual fixes; **(D-1)** and **(D-2)** illustrate how referencing the foreground can reveal and correct errors that are not readily visible in the raw image alone. These approaches provide a straightforward yet powerful way to refine 3D segmentations in a 2D interface.

Densely packed, touching instances without gaps: PlantSeg was originally designed for this scenario, as exemplified by the segmentation of cells in plant ovules. These cases are challenging because instances cannot be directly obtained from the semantic foreground mask from neural networks.

Mixed cases with both touching and non-touching instances: These scenarios, such as nuclei segmentation in noisy plant tissue images, are more complex. My extensions to PlantSeg enable it to effectively address these cases. Furthermore, accurate nuclear segmentation can be used to improve the instance segmentation of cells in the same tissue.

PlantSeg's original boundary-based approach was designed to amplify biological structure boundaries and use graph-based algorithms for instance segmentation. This method excels in densely packed 3D volumetric images, such as cells in plant tissues, where each cell wall separates two adjacent cells. Background, which borders the outermost tissue boundary, is easily identified and excluded. Predicting cell foregrounds in such scenarios would result in a single, large connected mask for the entire tissue. Instead, PlantSeg focuses on boundary segmentation to partition individual cells effectively.

Boundary-based segmentation becomes less effective when instances are not densely packed, such as nuclei separated by cytoplasm or extracellular spaces. In these cases, gaps between objects may be misclassified as part of the objects, leading to inaccurate segmentation. In [6], PlantSeg v1 successfully segments cell instances in leaf images because the gaps between cells are small and can be filtered out based on size. However, when segmenting nuclei, the gaps can be larger than the nuclei themselves, rendering the previous approach ineffective. To overcome this limitation, I introduced a dual-channel neural network output, incorporating both a boundary channel and a foreground channel.

In the foreground channel:

- Pixels inside nuclei are predicted as foreground (value close to 1).
- Pixels on nuclear membranes are predicted as boundary (value close to 1).
- All other pixels are classified as background (value close to 0).

This foreground channel enables direct segmentation of non-touching nuclei via thresholding. For touching nuclei, a more robust workflow involves:

1. Applying the watershed algorithm on the boundary channel to generate superpixels.
2. Filtering superpixels by their mean foreground probabilities.
3. Agglomerating the filtered superpixels into instances using algorithms like GASP.

This pipeline significantly improves robustness and enables accurate segmentation of nuclei in 3D volumetric images, even under challenging conditions.

The dual-channel network I trained — capable of predicting boundary and foreground channels for nuclei across various biological samples imaged with diverse light microscopy techniques — is fully integrated into PlantSeg. Users can run inference directly through PlantSeg's prediction API or Napari widget. This model was published as part of GoNuclear, detailed in chapter 2.

To filter superpixels by their mean foreground probabilities, I developed a foreground-guided false positive removal function, available in both PlantSeg's Python API and Napari widget.

This function can be applied before and/or after the agglomeration step. One critical issue it addresses is the merging of instance superpixels with background superpixels during agglomeration, due to weak boundary in the raw image or in the predicted boundary segmentation. Running this function on superpixels prior to agglomeration prevents such errors, ensuring cleaner segmentation outputs.

By integrating cell and nucleus segmentation, PlantSeg 2.0 provides researchers with powerful tools to analyze complex biological datasets, enabling workflows that were previously difficult or impossible to achieve.

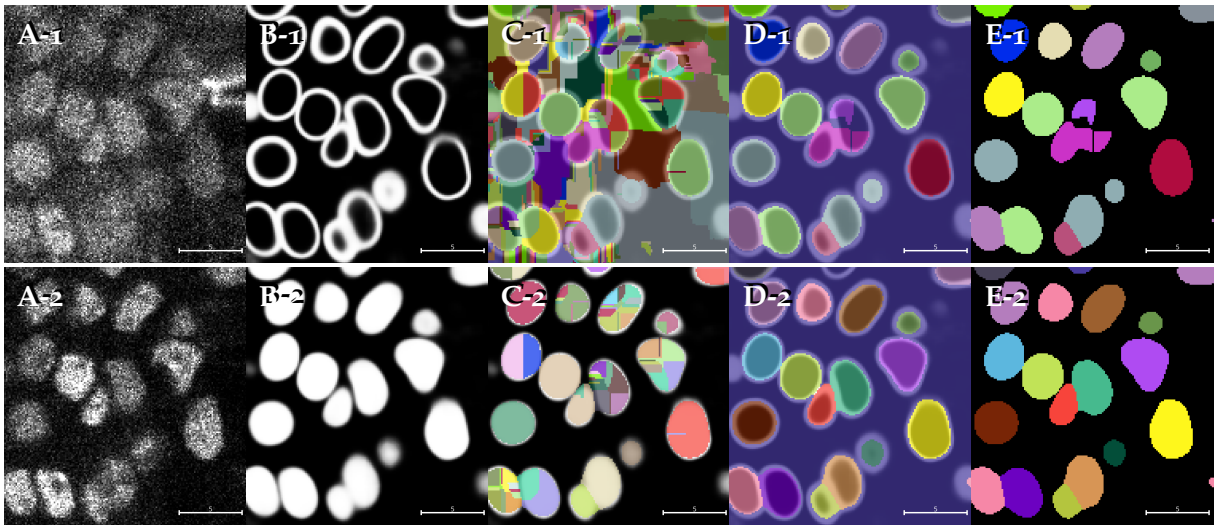


Figure 3.7: PlantSeg 2.0 - Sparse Instance Segmentation Using Foreground Filtering. Demonstration of two sparse instance segmentation pipelines on a noisy nuclear channel using PlantSeg. **(A-1)** Raw input with low signal-to-noise ratio; **(A-2)** cleaner nuclear image from the same sample (shown for reference only, not used in processing). **(B-1)** Boundary prediction from A-1; **(B-2)** foreground prediction from A-1. **(C-1)** Watershed on B-1, overlaid with boundary prediction; **(C-2)** result of filtering C-1 using the foreground mask B-2, overlaid with B-2. **(D-1)** GASP agglomeration on C-1; **(D-2)** GASP on C-2, both overlaid with B-1. **(E-1)** D-1 with the largest instance (background) removed; **(E-2)** D-2 with background removed. The baseline pipeline (**A-1** → **B-1** → **C-1** → **D-1** → **E-1**) performs instance segmentation without foreground filtering. The enhanced pipeline (**A-1** → **B-1** → **C-1** → **C-2** → **D-2** → **E-2**) applies foreground filtering twice, yielding more accurate segmentation of sparse and weakly stained nuclei.

3.2.6 Documentation

Documentation is a crucial component of any software project, serving as the bridge between developers and users. This is particularly true for PlantSeg, where the primary users are advanced biologists aiming to process microscopic images for scientific discoveries. Many of these users lack formal coding experience, making accessible and well-structured documentation essential. While a user-friendly and intuitive interface is important, it cannot alone guarantee the scientific rigor and reliability that researchers require. The documentation for PlantSeg provides clear and comprehensive guidance on installing, configuring, and effectively using the tool for bioimage segmentation tasks. It ensures that users, regardless of their technical expertise, can understand the software's capabilities and seamlessly integrate it into their workflows. High-quality documentation is vital for fostering user adoption, simplifying troubleshooting, and encouraging community contributions. By thoroughly documenting

PlantSeg, I empower researchers to unlock its full potential while ensuring scalability and adaptability for future advancements.

To address the documentation needs of PlantSeg, I created a dedicated website that significantly improved upon the incomplete repository wiki [80]. The legacy wiki briefly described an outdated configuration-creation interface that had been deprecated in PlantSeg v2. Recognising the limitations of GitHub wikis, I decided to migrate the documentation to Jupyter Book [81]. This migration was driven by several shortcomings of GitHub wikis, including: (1) lack of efficient and usable search functionality, (2) inability to version control alongside the codebase, (3) absence of responsive design for small screens, (4) lack of support for generating Python API or Napari widget documentation from code, and (5) inability to integrate Jupyter Notebooks. Jupyter Book addressed these challenges effectively, allowing me to provide a more robust documentation framework.

However, as the needs of PlantSeg evolved, I decided to migrate from Jupyter Book to MkDocs, specifically using the Material for MkDocs theme [82, 83]. While Jupyter Book excels in integrating executable code blocks and computational content, these features were unnecessary for PlantSeg's documentation. Instead, I prioritised a modern and aesthetically pleasing design combined with fast and efficient static site generation. Material for MkDocs offered the perfect balance, providing a professional appearance, advanced search capabilities, responsive design, and robust customization options that better aligned with the documentation needs of PlantSeg v2.

The documentation and migration processes presented challenges. For instance, automatically capturing screenshots of the Napari widget interface was not straightforward but proved essential for helping users relate the documentation to the software's interface. To achieve this, I manually adjusted Qt—the backend of Napari, a cross-platform application development framework—with the assistance of the bioimage analysis community[12].

By the time of writing, the documentation had evolved from an 11-page plain Markdown-based wiki (21 pages were planned, but 10 remained empty) to a 31-page modern website (with only 2 empty pages due to unfinished features in PlantSeg v2). It is version-controlled alongside the codebase and hosted using GitHub Pages, enabled by a custom GitHub Actions workflow for building and deployment. Users can easily navigate the documentation site, search efficiently for relevant information, and adjust the theme, with widget screenshots dynamically adapting to the selected theme. Python API and Napari widget documentation are automatically generated from the codebase, ensuring consistency and accuracy. The site also includes detailed installation instructions, troubleshooting guides, and contribution guidelines, empowering users to leverage PlantSeg effectively for their research. The improved documentation has received positive feedback from users, highlighting its clarity, organization, and user-friendly design. By enhancing the documentation, I have contributed to making PlantSeg more accessible, user-friendly, and reliable, ultimately supporting the scientific community in advancing their research through bioimage analysis.

[12] Scientific Community Image Forum - Where is `url("theme_{{ id }}:/drop_down_50.svg")` from Napari?

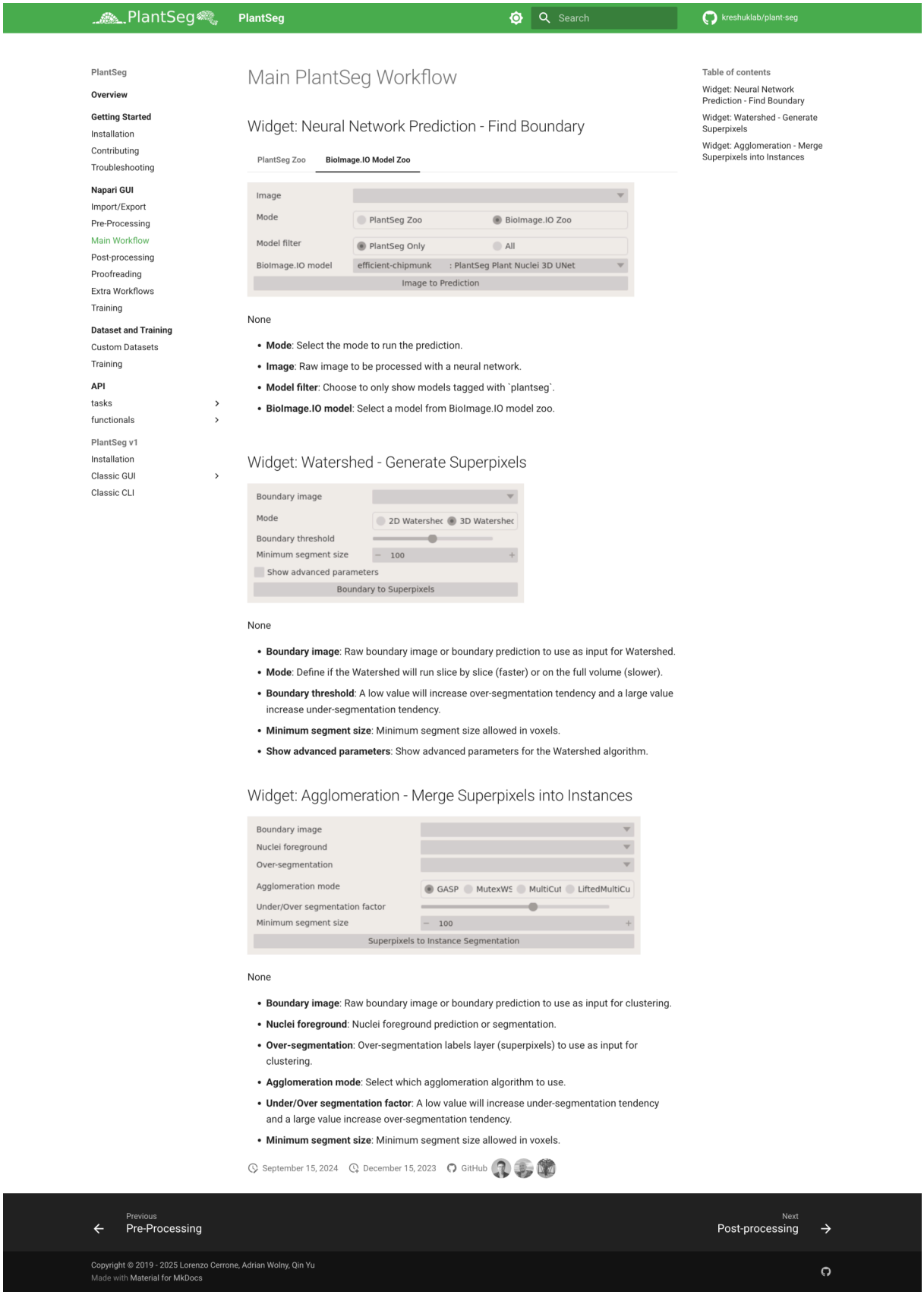


Figure 3.8: PlantSeg 2.0 - Automated Documentation Infrastructure. Documentation for PlantSeg 2.0 is generated automatically using GitHub Actions, which also handle package builds and testing. Any changes to the API or graphical interface are immediately propagated to the documentation site, ensuring consistency and up-to-date guidance for users.

3.3 Advancements Behind the Scenes

I also contributed to extensive architectural and technical improvements in PlantSeg that, while less visible to end users, are critical to the software's functionality, reliability, and maintainability. The architecture of PlantSeg is now organised into three hierarchical layers: functionals, tasks, and widgets. Additionally, core PlantSeg objects, such as `ModelZoo`, `PlantSegImage`, and `VoxelSize`, encapsulate data and metadata into structured and organised forms. My specific contributions included:

- **Multi-Channel and Inter-Channel Support**: I led the development of multi-channel data support and co-developed `PlantSegImage`, a module for managing data and metadata manipulations throughout user-defined workflows. This infrastructure enables the integration of GoNuclear and supports post-prediction and post-segmentation operations, such as Lifted Multicut and automated false-positive removal, leveraging multi-channel data.
- **Voxel Size Support**: I co-developed the `VoxelSize` class to handle voxel size information, enhancing metadata handling. This improvement ensures segmentation results are spatially accurate and biologically meaningful by incorporating scale-awareness into the processing pipeline.
- **PlantSeg Model Zoo and BioImage.IO Model Zoo Support**: As part of the core PlantSeg modules, I designed the `ModelZoo` class for managing models using object-oriented programming principles. This work enabled streamlined and reliable prediction and segmentation workflows, both with native PlantSeg inference and BioImage.IO Core inference.
- **Automatic Halo Computation and Padding Fixes**: I implemented algorithms for automatic halo size computation in UNet models and resolved critical padding bugs. These enhancements eliminated tiling artifacts and significantly improved segmentation accuracy across all PlantSeg models.
- **Hierarchical Functional Structure**: I co-developed the functional-task-widget framework, enabling modular software development. This framework improved typing, logging, and input/output handling, ensuring that the codebase remains maintainable, scalable, and robust.

3.4 Outlook

At the time of writing, PlantSeg v2 is nearing completion. The release of version 2.0.0a0 marks the successful transformation of PlantSeg into a modern, user-friendly, and powerful Napari-based bioimage segmentation tool. With version 2.0.0b0, BioImage.IO Core and the Model Zoo have been fully integrated, enhancing accessibility to state-of-the-art deep learning models. The upcoming release of version 2.0.0 will introduce human-in-the-loop training capabilities, further extending PlantSeg's functionality. The official release and publication of PlantSeg v2 are expected in the coming months, representing a significant milestone in the software's development. Moving forward, I will continue to contribute to its maintenance and further advancements.

PlantSeg 2.0 Workshop at I2K | Oct 2024 | Milano, Italia.

Brewing the SPOCO paper | Mar 2021 | Karlsruhe, Deutschland.

Chapter 4

Domain Adaptation and Efficient Clustering: Boosting Instance Segmentation Under Sparse Object-Level Supervision[1]

Instance segmentation is a fundamental task in computer vision, especially in biomedical imaging, where precisely delineating similar individual objects (e.g., cells, organelles, or anatomical structures) is crucial. Traditional instance segmentation models often require large collections of densely annotated data, a requirement that is particularly challenging in microscopy and other biomedical contexts, where expert labels are time-consuming and expensive to obtain.

To alleviate this constraint, *Sparse Object-Level Supervision for Instance Segmentation with Pixel Embeddings (SPOCO)* [1] was developed. SPOCO reduces the dependency on dense annotations by enabling effective learning from sparsely annotated object instances. This advantage is especially relevant in biomedical imaging, where annotation costs can be prohibitively high and dataset sizes are often limited.

My primary contribution to the SPOCO framework focuses on designing and implementing transfer learning strategies within this sparsely supervised paradigm. Transfer learning is crucial in medical and biological imaging scenarios, where annotated samples may be scarce, yet models must generalise effectively across diverse domains. I developed a transfer learning framework that enables domain adaptation with minimal supervision by leveraging pre-trained models, fine-tuning on sparse target-domain annotations, and incorporating an instance-level consistency loss. Experimental results demonstrated significant improvements in segmentation accuracy across multiple datasets, including both light and electron microscopy images.

Additionally, I explored the post-processing of pixel embeddings to improve efficiency and segmentation accuracy. My investigations focused on:

[1]The SPOCO study, published at CVPR 2022 [1], was co-authored by me (Qin Yu) and led by my former colleague Adrian Wolny. This chapter, while based on that publication, includes substantial unpublished content and focuses exclusively on my methodological contributions, omitting co-authors' parts. Specifically, I did not originate the differentiable instance selection or the consistency loss; however, I led the transfer learning strategy and conducted the systematic model evaluation and clustering method comparisons across diverse datasets. The text has been revised to accurately reflect my contributions. I have also added new insights not included in the original paper.

- **PCA for Embedding Dimensionality Reduction:** I demonstrated that projecting 16D pixel embeddings into a lower-dimensional PCA space (typically 4–6 principal components) can substantially reduce clustering computational cost while maintaining or even improving segmentation accuracy. The first few principal components encapsulate most of the variance in the embeddings, making it possible to discard the remaining, less informative dimensions without degrading clustering performance.
- **Comparison of Clustering Methods:** I evaluated multiple clustering approaches for embedding-based instance segmentation, identifying trade-offs between segmentation accuracy and computational efficiency. While Mutex Watershed is the fastest method, it sacrifices accuracy. Mean Shift, after simple filtering, emerges as the most semantically precise clustering method. HDBSCAN and consistency clustering offer robust results but with higher computational costs.
- **PCA-Projected Embeddings as Input for Instance Segmentation:** By treating the three leading PCA components as RGB channels, I explored the feasibility of using the *Segment Anything Model* (SAM) [84] to segment PCA-transformed embeddings instead of applying clustering. My results indicate that SAM can achieve reasonable instance segmentation from PCA-projected embeddings, particularly in cases where the objects are well separated. However, because PCA removes spatial coherence, the occlusion-aware advantages of high-dimensional embeddings are not fully preserved. Despite this, SAM remains computationally efficient and a viable alternative for simpler instance segmentation tasks.

My research highlights that embedding-based instance segmentation can be optimised through both dimensionality reduction and efficient clustering methods, and the high-dimensional embedding itself is redundant for post-processing though just enough for model training. Moreover, PCA-transformed embeddings can serve as direct input for instance segmentation models like SAM, offering a practical alternative for scenarios where computational efficiency is a priority. These insights expand the versatility of SPOCO, demonstrating its potential for large-scale biomedical image analysis while reducing annotation and computational burdens.

4.1 Sparse Object-Level Supervision for Instance Segmentation with Pixel Embeddings

4.1.1 Embedding-Based Instance Segmentation

Instance segmentation with pixel embeddings [85] offers a conceptually distinct approach compared to classical proposal-based or recurrent methods. Rather than predicting instance masks or bounding boxes directly, a neural network maps each pixel into an n-dimensional *embedding space*, where pixels of the same instance cluster together, and pixels of different instances lie farther apart. A post-processing step then clusters the pixel embeddings to reconstruct distinct instance masks in the original image space.

A core innovation in embedding-based segmentation is the introduction of *discriminative losses* that encourage desired properties in the embeddings:

- A **variance term** that pulls embeddings of the same instance closer together.
- A **distance term** that pushes embeddings of different instances apart.
- A **regularization term** that prevents uncontrolled divergence of embeddings.

These forces are typically margin-based or "hinged," activating only when embedding distances are within a specified threshold. Unlike methods that rely on large sets of proposals or iterative region-refinement, embedding-based instance segmentation directly learns a permutation-invariant representation well-suited for separating multiple objects, even when labels are partial or weak.

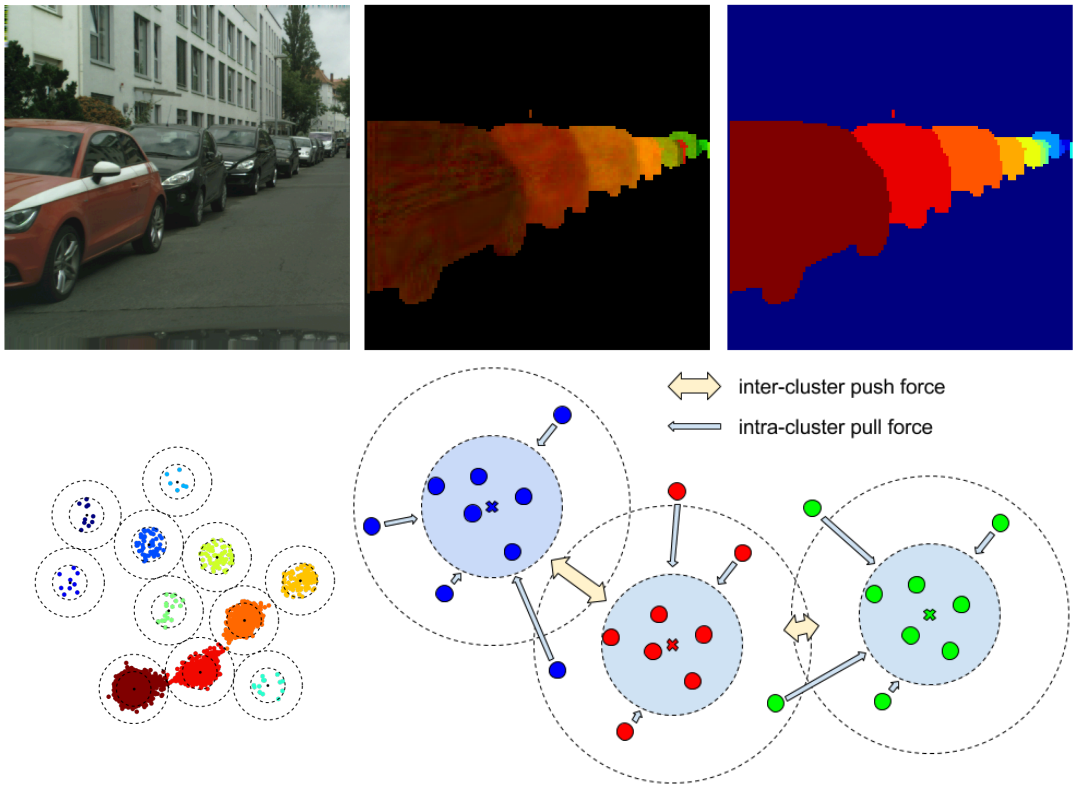


Figure 4.1: Discriminative Loss Functions. A schematic view of pixel-embedding instance segmentation. Each pixel is mapped to a point in feature space so that pixels of the same instance lie close to each other, while embeddings of different instances are far apart. The final step clusters pixels using a fast post-processing method. Top: input image, learned embeddings in 2D, and clustered output. Bottom Left: pixel embeddings forming clusters in embedding space. Bottom Right: illustration of the pull force (encouraging intra-cluster compactness) and the push force (encouraging inter-cluster separation) up to a specified margin (dotted circles). Figure adapted from [85].

In biomedical imaging, embedding-based methods excel because: (i) fine-grained, pixel-level annotations are often expensive to obtain; (ii) objects may appear blurred or overlapping; and (iii) robust generalization is required despite limited training data. SPOCO leverages these embedding-based ideas, adding additional mechanisms to cope with sparse annotations and transfer across domains.

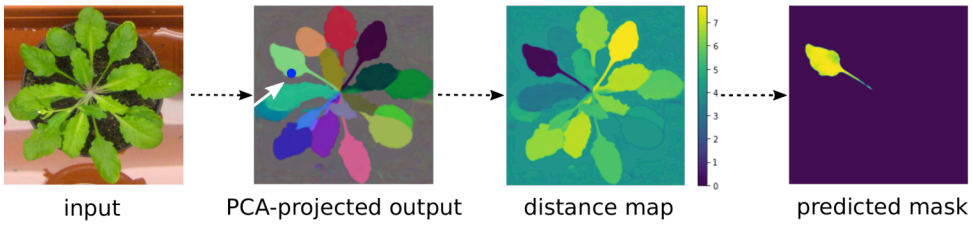


Figure 4.2: Differentiable instance selection for non-spatial embedding networks.. An anchor pixel is chosen randomly or based on the ground-truth instance. A distance map in the embedding space from that anchor to all pixels forms a soft mask of the instance. Figure adapted from [1], originally created by A. Wolny.

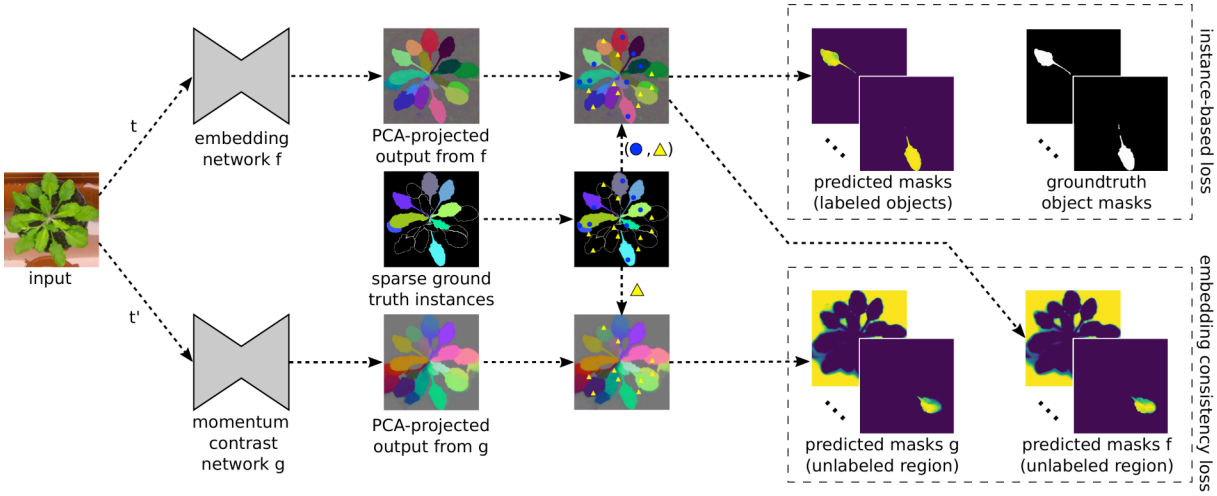


Figure 4.3: Overview of training procedure.. Two augmented views of an input image pass through two embedding networks $f(\cdot)$ and $g(\cdot)$. For labelled objects (blue dots), each anchor's instance is extracted (Fig. 4.2), and an instance loss aligns the prediction with the ground truth. For unlabelled regions (yellow triangles), pairs of corresponding instances from $f(\cdot)$ and $g(\cdot)$ must match via a consistency loss. Figure adapted from [1], created by A. Wolny and Q. Yu.

4.1.2 Sparse Object-Level Supervision

SPOCO [1] addresses the high cost of dense annotations by requiring only *sparse object-level supervision*. Rather than labeling every pixel, one only needs to label a small fraction of objects in each training image. Three key ideas underlie SPOCO's performance:

Sparse Supervision: SPOCO can learn from partial instance labels (a small subset of labelled objects per image) in a "positive unlabelled" setting, well-suited to biomedical images where many objects in a large field of view remain unlabelled.

Differentiable Instance Selection: Through a novel approach, SPOCO samples "anchor pixels" associated with labelled instances. A distance map in the embedding space then forms a soft mask for the corresponding instance. Because this mask selection is differentiable, instance-level losses can be applied end-to-end.

Consistency Loss for Unlabelled Regions: To avoid drift in unlabelled regions, SPOCO imposes a consistency loss that enforces similar embeddings across augmented views. Consequently, unlabelled pixels help stabilise the embedding space, enhancing discriminative features.

Figures 4.2 and 4.3 illustrate the differentiable selection procedure and the overall workflow, respectively. These ideas allow SPOCO to achieve state-of-the-art instance segmentation results in both 2D and 3D biomedical datasets, all while reducing the need for dense, labor-intensive annotations.

4.2 Transfer Learning in SPOCO

Within the SPOCO framework, my primary contribution focused on enhancing the transfer learning methodology to further boost model performance and adaptability across different datasets.

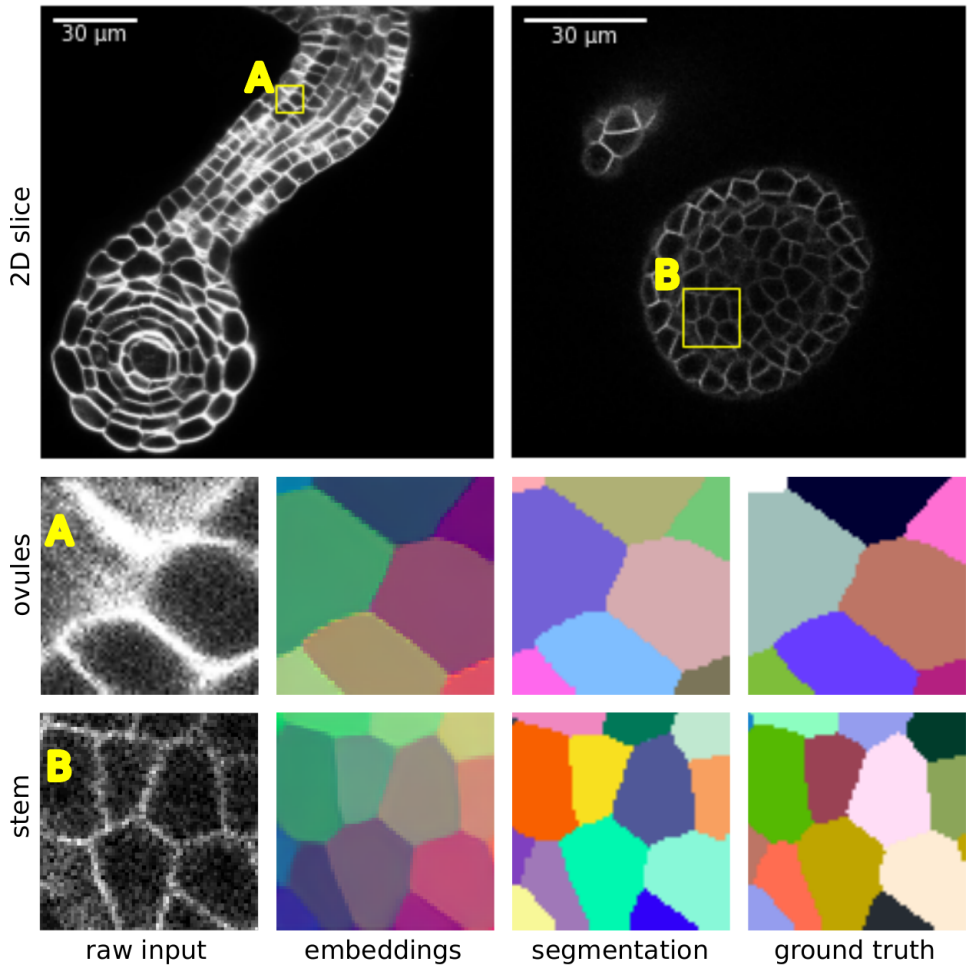


Figure 4.4: LM segmentation in standard and transfer learning settings.. Top) samples from the 3D Ovules (left) and Stem (right) datasets; **Middle)** segmentation of a selected patch (A) from the source domain; **Bottom)** output of the source (Ovules) network fine-tuned with 1% of instances from the target (Stem), and the corresponding segmentation of a selected patch (B). Figure adapted from [1], originally created by Q. Yu and A. Wolny.

4.2.1 Transfer Learning Methodology

The transfer learning strategy in SPOCO extends the model's capabilities by fine-tuning pre-trained models on new target domains using minimal additional training data. Key compo-

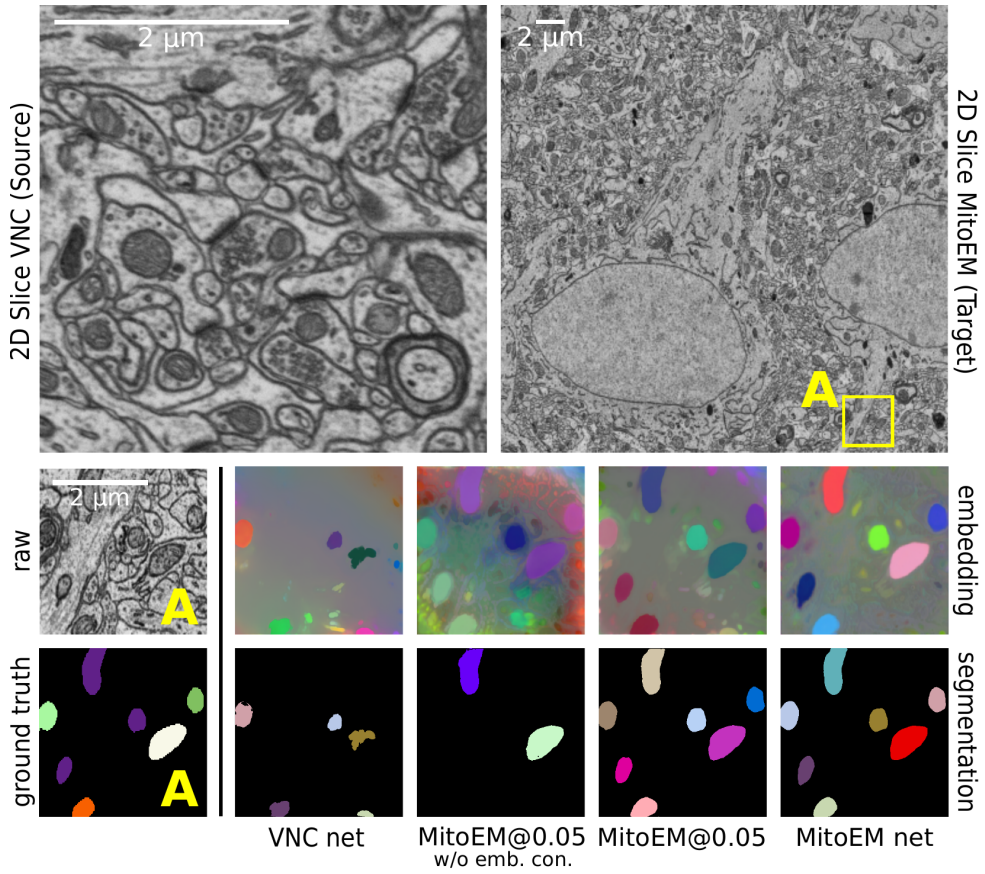


Figure 4.5: EM segmentation in the transfer learning setting.. Top) samples from the source (VNC, left) and target (MitoEM, right) datasets; **Middle)** the input image and the RGB-projected embeddings: trained on VNC only, VNC-pretrained + MitoEM@0.05-finetuned without embedding consistency, same but with embedding consistency, trained on MitoEM only; **Bottom)** ground truth and predicted segmentations. Figure adapted from [1], originally created by Q. Yu and A. Wolny.

nents include:

Source Domain Pre-training: Models are initially trained on fully annotated datasets in a source domain, such as the *Arabidopsis* ovule dataset, where comprehensive ground truth is available.

Target Domain Fine-Tuning: The pre-trained models are then adapted to new target domains using sparse annotations. This fine-tuning leverages SPOCO's weakly supervised framework, where only a small fraction of the available target-domain objects are labelled. The goal is to align and adapt the learned pixel embeddings from the source domain to the target, preserving or improving segmentation quality.

Consistency Loss Integration: To stabilise training and enhance performance in the target domain, the consistency loss is employed. This term ensures consistent pixel embeddings across augmented views, which is critical for accurate segmentation with sparse annotations.

4.2.2 Results and Impact

Incorporating transfer learning into SPOCO significantly boosted segmentation accuracy across diverse, challenging datasets, including both light microscopy (LM) and electron microscopy (EM):

Segmenting Cells in Light Microscopy Datasets: When transferring from the *Arabidopsis* ovule dataset to the stem dataset, segmentation accuracy almost doubled with just 5% of target-domain annotations, relative to models trained only on the source domain.

Segmenting Mitochondria in Electron Microscopy Datasets: For EM data, fine-tuning on the MitoEM dataset with just 1% of annotated objects yielded a 1.5-fold improvement in mean average precision (mAP) compared to models trained on the source domain alone.

These results underscore the versatility of SPOCO's transfer learning framework and its suitability for scenarios where dense annotations are impractical.

Method	AP@0.5	mAP
@0.01	0.368 ± 0.022	0.247 ± 0.022
@0.01 w/o L_{u_con}	0.306 ± 0.014	0.210 ± 0.008
@0.05	0.398 ± 0.007	0.277 ± 0.006
@0.05 w/o L_{u_con}	0.319 ± 0.002	0.227 ± 0.002
@0.10	0.389 ± 0.013	0.268 ± 0.007
@0.10 w/o L_{u_con}	0.301 ± 0.012	0.212 ± 0.007

Table 4.1: Ablation of the consistency term L_{u_con} in the transfer learning setting with 1%, 5%, 10% of ground truth objects (target domain).. Average precision for MitoEM mitochondria segmentation is reported. The VNC dataset serves as the source domain. Mean $\pm$ SD are reported across 3 random samplings of instances from the target dataset.

Method	ARand error
Stem only	0.074
Ovules only	0.227
Ovules+Stem@0.01	0.141 ± 0.002
Ovules+Stem@0.05	0.109 ± 0.002
Ovules+Stem@0.1	0.106 ± 0.004
Ovules+Stem@0.4	0.093 ± 0.003
Ovules+Stem@0.8	0.090 ± 0.003

Table 4.2: Evaluation on a 3D LM dataset in a transfer learning setting. The Ovules dataset acts as the source domain, Stem as the target.. Performance is shown as Adapted Rand Error (ARand); lower is better. Mean $\pm$ SD are reported across 3 random samplings of instances from the target dataset.

4.3 Efficient and Alternative Embedding Processing

As outlined in [1], one primary limitation of the original SPOCO framework is the absence of a universal clustering algorithm capable of consistently partitioning pixel embeddings into instance masks across diverse benchmarks. Multiple clustering algorithms have been proposed, yet none emerges as a clear winner in all scenarios. In this section, I examine several clustering methods and evaluate how dimensionality reduction, via principal component analysis (PCA), can boost both computational efficiency and, in some cases, segmentation accuracy. I also investigate whether the Segment Anything Model (SAM) [84] can serve as a universal clustering strategy for PCA-projected embeddings.

My experiments reveal that the 16-dimensional pixel embeddings used in SPOCO are essential during training but become somewhat redundant for post-processing steps such as clustering. During training, the discriminative loss encourages embeddings of the same instance to cluster tightly, while forcing those of different instances apart in a high-dimensional space where all directions are treated equally. This flexibility is necessary to accommodate the simultaneous forces of attraction and repulsion. In contrast, during post-processing, the embedding space is already well-structured, and each dimension contributes little to the task of separating instances. This structured feature space can thus be compressed into a lower-dimensional subspace with minimal information loss. A parallel yet fundamentally different observation was made in [86], where the authors introduce learnable "register" tokens in Vision Transformers (ViTs) to provide explicit computational storage. In both cases, redundancy in representational space is repurposed for computational utility, albeit in different architectural contexts.

4.3.1 Clustering Methods for Pixel Embeddings

Numerous clustering methods have been proposed to convert dense pixel embeddings into instance masks. Each method offers its own balance of segmentation quality, computational overhead, and robustness.

HDBSCAN. A hierarchical density-based algorithm that detects clusters by examining local density variations. This approach is suitable for crowded or overlapping objects because it can adapt to different instance sizes and filter out noise points. However, its high computational cost can be prohibitive for large datasets, and parameter tuning (*min_size*, etc.) can be challenging.

Mean Shift. Mean Shift is a mode-seeking algorithm that moves each pixel embedding toward the nearest local density peak, grouping pixel embeddings into coherent clusters. However, for embeddings with lower quality, it can over-segment near object boundaries and is computationally expensive relative to density-based approaches like HDBSCAN.

Consistency Clustering. Consistency clustering augments Mean Shift by requiring that stable regions be consistent between multiple augmented embeddings. By validating cluster assignments across pairs of transformations, it reduces over-segmentation but increases computational time.

Affinity-Based Clustering (Mutex Watershed). This graph-partitioning method models pixel relationships through an affinity graph, incorporating both short- and long-range connections. Mutex Watershed runs much faster (often thousands of times faster) than HDBSCAN or Mean Shift, although its segmentation quality is slightly lower in weakly supervised settings.

Since no single method prevails in all benchmarks, selecting a clustering approach depends on the trade-off between computational speed and segmentation precision.

4.3.2 Embedding Dimensionality Reduction

SPOCO produces 16-dimensional pixel embeddings, which can be expensive to cluster, especially for large datasets. Dimensionality reduction can substantially cut computational costs. However, it must preserve the essential features for distinguishing instances. Here, I explore PCA to maintain segmentation accuracy while reducing dimensionality.

PCA for Dimensionality Reduction. Principal component analysis (PCA) is a well-established technique for reducing the dimensionality of high-dimensional data while preserving most of its variance. It performs a linear transformation that reorients the original embedding space along orthogonal axes of maximal variance, known as *principal components*. Actual dimensionality reduction occurs only when a subset of these components is retained. Although many alternative dimensionality reduction methods exist, PCA strikes a particularly effective balance between simplicity, computational efficiency, and a non-destructive transformation, making it especially suitable for the aims of this study.

To apply PCA to the SPOCO embeddings, one projects the original 16-dimensional embedding vectors into a new 16-dimensional space of principal components. Selecting the first N components, which capture the most variance, effectively "projects" embeddings into a lower-dimensional subspace. If N is chosen so as to retain sufficient discriminative information, segmentation remains accurate while the computational cost decreases. Identifying an optimal N involves balancing these two objectives.

For visualisation, embeddings were always PCA-projected into three dimensions and represented as RGB images in the SPOCO paper. This technique proved so intuitive and effective that it was frequently cited independently of the main segmentation method. As shown in Figure 4.4 and Figure 4.5, PCA-projected embeddings provide valuable insights into the clustering of pixel embeddings and their role in instance segmentation. More intriguingly, these 3D PCA-projected embeddings often resemble segmentation masks, which led me to investigate whether they could be directly utilised for instance segmentation.

Hypothesis and Potential Applications. I hypothesised that the 16-dimensional pixel embeddings are somewhat redundant, which can be transformed into a smaller subset of crucial dimensions for instance separation. By applying PCA, these key features could be distilled into the leading principal components. In principle, focusing on the first few principal components might not only reduce computational cost but also remove noise, improving clustering robustness. Moreover, if the 3D PCA-projected embeddings retain enough discriminative structure, they could be fed directly into instance segmentation modules that takes RGB input,

for instance, SAM or other classical algorithms.

4.3.3 Application to the CVPPP Dataset

To test these ideas, I applied PCA-based dimensionality reduction to the CVPPP dataset, known for plant images with overlapping leaves (Figure 4.6 and Figure 4.7). In the original SPOCO workflow, clustering was done only on the full 16-dimensional embeddings. My experiments address whether compressing these embeddings into fewer principal components can preserve or even improve segmentation quality while markedly accelerating clustering. Results confirm the trade-offs between dimensionality reduction and segmentation performance, showcasing the potential of PCA-based embeddings as a more lightweight yet accurate alternative for instance segmentation.

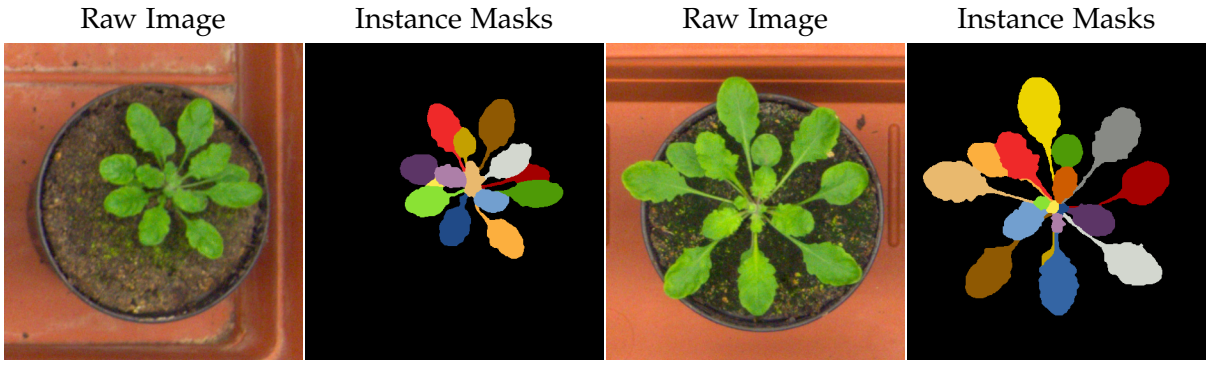


Figure 4.6: CVPPP Training Dataset. Left to right: raw RGB photo of plant #147 in the CVPPP dataset, the corresponding instance masks, raw RGB photo of plant #149 in the CVPPP dataset, the corresponding instance masks. In plant 147, the occluded leaf on the left (only a narrow portion is visible) is carefully annotated as a separate instance, whereas some smaller leaves in the centre are combined into a single instance. In plant 149, the bottom occluded leaf is annotated as a separate instance, but the middle leaf is merged, and smaller instances are annotated separately.

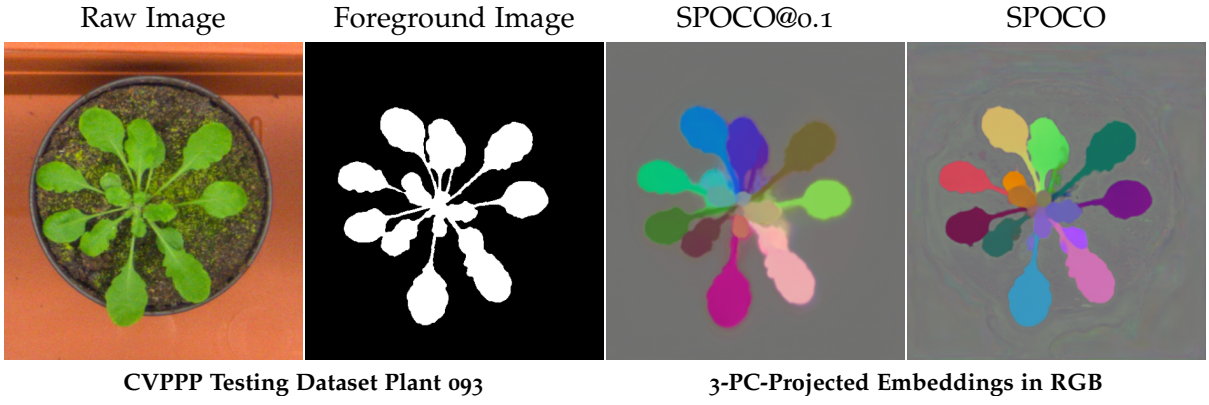


Figure 4.7: CVPPP Testing Dataset and SPOCO Embeddings. Left to right: raw RGB photo of plant #93 in CVPPP, the corresponding binary foreground (semantic) mask, the RGB PCA-projected embeddings from the SPOCO@0.1 model, and the RGB PCA-projected embeddings from the fully supervised SPOCO model. The test dataset does not provide instance masks.

The key difference between the N dimensions of pixel embeddings and the N dimensions of principal components lies in their roles and transformations. Pixel embeddings are directly influenced by the discriminative loss during training, meaning that they are actively pulled

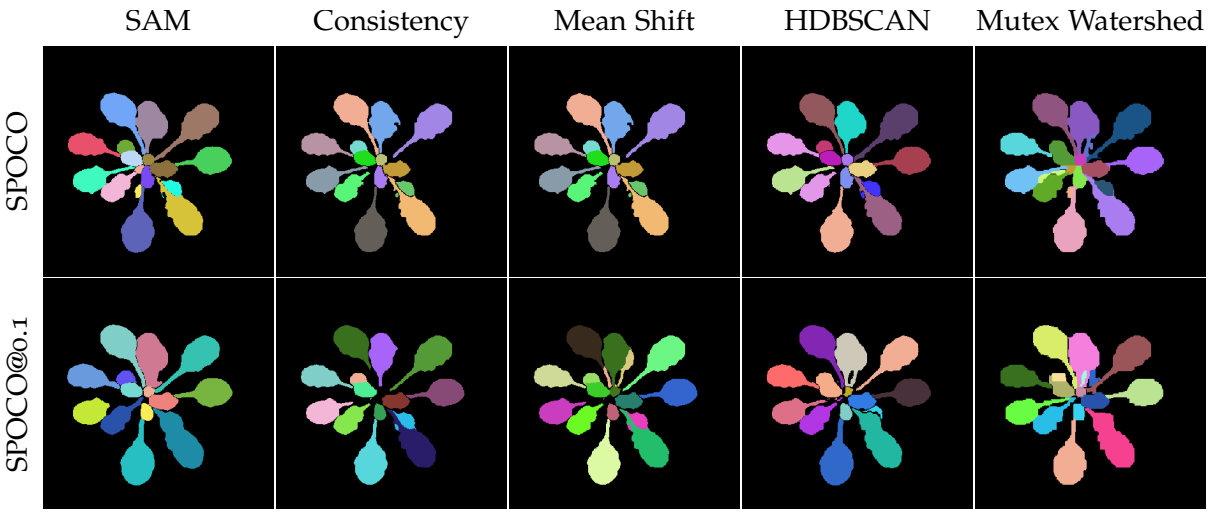


Figure 4.8: Clustered CVPPP Pixel Embeddings by Method. Segmentation (instance masks) of plant #93 from the CVPPP test set, obtained using SAM, consistency clustering, mean shift, HDBSCAN, and Mutex Watershed. The first row shows results for the fully supervised SPOCO model; the second row shows results from SPOCO trained with only 10% of the instance masks. Clusters are filtered by size and other shape-based heuristics.

closer within instances and pushed apart between different instances in the high-dimensional embedding space. This process structures the embedding space in a way that ensures spatial relationships between pixels correspond to instance identities, but it does not necessarily compress the data—each dimension remains equally informative for clustering.

In contrast, principal components are statistical transformations of the original embedding space that capture the directions of maximal variance. Instead of being learned to optimise instance discrimination, PCA identifies the most important axes of variation within the dataset, compressing the feature representation by aligning embeddings with these dominant variance directions. As shown in Figure 4.9, the signal-to-noise ratio across the original embedding dimensions is relatively uniform, meaning that each dimension contributes equally to defining instance relationships. However, Figure 4.10 reveals that the signal-to-noise ratio in the principal components is not uniform—variance is concentrated in the first few components, while later components contain increasingly less informative details.

This observation led me to hypothesise that only a small subset of principal components—specifically, the first 4–6—would be necessary to retain the key instance-discriminative information required for clustering. By discarding the remaining components, it should be possible to substantially reduce computational costs without sacrificing segmentation accuracy. This hypothesis was qualitatively validated in Figure 4.11 and Figure 4.12, where clustering results using reduced PCA embeddings closely matched those obtained with the full 16-dimensional pixel embeddings. Quantitative evidence can be found in the next section. This finding suggests that PCA can serve as an effective dimensionality reduction tool for embedding-based instance segmentation, making clustering more efficient while preserving accuracy.

Furthermore, this insight motivated an exploration of using PCA-projected embeddings as input for instance segmentation models like SAM. Since PCA reduces dimensionality while preserving instance structure, treating the first three principal components as pseudo-RGB channels for SAM segmentation is a logical extension. However, as discussed in subsec-

tion 4.4.4, PCA projection removes spatial coherence to some extent, limiting SAM's ability to fully leverage the occlusion-aware advantages of embedding-based segmentation. Nevertheless, SAM remains computationally efficient and offers a complementary approach to clustering methods like Mutex Watershed when working with PCA-transformed embeddings.

These experiments suggest that substituting the original 16D embeddings with a smaller number of PCA components is a viable strategy for large-scale or time-sensitive applications. In principle, one could also treat the 3D PCA-projected embeddings like ordinary RGB channels and feed them into another segmentation model (e.g., SAM), though the inherent spatial coherence of the original high-dimensional embeddings is partially lost. I return to this point in Section 4.4.4.

4.4 Benchmarking and Comparative Analysis of Embedding Processing

This section compares various clustering algorithms for SPOCO embeddings, focusing on both computational efficiency and segmentation accuracy. I further assess whether PCA can compress 16D embeddings without degrading results. Lastly, I compare embedding-based clustering approaches with the Segment Anything Model (SAM) [84], highlighting trade-offs between speed and accuracy.

4.4.1 Computational Efficiency Assessment

The standard SPOCO pipeline clusters 16-dimensional embeddings to derive instance masks, but clustering in high-dimensional spaces can be computationally heavy. To mitigate this, I applied PCA to reduce dimensionality before clustering, then measured runtime with four clustering algorithms under different numbers of principal components.

Figure 4.15 confirms that runtime grows with embedding dimensionality for all methods, especially for HDBSCAN, mean shift, and consistency clustering. Mutex Watershed remains faster, though it yields coarser segmentation. Crucially, using only four or five principal components significantly cuts runtime with little or no accuracy loss.

Additionally, I tested SAM on GPU. SAM took only a few seconds per image, faster than most clustering methods except Mutex Watershed. However, as shown later, SAM's segmentation accuracy also falls between that of Mutex Watershed (fastest but less accurate) and the more computationally demanding clustering methods.

4.4.2 Evaluation Metrics and Strategies

The CVPPP dataset provides raw RGB images, instance masks, and semantic masks in the training set, while the test set includes only raw RGB images and semantic masks. To evaluate the segmentation accuracy of clustering methods, I use the Symmetric Best Dice (SBD) metric. Since the best-performing SPOCO models—those trained on all available training data—are used for evaluation, there is no instance mask as ground truth for direct comparison.

Although quantitative scores are commonly reported in publications for rigour, they can sometimes obscure the actual understanding of methods and results. This is because no ground truth annotation is perfectly accurate for precise benchmarking. While it is useful to approximate how well a method aligns with the available annotations, the absolute difference between a segmentation and a given ground truth becomes less informative when both are close to the actual biological truth. Errors in annotation, segmentation, and evaluation contribute to discrepancies that may not necessarily reflect meaningful differences. Figure 4.6 illustrates this point: while the official ground truth instance masks accurately classify pixels as belonging to leaves, they often fail to correctly distinguish between individual leaf instances.

To ensure a meaningful evaluation, I compare the clustered embeddings, or instance segmentations, against the ground truth semantic masks. This approach allows assessment of how well the segmentation captures leaf regions, which is sufficient for the scientific discussion in this section.

For the comparison with SAM segmentation, instead of evaluating against semantic masks, I compare SAM's output with segmentations produced by the clustering methods to highlight differences in instance-level segmentation. The key distinction between SAM and clustering-based methods is that SAM operates on 2D RGB images and does not inherently account for occlusions or overlapping instances. In contrast, clustering methods leverage 16D pixel embeddings, which provide a spatially meaningful representation of instances and are specifically designed to handle occlusions. Since these 16D embeddings are projections of 2D instances into a high-dimensional space, they encode instance relationships more effectively. The instance-level SBD metric highlights these differences.

The Symmetric Best Dice (SBD) metric is used to estimate the average leaf instance segmentation accuracy. It is derived from the Dice-Sørensen coefficient, a statistical measure of similarity between binary or semantic segmentation masks. Given two multi-instance segmentations, A and B, where $A = \{A_1, A_2, \ldots, A_M\}$ and $B = \{B_1, B_2, \ldots, B_N\}$ represent the sets of instances in segmentations A and B, respectively, the Dice coefficient between two instances A_i and B_j is defined as:

$$\mathrm{Dice}(A_i, B_j) = \frac{2|A_i \cap B_j|}{|A_i| + |B_j|} = \frac{2\ \mathrm{TP}}{2\ \mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

where TP (true positives), FP (false positives), and FN (false negatives) denote the overlap and misclassification areas.

The best Dice score for each instance in A with respect to B is:

$$D_A = \frac{1}{M} \sum_{i=1}^{M} \max_{j=1,\ldots,N} \mathrm{Dice}(A_i, B_j)$$

Similarly, the best Dice score for each instance in B with respect to A is:

$$D_B = \frac{1}{N} \sum_{j=1}^{N} \max_{i=1,\ldots,M} \mathrm{Dice}(A_i, B_j)$$

The final SBD score is computed as:

$$\mathrm{SBD}(A, B) = \min\left(D_A, D_B\right)$$

For semantic masks, the SBD reduces to:

$$\mathrm{SBD}(A, B) = \mathrm{Dice}(A_1, B_1)$$

It is worth noting that the Dice coefficient is conceptually similar to, but different from, Intersection over Union (IoU), which I introduce in subsection 2.4.2. Additionally, in the context of semantic mask comparison, the Dice coefficient is mathematically equivalent to the F1 score.

4.4.3 Impact of PCA-Projected Embeddings on Clustering Methods

In the original SPOCO paper [1], clustering was performed using the full 16-dimensional embeddings. One of my key aims was to investigate whether dimensionality reduction, specifically via PCA, could preserve or even improve the clustering performance while substantially reducing computational overhead. To this end, I systematically evaluated four clustering algorithms—mean shift, consistency clustering, HDBSCAN, and mutex watershed—using PCA-projected embeddings with varying numbers of principal components.

Dimensionality Reduction Benefits. As illustrated in Figure 4.10, Figure 4.13 and Figure 4.15, PCA can compress the original 16-dimensional embeddings into fewer components without sacrificing key instance-discriminative information. By selecting only the first N principal components—where N typically ranges from 4 to 6—it is possible to achieve near-optimal instance segmentation results while substantially cutting down clustering time. This speed-up is most pronounced for HDBSCAN and mean shift, both of which become prohibitively slow in higher dimensions. Even mutex watershed, already fast relative to other methods, also benefits from the lower dimensionality.

Algorithmic Observations. Figure 4.13 and Figure 4.14 illustrate that each clustering method has distinct pros and cons:

Mean Shift Often over-segments small boundary fragments, but a simple post-processing filter can remove these tiny clusters, thereby yielding top-tier accuracy. Mean Shift requires only 4–5 PCA components for near-optimal performance.

Consistency Clustering Achieves the highest raw accuracy (without extra filtering) but at considerable computational expense. Reducing to 4–5 PCA components lowers runtime but still leaves it slower than other methods.

HDBSCAN Includes a built-in minimum cluster size, filtering out noise or boundary artifacts, so it often needs less manual filtering. However, it remains computationally heavy relative to mutex watershed. PCA helps significantly, especially for embeddings trained with fewer annotations (e.g., SPOCO@0.1).

Mutex Watershed Extremely fast but prone to merging occluded leaves or capturing background noise as small clusters. Limiting dimensionality to 4–5 PCA components provides a good speed/accuracy trade-off.

Overall, PCA-based dimensionality reduction integrates seamlessly with SPOCO's pixel embeddings and clustering pipeline. Even a small subset of principal components typically captures enough information for high-quality segmentations and faster runtimes.

4.4.4 Segment Anything vs. High-Dimensional Clustering Methods

In addition to classic clustering algorithms, I explored the feasibility of using the Segment Anything Model (SAM) [84] as a post-processing tool for PCA-projected embeddings. Whereas SAM operates directly on RGB images and can segment an unconstrained variety of objects, its generic approach may overlook occluded instances in scenes with overlapping structures. By contrast, pixel embeddings from SPOCO explicitly encode instance identity within a high-dimensional space, which is crucial for resolving occlusions. However, once embeddings are PCA-projected, they are no longer spatially structured in a way that preserves occlusion-aware properties. This loss of spatial coherence fundamentally limits SAM's ability to leverage the occlusion-handling strengths of embedding-based methods. However, SAM itself resolves occlusions to some extent as shown in Figure 4.8 SPOCO-SAM segmentation: both parts of the occluded leaf on the bottom right is identified as the same instance (not just the colour but the actual instance label values are the same).

Adapting SAM to PCA-Projected Embeddings. A practical way to harness SAM for instance segmentation is to treat the three leading principal components of a SPOCO embedding as if they were standard RGB channels. This setup effectively "prompts" SAM to find objects in the PCA-projected space. While the original embeddings maintain spatial coherence, PCA transforms them into a new coordinate system where instance-related pixels may no longer form contiguous regions. Although SAM interprets PCA-projected embeddings purely as a colour-based segmentation task, it does recognise that spatially disconnected regions might belong to the same instance.

Comparison with Clustering Methods. Figure 4.16 measures the similarity (via the instance-level metric SBD) between SAM-generated masks and those obtained from HDBSCAN, mean shift, consistency clustering, or mutex watershed. The results, corroborated by Figure 4.15, reveal the following:

- SAM is computationally efficient, running faster than mean shift, HDBSCAN, or consistency clustering, but still slower than mutex watershed. This might be improved by unified memory architectures which is growing in popularity in recent years.
- On simple 2D structures where object instances are well-separated, SAM performs comparably to clustering-based methods on PCA-projected embeddings. Segmentation details could be improved with SAM derivatives such as SAM-HQ [87].

- Unlike embedding-based methods, SAM does not inherently model occlusions. In PC space, instance structure is no longer spatially coherent. However, SAM still tries to distinguish between overlapping objects.
- Among the four clustering methods, mutex watershed's results resemble SAM most closely.

4.5 Conclusion

SPOCO addresses the high annotation demands typical of instance segmentation by leveraging sparse, object-level supervision and embedding-based methods. My contributions focus on transfer learning, extending SPOCO's usefulness across diverse imaging modalities, and on efficient embedding post-processing strategies. Through extensive experiments on the CVPPP dataset, I demonstrate that:

- **PCA-Based Dimensionality Reduction** preserves or even enhances segmentation quality while substantially reducing clustering time.
- **Choice of Clustering Algorithm** depends on the specific balance between speed and accuracy. Mean Shift (with simple filtering) and Consistency Clustering yield the best results at the expense of computational cost, while Mutex Watershed is exceptionally fast but less precise.
- **Segment Anything (SAM)** applied to PCA-projected embeddings can be a practical, lightweight alternative for simpler 2D scenarios, though it may struggle in complex scenes with severe occlusions.

Overall, these findings highlight SPOCO's adaptability: from training with minimal object-level labels to harnessing PCA and clustering algorithms, it achieves high-quality instance segmentation in biomedical settings where dense annotations are prohibitively expensive.

4.6 Challenges and Future Directions

While SPOCO has demonstrated remarkable versatility and efficiency in instance segmentation, several challenges and opportunities remain that warrant further exploration.

Hierarchical Embeddings for Topological Constraints. The current SPOCO framework could be extended to support the joint segmentation of multi-channel datasets. For instance, in the GoNuclear dataset (Figure 2.2), which captures both cell walls and nuclei, a hierarchical embedding strategy could be beneficial. Instead of training separate networks for each channel, a single embedding model could simultaneously enforce both instance-level grouping and hierarchical constraints. Specifically, embeddings could be structured so that (i) all pixels belonging to the same cell cluster together onto a high-dimensional sphere, and (ii) nucleus embeddings nest within their corresponding cell embeddings. This hierarchical

approach would facilitate accurate cell segmentation through conventional clustering while simplifying nucleus segmentation by restricting it to cell-specific embedding regions.

For a broader discussion on multi-channel segmentation strategies and algorithms, see chapter 5.

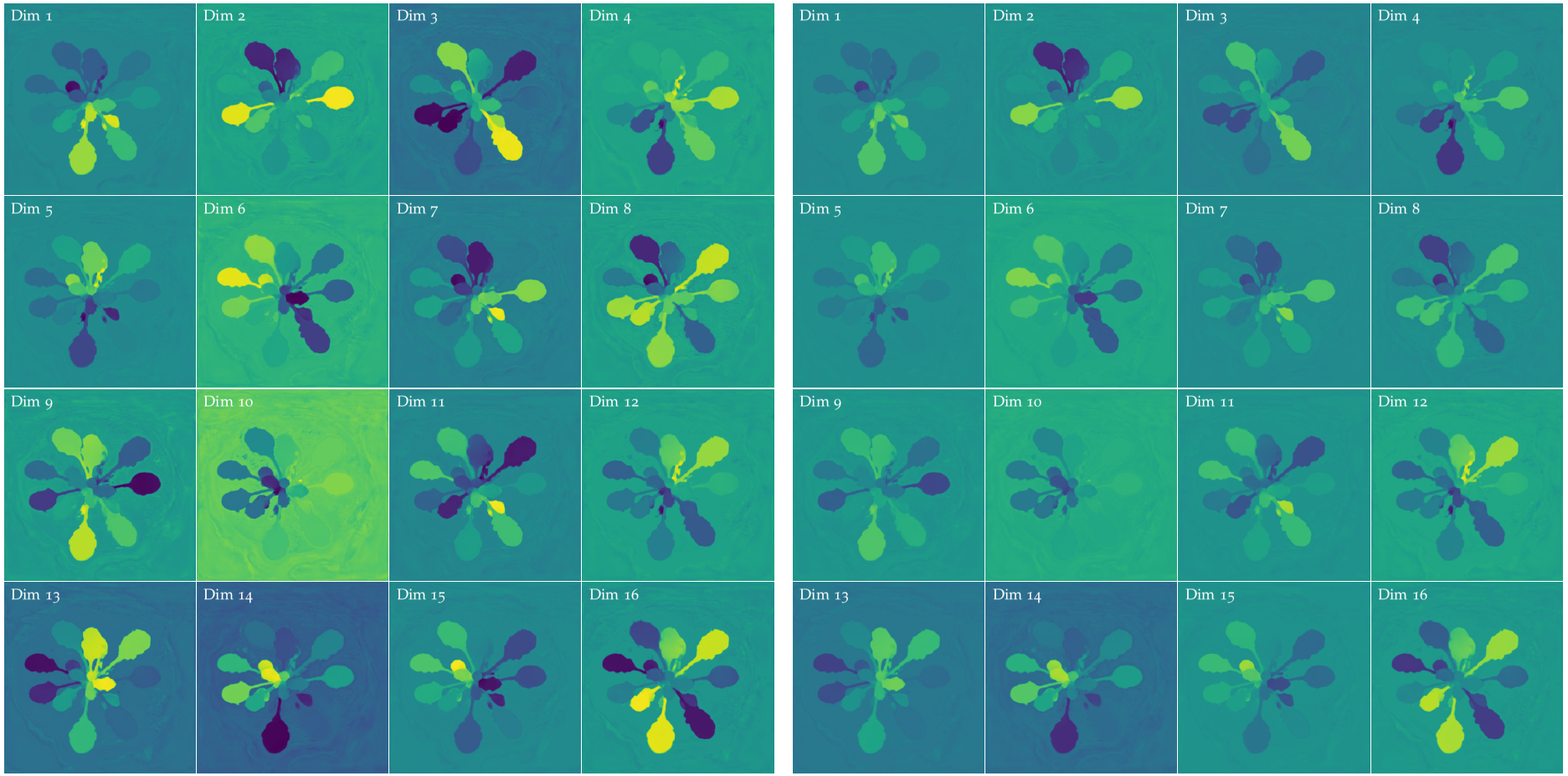
Accessibility and User-Friendliness. Unlike GoNuclear (chapter 2) and PlantSeg (chapter 3), SPOCO does not currently offer a user-friendly interface or pre-trained models for easy adoption. As an advanced deep-learning framework primarily aimed at experienced users, it lacks the accessibility features that would enable wider adoption by biologists and researchers. However, the training and transfer learning capabilities of SPOCO could be integrated into the PlantSeg pipeline to provide a more accessible and user-friendly interface. This integration is planned for the release of PlantSeg 2.0.

Additionally, while Mutex Watershed is available in a separate repository, its incorporation into SPOCO would streamline the clustering workflow and enhance usability. This integration is a planned future development.

SAM-HQ as an Alternative Clustering Method. Recent advances in foundational segmentation models, such as SAM-HQ [87], present an opportunity to explore alternative clustering strategies for PCA-projected embeddings. Building upon the original Segment Anything Model (SAM), SAM-HQ produces segmentation masks with significantly improved spatial accuracy, sharper object boundaries, and enhanced fine-grained details. Although this method incurs slightly increased computational costs due to its refined segmentation process, the trade-off may offer substantial improvements in segmentation quality.

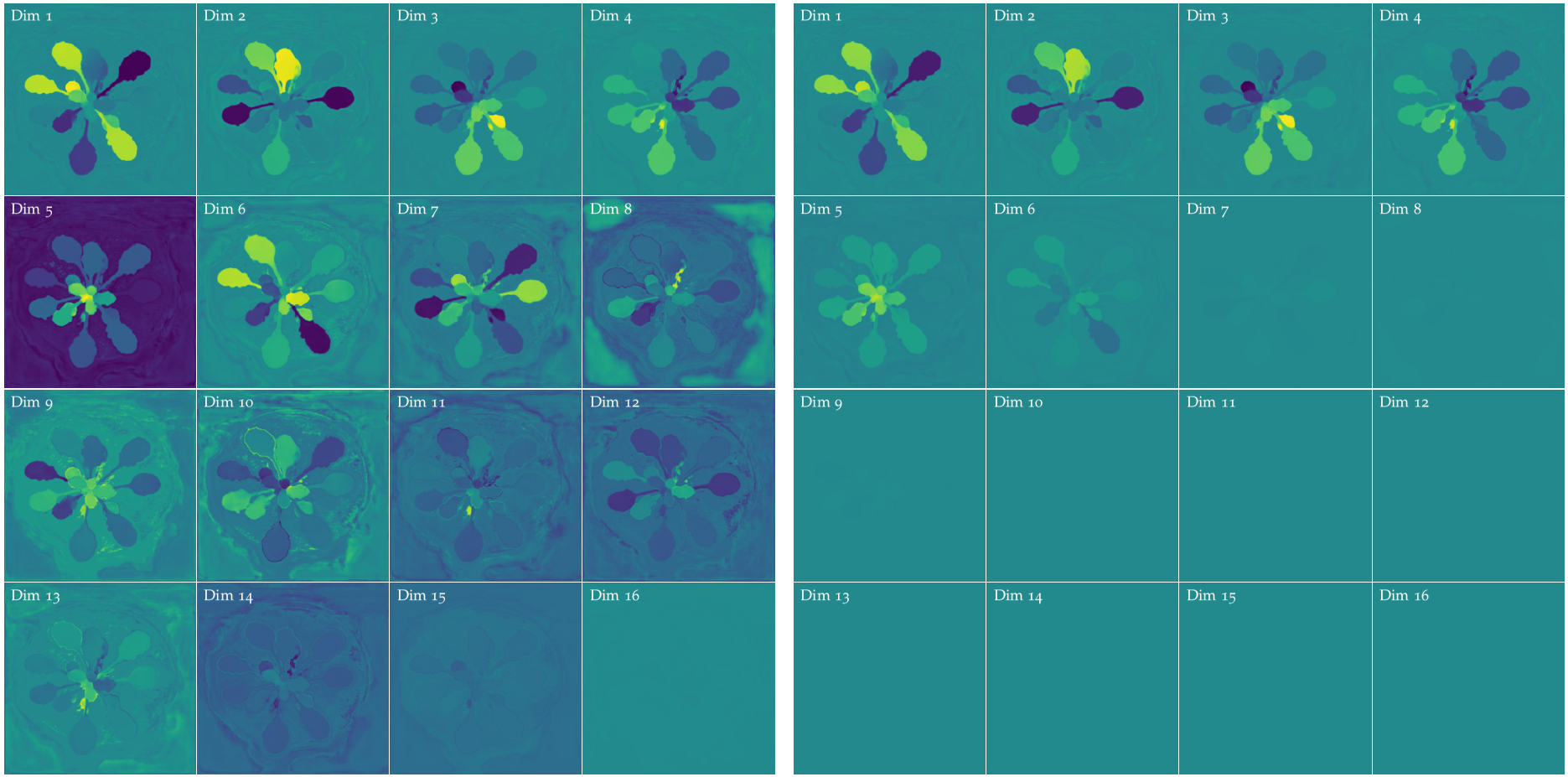
Future work will involve evaluating SAM-HQ's effectiveness as a clustering method for PCA-projected embeddings to determine its feasibility as an alternative to traditional embedding-based clustering approaches. If successful, SAM-HQ could still reduce computational demands while boosting segmentation accuracy.

These future directions highlight the continued evolution of embedding-based instance segmentation. By enhancing the hierarchical structure of embeddings, improving accessibility through integration with user-friendly pipelines, and leveraging advances in generalised segmentation models, SPOCO can be further refined to meet the needs of diverse biomedical imaging applications.



Embeddings Visualised with Local Normalisation **Embeddings Visualised with Global Normalisation**

Figure 4.9: SPOCO Embeddings by Dimension. The 16-dimensional pixel embeddings produced by the fully supervised SPOCO model, visualised dimension by dimension in a 4×4 grid. The left side uses per-dimension normalisation, the right side a global normalisation. The non-normalised version demonstrates that the signal-to-noise ratio is uniform across dimensions, whereas the normalised version highlights that the feature distribution is balanced, indicating that all dimensions should contribute equally to clustering.



Principal Components Visualised with Local Normalisation **Principal Components Visualised with Global Normalisation**

Figure 4.10: PCA-Projected SPOCO Embeddings. All 16 principal components of the fully supervised SPOCO embeddings, visualised dimension by dimension. The left side shows per-component normalisation, the right side uses global normalisation. The non-normalised version highlights how noise becomes more dominant along the principal components, as the first few components encapsulate most of the variance and distinct features of the embeddings. The normalised version further illustrates that the distribution of features across dimensions is not uniform, with the first principal components carrying higher contrast and therefore contributing more significantly to clustering.

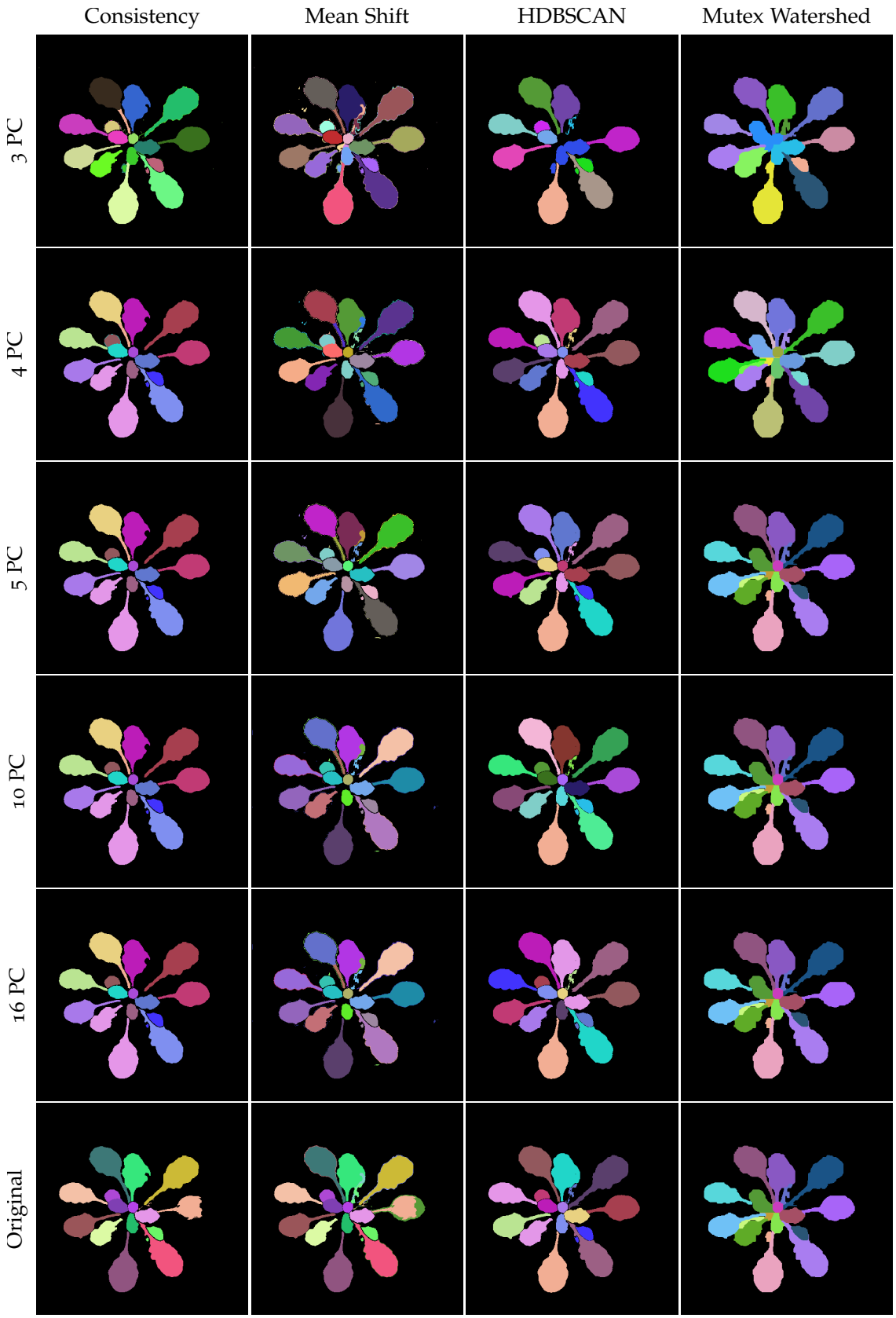


Figure 4.11: Qualitative Comparison: Number of Principal Components for the Fully Supervised SPOCO Model. Each column shows a different clustering method (consistency clustering, mean shift, HDBSCAN, or Mutex Watershed) applied to the first N principal components ($N = 3, 4, 5, 10, 16$). Reducing dimensionality to only 4–5 components preserves most of the segmentation quality while cutting computational cost.

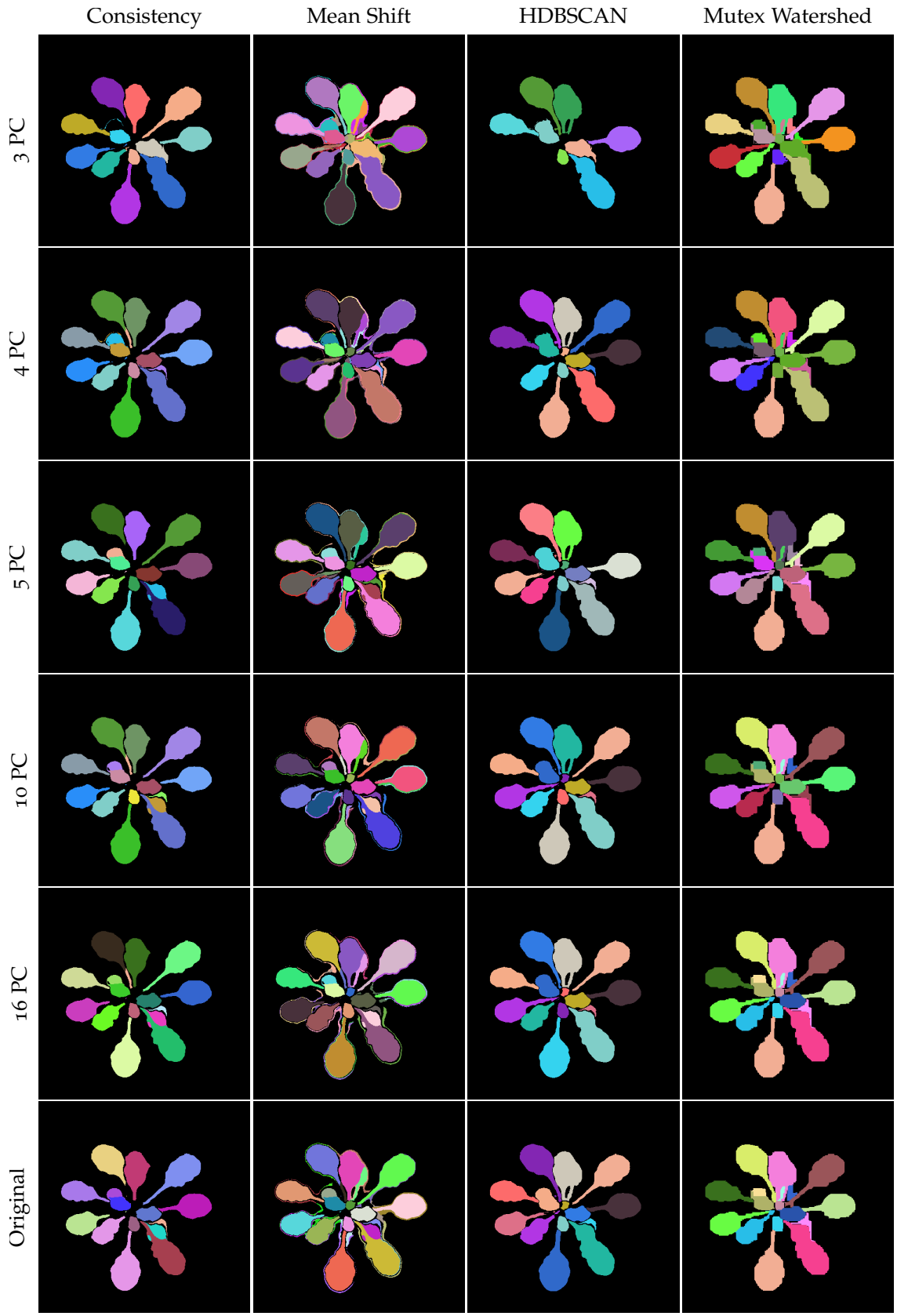


Figure 4.12: Qualitative Comparison: Number of Principal Components for SPOCO@0.1. Similar to Fig. 4.11, but using a SPOCO model trained with only 10% of the instance masks. Clustering remains relatively robust even at lower dimensionalities (N = 4 or 5).

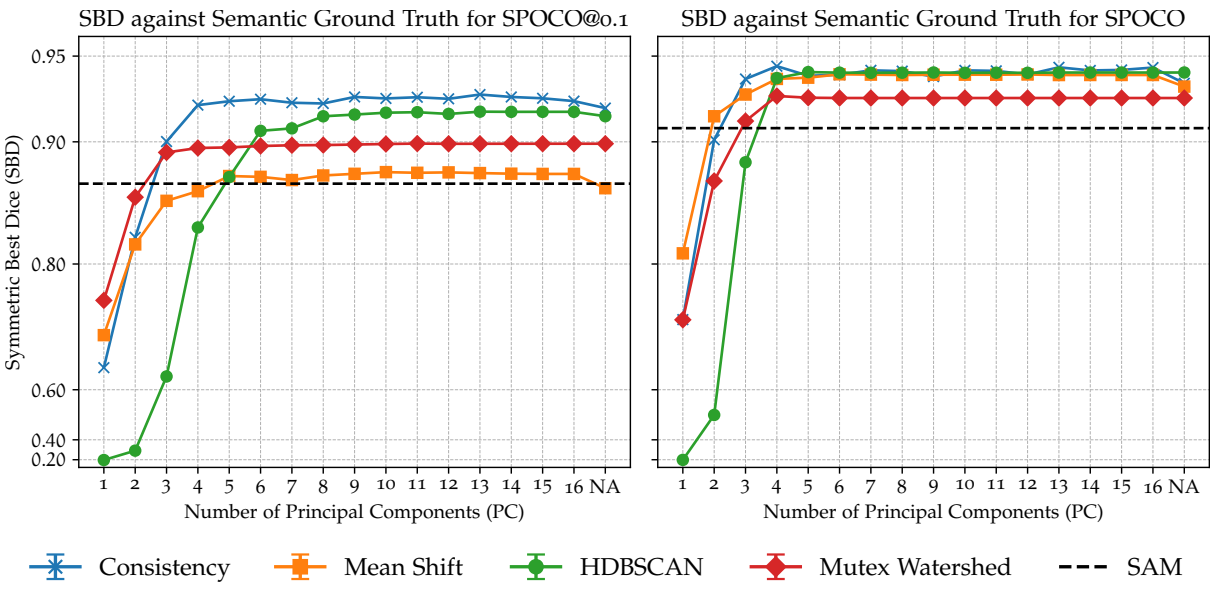


Figure 4.13: Pixel Embedding Clustering Accuracy. SBD(*ground truth*, *binary segmentation*) is plotted for the SPOCO@0.1 model (left) and the fully supervised SPOCO (right). Increasing the number of PCA components does not necessarily improve semantic accuracy. Notably, 4–5 components already suffice in most cases. A version with error bars is in Figure A.5.

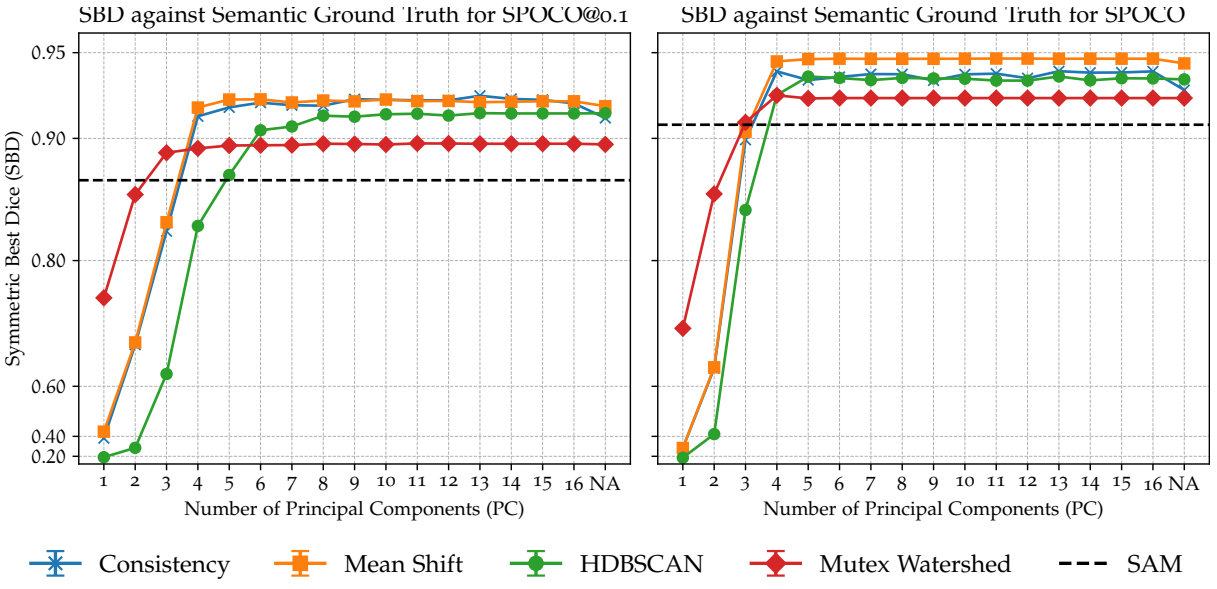


Figure 4.14: Pixel Embedding Clustering Accuracy (Filtered). Same as Figure 4.13, but filtering out small boundary artifacts for Mean Shift greatly boosts its performance, making it the highest-scoring clustering method. SAM has lower semantic accuracy than the embedding-based methods, though it remains much faster than all except Mutex Watershed.

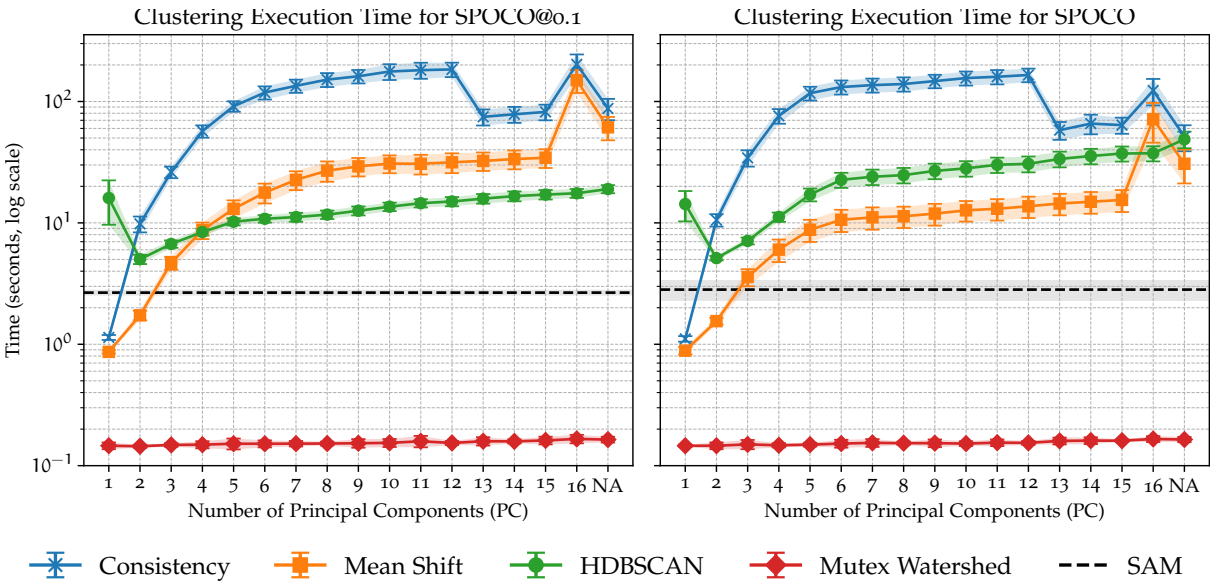


Figure 4.15: Pixel Embedding Clustering: Computational Efficiency. The left plot shows SPOCO@0.1 (trained with 10% of instance masks), the right plot shows the fully supervised SPOCO. In both, higher dimensionality inflates runtime for HDBSCAN, mean shift, and consistency clustering. Mutex Watershed is consistently faster, and SAM (on GPU) completes each image in seconds, about an order of magnitude slower than Mutex Watershed but far faster than the other algorithms. A linear-scale version is in Figure A.4.

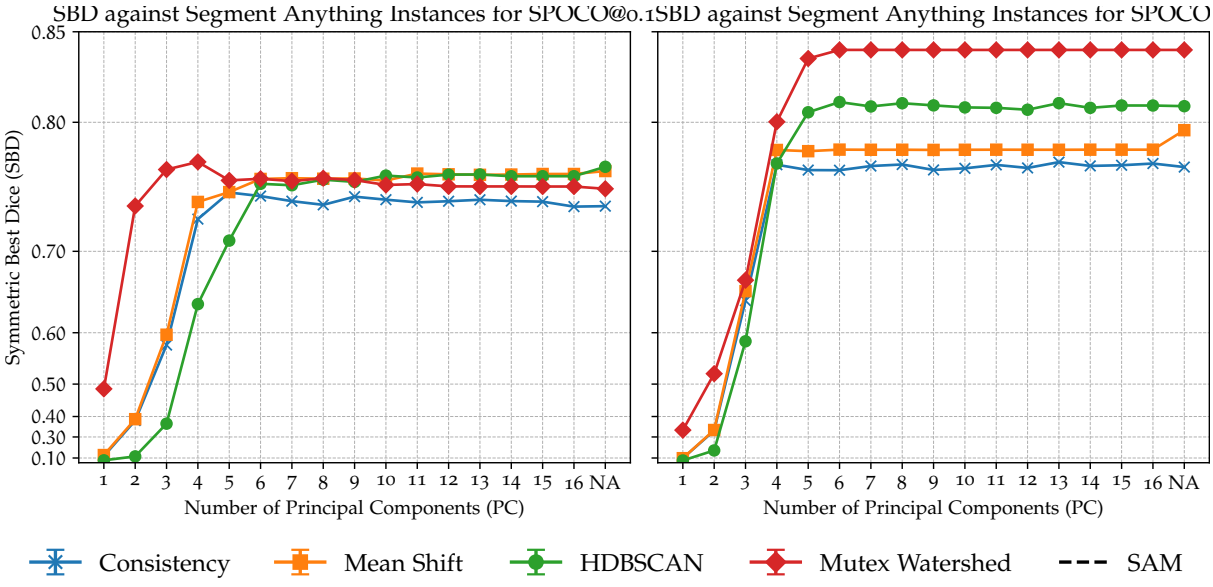


Figure 4.16: Clustering Similarity with SAM. Y-axis: SBD(*SAM instance segmentation*, *clustering results*). Smaller fragments and hollow instances are removed. Left: SPOCO@0.1; Right: fully supervised SPOCO. For high-quality embeddings, SAM's instance masks are moderately close to those of the embedding-based methods. Among the four, Mutex Watershed resembles SAM most closely.

Multi-Channel Imaging - Summer | Aug 2023 | EMBL Heidelberg.

Multi-Channel Imaging - Winter | Dec 2020 | EMBL Heidelberg.

Chapter 5

Multi-Channel Bioimage Analysis: Strategies and Algorithms for Leveraging Additional Information

Multi-channel bioimage analysis provides additional dimensions of information, substantially improving the accuracy and interpretability of biological image analysis. This chapter presents advanced strategies for leveraging multi-channel data, including complementary chemical stains, multiple imaging modalities, temporal information, and combined biological structures within single samples. Each section demonstrates how innovative algorithmic strategies exploit these additional data channels to enhance segmentation and tracking tasks in bioimaging.

5.1 Multi-Label Bioimage Segmentation: Exploiting Complementary Chemical Stains

In bioimage analysis, the term "label" can signify different concepts depending on the context—it may refer either to a chemical stain or to a ground-truth annotation. Here, I use "label" to denote a chemical stain that highlights specific biological structures or molecules in light microscopy images. In multi-label bioimage segmentation, multiple chemical stains are employed to highlight either the same structure or distinct structures within the same biological sample. By integrating the complementary information across these channels, segmentation performance can be improved, leading to more precise structural delineation and deeper biological insights. In subsection 5.1.1, I demonstrate how leveraging the distinct noise profiles of different channels enhances the segmentation of cell surfaces. In subsection 5.1.2 and chapter 2, I present how paired fluorescence stains facilitate both ground-truth generation and the design of human-in-the-loop (HITL) workflows.

5.1.1 Multi-Channel Noise Mitigation for Cell Surface Analysis

In this project, I aim to perform semantic segmentation of cell surfaces within 3D multi-channel images. Although there is currently no established method for accurately segmenting the cell surface, I will demonstrate how leveraging multi-channel information can achieve high-quality results for subsequent surface analyses. Traditional image processing methods often fail to provide reliable segmentations, particularly under significant noise in the actin

channel. While ilastik can be quickly trained with a few expert annotations from collaborators, its segmentations are typically noisy. Despite this imperfect starting point, I used these preliminary ilastik outputs to train deep neural networks.

The adage "rubbish in, rubbish out" remains true: using noisy ilastik labels leads to models that reproduce this noise. However, by introducing another channel with a distinctly different noise profile, I prevented the network from overfitting to the actin-channel noise when segmenting new volumes. As a result, the multi-channel approach demonstrates an effective strategy for attaining high-quality cell surface segmentations.

Biological Motivation: Membrane-cortex attachment is crucial for processes such as cell migration, division, and development, as well as for maintaining cell shape and polarity. The actin cytoskeleton is a dynamic network of filaments regulated by numerous actin-binding proteins. Additionally, various proteins of different sizes can serve as membrane-cortex attachment proteins. To study the effect of linker size in cells, genetically engineered linkers of varying lengths were developed; these linkers appear to influence actin cortex polarization and membrane curvature in a dose- and length-dependent manner. Moreover, membrane viscosity also seems to affect their distribution. Together, these factors necessitate high-fidelity, detailed 3D analyses of the cell surface. Traditional 2D or spherical projections of 3D data are insufficient, motivating the use of comprehensive 3D semantic segmentation.

Technical Motivation: When segmenting the actin channel with ilastik, the results are often noisy due to significant signal disturbances. Furthermore, the negative channel lacks clear detail. This study investigates how deep learning models can be employed to improve segmentation accuracy.

Data. I currently have 9 volumetric datasets, each with three channels (actin, linker, and environment) and three versions of labels:

- **Raw Channels:** Imaged by Ruben Tesoro Moreno, containing the following channels:

  Negative: Anything not part of the cell

  Linker: Artificial linkers tethering actin to the membrane

  Actin: Actin filaments inside the cell

- **Label Channels:**

  Ilastik: Created by manually annotating the actin channel in ilastik. These segmentations are noisy but served as initial training data for deep learning models.

  Initial: Produced by running an early model on the negative channel and thresholding its combined foreground and boundary probabilities at 0.5. Used for a second round of experiments and initial validation steps.

  Initial-thin: Generated by applying the "initial plus" model on the negative channel and thresholding the foreground at 0.5. Used for a third round of experiments.

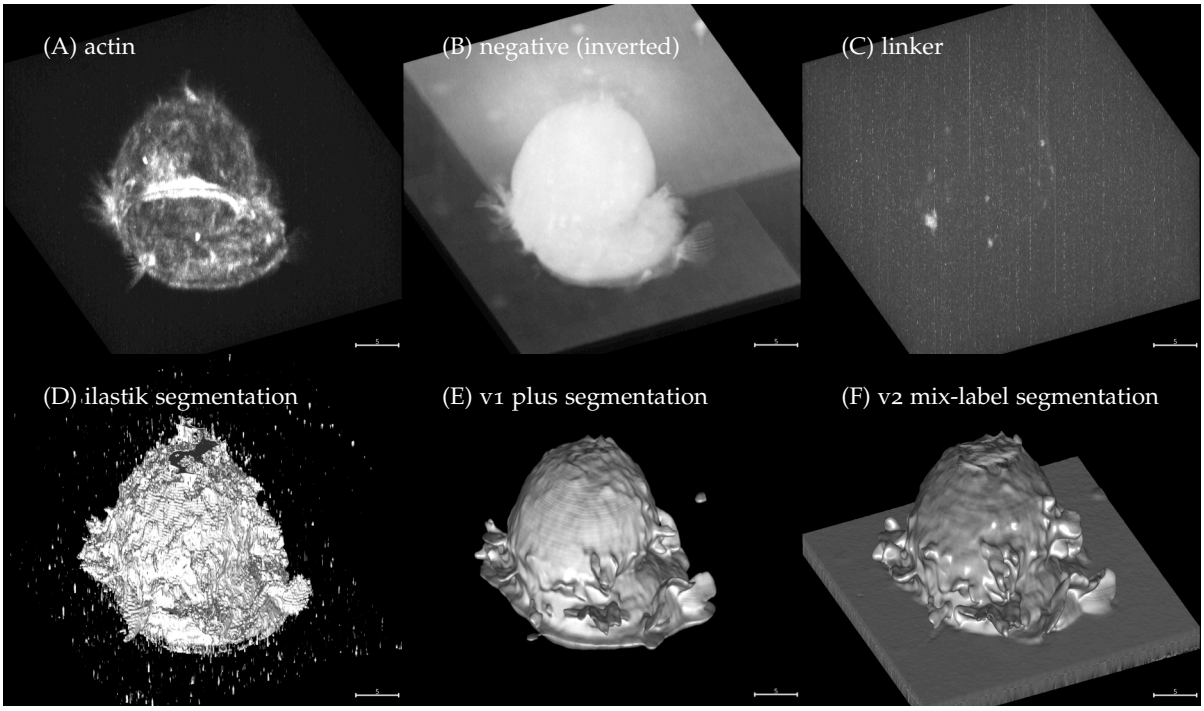


Figure 5.1: Multi-Channel 3D Images, Label, and Outcome. (A–C) multi-channel 3D images of actin, negative, and linker. Negative channel is inverted for visualisation. (D) ilastik segmentation. (E) v1 plus model segmentation. (F) v2 mix-label model segmentation. Scale bars: 5 µm.

Training Strategies. I investigated a series of training strategies and their outcomes, as summarised in Table 5.1. Initially, I received 4 volumes with three channels (negative, linker, actin) and the ilastik label. To test whether a model could learn to ignore out-of-distribution noise, I first trained the "initial" model with only the negative channel as input and the ilastik segmentation as label. Once this model produced promising results, I convinced collaborators to acquire 5 additional volumes under the same imaging conditions.

Subsequently, I trained 14 more models with various combinations of:

1. Different numbers of volumes (4 vs. 9) with ilastik labels;
2. Incorporating segmentations from the initial model as new training labels;
3. Validating against only the initial model's segmentation;
4. Adding the original actin channel as an additional input;
5. Testing whether the same training/validation set but different random seeds could produce different results;
6. Training with mixed labels (both ilastik and initial) in the same dataset.

Results. Qualitative comparisons (Figure 5.2) show that:

1. Training with more volumes (9 instead of 4) labelled with ilastik does improve segmentation results.

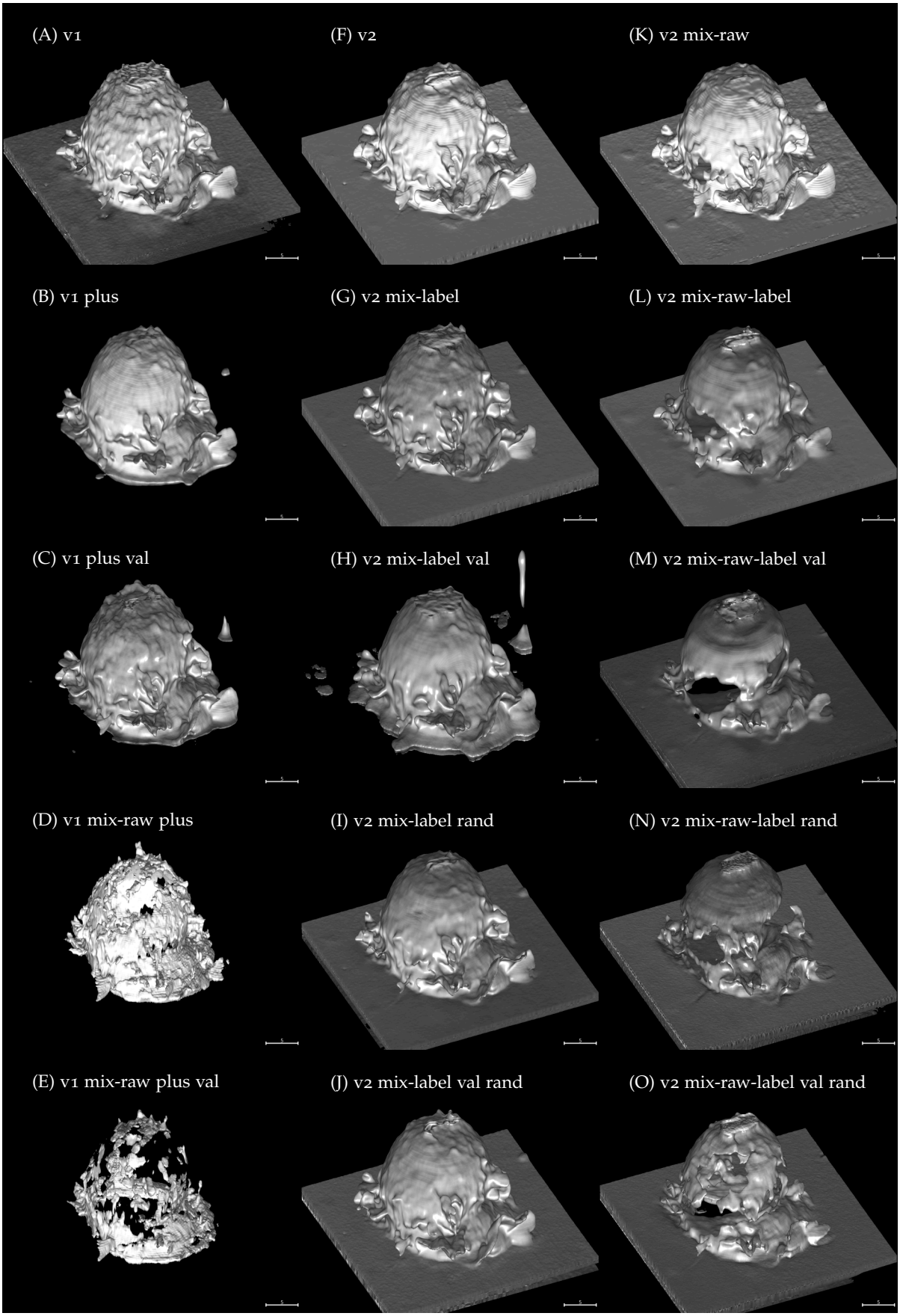


Figure 5.2: Qualitative Comparison of Cell Surface Models. The first column shows v1 models trained solely on ilastik segmentations. The second column shows v2 models trained on `initial` or `mix-label` segmentations. The third column includes models using both actin and negative channels ("mix-raw"). Training configurations are detailed in Table 5.1. Scale bars: 5 µm.

N	↓ Label	Raw →	`negative`	`negative` + `actin` (mix-raw)
4	`ilastik`		initial	~~initial mix-raw~~
9	`ilastik`		initial plus	initial mix-raw plus
9	`ilastik`		initial plus val	initial mix-raw plus val
9	`initial`		second (val)	second mix-raw (val)
9	`ilastik`	`+initial`	second mix-label	second mix-raw mix-label
9	`ilastik`	`+initial`	second mix-label rand	second mix-raw mix-label rand
9	`ilastik`	`+initial`	second mix-label val	second mix-raw mix-label val
9	`ilastik`	`+initial`	second mix-label rand val	second mix-raw mix-label rand val

Table 5.1: All Actin Models. Terminologies: the `initial`(-round) model is trained with 4 pairs of (negative, ilastik). `plus` models are initial models trained with 5 more pairs. `val` models only use the `initial` model's segmentation for validation. `mix-raw` models add the actin channel as input. `mix-label` models are `second`(-round) models using both the `initial` and ilastik labels. `rand` models use a randomised file-path order (with an ilastik label first), whereas the other models alternate between initial and ilastik labels. For further details, see Table A.3.

2. Even when using the same training/validation sets, the training process can be unstable; random seeds often produce noticeably different segmentations (Figure 5.3).
3. Adding the actin channel as an extra input ("mix-raw") can degrade performance compared to using negative-only inputs, suggesting the differing noise profile complicates the learning task.
4. Validating exclusively against initial model outputs can harm performance, possibly because it constrains the training objective to match imperfect segmentations.
5. Combining ilastik and initial labels ("mix-label") does not consistently improve model quality.

These observations suggest that more volumes are still needed to sufficiently fill the model's capacity and stabilise training. Moreover, providing a channel with a different noise profile than the labels (as the negative channel does) helps the network learn to ignore the original label noise.

After evaluating multiple models, I may chose the "initial plus" model for detailed surface analysis. Although it loses some resolution in thin regions such as filopodia and lamellipodia, its overall accuracy across the surface is superior. By contrast, models such as "second mix-label" generate thicker, overfilled segmentations that compromise fine surface details. Collaborators specialising in surface analysis recommended prioritising specificity to avoid overestimating surface features, even at the cost of losing some finer structures.

Next Steps. This is an ongonig project. Next, I will further refine the "initial plus" model to better capture filopodia and lamellipodia resolution by training on additional volumes. These new datasets will again use the negative channel as input and updated thresholded outputs of the "initial plus" model as labels, as shown in Table 5.2. This strategy is expected to yield the most accurate cell surface model for further analyses.

After producing the final model, I will use its segmentations for downstream studies of membrane-cortex attachment. Software packages including `u-shape3D`, `u-signal3D`,

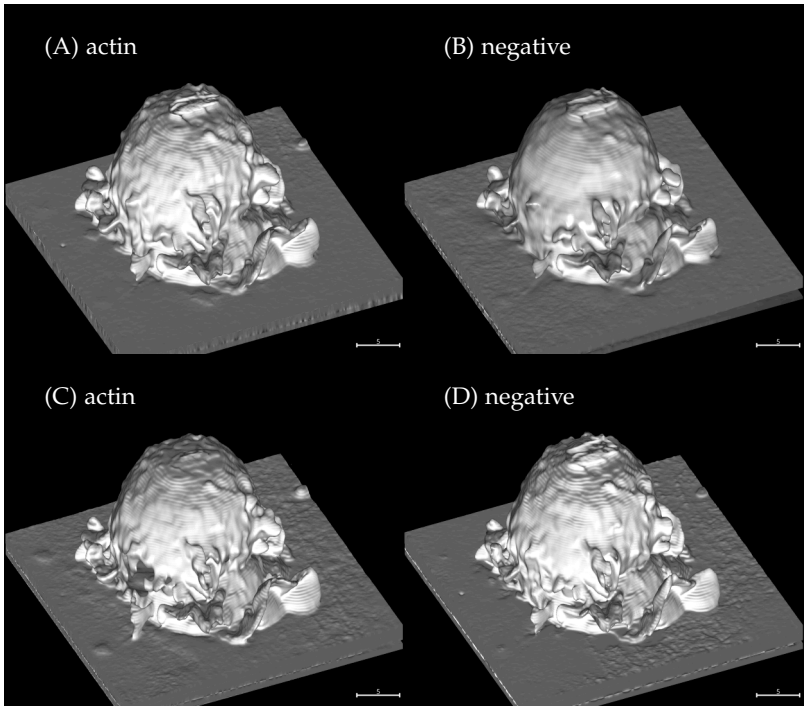


Figure 5.3: Qualitative Comparison under Identical Conditions. When training the same dataset with different random seeds, the resulting models can produce visibly different segmentations. Scale bars: 5 µm.

and `u-unwrap3D` [88–90] will be employed to quantify and investigate membrane-cortex interactions in 3D.

N	↓ Label	Raw →	`negative`
> 9	`initial plus`		third val
> 9	`ilastik + initial plus`		third mix-label
> 9	`ilastik + initial plus`		third mix-label val

Table 5.2: Future Actin Models. "Third"-round models use additional (negative, initial plus) pairs. "Mix-label" indicates including both ilastik and initial plus labels. "val" models are validated only against `initial plus` segmentations. For naming conventions, see Table A.3.

5.1.2 GoNuclear: Paired Fluorescence Staining for Nuclear Segmentation

This strategy is exemplified in chapter 2 and summarised in Figure 2.1.

A dual-fluorescence staining approach, wherein one channel exhibits high signal clarity while the other is noisy but biologically relevant, can be leveraged to train a deep learning model for segmentation. The high-quality channel provides the ground truth annotations, while the noisy channel serves as the input for the model. This strategy is particularly advantageous when the noisy channel suffers from low contrast or substantial background interference, whereas the high-quality channel is either costly to acquire or unavailable in all experimental conditions.

In live-cell imaging, obtaining a high-fidelity signal for certain structures is challenging due

to phototoxicity and photobleaching, which impose constraints on fluorescence intensity and exposure time. To circumvent these limitations, an alternative approach involves post-experiment fixation and restaining with more robust fluorophores, higher fluorophore concentrations, or imaging at higher laser power. Once the biological sample is no longer viable, it can tolerate stronger chemical treatments, enhancing signal intensity and contrast. The resultant high-contrast channel facilitates the generation of precise segmentation labels, which can then be used to train a machine learning model capable of segmenting the weaker channel acquired during live imaging.

To ensure accurate alignment between the live and post-fixation images, image registration techniques may be required. However, in the case of rigid biological structures such as plant tissues, registration is typically not the major bottleneck in image analysis.

5.2 Temporal Signal Integration: Blending Space and Time in Live Imaging

What is an image or a video? At its core, a 2D image is simply a matrix of pixel values, while a 3D image can be viewed as a stack of 2D images or a tensor of voxel intensities. Beyond three dimensions, additional axes—such as time or channels—can be appended, resulting in 4D, 5D, or higher-dimensional datasets. Although we do not perceive space beyond three dimensions, computational algorithms can theoretically handle such data extensions. A method designed for 2D or 3D data often generalises to higher dimensions if implemented appropriately.

In this section, I explore how multi-dimensional datasets can be reframed to simplify complex bioimage analysis tasks. Specifically, in subsection 5.2.1, I reinterpret a 2D time-lapse video as a 3D volume, enabling tracking to be solved as a segmentation problem using PlantSeg. In subsection 5.2.2, I segment and track cells in 3D time-lapse microscopy data using Ultrack, an algorithm that builds upon and extends the classical 3D reconstruction method of Funke *et al.* [91] to 4D. While Ultrack leverages modern computational resources to jointly perform 3D segmentation and cell tracking over time, it does not support full 4D segmentation, where time is treated as an additional spatial axis.

The novelty of my approach lies in the conceptual shift: by treating an N-dimensional video as an $N+1$-dimensional volume, I reduce challenging tracking problems to classic 3D segmentation problems. This stands in contrast to the other methods, [91] and [92], which aimed to keep the segmentation dimensionality low due to computational limitations at the time. By capitalising on increased computational power, I adopt the opposite strategy—elevating problem dimensionality to simplify algorithmic design and implementation.

5.2.1 Treating Tracking of 2D Video of Mobile *Planarian* as 3D Segmentation

I demonstrate how treating time-lapse video as 3D volumetric data enables robust tracking of regenerating planarian fragments. This innovative reframing simplifies identity tracking through direct volumetric segmentation, reducing complexity and improving consistency.

Biological Motivation: Regeneration ability and strategy vary significantly across animal

species. *Phagocata velata*, a freshwater planarian, can regenerate via two distinct strategies: blastema-mediated regeneration (where neoblasts proliferate and differentiate to form a blastema) and cyst-mediated regeneration (where fragments secrete mucus to encapsulate themselves). The choice between these strategies depends on both intrinsic and environmental factors, as well as on fragment mobility: mobile fragments typically form blastemas, whereas immobile ones tend to form cysts. Quantifying how mobility and morphology correlate with regeneration outcomes requires reliable tracking of individual fragments in time-lapse videos.

Technical Motivation: Traditional methods that segment each frame and then link objects across frames can become cumbersome, requiring sophisticated algorithms to maintain consistent object identities. Handling fragments that disappear and reappear in different frames further complicates the analysis. In this study, I aim to develop a more robust framework capable of tracking planarian fragments throughout regeneration, even when overlap or morphological changes occur.

Data. I worked with time-lapse videos of regenerating planarian fragments kindly provided by Shuchang Hu from the Vu group at EMBL Heidelberg. These videos were acquired as sequences of RGB images using consumer cameras. Shuchang achieved satisfactory instance segmentation for most frames, which serves as a valuable starting point.

Tracking Strategy. Although instance segmentation on a per-frame basis is relatively straightforward with standard workflows, maintaining consistent tracking over time often requires extensive human correction. After close inspection, I noticed that the segmentations themselves are generally reliable, but certain fragments are missing in some frames, causing tracking disruptions.

My primary research focuses on 3D bioimage analysis, where an additional spatial dimension often clarifies context for both human observers and computational algorithms. By analogy, rather than tracking 2D objects frame by frame, I propose to treat the time-lapse video as a 3D volume by stacking frames along the z-axis (time). In this 3D representation, 2D objects in consecutive frames become continuous 3D objects, making tracking implicit.

Shuchang's preliminary segmentation allows me to convert the existing instance masks into object boundaries and then apply watershed and GASP [61] to partition the 3D volume, following standard PlantSeg post-processing for boundary-based segmentation (see chapter 3). In effect, this reduces the 2D tracking problem to a 3D segmentation problem.

To demonstrate the feasibility of this approach, I first combined the per-frame instance masks into a boundary map, stacked the frames to create a 3D volume, and applied watershed and GASP to obtain a partitioned volume whose labels remained consistent through time. Encouraged by these initial results, I refined the pipeline further, producing boundary maps directly from raw videos instead of deriving them from instance segmentation. This approach is more efficient and can help avoid errors introduced by additional intermediate steps.

I used circular region detection (via the Hough Circle Transform) to mask out background wells, thereby restricting subsequent analysis to the relevant region of interest. A locally adaptive thresholding-based method then provides a semantic segmentation of worm fragments, from which boundary maps can be generated (e.g., via dilation). Although this step

sometimes retains the outline of the well border as a ring-shaped artifact, I employ an erosion step to remove extraneous boundaries while still preserving a border for worms near the wall. After partitioning the resulting boundary map with watershed and GASP, I obtain robust segmentation labels for each fragment across time, effectively yielding consistent tracking results.

Discussion. At this stage, I have not employed deep learning or other machine learning approaches; I fine-tuned the parameters of each step by hand. Nevertheless, I plan to train a deep learning model to predict boundary maps directly from raw videos, using the existing 3D segmentations as ground truth. Although I treat the stack of 2D frames as a 3D volume, the data remain fundamentally two-dimensional in each time frame: if fragments truly overlap in a single frame, there is no straightforward way to assign multiple labels to the same pixel. Fortunately, such overlaps are rare in this dataset and do not impede the core biological analysis.

When visualised in 3D, object splits appear as "Y" shapes, while unchanging objects appear as "I" shapes. Overlapping fragments that separate after a short time would form an "X" shape, which is clearly different from a split or an intact fragment. In this study, overlapping worms are rare and the worms do not split, so consistently maintaining an "I" shape is sufficient. If a label becomes disconnected in the 3D view, I simply remove that label from the video. This strategy ensures that my final segmentations track individual fragments in a way that facilitates subsequent morphological and behavioral analyses.

5.2.2 Cell and Nuclear Segmentation in Live *Drosophila* Embryos: The RIKEN Dataset

In this subsection, I tackle the complex segmentation and tracking challenges posed by live imaging of *Drosophila* embryos, introducing a combined strategy of denoising, boundary prediction, and integer linear programming-based tracking. This approach leverages modern computational techniques to deliver stable, high-quality tracking in noisy, dynamically changing conditions.

Biological Motivation: Gastrulation in *Drosophila* is a highly dynamic process in which a simple epithelial blastula transforms into a multilayered gastrula, laying the foundation for the organism's body plan. A key open question is whether nuclei act as a "battery," storing and releasing energy to drive these morphological changes. Investigating this hypothesis requires high-precision tracking of both nuclear and membrane structures over time.

Technical Motivation: The 3D time-lapse dataset from the RIKEN Center for Biosystems Dynamics Research presents multiple challenges: heavy noise, diminishing signal intensity along the z-axis, and partially open membranes in early embryonic stages. Furthermore, ground-truth annotations are only available for a subset of z-slices, making standard supervised learning approaches impractical. These constraints necessitate a robust pipeline for accurate segmentation and temporally coherent tracking.

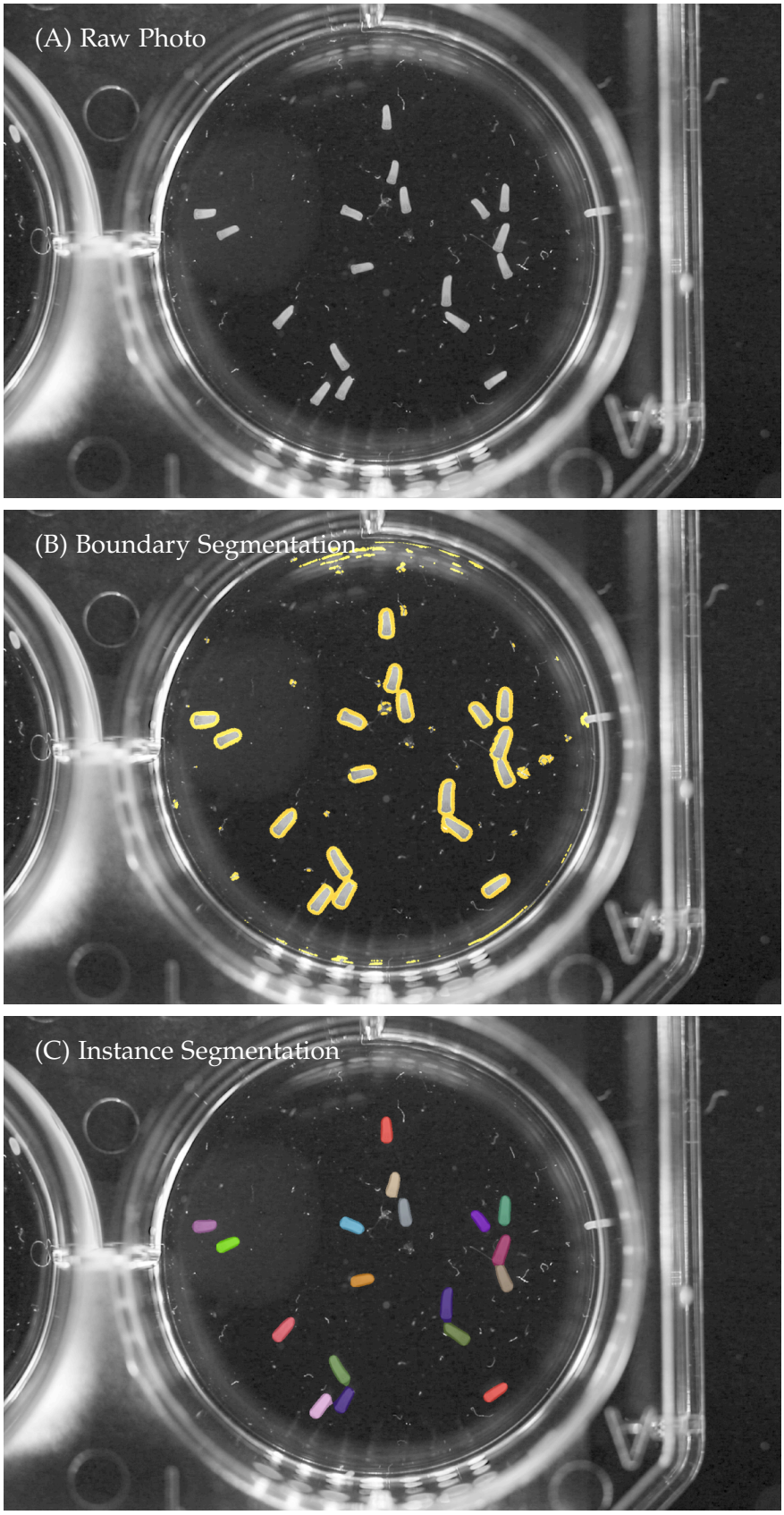


Figure 5.4: A Single Frame of the 2D Video. (A) The raw grayscale "red" channel, (B) the boundary segmentation, and (C) the resulting instance segmentation for a single frame of the mobile planarian video.

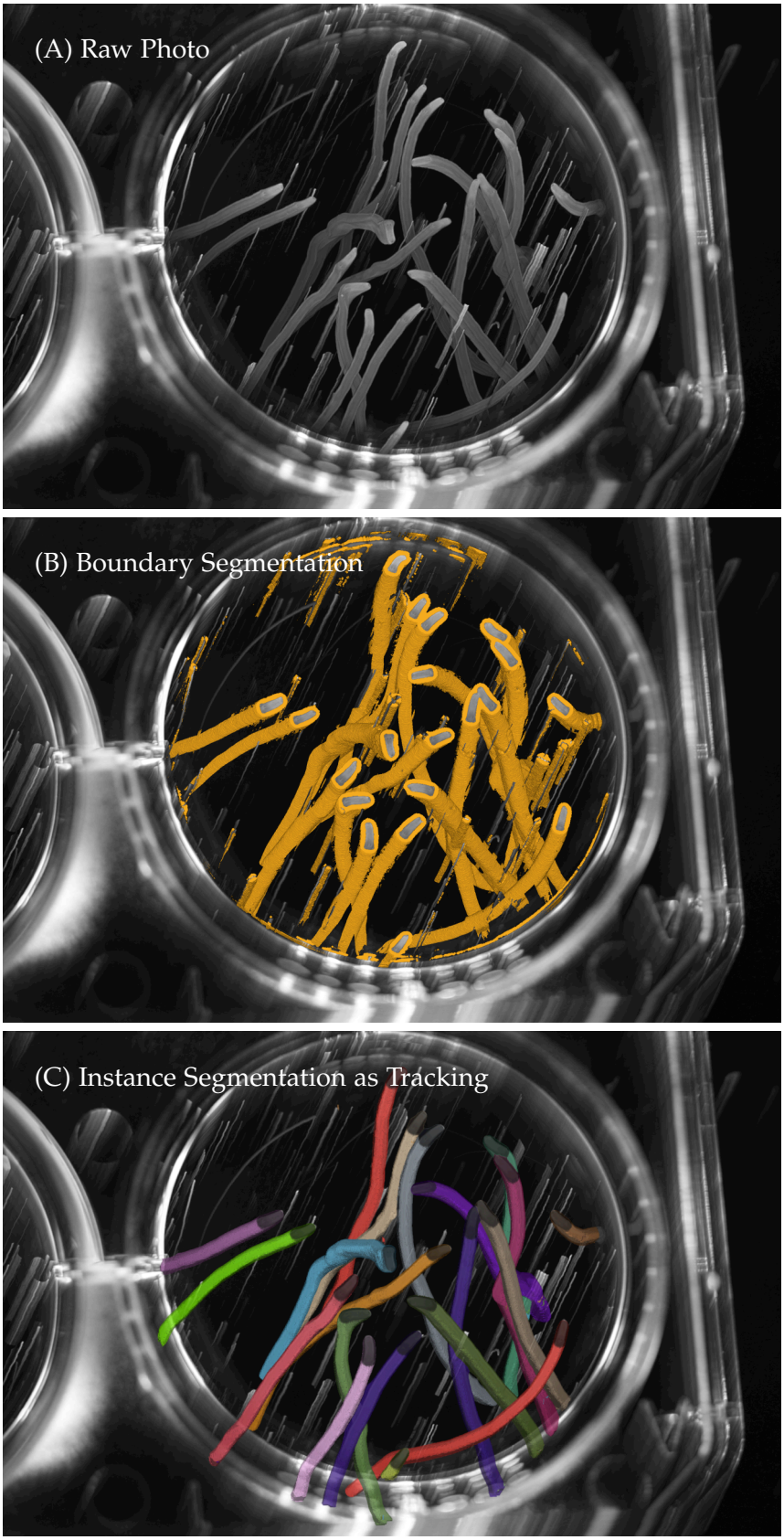


Figure 5.5: 3D View of 2D Video of Mobile Planarian. The time-lapse video frames are stacked to form a 3D volume, where (A) shows a raw frame, (B) shows the corresponding boundary segmentation, and (C) illustrates the resulting instance segmentation that also acts as a tracking solution through time.

Data. The dataset consists of live-imaged *Drosophila* embryos labelled with fluorophores marking nuclei and cell membranes. Acquired via two-photon microscopy, the image quality degrades at deeper imaging depths, leading to reduced membrane visibility. Moreover, only half of the z-slices contain reliable ground truth, requiring selective masking of unreliable regions during training to prevent biased learning.

Proposed Solutions and Exploration. Initially, I explored whether nuclear instance masks could serve as seeds for watershed-based segmentation of cell volumes. While theoretically sound, this approach suffered from under-segmentation and boundary misclassification due to intense noise and incomplete membrane signals.

I also tested a strategy where dilated nuclear segmentation masks were used as pseudo-cell instance masks for training a boundary segmentation model. However, qualitative evaluation revealed that this approach was less accurate than using stitched 2D Cellpose-based segmentations as training labels.

To mitigate the effects of noise, I incorporated denoising with Noise2Void [13], which significantly enhanced membrane visibility. While deep-learning models can learn to ignore noise given sufficiently large datasets, the limited and imperfectly annotated data in this case made explicit denoising a crucial preprocessing step.

Boundary Segmentation and ILP-Based Tracking with Ultrack. My final approach follows a two-stage pipeline, akin to `PlantSeg` (see chapter 3). First, I predict boundary probability maps for the membrane channel. Second, I leverage `Ultrack` [92], which jointly evaluates segmentation and tracking by formulating the problem as an Integer Linear Programming (ILP) optimization problem.

Unlike traditional frame-by-frame linking, `Ultrack` generates multiple segmentation hypotheses per time step using ultrametric contour maps (UCMs), which encode hierarchical segmentations at different thresholds. The ILP solver then selects the most temporally consistent segmentation while enforcing biologically plausible constraints such as valid cell divisions and prohibiting non-physical cell merges. Additionally, `Ultrack` integrates vector-field registration to account for motion artifacts, improving the stability of tracking across time.

This formulation significantly reduces the accumulation of tracking errors that arise in conventional linking-based approaches, allowing robust tracking of nuclei and cells even in the presence of noise and incomplete membranes.

Contributions to Ultrack, GoNuclear, and PlantSeg. To adapt `Ultrack` for the RIKEN dataset, I contributed improvements to dependency management[1][2] and tracking data export[3]. I also refined `GoNuclear` (see chapter 2) and integrated `PlantSeg` for boundary segmentation. Figures 5.6 and 5.7 illustrate the segmentation challenges and the improvements achieved through denoising, boundary prediction, and temporal consistency.

[1] GitHub – royerlab/ultrack/pull/85
[2] GitHub – royerlab/ultrack/pull/88
[3] GitHub – royerlab/ultrack/pull/90

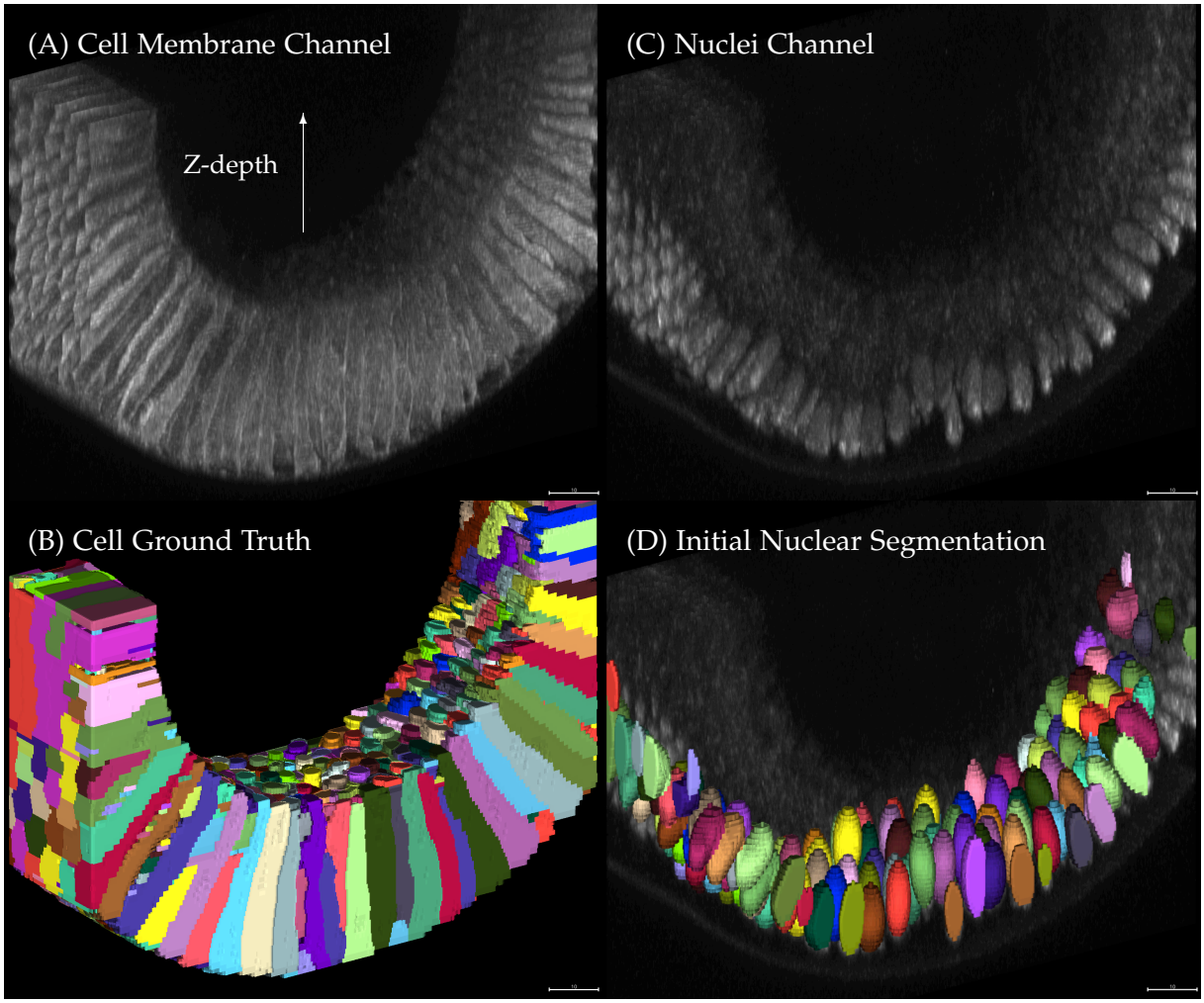


Figure 5.6: Initial Challenges in Live *Drosophila* Embryo Video. (A) Two-photon microscopy image of the cell membrane channel, showing signal degradation at deeper imaging depths. The left side illustrates poor membrane visibility due to limited z-resolution. (B) Ground-truth segmentation generated by collaborators using 2D Cellpose, exhibiting frequent over- and under-segmentation due to noise and incomplete membranes. (C) Nuclear channel with extensive background noise and uneven signal distribution. (D) Initial nuclear segmentation using `GoNuclear`.

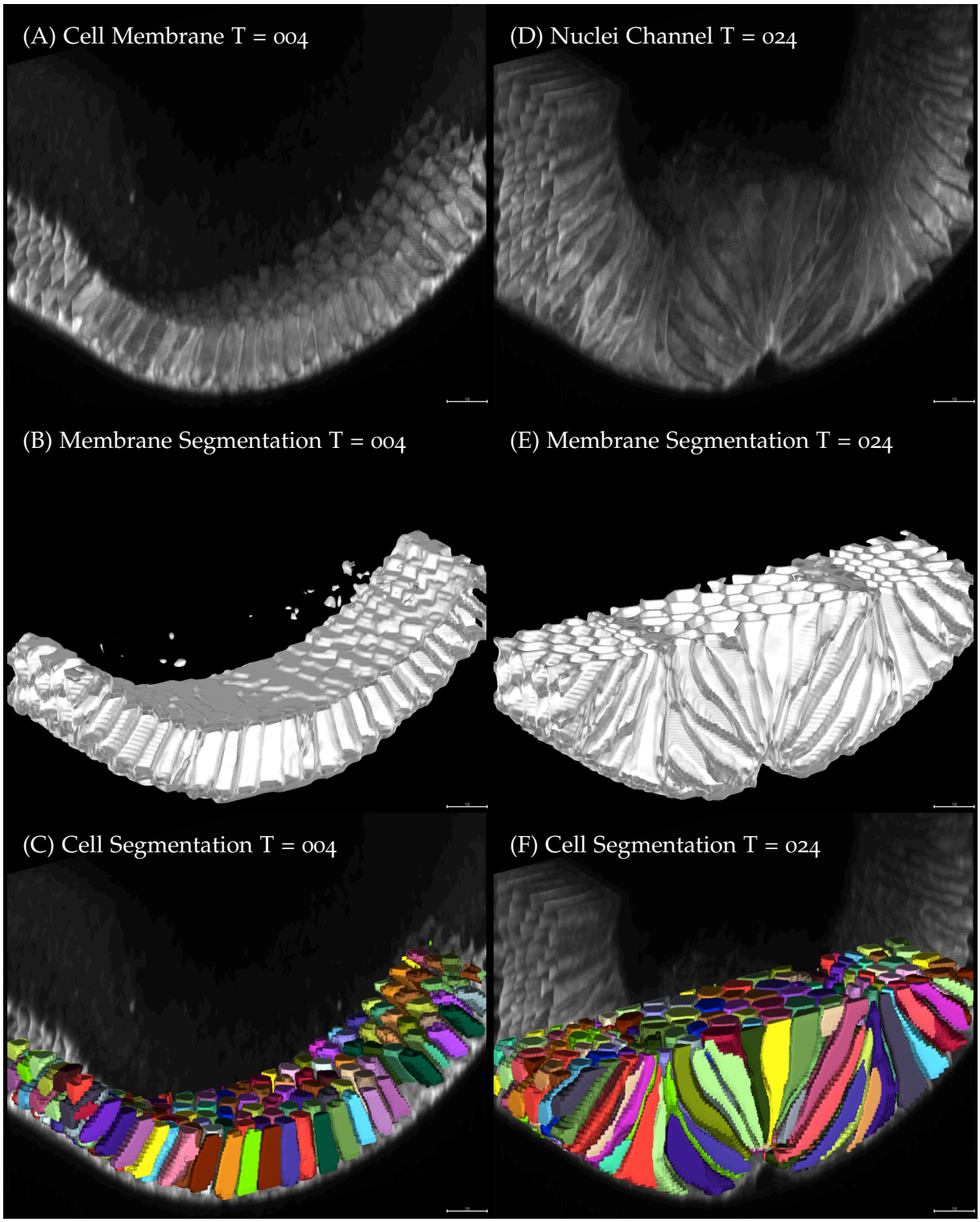


Figure 5.7: Improved Segmentation in Live *Drosophila* Embryo Video. (A, D) Noise2Void (N2V)-denoised raw membrane and nuclear channels, demonstrating improved contrast and clarity. (B, E) Predicted boundary maps generated from a model trained only on regions with reliable ground-truth annotations. Poor-quality slices were masked to prevent their influence on training. (C, F) Final cell segmentation obtained using `Ultrack`. The images are cropped to show high-confidence z-slices only.

5.3 Cross-Channel Prompting: Heuristics for Segment Anything Model (SAM)[4]

In multi-channel bioimage analysis, certain imaging channels often exhibit higher signal-to-noise ratios or better staining specificity, making them easier to segment. By contrast, other channels—while biologically critical—may suffer from low contrast, noise, or weak signal intensity, rendering them challenging for direct segmentation. This section introduces *cross-channel prompting*, a heuristic-based strategy where annotations from an easily segmentable channel are used to guide the segmentation of a more difficult channel.

To investigate this approach, I conducted extensive experiments using two large foundation models for image segmentation: the Segment Anything Model (SAM) [84, 93] and microSAM (µSAM) [10]. Both models have demonstrated strong generalisation capabilities across diverse image types, yet their application to bioimaging data—especially multi-channel microscopy images—remains underexplored. Through these experiments, I illustrate both the potential and limitations of using such models in a cross-channel context.

SAM is a versatile model trained on a broad range of natural images. However, it does not operate in a fully autonomous manner and requires external *prompts* to delineate specific objects. Prompts may take various forms, from simple spatial cues such as points and bounding boxes to more advanced polygonal inputs, as introduced in SAM 2 [93]. SAM's default strategy for automated segmentation is the *automatic mask generator* (AMG), which systematically samples single-point prompts across a grid over the input image. For each sampled location, SAM generates one or more segmentation masks, each accompanied by a confidence score. These proposals are then filtered using non-maximal suppression (NMS) and a confidence threshold to obtain a set of candidate masks. While AMG offers a fast and interaction-free approach to generating instance masks, it is highly sensitive to grid density and filtering parameters, and can produce numerous incomplete or extraneous masks.

In the context of multi-channel bioimages, cross-channel prompting offers a practical solution to overcome challenges associated with difficult channels. Specifically, I employ annotations (e.g., bounding boxes or point prompts) derived from a well-segmented reference channel to guide the segmentation of another channel that is harder to process due to noise, poor contrast, or biological variability.

The remainder of this section details my implementation of cross-channel prompting on the multi-channel ovules dataset described in chapter 2. I systematically assess how different prompting strategies affect segmentation outcomes in SAM and µSAM, comparing auto-prompting (prompts derived from the same channel) to cross-prompting (prompts derived from a different channel). These experiments reveal practical considerations for deploying large segmentation models on bioimage data and highlight scenarios where cross-channel heuristics can significantly enhance segmentation accuracy.

[4] At the time of these experiments, SAM [84] was newly released, and µSAM [26] was in early development. For completeness, my qualitative evaluation of µSAM has been updated from v1 to v2 [10], but the SAM 2 [93] has not been explored, as it is less biologically relevant here. Scale bars are omitted from the figures in this section; please refer to Figure 2.2 and Figure 2.2 for relevant size references. I also note two key practical constraints: (1) SAM resizes inputs to 1024×1024 pixels; (2) SAM only processes 2D images.

Experimental Design. I tested multiple models, including SAM's base/large/huge models and µSAM's light-microscopy/electron-microscopy variants (both v1 and v2). For clarity, I mainly show results from SAM large and µSAM v2 light microscopy large to keep the comparison somewhat consistent (noting that SAM huge tends to outperform SAM large). I also tried fine-tuning SAM on the ovules dataset, but the results were no better than the original pretrained weights—likely due to the difficulty of replicating Meta AI's full training recipe. Meanwhile, µSAM is fully open source, but v1 failed entirely for point prompts, so its point-prompting results are omitted from Table 5.3. More recent models, like SAM-HQ [87], SAM 2 [93] and µSAM v2 [10], do not alter the key conclusions, so I have not re-updated every entry in the table to reflect them.

I explored three raw channels in the ovules dataset:

- cell `wall`
- `noisy` nuclei
- `clear` nuclei

and employed two primary prompting strategies:

- *Cross-prompting:* Use ground-truth annotations from one channel (e.g. bounding boxes or point prompts) to segment objects in a *different* channel.
- *Auto-prompting:* Use a channel's own ground-truth annotations to prompt segmentation in the *same* channel (for baseline or sanity checking).

I tested bounding-box and point prompts, and also combined all three channels into a single "RGB-like" image for SAM or µSAM to process simultaneously. The experiments revolve around four major topics: (1) automatic mask generators, (2) bounding-box prompting, (3) point prompting, and (4) true multi-channel "RGB" input.

Automatic Mask Generators (AMG). By default, the *automatic mask generator* (AMG) method samples single-point prompts across a grid, then filters out or merges overlapping mask proposals. Figure 5.8 shows AMG results on three channels. SAM AMG finds most cells in the cell wall channel, some nuclei in the noisy channel, and most nuclei in the clearer channel—though it can also pick up cell-wall structures in the latter. µSAM AMG behaves similarly but often shows improved sensitivity to smaller objects, reflecting its training focus on microscopy. AMG can be a convenient way to bootstrap large annotation sets with minimal labor, albeit with typical automated-segmentation caveats (missed objects, partial objects, etc.).

Bounding Box Prompting. Next, I considered bounding-box prompting. Figure 5.9 shows *auto-prompting*, where ground-truth bounding boxes from the same channel are used as prompts; SAM and µSAM both reliably segment the targeted object. Although bounding-box prompting can seem nearly trivial, Table 5.3 shows that even then, SAM may not always yield high mAP scores across all instances. For example, on the noisy-nuclei channel, SAM's huge model reached $\sim$ 86% mAP^{50} in some settings, while µSAM v1 achieved $\sim$ 96%. Given the computational cost and complexity of large foundation models, simpler or more

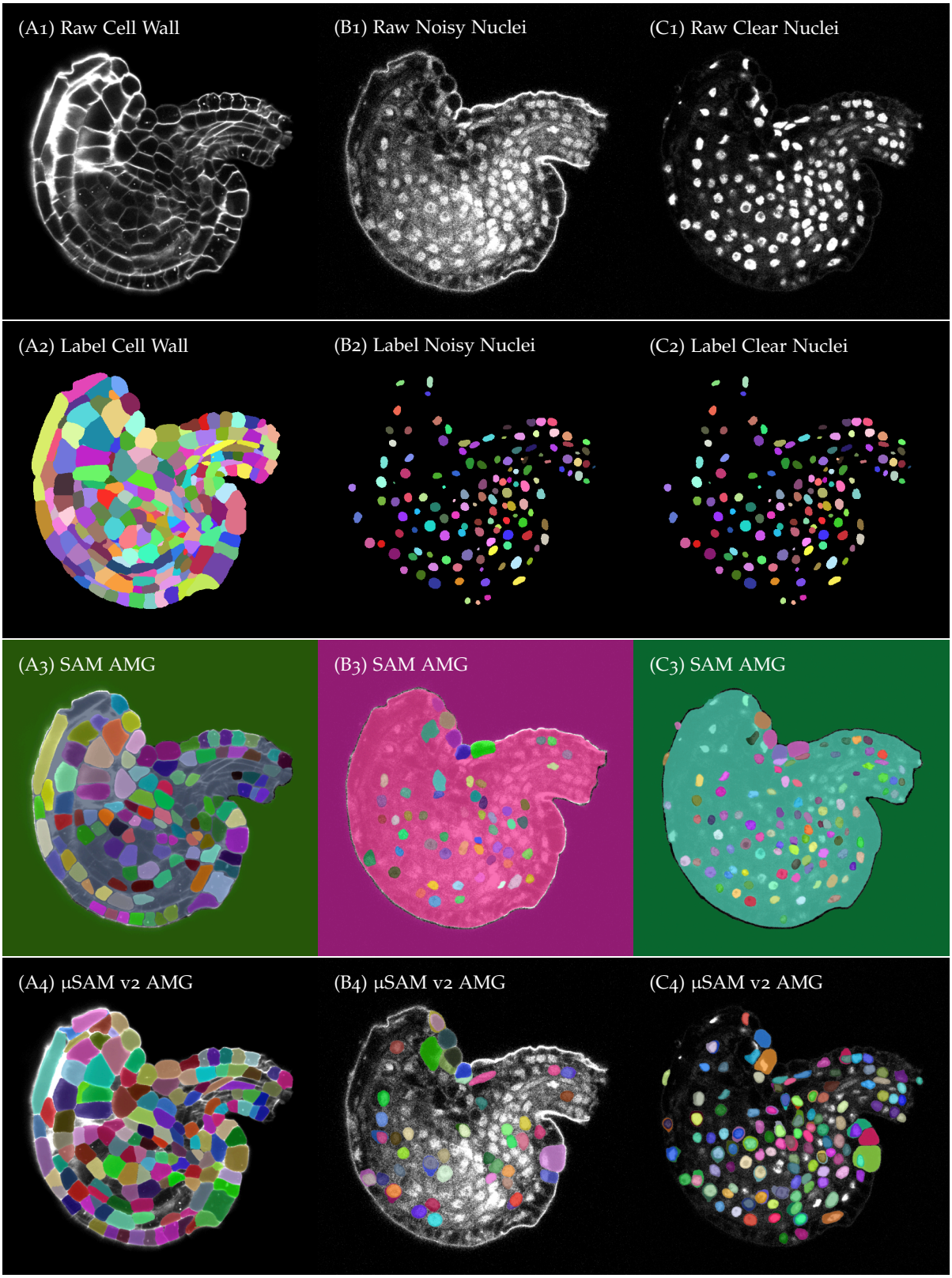


Figure 5.8: AMG Results in Different Channels. Columns A, B, and C correspond to the cell wall, noisy nuclei, and clear nuclei channels. Rows 1–2 show raw images and ground truths; rows 3–4 compare SAM AMG and µSAM v2 AMG segmentations. While cell walls and nuclei are often detected, partial or extraneous objects appear as well.

domain-specific tools (PlantSeg, Cellpose, ilastik) could achieve comparable performance with lower computational overhead.

Figure 5.10 then illustrates *cross-prompting* with bounding boxes from a *different* channel. In such cases, both SAM and µSAM dutifully force an object mask in each provided box, regardless of whether the channel's content aligns with it. Sometimes µSAM produces near-rectangular masks, closely following the bounding box while ignoring image structure. One can try *adjusting* bounding-box sizes (Figure 5.11) to better match the scale of objects in the new channel. For instance, inflating nuclear boxes may help discover entire cells, while shrinking cell boxes might capture nuclei. However, this ad-hoc process undermines the intent that SAM be "scale-agnostic." If bounding boxes must be repeatedly resized to coax the model into a correct result, one might question whether specifically training a segmentation model per channel is more direct than heavily manipulating prompts.

Point Prompting. Whereas bounding boxes can be easily derived from instance masks, deriving accurate point prompts is more nuanced in bioimage segmentation: a positive point should land inside the target, and negative points should lie outside. However, instance-mask centroids or random points risk falling on boundaries or ambiguous regions.

Figure 5.12 and Figure 5.13 show various multi-output scenarios where SAM or µSAM returns multiple top-confidence masks per prompt set. SAM will often ignore negative prompts if they conflict strongly with its learned representation, whereas µSAM can be over-swayed by negative prompts, sometimes chopping away part of an object.

In single-output mode (e.g. retrieving only the highest-scoring mask), both SAM and µSAM can be more reliable, as seen in Figure 5.14 and Figure 5.15. One positive point plus one well-placed negative point typically suffices in simpler cases, though µSAM's bias for microscopy can yield "fuller" masks (e.g. including a cell wall) than SAM might produce. Whether that is beneficial depends on the user's specific segmentation goal (e.g. entire cell vs. cytoplasm only).

Real Multi-Channel Prompting (RGB Input). Although SAM was primarily designed for 2D RGB images, one could pack the three ovule channels (`cell wall`, `noisy nuclei`, `clear nuclei`) into a single "RGB" image. As Figure 5.16 shows, AMG on this merged image typically detects some cell and nuclear structures but can also miss or partially segment them due to the increased complexity. Bounding-box prompting on an RGB combination can still produce decent masks, whereas point prompting might yield overly large or unconnected segments, depending on which channel the model "locks onto."

Conclusion. Overall, µSAM—trained specifically on microscopy data—shows more natural affinity for cell and nuclear structures than the original SAM, which sometimes prefers large "natural scene" objects (e.g. entire ovules). This makes µSAM particularly appealing for minimal-interaction workflows (see also subsection 2.3.2), where users only provide a bounding box or point prompt. Still, both methods are constrained to 2D images, complicating volumetric data analysis. If prompts must be heavily manipulated to "force" cross-channel segmentation, one might instead prefer a dedicated model trained for that channel.

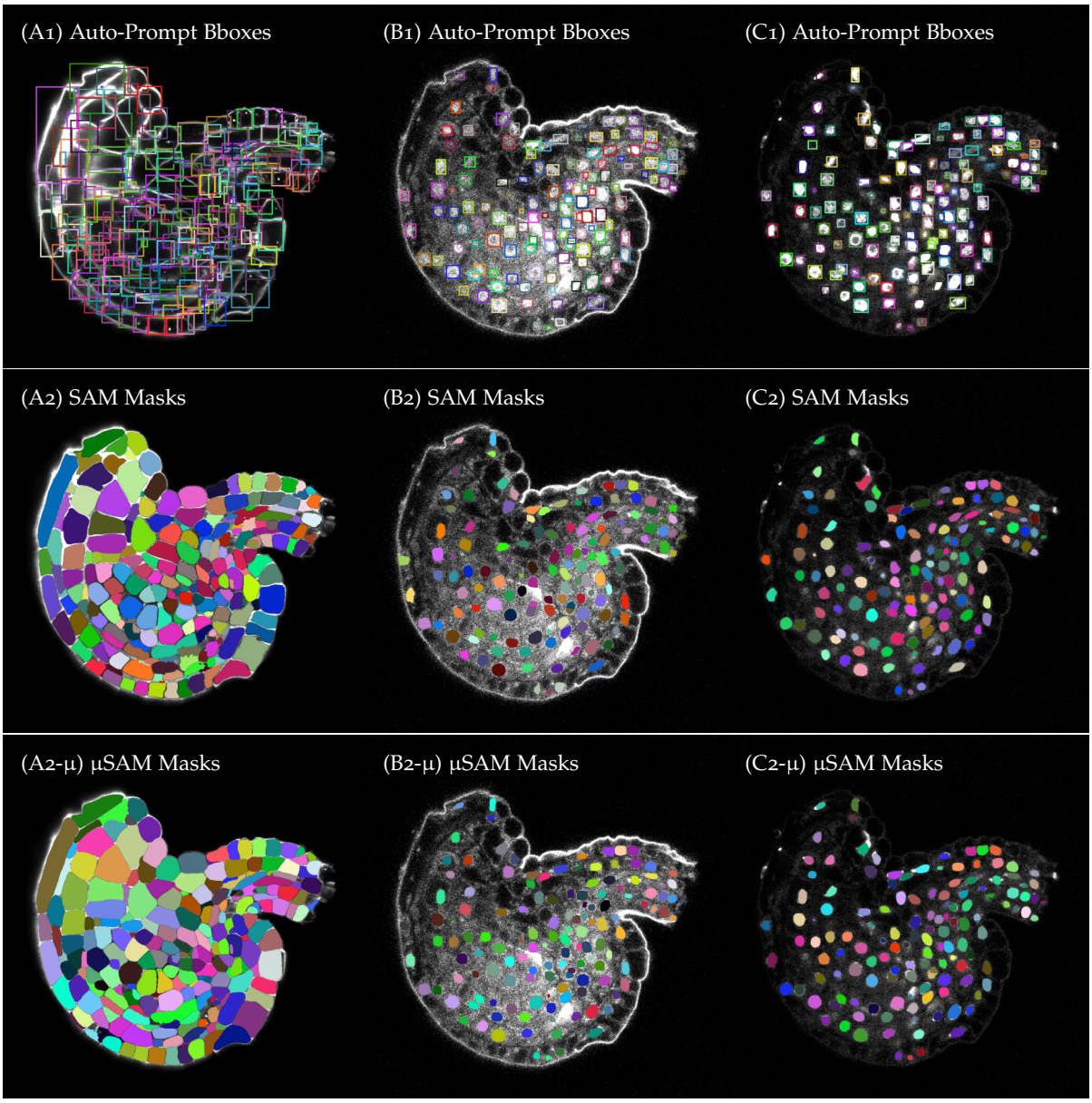


Figure 5.9: Bounding Box Prompting (Auto-Prompting). Columns A, B, and C show the cell wall, noisy nuclei, and clear nuclei channels. Row 1 displays bounding boxes from the same channel's ground truth, and rows 2–3 show how SAM and μSAM segment these boxes. This "auto-prompting" verifies that if a bounding box is accurate, the model can typically recover the object.

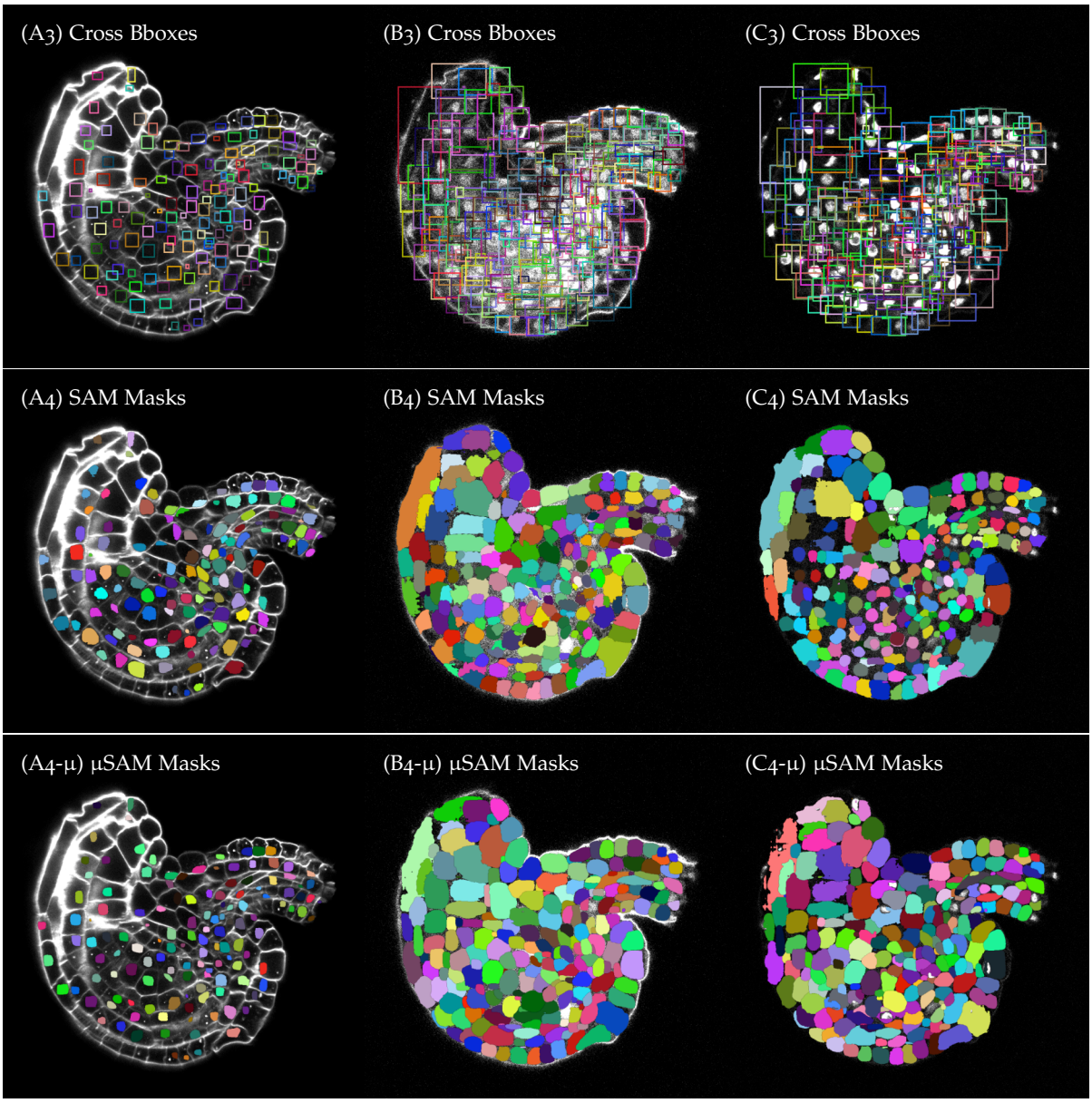


Figure 5.10: Bounding Box Prompting (Cross-Prompting). Bounding boxes from a different channel's ground truth are placed on each image. SAM and μSAM still force a mask within each bounding box, even if the underlying image content does not match. In some cases, μSAM produces near-rectangular masks that follow the bounding box more than the actual image.

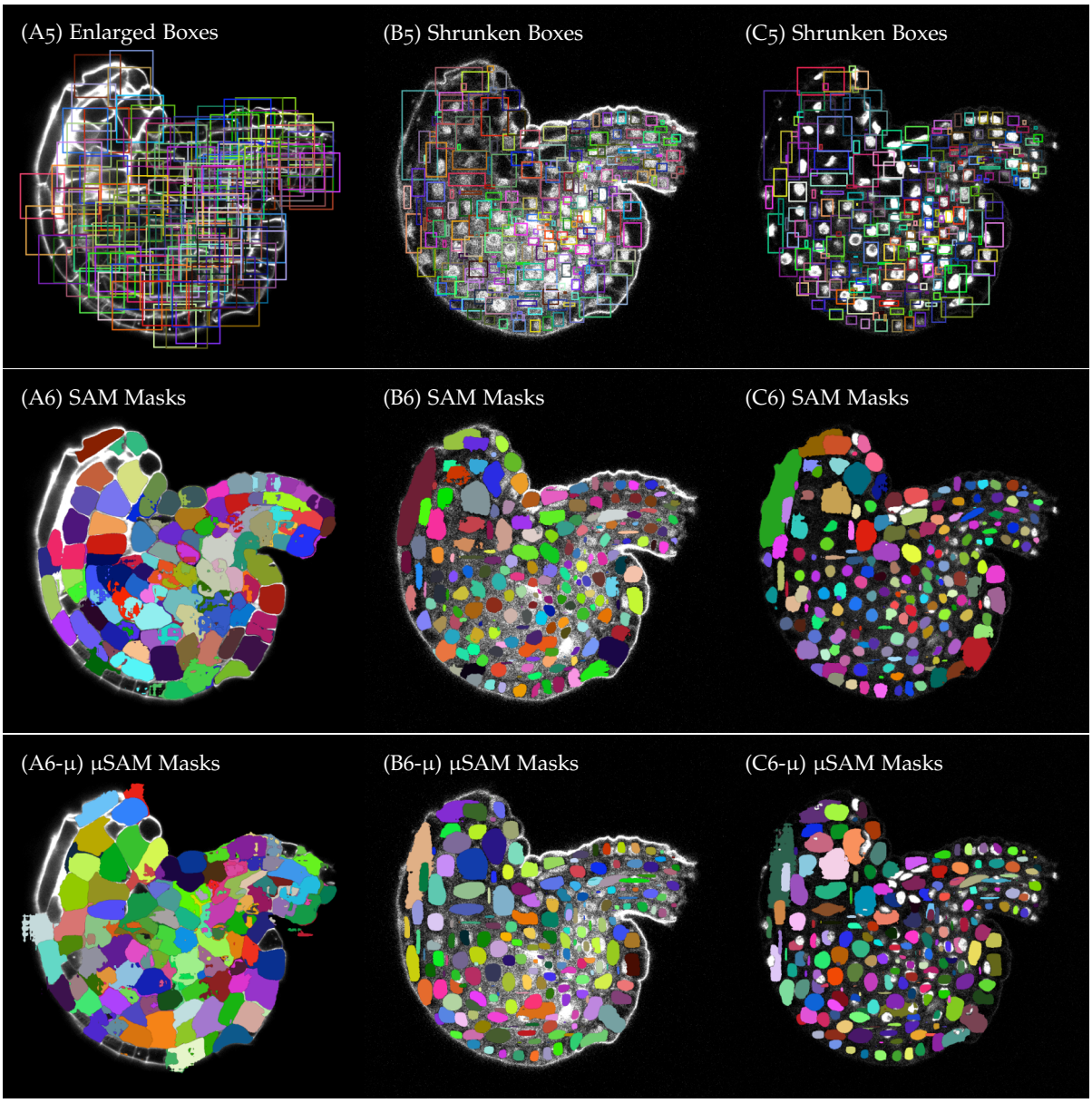


Figure 5.11: Adjusting Bounding Boxes for Cross-Prompting. In these examples, nuclear bounding boxes are enlarged by 100 px in both x and y directions to capture entire cells, while cell bounding boxes are shrunk by 20 px to better match nuclei. Although this can partially correct scale mismatches, excessive box manipulation becomes guesswork and undermines SAM's "any scale" premise.

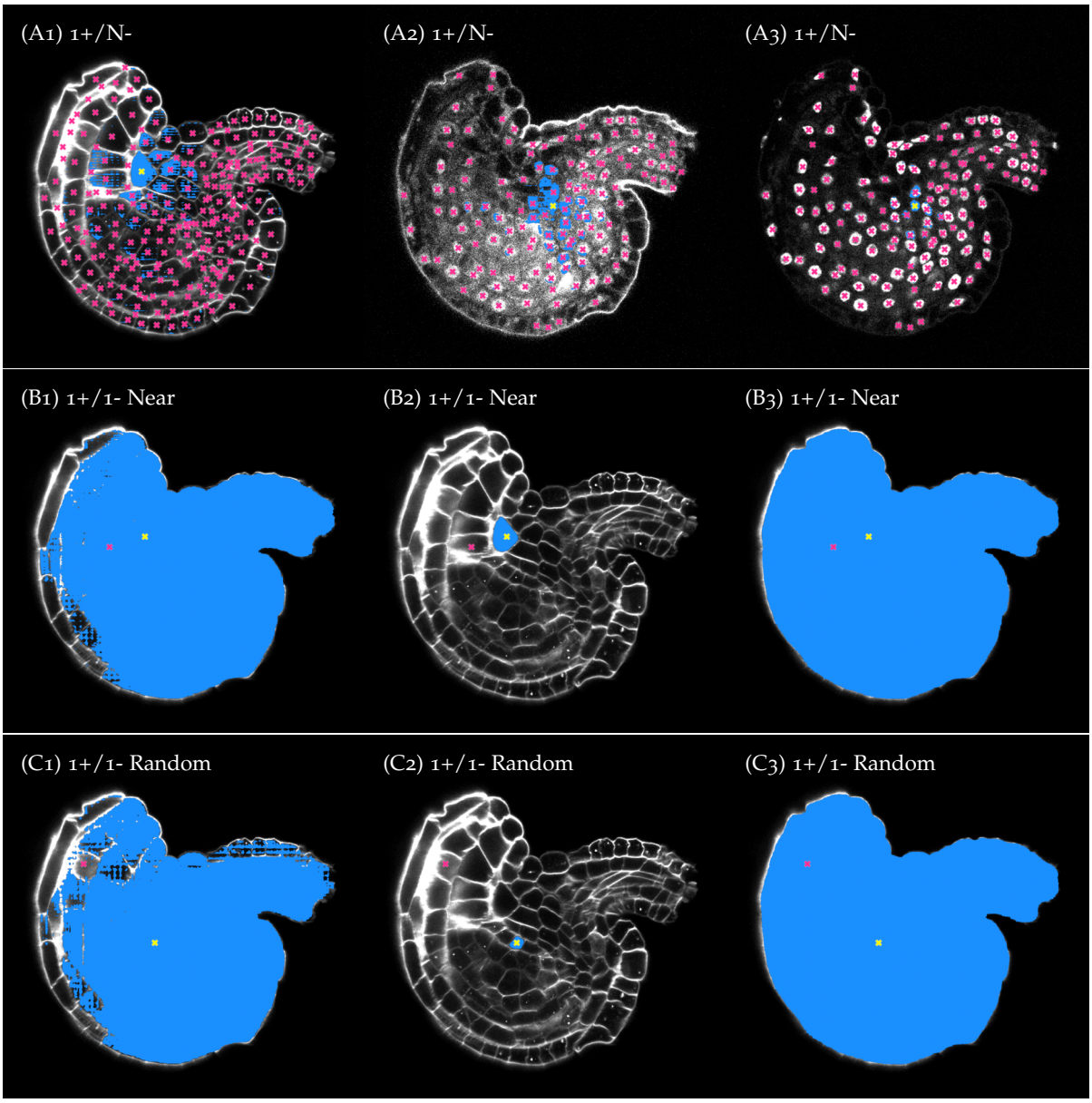


Figure 5.12: SAM Point-Prompting Strategies (Multi-Output). Here, we show three different sets of point prompts for SAM's multi-output mode: (A) one positive point + multiple negative points; (B) one positive + one negative in a nearby object; (C) one positive + one negative in a random object. SAM sometimes ignores negative prompts it deems inconsistent, returning multiple plausible masks.

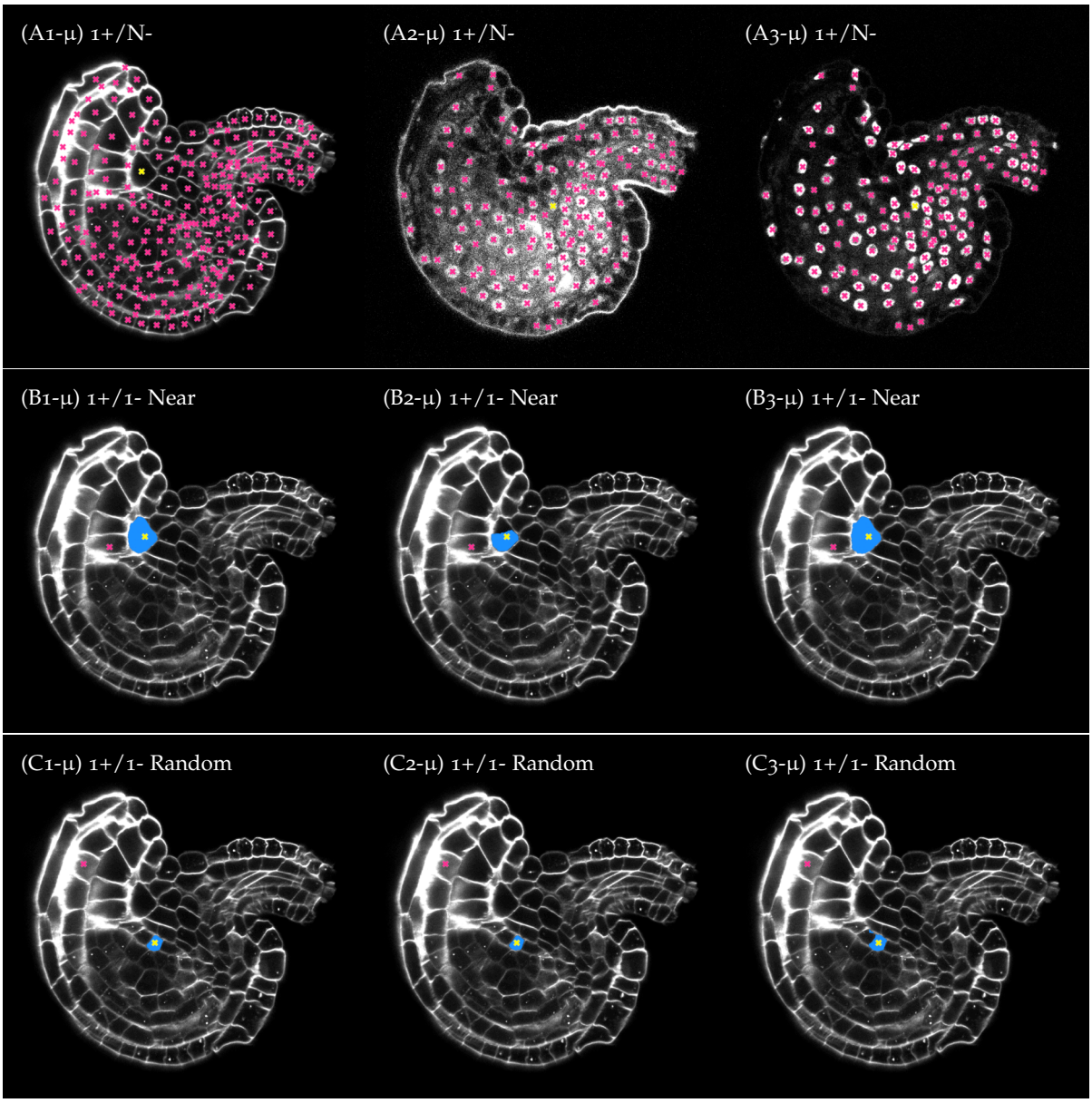


Figure 5.13: μSAM Point-Prompting Strategies (Multi-Output). Using the same prompt sets as in Figure 5.12, μSAM often reacts more strongly to negative prompts, sometimes "carving out" large regions. Still, it is generally biased toward segmenting biological structures (e.g. more complete cells), reflecting its microscopy-oriented training.

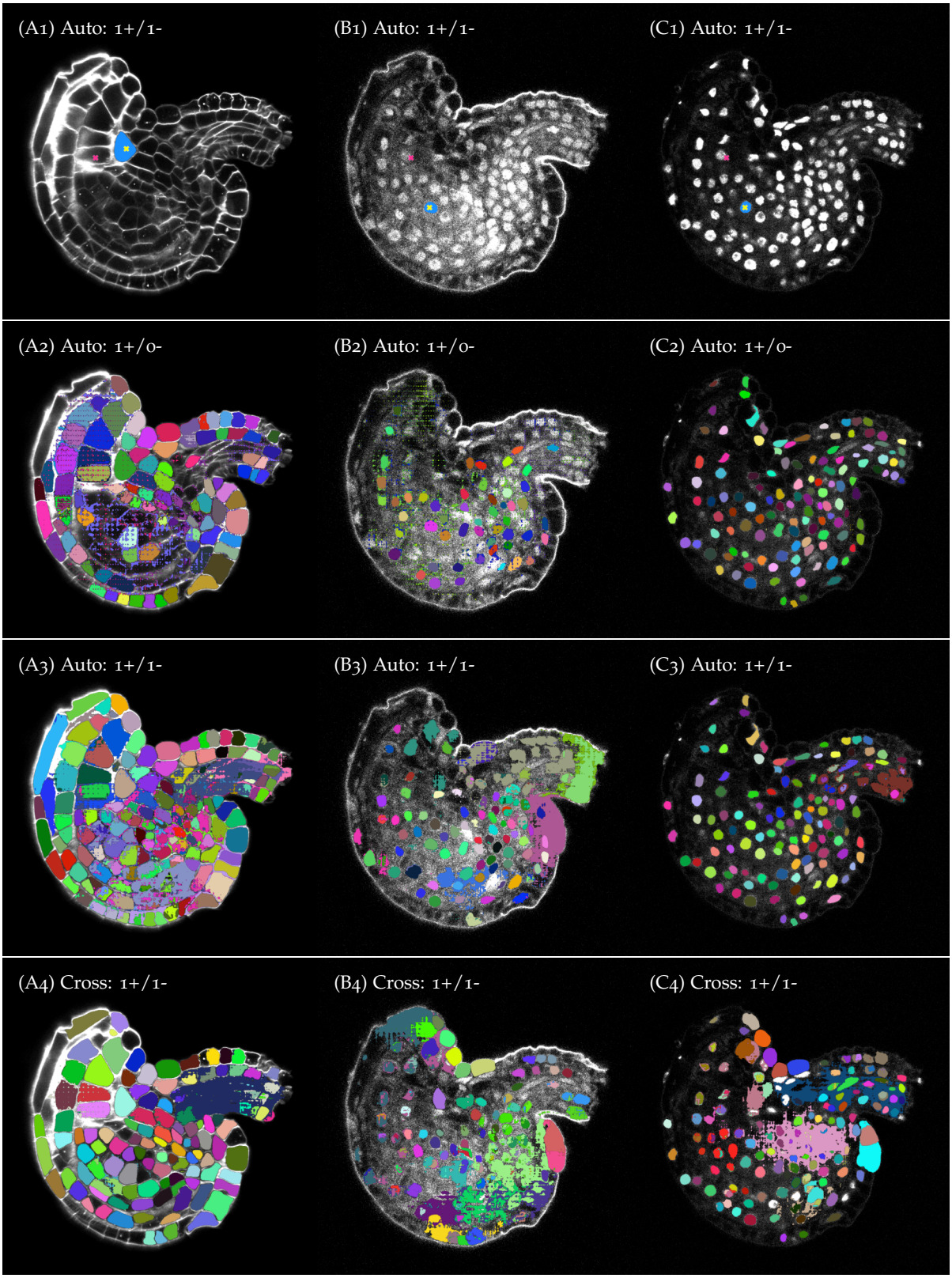


Figure 5.14: Single-Output SAM Point Prompting. Columns A, B, and C are the cell wall, noisy nuclei, and clear nuclei channels. The top three rows show auto-prompting with different numbers of negative points. The last row demonstrates cross-channel prompts (e.g. nuclear prompts on the cell-wall channel). SAM often ignores negative points if they conflict strongly with its learned representation.

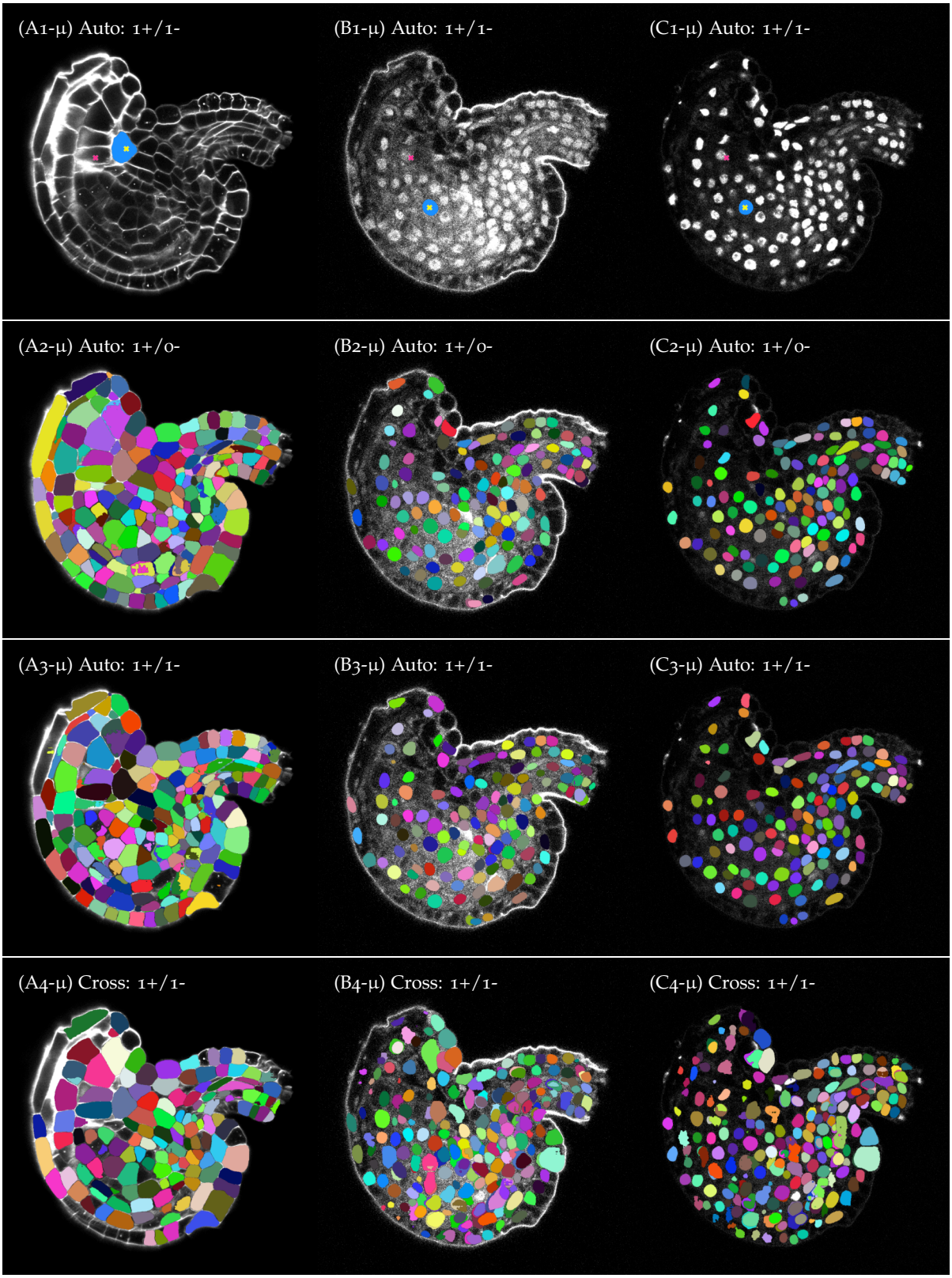


Figure 5.15: Single-Output μSAM Point Prompting. Same structure as Figure 5.14 but using μSAM v2. Generally, μSAM segments more of the cell or nucleus than SAM, reflecting its microscopy-oriented training. In some cases, this may include regions the user does not wish to label (e.g. cell walls), so manual curation might still be needed.

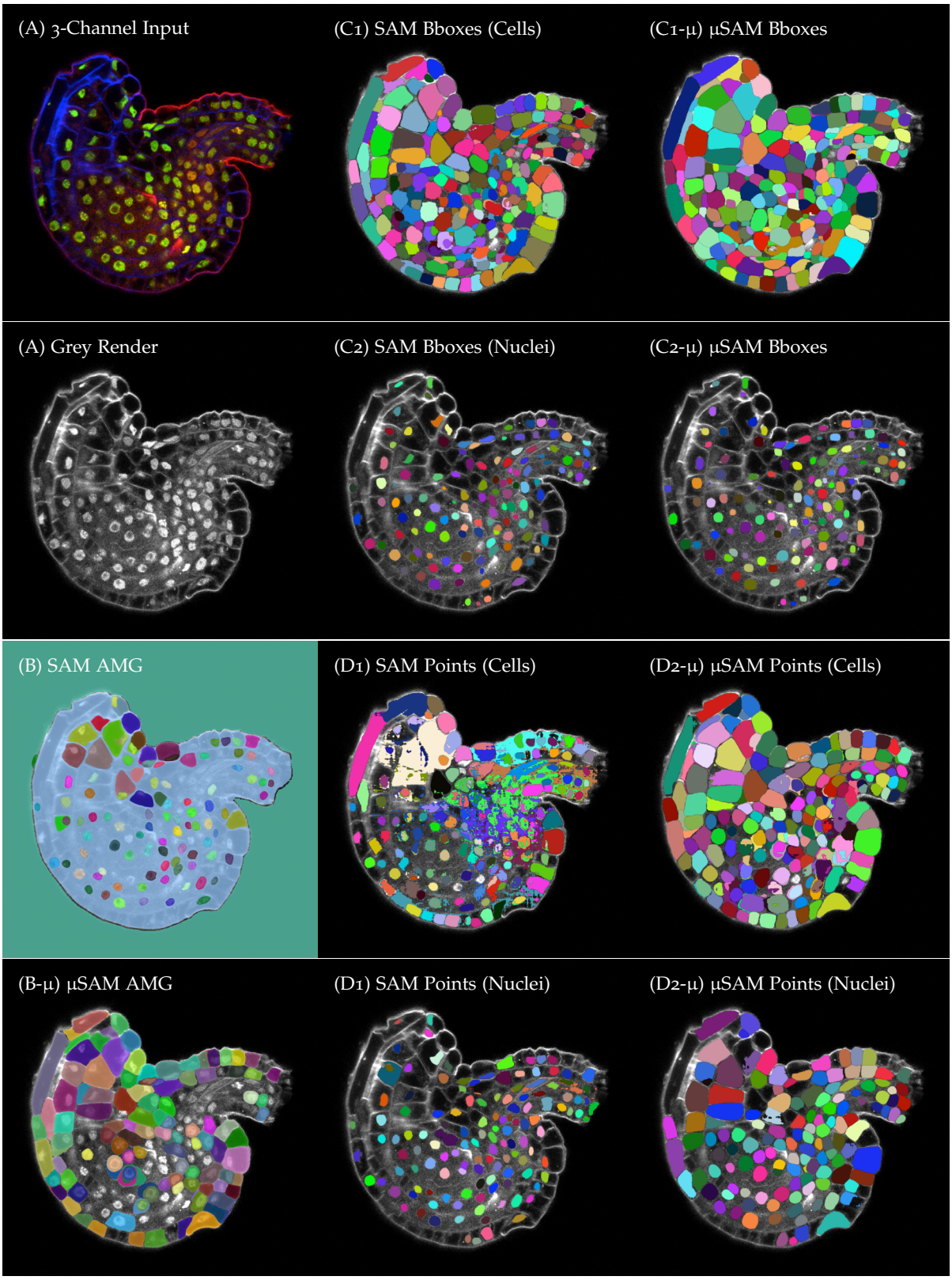


Figure 5.16: Combining All Channels into a Single RGB Image. (A) The three ovule channels are merged into one RGB image for SAM/μSAM to process at once. (B) AMG can only detect some cells and nuclei simultaneously. (C) Bounding-box prompts remain fairly robust if one has correct boxes for each target structure. (D) Point prompting can produce mixed masks, depending on which channel's features the model emphasises.

Average Precision		0.5	...	0.75	0.8	0.85	0.9	0.95	mAP
AMG	Base	27.64%	...	19.54%	14.20%	7.99%	2.31%	0.06%	17.22%
	Large	34.37%	...	24.87%	19.86%	13.11%	5.21%	0.19%	22.20%
	Huge	35.11%	...	24.31%	19.03%	12.48%	4.71%	0.16%	22.09%
Bounding Boxes	Base	88.99%	...	49.34%	34.79%	18.93%	5.33%	0.11%	49.65%
	Large	84.24%	...	44.47%	32.96%	19.80%	6.96%	0.24%	45.79%
	Huge	86.13%	...	46.83%	33.40%	19.78%	6.62%	0.20%	47.55%
Bounding Boxes	Base LM	**95.43%**	...	78.71%	63.67%	37.55%	10.83%	0.25%	**64.99%**
	Huge LM	**95.92%**	...	81.12%	69.21%	46.42%	16.38%	0.54%	**67.76%**
Bounding Boxes	Base EM	**95.55%**	...	74.70%	55.72%	26.81%	6.00%	0.17%	**62.05%**
	Huge EM	**96.18%**	...	80.14%	66.36%	40.11%	9.97%	0.17%	**66.18%**

Table 5.3: Benchmarking SAM Cross-Channel Prompting. This table summarises the performance of SAM and µSAM v1 on the GoNuclear ovules dataset (chapter 2). Different prompting strategies are tested on the noisy nuclei channel, and performance is reported as mean average precision (mAP) at various IoU thresholds.

Next Steps. Future directions could include:

- Deeper testing or fine-tuning of µSAM v2 (or a specialised "µSAM-HQ," inspired by SAM-HQ [87]) on microscopy data.
- Pairing heuristic bioimage workflows (e.g. ilastik) with large foundation models to generate more informed "smart seeds" for AMG.
- Explore the possibility of using SAM 2 models with video segmentation abilities to segment 3D images by treating them as videos.

Such approaches may further reduce annotation effort while preserving segmentation quality, especially for large-scale or high-dimensional imaging data.

5.4 Outlook: Context-Aware Segmentation in Multi-Channel Bioimaging

While multi-channel bioimages inherently contain complementary information across different stains or markers, leveraging these additional channels within a single model remains challenging. In my experiments, straightforward multi-channel, multi-head convolutional networks yielded only marginal improvements unless the model was explicitly guided to integrate cross-channel cues. Empirically, pipelines that segment one structure (e.g., nuclei) to refine another (e.g., cells), and vice versa [2], can effectively exploit inter-channel dependencies. However, truly context-aware modelling within a unified network remains an open research problem. My experiments with multi-channel, multi-head, and multi-prediction networks are detailed in subsection A.1.1.

As a final observation, the scale of training data can precipitate a qualitative shift in performance. For instance, DINO [94, 95], a self-supervised model from the broader computer vision

domain, learns to segment objects without explicit supervision when trained on 1.3 million images (and 142 million in DINOv2). This illustrates that, given sufficient data, context-aware segmentation might emerge naturally—even in the absence of hand-crafted guidance.

I propose three avenues for further exploration:

1. *Hierarchical Discriminative Loss* for multi-channel instance embedding (section 4.6);
2. *Inter-Prediction Loss* to enforce consistency across predicted channels (subsection A.1.1); and
3. *Topological Constraints*, such as containment and exclusion, to encode structural rules into the segmentation process (subsection A.1.2).

In particular, inter-prediction loss aims to capture how one channel's segmentation can inform and refine another by explicitly penalising or rewarding cross-channel consistency. However, naïve formulations risk overexpansion of the predicted regions. Future work could investigate adaptive normalisation schemes or instance-specific metrics to mitigate such effects.

Topological constraints, on the other hand, address global relationships such as "nuclei must reside inside cells" or "cell boundaries must be closed"—relationships that are difficult to learn via local, per-pixel losses alone. Incorporating such constraints into training, possibly through differentiable instance-level strategies like those explored in chapter 4, may improve multi-channel performance where structural accuracy or boundary completeness is critical. Detailed discussions can be found in section 4.6 and subsection A.1.2.

Overall, systematically harnessing the complementary signals in multi-channel data remains an exciting frontier. By integrating higher-level structural priors, inter-channel dependencies, and instance-aware constraints, future approaches may achieve more robust and biologically consistent segmentations across diverse bioimaging modalities.

Chapter 6

The Future of Bioimage Analysis

In summary, bioimage segmentation remains pivotal yet intrinsically challenging. The diversity of imaging modalities, evolving microscopy techniques, and heterogeneous biological contexts demand analytical tools that are not only accurate but also generalisable, interpretable, and adaptive. While deep learning has driven remarkable progress, persistent issues such as data scarcity, inconsistent imaging protocols, and the inherent complexity of biological samples preclude any universal solution. My thesis addresses these challenges through novel algorithmic contributions, robust software engineering, and practical case studies designed to push the limits of current computational methodologies.

Although I have largely avoided the term "AI" in this thesis—preferring the more precise nomenclature of deep learning or computer vision—the end of my PhD coincided with the rise of large language models (LLMs), marking a turning point in the broader AI landscape. Initially impractical for academic use due to financial and hardware constraints, the release of efficient and open-source models such as DeepSeek-V3 [96] during my time of writing this thesis, which employ mixture-of-experts architectures and FP8 quantisation, has changed the playing field. Beyond text generation, these LLMs demonstrate state-of-the-art capabilities in reasoning, coding, and tool use, thus offering profound implications for bioimage analysis.

Recent developments suggest that LLMs are being repurposed as multimodal agents capable of reasoning across image, text, and structured data. For example, the *Omega* plugin [97] integrates LLMs into *napari*, enabling users to issue image-analysis tasks via natural language. The *BioImage.IO Chatbot* [98] extends this vision, coupling LLMs with model repositories to generate executable code, guide workflows, and perform tool-based reasoning. These systems illustrate how future human–machine collaboration in image analysis may be mediated by dialogue rather than scripts or graphical user interfaces.

The emergence of multimodal LLMs (MLLMs) marks a significant frontier [99]. MLLMs integrate multiple inputs—images, text, and omics data—to generate coherent, context-aware outputs. Their emergent capabilities allow them to generalise beyond explicit training domains, while mixture-of-experts and tool-calling modules enable them to handle diverse biological images, propose hypotheses, and even control instruments.

Researchers have just demonstrated in very recent publications [100] how scaling and fine-tuning vision–language models (VLMs) on scientific biomedical data (through techniques like LoRA, FlashAttention-2, and cross-modal projector alignment) substantially improves both reasoning accuracy and factual consistency in biomedical visual question answering (VQA).

Their tuned models achieve higher ROUGE scores, reduced hallucination, and stronger alignment with ground-truth data than base models. This suggests that, with sufficient domain-specific adaptation, generalist VLMs can evolve into robust scientific assistants.

In parallel, *MicroVQA* [101] was introduced when I started to write this chapter, a VQA benchmark designed to probe scientific reasoning capacities of MLLMs in microscopy. This benchmark includes tasks such as expert image interpretation, hypothesis generation, and experimental design, drawn from real biological practice. State-of-the-art models performed moderately, revealing that multimodal reasoning in microscopy remains a frontier. Crucially, errors often stemmed from perceptual failures rather than language or logic, indicating a pressing need to improve visual representation learning.

A benchmark for evaluating LLMs on code generation for bioimage analysis was proposed [102]. Using a curated set of Python tasks and unit tests, they assessed functional correctness across models. While top models (such as GPT-4 and Claude-3) are promising, their study highlights the need for domain-specific tuning, broader support for scientific libraries (like `scikit-image` or `pyclesperanto`), and benchmarks reflecting real bioimage-analysis workflows.

Together, these studies suggest that foundational models and community-driven infrastructure will co-shape the future of bioimage analysis. Hallucinations, shallow reasoning, and alignment mismatches remain problematic in scientific contexts, but approaches like retrieval-augmented generation (RAG) and parameter-efficient fine-tuning (PEFT) can help contain them, while LoRA offers mechanisms to update specific modules without retraining the entire network. In parallel, open, FAIR resources and active community engagement will be essential to integrate these AI tools into real-world experimental pipelines.

Ultimately, bioimage analysis is evolving from a niche engineering discipline into a testing ground for general-purpose scientific AI. The inherent complexity of biological images—spanning semantic ambiguity, multiscale structure, and imaging artifacts—poses uniquely rigorous challenges for machine intelligence. As AI systems become more capable, they should be guided by domain expertise and developed with the rigor that accompanies experimental design. Rather than replacing human insight, the path forward lies in amplifying it.

Chapter A

Supplimentary Material

A.1 Context-Aware Segmentation: Using One Biological Structure to Guide Another

This section extends chapter 5 from the main text, offering theoretical insights, software tools, and preliminary experiments on multi-channel and multi-head networks with inter-prediction loss, as well as conceptual foundations for incorporating topological interactions into bioimage segmentation tasks.

This section outlines prospective directions to enhance multi-channel bioimage segmentation by explicitly modeling context-aware interactions among biological structures. I describe several conceptual approaches—some of which I have explored in preliminary experiments, others that remain theoretical—that aim to improve segmentation accuracy by leveraging the inherent biological context present across multiple data channels.

To integrate multi-channel information and jointly segment different biological structures within the same sample, I propose three strategies:

1. A hierarchical discriminative loss function for pixel-embedding instance segmentation of multi-channel images (section 4.6);
2. An inter-prediction loss function for multi-channel, multi-head networks (subsection A.1.1); and
3. An extended application of topological constraints—containment and exclusion—for boundary segmentation (subsection A.1.2).

A.1.1 Multi-Channel, Multi-Head Networks

Convolutional neural networks (CNNs) are well-suited for computer vision tasks because they leverage local receptive fields and weight sharing to capture spatially correlated features, thereby introducing a spatial hierarchy. By contrast, transformers rely on self-attention mechanisms that naturally handle long-range dependencies and dynamically focus on different parts of the input. Since transformers are permutation-invariant, they need positional encodings to maintain information about sequence ordering. Structurally, CNNs offer translation-invariant

inductive biases, whereas transformers introduce biases aligned with attention-based feature aggregation.

Multi-Channel Architectures in Bioimage Analysis. In bioimaging, multi-channel data are extremely common, with each channel representing a distinct marker or stain. I experimented with multi-channel, multi-head networks that can process multiple input channels and predict multiple outputs simultaneously. However, my experiments did not reveal substantial benefits from merging channels in the model architecture unless specific design constraints were imposed. Qualitative results suggest little improvement in areas that remain difficult to segment, even though expert human annotators often resolve these regions more easily by using complementary signals across channels.

In fact, chapter 2 and chapter 3 showcase state-of-the-art multi-channel pipelines that adopt a two-step process: one channel (e.g., a nucleus channel) is segmented first and is then used to refine another channel (e.g., a cell channel), and vice versa. In this manner, nuclear segmentation supports the merging or splitting of cells, while cell segmentation guides the detection or exclusion of nuclei. But these are not directly integrating the multi-channel information into the model representation.

From my observations, when fluorescence microscopy is employed and the structures of interest are already distinctly labelled, adding a noisier channel—such as an alternative stain of the same structure—does not necessarily improve learning. This holds especially true when annotated training data are scarce, because the less reliable channel can distract the model. In some bioimage segmentation contexts, a certain degree of overfitting to the primary channel(s) is even desirable. Similarly, appending a channel for an entirely different structure may be unhelpful unless the model is explicitly guided to integrate that additional information meaningfully.

My early experiments on multi-channel, multi-head networks started before 3D multi-channel datasets (Table A.1) were labelled. Around the same time, Mesmer [23] (a 2D cell-nuclear instance segmentation method) was released as a preprint. Mesmer uses a ResNet50 backbone with two semantic segmentation heads for joint cell and nuclear predictions. In contrast, my residual U-Net implementation[1] based on `pytorch-3dunet` [103] is theoretically similar; it covered architectures 1, 2, 4, and 5 in Table A.2. Rather than duplicating similar published experiments, I therefore focused on an *inter-prediction loss* function for enhancing the performance of multi-channel, multi-head networks.

Although the field has largely moved away from this specific optimisation strategy, by the time of writing, Cellpose-style multi-prediction of nuclear and cell flows has been thoroughly explored [104], yielding only marginal improvements. Should further experiments be conducted, Table A.2 and Table A.1 summarise the relevant architectural variants and benchmarking datasets, respectively.

Inter-Channel Loss. I next considered a loss function aimed at enforcing consistent interplay between different predicted channels. Standard losses such as the Dice loss operate on each output channel individually, focusing on pixel-wise overlap with the ground truth. In

[1] GitHub – qin-yu/segmenu/pull/3

Dimension	Dataset 1	Dataset 2
2D	COVID-IF	
3D	FOR2581	Sponge
4D	RIKEN	

Table A.1: Multi-Channel Datasets for Benchmarking Architectures. These datasets can be used to quantitatively benchmark the architectures: COVID-IF [105], FOR2581/GoNuclear [2], Sponge (private), and RIKEN (private) at EMBL.

	Input Head 1		Input Head 2		Output Head 1		Output Head 2	
	c1	c2	c1	c2	c1	c2	c1	c2
1a	Nuclei				Nuclei			
1b	Cells				Cells			
2a	Nuclei	Cells			Nuclei			
2b	Nuclei	Cells			Cells			
3	Nuclei	Cells			Nuclei + Cells			
4	Nuclei	Cells			Nuclei	Cells		
5	Nuclei	Cells			Nuclei		Cells	
2a′	Nuclei		Cells		Nuclei			
2b′	Nuclei		Cells		Cells			
3′	Nuclei		Cells		Nuclei + Cells			
4′	Nuclei		Cells		Nuclei	Cells		
5′	Nuclei		Cells		Nuclei		Cells	

Table A.2: Architectures of Baselines. These are architectural variants of the U-Net from [6, 103] for multi-channel, multi-head input and output. Bold entries are covered by my implementation and experiments. The results of these multi-channel/-head U-Net architectures on multi-channel datasets are not qualitatively better than single-channel input, single-channel output networks.

semantic segmentation, the Dice loss is typically:

$$L_{\text{dice}}(P, G) \;=\; 1 - \frac{2\sum_{i=1}^{N} p_i\, g_i}{\sum_{i=1}^{N} p_i \;+\; \sum_{i=1}^{N} g_i},$$

where P is the predicted segmentation and G the ground truth, each consisting of N pixels. While minimising $L_{\text{dice}}(P, G)$ aligns P with G, it does not constrain how *different* predicted channels should interact.

For example, if we want to penalise the overlap between two *predicted* channels, such as nuclei P and cell membranes Q, simply setting

$$L_{\text{inter-channel}} \;=\; \text{Dice}(P, Q)$$

and minimising it can inadvertently force both channels to expand while attempting to remain mutually exclusive. This phenomenon can lead to more background being classified as foreground[2], as shown in Figure A.1. My experimentation ended there, but a promising direction might be an inter-channel loss of the form

$$L_{\text{inter-channel}}(P, Q) = \frac{\sum_{i=1}^{N} p_i\, q_i}{f(P, Q)},$$

[2]GitHub – qin-yu/segmenu/issues/6

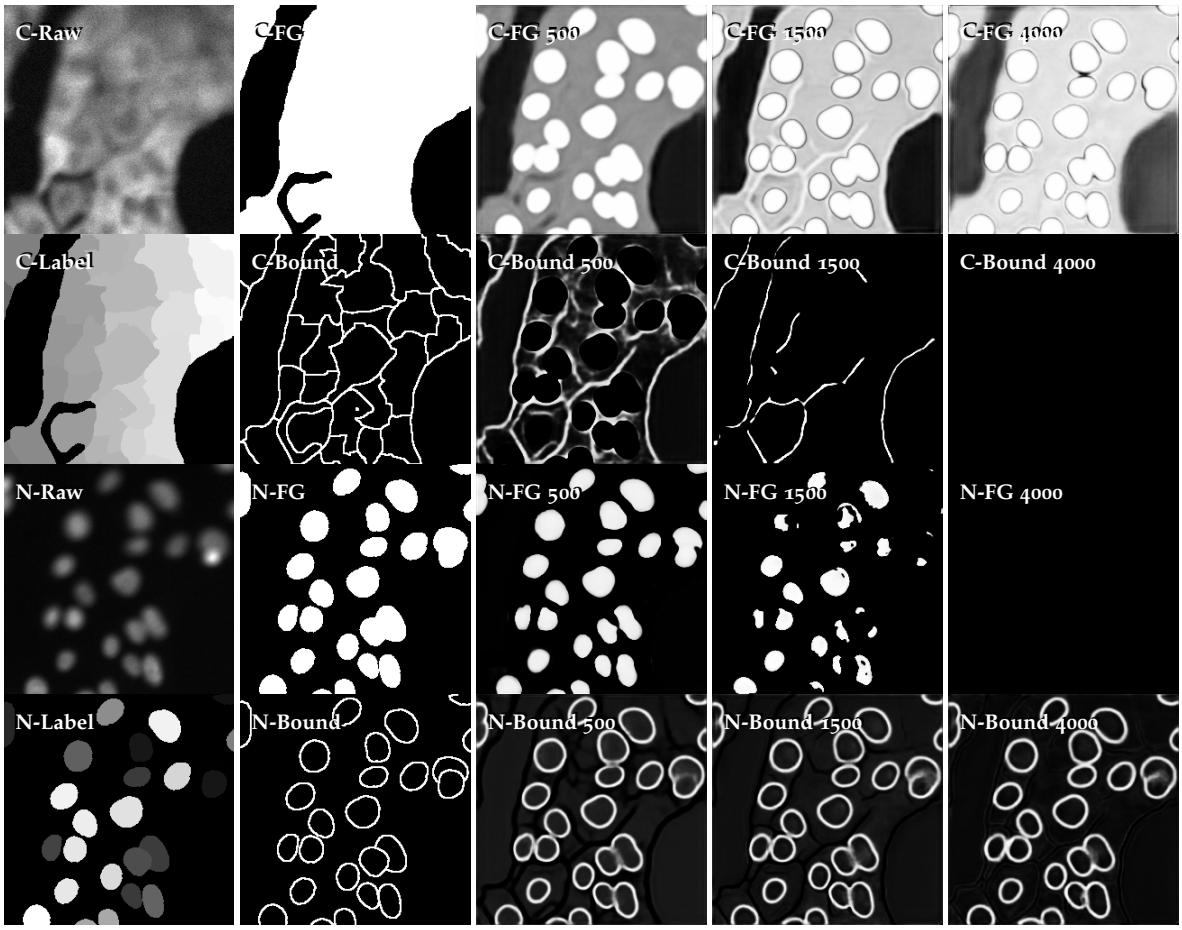


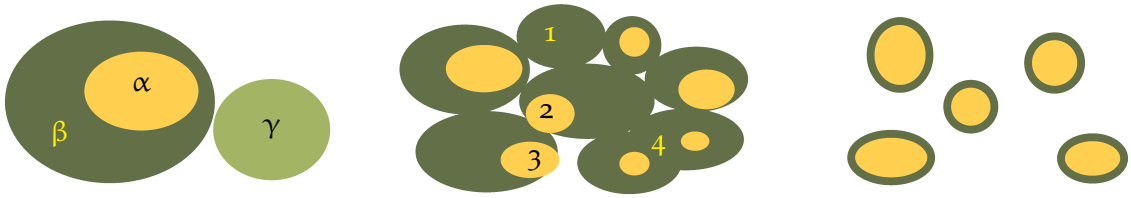
Figure A.1: Empirical Investigation of Inter-Channel Loss. Minimising a simple inter-channel Dice term encourages both channels to expand their predicted foregrounds while attempting to remain mutually exclusive, inadvertently leading to background regions being misclassified as foreground. C = Cell, N = Nucleus, FG = Foreground, Label = Ground Truth, Bound = Boundary. Image patches are taken from a 2-output-head U-Net trained on the 2-channel COVID-IF dataset [105]. Numbers indicate the training iteration.

where $f(P, Q)$ serves as an appropriate normalization. Selecting or designing such an $f(P, Q)$ to avoid excessive channel expansion remains an open challenge.

A.1.2 Topological Interaction for Instance Segmentation

It is now clear that a fundamental limitation of many deep learning approaches is their difficulty in capturing topological interactions among different classes. Standard per-pixel losses do not ensure global structural constraints, such as the requirement that nuclei must reside inside cells. In subsection A.1.1, I discussed my idea for an inter-channel loss. In related work, Gupta *et al.* introduced a topological interaction module that encodes global structure into deep neural networks [106].

Topological Interactions: Containment and Exclusion. Their key insight is that two important topological interactions, *containment* and *exclusion*, can be translated into constraints on allowable adjacent pixel labels. In other words, requiring that one class enclose another (con-



(A) Topological Interactions (B) Cells and Nuclei (C) Boundary and Foreground

Figure A.2: Illustration of topological constraints in segmentation. **(A)** Containment and exclusion constraints: β contains α ($\beta \rightarrow \alpha$); α and γ are mutually exclusive ($\alpha \nleftrightarrow \gamma$). **(B)** Examples of segmentation errors that containment or exclusion alone cannot fix: 1. Containment does not imply existence, 2. Exclusion *does* prevent nuclei outside cells, but 3. A nucleus overlapping two cells may still satisfy containment, 4. Semantic topological constraints alone do not fix merged cells. **(C)** Possible application of topological constraints for boundary segmentation in sparse instances; one may artificially add a boundary class and enforce containment/exclusion to improve segmentation quality.

tainment) or that two classes never touch (exclusion) can be enforced by penalising "illegal" label pairs. As illustrated in Figure A.2 (A), β contains α means that a pixel labelled α cannot be adjacent to any label other than α or β, while α and γ being mutually exclusive means adjacent pixels cannot form the pair (α, γ) or (γ, α).

To implement these constraints purely through convolutional operations, they propose a post-dilation strategy: each semantic class is dilated, and any overlapping region forms a "critical region" that should not exist if the global constraints hold. The model is then penalised for mismatches in these critical regions, effectively encoding local adjacency rules that capture global structural constraints.

Containment and Exclusion in Multi-Channel Bioimage Instance Segmentation. Although promising, this approach does not straightforwardly extend to instance segmentation in bioimaging. In many medical segmentation settings, it is sufficient to have one class enclose another or to enforce mutual exclusivity. However, in bioimage analysis, each cell often must contain exactly one nucleus, and a network might benefit from learning such finer-grained constraints when boundary signals are faint or the nuclear stain is weak. As shown in Figure A.2 (B), certain segmentation errors cannot be corrected by containment or exclusion alone.

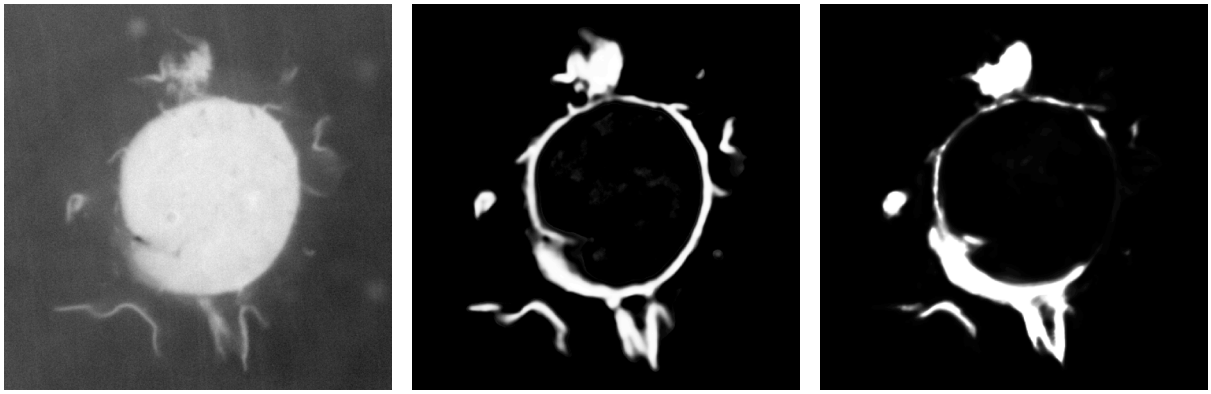
Two main difficulties arise. First, the transition from semantic to instance segmentation is generally not differentiable, complicating back-propagation of errors. Second, instance matching (i.e., pairing a particular cell instance with its nucleus) is also not differentiable. Nonetheless, progress has been made. For example, the SPOCO framework (chapter 4) employs a differentiable method for selecting individual instances and applying auxiliary losses at the instance level, suggesting a possible path to incorporate topological constraints into instance-level training.

Containment and Exclusion for Boundary Segmentation. Beyond joint cell-nucleus segmentation, topological constraints can also enhance boundary segmentation, as illustrated in Figure A.2 (C). High-quality boundary delineation is critical for two-stage instance-segmentation pipelines such as PlantSeg (chapter 3), where accurate boundaries are a prerequisite for correct instance separation. By enforcing containment constraints between

boundary and foreground predictions, clearer and more robust boundary segmentation can be achieved. This is particularly useful in sparse-instance scenarios, such as those in GoNuclear (chapter 2).

For instance, enforcing a constraint that the cytoplasm must be fully enclosed by the cell membrane prediction can prevent gaps or inconsistencies in boundary segmentation. Without such constraints, deep learning models may predict broken or incomplete boundaries, as observed in Figure A.3. Ensuring that boundaries separate different compartments correctly is essential for downstream analysis, particularly in cases where precise membrane delineation is required to infer cell morphology, interactions, or subcellular organization.

In summary, topological interaction constraints offer a promising direction for improving multi-channel bioimage segmentation, particularly in tasks that demand strong inter-channel relationships or accurate structural delineations. While formulating these constraints for fully differentiable instance segmentation remains challenging, further research in this direction holds significant potential.



(A) Cell Surface Image (B) Closed Membrane Prediction (C) Open Membrane Prediction

Figure A.3: Potential effect of topological constraints on boundary segmentation. Topological constraints ensure that the cytoplasm is fully enclosed by the predicted cell membrane, preventing gaps or inconsistencies. **(A)** A 2D Z-slice of the inverted negative channel from Figure 5.1. **(B)** A 2D example of a mostly well-formed cell boundary prediction from the "v1 plus" model in Figure 5.2, though a small gap remains. **(C)** A 2D example of a broken cell boundary prediction from the "v2 mix-raw-label val" model in Figure 5.2, where the missing boundary violates containment constraints.

A.2 Contribution to PlantSeg

Since my participation as a core developer of PlantSeg, the rate of GitHub Star acquisition has increased by 34.34% (Statistics by the end of 2024).

Channel Support

- Backward compatible `key` & `channel` params for config pipeline #165
- Fix Flow in GUI & Napari: key, channel, LMC in GUI; multi-channel UNet layers in Napari #184
- Headless: Handle multi-channel prediction output and add docs #239

Prediction Support

- `feat(pred)`: check and free VRAM for prediction with oversized patch #272
- Automatic patch shape recommendation #302
- Compute best patch and halo shapes by default, reduce user effort #307
- Fix size finder and widgets for 2D UNet #313
- Improve model filters | Use RadioButtons | Custom modality #314
- `fix`: allow more runtime errors in `find_batch_size()` #359
- `fix`: allow but warn all runtime errors in 3D `find_a_max_patch_shape()` #361
- PlantSeg v2: full codebase refactor with unified task framework, enhanced prediction/proofreading, and metadata handling #285
  - `fix(enum)`: improve use of Enum in prediction widget
  - `fix(prediction)!`: properly handle multichannel output in prediction task
  - `fix(task)!`: properly handle multiple outputs
  - `feat(widget-pred)`: create widget for adding custom models
  - `fix(test-zoo)`: update use of `torch.load()`

Halo Support

- Add validation, adaptation and halo to prediction widgets #211
- Fix halo implementation and tiling artefact #220
- Fix Tiling Artefact: intensity normalisation was correct #223
- Recommend users a halo for U-Nets #252
- PlantSeg v2: full codebase refactor with unified task framework, enhanced prediction/proofreading, and metadata handling #285
  - `fix(halo)`: fix halo not used in v2

GoNuclear Support

- Add Zenodo record for generic plant nuclei model #169
- Rename generic plant nuclear model for paper submission #193

BioImage.IO Support

- Central model records #216
- Fix `CYX`/`CZYX` pad/halo, fix `io.zarr`/tests, fix typing, refactor model load #233
- BioImage.IO Core: Execute PlantSeg-compatible models by nickname #247
- BioImage.IO Core: Improve interface for PlantSeg models #248
- Add inference API and task for all BioImage.IO Model Zoo models #338

Documentation

- Add documentation folder and Actions for https://hci-unihd.github.io/plant-seg/ #179
- Redirect all Wiki links to new doc and improve READMEs #191
- Documentation: Migrate to MkDocs; update installation, README, and logo #230
- Doc: Data/models, train/add new models, Jupyter Book remnant #240
- Doc: Fix and update rescaling widget screenshot docs #259
- Doc: Add revision date and committer plugins #260
- `fix`: links in docs, docstring warning and theme override #269
- **`fix(docs)`: teach Qt to find napari icons** #271
- Fix and document distance transform watershed for 2D pmaps/images #318

- PlantSeg v2 docs base #320
- Recover and document over-/under-seg fix by nuclei #321
- PlantSeg v2: Finalise docs #322
- Document missing button issue #327
- `ci(docs)`: fetch all git history/tags/branches for MkDocs dates #366

Widgets

- Add widget: remove objects using a probability map #202
- Add important widgets and improve logger further #297
- Recover and document over-/under-seg fix by nuclei #321
- Add un-/re-do and save-/load-states for proofreading tool #323
- Add image pair operation widget/task/functionals/docs #370
- PlantSeg v2: full codebase refactor with unified task framework, enhanced prediction/proofreading, and metadata handling #285
  - **Proofreading**
    - `feat(proofread)`: add proofreading code to new location
    - `feat(widget-proofreading)`: create task and widget for removing false positives
  - **Segmentation**
    - `fix(task-seg)`: fix args and raise in agglomerative seg task
    - `fix(widget-seg)`: fix args name conflict with `magicgui`
    - `feat(widget-seg)`: create `dt-watershed` widget
    - `feat(widget-seg)`: create lifted-multicut (LMC) task and widget

Usability

- Add option to only run official models in "Try all Available Models" #207
- **`refactor: use magicgui.types.Separator`** #311
- Fix TIFF default layer name and hide key widget #312
- Improve model filters | Use RadioButtons | Custom modality #314
- Fix and document distance transform watershed for 2D pmaps/images #318
- Fix and integrate lifted multicut widget into agglomeration widget #328
- Refactor and recover widgets: cropping and label processing #330
- PlantSeg v2: Final polishment of GUI #326
- `refactor(GUI)`: rename and reorder widgets for intuitive sequential flow #331
- Fix smart load | Make extension check case-insensitive #368

Maintainability and Reliability

- Add validation, adaptation and halo to prediction widgets #211
- Fix Legacy GUI after v1.7.0 #237
- Group widgets for batch `.hide() / .show()` #317
- `fix`: widgets should have optional arg for updating other widgets #362

Refactoring Core

- Create `PlantSeg.Core`: `ModelZoo`, `PlantSegImage`, and `VoxelSize` #298
  - `feat(core)!`: collect key custom classes into `plantseg/core/`
  - `fix(PlantSegImage)`: fix dataset name (key) mismatch

  - `fix(PlantSegImage)`: improve `_load_data()`, explicit key, better `.zarr` check
- Test and refactor `.core` and `functionals.dataprocessing` #303
- Make `VoxelSize` iterable and array-like, remove `.is_valid` #310
- Fix voxel size unit #343
  - `fix(IO)`: allow `"micron"` as a voxel size unit
  - `feat(IO)`: allow ImageJ-like unit name parsing, related to #288
  - `fix(VoxelSize)`: `.io.utils` is not used by `.io` but a module for `VoxelSize`
- PlantSeg v2: full codebase refactor with unified task framework, enhanced prediction/proofreading, and metadata handling #285

Fix File I/O

- `fix(widget-IO)`: temp fix #282 and auto-generate layer name #304
- **Properly refresh ComboBox choices with functions** #305

Logging

- Unify and standardise logging in PlantSeg v2 #296
- Add important widgets and improve logger further #297
  - `fix(logging)`: replace deprecated `warn()`s
  - `feat(logging)`: unify logging
  - `feat(logging)!`: unify logging using simpler logger hierarchy
  - `chore(logging)`: use module-level logger instead of `print()`
  - `fix(logging)`: improve and make compatible v1 PlantSeg Napari logger

Typing and Input Checking

- Type Fix: `np.array` is not a type, change to `np.ndarray` #231
- Fix `CYX`/`CZYX` pad/halo, fix `io.zarr`/tests, fix typing, refactor model load #233
- Improve use of `Enum` as widget choices and fix types #256
- `fix(cropping)`: use `RangeSlider` to fix #278 | handle `None` image #279
- PlantSeg v2: full codebase refactor with unified task framework, enhanced prediction/proofreading, and metadata handling #285
  - **Type and Name**
    - `fix(ImageType)`: match `ImageType` to layer type
    - `chore(trainer)`: improve types and variable names
    - `chore(zoo)`: replace constants with `Enum`
    - `fix!`: do not use label, labels or _labels for widget names
    - `fix(typing)`: `Tuple` and `List` are deprecated, use `tuple` and `list`
    - `fix(typing)`: avoid mixed `|` and `Optional`, update docstring
  - **Axis Order**
    - `fix(IO)`: use consistent ZYX axis order, fix #289
    - `fix(ImageLayout)`: change axis order __XY to __YX, fix #289
    - `fix(rescale)`: fix non-existent `ImageLayout`
    - `fix(IO)`: make ZCYX images CZYX at import, fix #294
  - **Tasks**
    - `fix(task)`: avoid unnecessary repetition in layer suggestion
    - `fix(task)`: properly handle `None` layer suggestion

Testing

- Tests for rescaling widget #258
- Test: Fix always-skipped test and misleading name #262
- Tests: Fix all existing tests #299
- Test and refactor `.core` and `functionals.dataprocessing` #303

CI

- Use `pre-commit` #263
- `chore`: migrate from `bumpversion` to `bump-my-version` #266
- `chore(pre-commit)`: add conventional commits #267
- Use Codecov #300
- PlantSeg v2: full codebase refactor with unified task framework, enhanced prediction/proofreading, and metadata handling #285
  - `chore(ruff)`: enable isort (`I`) for import
  - `docs(ci)`: check CI before writing tests for v2

Installation & Environment

- **Dev Env**: Add BioImage.IO Core dependency and fix CUDA installation #229
- `fix(installation)`: remove old configs if `KeyError` | macOS installation #277
- Fix version related issues: bump version, comparison and Anaconda upload #333
- `fix(env)`: Pydantic v2.10 causes `bioimageio.spec` issues #363
- Fix and promote PlantSeg v1 installation #364
- PlantSeg v2: full codebase refactor with unified task framework, enhanced prediction/proofreading, and metadata handling #285
  - `chore(env)`: improve development environment configuration file
  - `chore(env)`: fix environment due to Anaconda's new policy for EMBL
  - `chore(env)`: improve environment configuration with default channel absent

GUI

- Napari GUI: Fix voxel size precision, font size, and rearrange widgets #227
- `fix`: avoid reading logo file from docs folder #336
- Match conda recipe and include `package_data` logo #348

A.2.1 Contribution to PlantSeg by Others

Bug Fixes

- `fix`: `remove_false_positives_by_foreground_probability` only works in 3D #353 #354

Training

- Add support for training #159

I/O

- Extend channel selection #212
- Fix `h5` export #335
- New export widget #341

Proofreading

- Proofreading improvements #355

Headless

- Improved Headless + GUI #356

Documentation

- API docs | Docs screenshots | Refactor rescaling | Rearrange widgets #249
- Fix docs build in the CI #257

Installation

- Fixes PlantSeg crash on Apple Silicon #273 #275
- Fix performance on Mac #337
- `fix` (#344): update Windows install instructions #347

Removed Features

- Add widget manager for extra-pred and extra-seg #221

A.3 Complementary Figures and Tables

N	↓ Label	Raw →	`negative`	`negative` + `actin` (pro)
4	`ilastik`		initial	~~initial pro~~
9	`ilastik`		initial plus	initial pro plus
9	`ilastik`		initial plus val	initial pro plus val
9	`initial`		second (val)	second pro (val)
9	`ilastik`	`+initial`	second max	second pro max
9	`ilastik`	`+initial`	second max rand	second pro max rand
9	`ilastik`	`+initial`	second max val	second pro max val
9	`ilastik`	`+initial`	second max rand val	second pro max rand val

Table A.3: All Actin Models with Old Names. This version matches files on EMBL storage. Terminologies: the `initial` model is trained with 4 pairs of negative-ilastik images as raw-label input; `plus` models are initial models trained with 5 more raw-label image pairs; `val` models are only validated with segmentation generated from the `initial` model; `pro` models use an extra `actin` channel; `max` models are `second` models using segmentation generated from the `initial` model in addition to ilastik labels used in training `initial` models; `rand` models have training configurations with randomised order of file paths where an ilastik label is first, other models have alternating initial-ilastik labels.

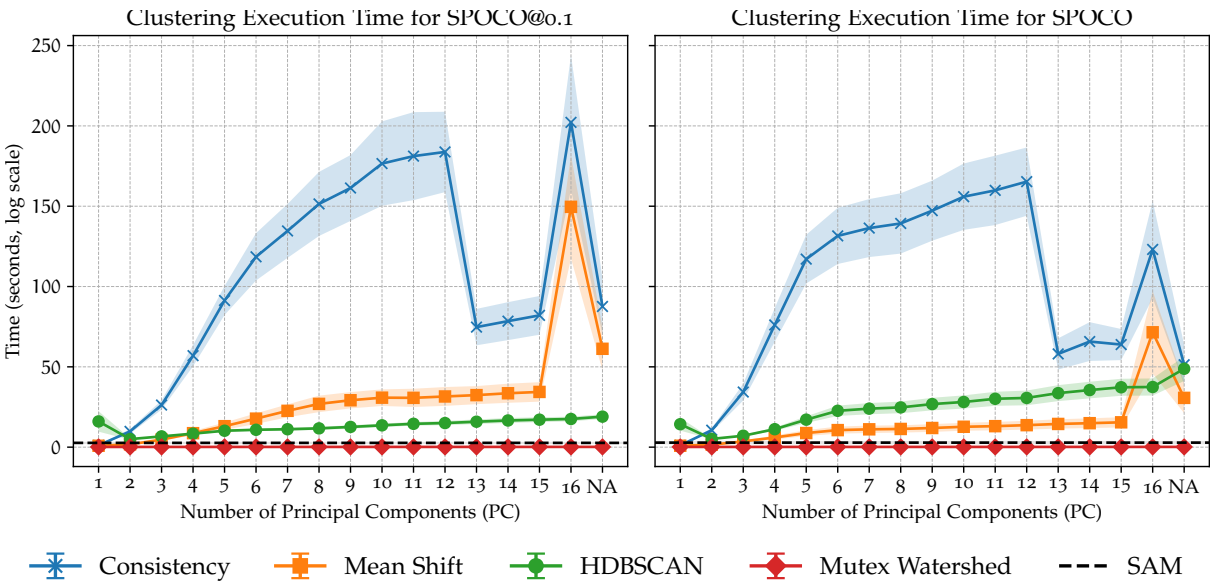


Figure A.4: Pixel Embedding Clustering: Computational Efficiency. The left plot shows SPOCO@0.1 (trained with 10% of instance masks), the right plot shows the fully supervised SPOCO. In both, higher dimensionality inflates runtime for HDBSCAN, mean shift, and consistency clustering. Mutex Watershed is consistently faster, and SAM (on GPU) completes each image in seconds, about an order of magnitude slower than Mutex Watershed but far faster than the other algorithms. An identical plot with log y scale is in Figure 4.15.

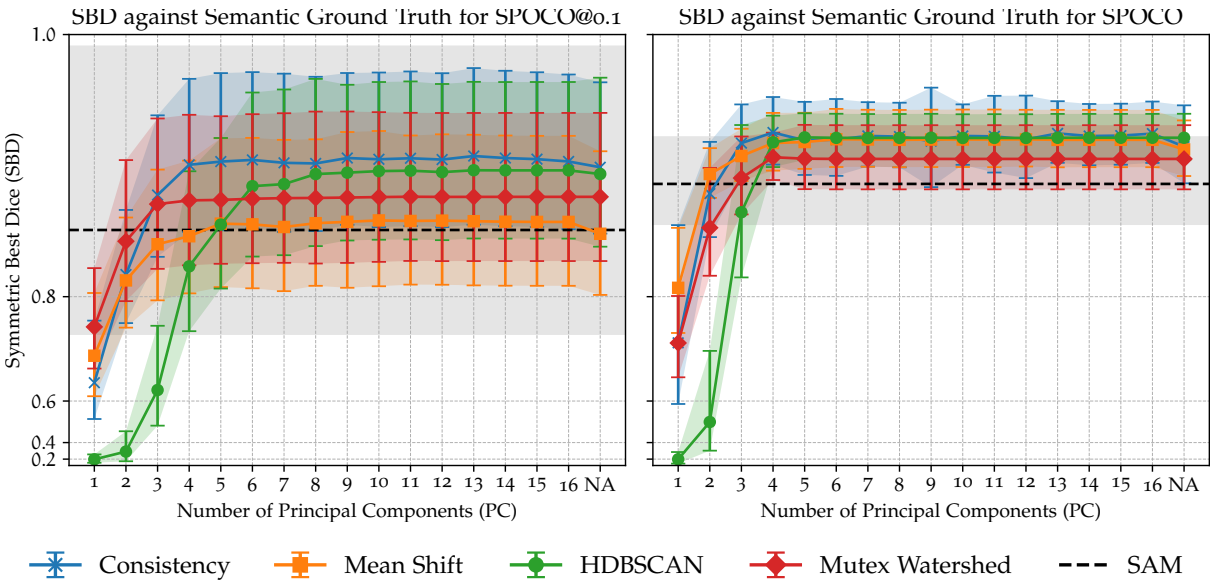


Figure A.5: Pixel Embedding Clustering: Accuracy. SBD(*ground truth*, *binary segmentation*) is plotted for the SPOCO@0.1 model (left) and the fully supervised SPOCO (right). Increasing the number of PCA components does not necessarily improve semantic accuracy. Notably, 4–5 components already suffice in most cases. A simplified clear version is in Figure 4.13.

Scores AP0.5		Cellpose						StarDist												PlantSeg						Scores AP0.5	
		fintune nuclei		finetune cyto2		scratch		ResNet		ResNet grid 244		U-Net		U-Net grid 244		ResNet large		ResNet large grid 244		small		large		max			
		1135	5-fold	1135	5-fold	1135	5-fold	1135	5-fold	1135	5-fold	1135	5-fold	1135	5-fold	1135	5-fold	1135	5-fold	1135	5-fold	1135	5-fold	1135	5-fold		
isotropic XY, YZ, ZX planes	Initial	87.03%	86.78%	89.04%		90.34%		88.60%	91.36%			91.73%								88.99%	88.82%	87.90%				Initial	3D rotate isotropic
	Gold	88.89%	84.30%	88.13%	84.13%	88.51%	83.98%	89.56%	90.92%	95.99%	96.03%									94.78%	94.57%					Gold	
isotropic XY planes	Initial											91.97%										91.74%				Initial	2D rotate isotropic
	Gold							89.51%	92.47%	96.31%	96.18%									95.37%	94.79%					Gold	
original XY plane	Initial							91.00%	92.21%	96.54%	95.92%	92.78%								90.71%	89.94%	94.69%				Initial	2D rotate original
	Gold	74.15%	74.43%	73.81%	74.27%	76.98%	74.60%	93.13%	93.69%	96.97%	97.30%	96.64%	96.52%	96.30%	96.98%	96.80%	96.65%	96.81%	97.04%	96.53%	96.74%			96.36%	95.79%	Gold	

Scores mAP		Cellpose						StarDist												PlantSeg						Scores mAP	
		fintune nuclei		finetune cyto2		scratch		ResNet		ResNet grid 244		U-Net		U-Net grid 244		ResNet large		ResNet large grid 244		small		large		max			
		1135	5-fold	1135	5-fold	1135	5-fold	1135	5-fold	1135	5-fold	1135	5-fold	1135	5-fold	1135	5-fold	1135	5-fold	1135	5-fold	1135	5-fold	1135	5-fold		
isotropic XY, YZ, ZX planes	Initial	53.09%	43.64%	55.96%		57.36%		55.77%	56.61%			57.71%								56.48%	55.34%	54.63%				Initial	3D rotate isotropic
	Gold	53.38%	51.96%	52.49%	51.05%	53.42%	51.26%	55.02%	54.07%	61.24%	62.78%									68.73%	68.42%					Gold	
isotropic XY planes	Initial											59.81%										60.00%				Initial	2D rotate isotropic
	Gold							57.26%	58.47%	65.97%	65.79%									68.45%	68.25%					Gold	
original XY plane	Initial							68.07%	66.19%	58.03%	67.61%	68.04%								60.43%	57.40%	66.12%				Initial	2D rotate original
	Gold	43.16%	44.12%	43.01%	44.10%	46.10%	44.54%	66.86%	69.70%	77.31%	78.33%	75.81%	75.79%	77.60%	78.25%	65.93%	67.32%	78.09%	77.63%	78.39%	78.80%			78.02%	76.96%	Gold	

Additional AP0.5		PS-CP Nuclei Default	
		1135	5-fold
original	Initial	82.80%	88.41%
	Gold	91.30%	93.19%

Additional mAP		PS-CP Nuclei Default	
		1135	5-fold
original	Initial	37.10%	50.64%
	Gold	60.32%	61.68%

Table A.4: Complete mAP^{50} and $mAP^{50:5:95}$ Scores for All Evaluated Models in the GoNuclear Study. To identify optimal training strategies for 3D segmentation of weakly stained nuclei, I systematically benchmarked PlantSeg, StarDist, and Cellpose under varying conditions. These included: (i) anisotropic vs. isotropic voxel resampling, (ii) 2D vs. 3D rotation augmentations, (iii) dimensional merging strategies in Cellpose, (iv) object-size calibration in StarDist, and (v) backbone architecture choice. Shown here are two complementary metrics: mAP^{50}, reflecting object detection accuracy, and $mAP^{50:5:95}$, a more stringent instance segmentation score. Results are averaged over fivefold cross-validation (5 models per configuration). Table cell colours aid visual interpretation but are not essential for reading the results.

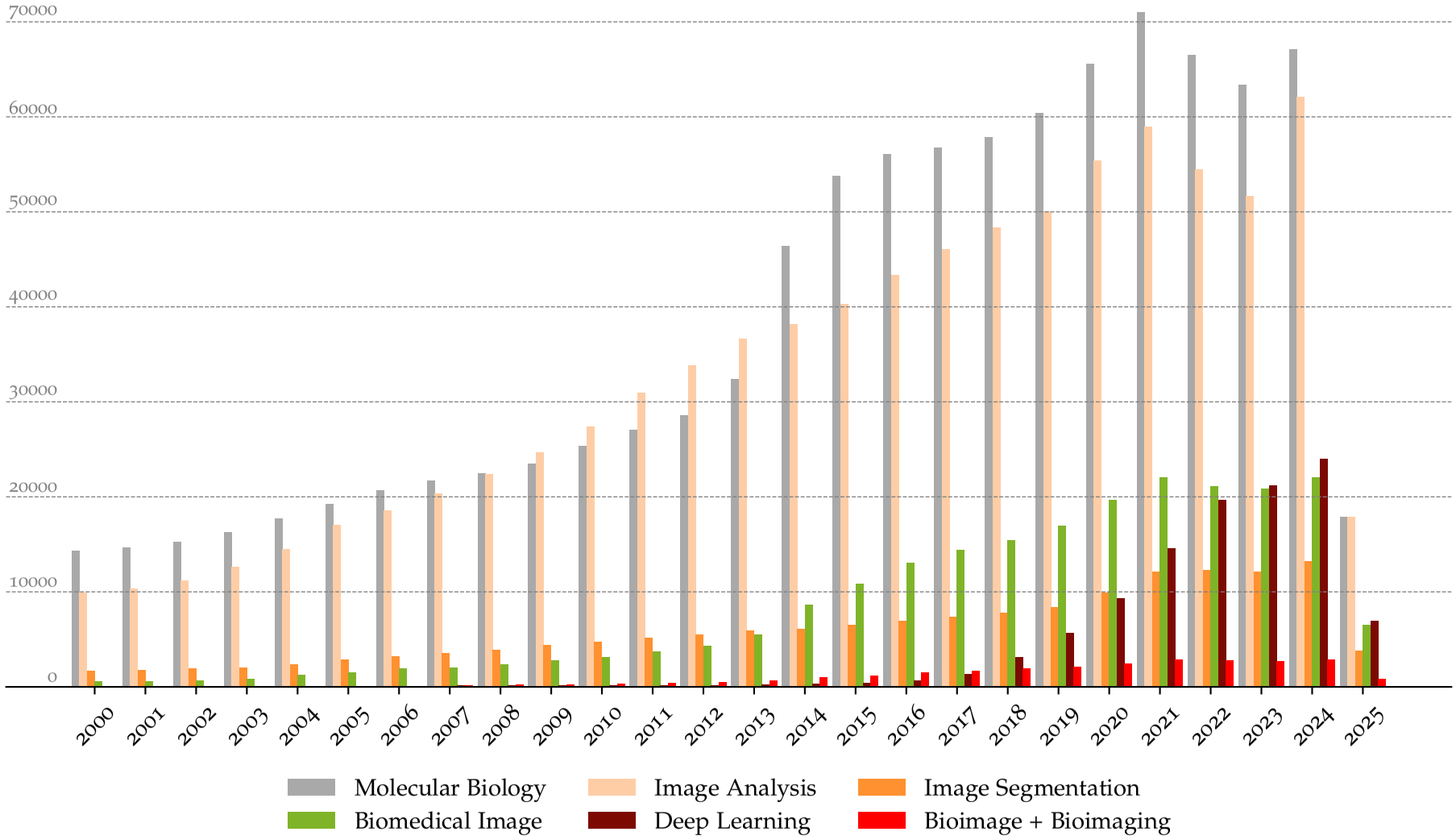


Figure A.6: PubMed Publication Counts by Keywords (2000–2024). This figure illustrates the growth of publications indexed in PubMed from 2000 to 2024 for multiple research keywords: "Molecular Biology," "Image Analysis," "Image Segmentation," "Biomedical Image," "Deep Learning," and "Bioimage" (grouped with "Bioimaging"). While "deep learning" skyrocketed from 384 papers in 2015 to 23,951 in 2024, only 146 papers explicitly referenced "bioimage" or "bio-image" in 2024. In contrast, 2,685 publications used "bioimaging" and 22,004 used "biomedical image." The term "bioimage analysis" is even rarer. This highlights the niche yet growing nature of bioimage analysis at the intersection of biology, imaging, and AI.

References

[1] A. Wolny, Q. Yu, C. Pape, and A. Kreshuk, "Sparse object-level supervision for instance segmentation with pixel embeddings," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA: IEEE, Jun. 2022, pp. 4392–4401. DOI: 10.1109/CVPR52688.2022.00436 (cit. on pp. 1, 7–9, 63, 66–68, 70, 76).

[2] A. Vijayan, T. A. Mody, Q. Yu, *et al.*, "A deep learning-based toolkit for 3d nuclei segmentation and quantitative analysis in cellular and tissue context," *Development*, vol. 151, no. 14, dev202800, Jul. 18, 2024. DOI: 10.1242/dev.202800 (cit. on pp. 1, 3, 7, 15, 21, 23, 36, 37, 113, 119).

[3] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 1, 2001. DOI: 10.1023/A:1010933404324 (cit. on p. 5).

[4] S. Berg, D. Kutra, T. Kroeger, *et al.*, "Ilastik: Interactive machine learning for (bio)image analysis," *Nature Methods*, vol. 16, no. 12, pp. 1226–1232, Dec. 2019. DOI: 10.1038/s41592-019-0582-9 (cit. on pp. 5, 6, 10).

[5] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28 (cit. on pp. 5, 6).

[6] A. Wolny, L. Cerrone, A. Vijayan, *et al.*, "Accurate and versatile 3d segmentation of plant tissues at cellular resolution," *eLife*, vol. 9, C. S. Hardtke, D. C. Bergmann, D. C. Bergmann, and M. Graeff, Eds., e57613, Jul. 29, 2020. DOI: 10.7554/eLife.57613 (cit. on pp. 6, 8, 11, 18, 41, 45, 50, 56, 119).

[7] M. Weigert, U. Schmidt, R. Haase, K. Sugawara, and G. Myers, "Star-convex polyhedra for 3d object detection and segmentation in microscopy," in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Snowmass Village, CO, USA: IEEE, Mar. 2020, pp. 3655–3662. DOI: 10.1109/WACV45572.2020.9093435 (cit. on pp. 6, 8, 11, 15, 18).

[8] C. Stringer and M. Pachitariu, "Cellpose 2.0: How to train your own model," Bioinformatics, preprint, Apr. 5, 2022. DOI: 10.1101/2022.04.01.486764 (cit. on pp. 6–9, 11, 22).

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, *An image is worth 16x16 words: Transformers for image recognition at scale*, Jun. 3, 2021. DOI: 10.48550/arXiv.2010.11929. arXiv: 2010.11929[cs] (cit. on p. 6).

[10] A. Archit, L. Freckmann, S. Nair, *et al.*, "Segment anything for microscopy," *Nature Methods*, pp. 1–13, Feb. 12, 2025. DOI: 10.1038/s41592-024-02580-4 (cit. on pp. 6, 9, 101, 102).

[11] J. Lehtinen, J. Munkberg, J. Hasselgren, *et al.*, *Noise2noise: Learning image restoration without clean data*, Oct. 29, 2018. DOI: 10.48550/arXiv.1803.04189. arXiv: 1803.04189[cs] (cit. on pp. 6, 9).

[12] J. Batson and L. Royer, "Noise2self: Blind denoising by self-supervision," in *Proceedings of the 36th International Conference on Machine Learning*, PMLR, May 24, 2019, pp. 524–533 (cit. on pp. 6, 9).

[13] A. Krull, T.-O. Buchholz, and F. Jug, *Noise2void - learning denoising from single noisy images*, Apr. 5, 2019. DOI: 10.48550/arXiv.1811.10980. arXiv: 1811.10980[cs] (cit. on pp. 6, 98).

[14] H. Kobayashi, K. C. Cheveralls, M. D. Leonetti, and L. A. Royer, "Self-supervised deep learning encodes high-resolution features of protein subcellular localization," *Nature Methods*, vol. 19, no. 8, pp. 995–1003, Aug. 2022. DOI: 10.1038/s41592-022-01541-z (cit. on pp. 6, 9).

[15] A. Gupta, Z. Wefers, K. Kahnert, *et al.*, *SubCell: Vision foundation models for microscopy capture single-cell biology*, Dec. 8, 2024. DOI: 10.1101/2024.12.06.627299 (cit. on p. 6).

[16] O. Kraus, K. Kenyon-Dean, S. Saberian, *et al.*, "Masked autoencoders for microscopy are scalable learners of cellular biology," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA: IEEE, Jun. 16, 2024, pp. 11 757–11 768. DOI: 10.1109/CVPR52733.2024.01117 (cit. on p. 6).

[17] M. Doron, T. Moutakanni, Z. S. Chen, *et al.*, *Unbiased single-cell morphology with self-supervised vision transformers*, Jun. 18, 2023. DOI: 10.1101/2023.06.16.545359 (cit. on p. 6).

[18] V. Zinchenko, J. Hugger, V. Uhlmann, D. Arendt, and A. Kreshuk, "MorphoFeatures for unsupervised exploration of cell types, tissues, and organs in volume electron microscopy," *eLife*, vol. 12, L. K. Scheffer, C. Desplan, L. K. Scheffer, and J. Funke, Eds., e80918, Feb. 16, 2023. DOI: 10.7554/eLife.80918 (cit. on p. 6).

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cit. on p. 7).

[20] M. Hu, J. Zhang, L. Matkovic, T. Liu, and X. Yang, "Reinforcement learning in medical image analysis: Concepts, applications, challenges, and future directions," *Journal of Applied Clinical Medical Physics*, vol. 24, no. 2, e13898, Jan. 10, 2023. DOI: 10.1002/acm2.13898 (cit. on p. 7).

[21] M. Hartley, G. J. Kleywegt, A. Patwardhan, U. Sarkans, J. R. Swedlow, and A. Brazma, "The BioImage archive – building a home for life-sciences microscopy data," *Journal of Molecular Biology*, Computation Resources for Molecular Biology, vol. 434, no. 11, p. 167 505, Jun. 15, 2022. DOI: 10.1016/j.jmb.2022.167505 (cit. on p. 9).

[22] C. Edlund, T. R. Jackson, N. Khalid, *et al.*, "LIVECell—a large-scale dataset for label-free live cell segmentation," *Nature Methods*, vol. 18, no. 9, pp. 1038–1045, Sep. 2021. DOI: 10.1038/s41592-021-01249-6 (cit. on p. 9).

[23] N. F. Greenwald, G. Miller, E. Moen, *et al.*, "Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning," *Nature Biotechnology*, vol. 40, no. 4, pp. 555–565, Apr. 2022. DOI: 10.1038/s41587-021-01094-0 (cit. on pp. 9, 118).

[24] C. Spahn, E. Gómez-de-Mariscal, R. F. Laine, *et al.*, "DeepBacs for multi-task bacterial image analysis using open-source deep learning approaches," *Communications Biology*, vol. 5, no. 1, pp. 1–18, Jul. 9, 2022. DOI: 10.1038/s42003-022-03634-z (cit. on p. 9).

[25] C. Stringer and M. Pachitariu, "Cellpose3: One-click image restoration for improved cellular segmentation," *Nature Methods*, vol. 22, no. 3, pp. 592–599, Mar. 2025. DOI: 10.1038/s41592-025-02595-5 (cit. on p. 9).

[26] A. Archit, S. Nair, N. Khalid, *et al.*, *Segment anything for microscopy*, Aug. 22, 2023. DOI: 10.1101/2023.08.21.554208 (cit. on pp. 11, 101).

[29] 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, Jan. 2021. DOI: 10.1038/s41592-020-01018-x (cit. on pp. 15, 18, 22).

[30] P. Barbier de Reuille, A.-L. Routier-Kierzkowska, D. Kierzkowski, *et al.*, "MorphoGraphX: A platform for quantifying morphogenesis in 4d," *eLife*, vol. 4, D. C. Bergmann, Ed., e05864, May 6, 2015. DOI: 10.7554/eLife.05864 (cit. on pp. 16, 18).

[31] G. W. Bassel, P. Stamm, G. Mosca, *et al.*, "Mechanical constraints imposed by 3d cellular geometry and arrangement modulate growth patterns in the arabidopsis embryo," *Proceedings of the National Academy of Sciences*, vol. 111, no. 23, pp. 8685–8690, Jun. 10, 2014. DOI: 10.1073/pnas.1404616111 (cit. on p. 16).

[32] Y. Fridman, S. Strauss, G. Horev, *et al.*, "The root meristem is shaped by brassinosteroid control of cell geometry," *Nature Plants*, vol. 7, no. 11, pp. 1475–1484, Nov. 2021. DOI: 10.1038/s41477-021-01014-9 (cit. on p. 16).

[33] M. Graeff, S. Rana, J. R. Wendrich, *et al.*, "A single-cell morpho-transcriptomic map of brassinosteroid action in the *Arabidopsis* root," *Molecular Plant*, vol. 14, no. 12, pp. 1985–1999, Dec. 6, 2021. DOI: 10.1016/j.molp.2021.07.021 (cit. on p. 16).

[34] E. Hernandez-Lagana, G. Mosca, E. Mendocilla-Sato, *et al.*, "Organ geometry channels reproductive cell fate in the arabidopsis ovule primordium," *eLife*, vol. 10, S. McCormick and J. Kleine-Vehn, Eds., e66031, May 7, 2021. DOI: 10.7554/eLife.66031 (cit. on p. 16).

[35] J. Lora, M. Herrero, M. R. Tucker, and J. I. Hormaza, "The transition from somatic to germline identity shows conserved and specialized features during angiosperm evolution," *New Phytologist*, vol. 216, no. 2, pp. 495–509, 2017. DOI: 10.1111/nph.14330 (cit. on p. 16).

[36] T. D. Montenegro-Johnson, P. Stamm, S. Strauss, *et al.*, "Digital single-cell analysis of plant organ development using 3dcellatlas," *The Plant Cell*, vol. 27, no. 4, pp. 1018–1033, Apr. 1, 2015. DOI: 10.1105/tpc.15.00175 (cit. on p. 16).

[37] I. Ouedraogo, M. Lartaud, C. Baroux, *et al.*, "3d cellular morphometrics of ovule primordium development in zea mays reveal differential division and growth dynamics specifying megaspore mother cell singleness," *Frontiers in Plant Science*, vol. 14, May 12, 2023. DOI: 10.3389/fpls.2023.1174171 (cit. on p. 16).

[38] T. Pasternak, T. Haser, T. Falk, O. Ronneberger, K. Palme, and L. Otten, "A 3d digital atlas of the nicotiana tabacum root tip and its use to investigate changes in the root apical meristem induced by the agrobacterium 6b oncogene," *The Plant Journal*, vol. 92, no. 1, pp. 31–42, 2017. DOI: 10.1111/tpj.13631 (cit. on p. 16).

[39] T. Schmidt, T. Pasternak, K. Liu, *et al.*, "The iRoCS toolbox – 3d analysis of the plant root apical meristem at cellular resolution," *The Plant Journal*, vol. 77, no. 5, pp. 806–814, 2014. DOI: 10.1111/tpj.12429 (cit. on pp. 16, 18).

[40] A. Vijayan, R. Tofanelli, S. Strauss, *et al.*, "A digital 3d reference atlas reveals cellular growth patterns shaping the arabidopsis ovule," *eLife*, vol. 10, S. McCormick, C. S. Hardtke, S. McCormick, and D. Weijers, Eds., e63262, Jan. 6, 2021. DOI: 10.7554/eLife.63262 (cit. on pp. 16, 18, 22).

[41] S. Yoshida, P. Barbier de Reuille, B. Lane, *et al.*, "Genetic control of plant development by overriding a geometric division rule," *Developmental Cell*, vol. 29, no. 1, pp. 75–87, Apr. 14, 2014. DOI: 10.1016/j.devcel.2014.02.002 (cit. on p. 16).

[42] H. Cantwell and P. Nurse, "Unravelling nuclear size control," *Current Genetics*, vol. 65, no. 6, pp. 1281–1285, Dec. 1, 2019. DOI: 10.1007/s00294-019-00999-3 (cit. on p. 18).

[43] F. Federici, L. Dupuy, L. Laplaze, M. Heisler, and J. Haseloff, "Integrated genetic and computation methods for in planta cytometry," *Nature Methods*, vol. 9, no. 5, pp. 483–485, May 2012. DOI: 10.1038/nmeth.1940 (cit. on p. 18).

[44] D. Eschweiler, T. V. Spina, R. C. Choudhury, E. Meyerowitz, A. Cunha, and J. Stegmaier, "CNN-based preprocessing to optimize watershed-based cell segmentation in 3d confocal microscopy images," in *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, Apr. 2019, pp. 223–227. DOI: 10.1109/ISBI.2019.8759242 (cit. on p. 18).

[45] R. Fernandez, P. Das, V. Mirabet, *et al.*, "Imaging plant growth in 4d: Robust tissue reconstruction and lineaging at cell resolution," *Nature Methods*, vol. 7, no. 7, pp. 547–553, Jul. 2010. DOI: 10.1038/nmeth.1472 (cit. on p. 18).

[46] C. Sommer, C. Straehle, U. Köthe, and F. A. Hamprecht, "Ilastik: Interactive learning and segmentation toolkit," in *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, Mar. 2011, pp. 230–233. DOI: 10.1109/ISBI.2011.5872394 (cit. on p. 18).

[47] J. Stegmaier, F. Amat, W. C. Lemon, *et al.*, "Real-time three-dimensional cell segmentation in large-scale microscopy data of developing embryos," *Developmental Cell*, vol. 36, no. 2, pp. 225–240, Jan. 25, 2016. DOI: 10.1016/j.devcel.2015.12.028 (cit. on p. 18).

[48] S. Strauss, A. Runions, B. Lane, *et al.*, "Using positional information to provide context for biological image analysis with MorphoGraphX 2.0," *eLife*, vol. 11, D. C. Bergmann, D. Weigel, R. M. Merks, and N. Nakayama, Eds., e72601, May 5, 2022. DOI: 10.7554/eLife.72601 (cit. on p. 18).

[49] D. Kurihara, Y. Mizuta, Y. Sato, and T. Higashiyama, "ClearSee: A rapid optical clearing reagent for whole-plant fluorescence imaging," *Development*, vol. 142, no. 23, pp. 4168–4179, Dec. 1, 2015. DOI: 10.1242/dev.127613 (cit. on p. 18).

[50] T. J. Musielak, L. Schenkel, M. Kolb, A. Henschen, and M. Bayer, "A simple and versatile cell wall staining protocol to study plant reproduction," *Plant Reproduction*, vol. 28, no. 3, pp. 161–169, Dec. 1, 2015. DOI: 10.1007/s00497-015-0267-1 (cit. on p. 18).

[51] R. Tofanelli, A. Vijayan, S. Scholz, and K. Schneitz, "Protocol for rapid clearing and staining of fixed arabidopsis ovules for improved imaging by confocal laser scanning microscopy," *Plant Methods*, vol. 15, no. 1, p. 120, Oct. 25, 2019. DOI: 10.1186/s13007-019-0505-x (cit. on p. 18).

[52] R. Ursache, T. G. Andersen, P. Marhavý, and N. Geldner, "A protocol for combining fluorescent proteins with histological stains for diverse cell wall components," *The Plant Journal*, vol. 93, no. 2, pp. 399–412, 2018. DOI: 10.1111/tpj.13784 (cit. on p. 18).

[53] K. Harris, D. Crabb, I. M. Young, *et al.*, "*In situ* visualisation of fungi in soil thin sections: Problems with crystallisation of the fluorochrome FB 28 (calcofluor m2r) and improved staining by SCRI renaissance 2200," *Mycological Research*, vol. 106, no. 3, pp. 293–297, Mar. 1, 2002. DOI: 10.1017/S0953756202005749 (cit. on p. 18).

[54] K. Bink, A. Walch, A. Feuchtinger, *et al.*, "TO-PRO-3 is an optimal fluorescent dye for nuclear counterstaining in dual-colour FISH on paraffin sections," *Histochemistry and Cell Biology*, vol. 115, no. 4, pp. 293–299, Apr. 1, 2001. DOI: 10.1007/s004180100254 (cit. on p. 18).

[55] C. A. E. M. Van Hooijdonk, C. P. Glade, and P. E. J. Van Erp, "TO-PRO-3 iodide: A novel HeNe laser-excitable DNA stain as an alternative for propidium iodide in multiparameter flow cytometry," *Cytometry*, vol. 17, no. 2, pp. 185–189, 1994. DOI: 10.1002/cyto.990170212 (cit. on p. 18).

[57] J. Moore, D. Basurto-Lozada, S. Besson, *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, Sep. 1, 2023. DOI: 10.1007/s00418-023-02209-1 (cit. on p. 21).

[58] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, *et al.*, "The FAIR guiding principles for scientific data management and stewardship," *Scientific Data*, vol. 3, no. 1, p. 160 018, Mar. 15, 2016. DOI: 10.1038/sdata.2016.18 (cit. on p. 21).

[59] J. C. Caicedo, A. Goodman, K. W. Karhohs, *et al.*, "Nucleus segmentation across imaging experiments: The 2018 data science bowl," *Nature Methods*, vol. 16, no. 12, pp. 1247–1253, Dec. 2019. DOI: 10.1038/s41592-019-0612-7 (cit. on p. 27).

[60] D. Hirling, E. Tasnadi, J. Caicedo, *et al.*, "Segmentation metric misinterpretations in bioimage analysis | nature methods," *Nature Methods*, Jul. 27, 2023. DOI: 10.1038/s41592-023-01942-8 (cit. on p. 27).

[61] A. Bailoni, C. Pape, N. Hütsch, *et al.*, "GASP, a generalized framework for agglomerative clustering of signed graphs and its application to instance segmentation," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022, pp. 11 635–11 645. DOI: 10.1109/CVPR52688.2022.01135 (cit. on pp. 33, 44, 94).

[62] H. M. Meyer, J. Teles, P. Formosa-Jordan, *et al.*, "Fluctuations of the transcription factor ATML1 generate the pattern of giant cells in the arabidopsis sepal," *eLife*, vol. 6, D. C. Bergmann, Ed., e19131, Feb. 1, 2017. DOI: 10.7554/eLife.19131 (cit. on p. 35).

[63] M. I. Rast-Somssich, S. Broholm, H. Jenkins, *et al.*, "Alternate wiring of a KNOXI genetic network underlies differences in leaf development of a. thaliana and c. hirsuta," *Genes & Development*, vol. 29, no. 22, pp. 2391–2404, Nov. 15, 2015. DOI: 10.1101/gad.269050.115 (cit. on p. 35).

[64] M. Cerise, V. da Silveira Falavigna, G. Rodríguez-Maroto, *et al.*, "Two modes of gene regulation by TFL1 mediate its dual function in flowering time and shoot determinacy of arabidopsis," *Development*, vol. 150, no. 23, dev202089, Dec. 7, 2023. DOI: 10.1242/dev.202089 (cit. on p. 35).

[65] D. Martignago, V. d. S. Falavigna, A. Lombardi, *et al.*, "The bZIP transcription factor AREB3 mediates FT signalling and floral transition at the arabidopsis shoot apical meristem," *PLOS Genetics*, vol. 19, no. 5, e1010766, May 15, 2023. DOI: 10.1371/journal.pgen.1010766 (cit. on p. 35).

[66] H. Nunley, B. Shao, D. Denberg, *et al.*, *Nuclear instance segmentation and tracking for preimplantation mouse embryos*, Feb. 23, 2024. DOI: 10.1101/2023.03.14.532646 (cit. on p. 35).

[67] M. Maška, V. Ulman, P. Delgado-Rodriguez, *et al.*, "The cell tracking challenge: 10 years of objective benchmarking," *Nature Methods*, vol. 20, no. 7, pp. 1010–1020, Jul. 2023. DOI: 10.1038/s41592-023-01879-y (cit. on p. 38).

[68] A. Coghlan and D. Stufft. "PEP 440 – version identification and dependency specification," Python Enhancement Proposals (PEPs). (Mar. 18, 2013), https://peps.python.org/pep-0440/ (cit. on p. 41).

[69] J. B. T. M. Roerdink and A. Meijster, *The watershed transform: Definitions, algorithms and parallelization strategies*, 2001 (cit. on p. 44).

[70] S. Wolf, C. Pape, A. Bailoni, *et al.*, "The mutex watershed: Efficient, parameter-free image partitioning," presented at the Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 546–562 (cit. on p. 44).

[71] 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*, Y. Boykov, F. Kahl, V. Lempitsky, and F. R. Schmidt, Eds., Berlin, Heidelberg: Springer, 2011, pp. 31–44. DOI: 10.1007/978-3-642-23094-3_3 (cit. on p. 44).

[72] W. Ouyang, F. Beuttenmueller, E. Gómez-de-Mariscal, *et al.*, *BioImage model zoo: A community-driven resource for accessible deep learning in BioImage analysis*, Jun. 8, 2022. DOI: 10.1101/2022.06.07.495102 (cit. on p. 45).

[74] M. Majurski and P. Bajcsy, "Exact tile-based segmentation inference for images larger than GPU memory," *Journal of Research of the National Institute of Standards and Technology*, vol. 126, p. 126 009, Jun. 3, 2021. DOI: 10.6028/jres.126.009 (cit. on p. 51).

[75] A. Araujo, W. Norris, and J. Sim, "Computing receptive fields of convolutional neural networks," *Distill*, vol. 4, no. 11, e21, Nov. 4, 2019. DOI: 10.23915/distill.00021 (cit. on p. 51).

[80] GitHub. "About wikis," GitHub Docs. (Mar. 29, 2021), https://docs.github.com/en/communities/documenting-your-project-with-wikis/about-wikis (cit. on p. 58).

[84] A. Kirillov, E. Mintun, N. Ravi, *et al.*, "Segment anything," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, Paris, France: IEEE, Oct. 1, 2023, pp. 3992–4003. DOI: 10.1109/ICCV51070.2023.00371 (cit. on pp. 64, 70, 74, 77, 101).

[85] B. De Brabandere, D. Neven, and L. Van Gool, "Semantic instance segmentation with a discriminative loss function," in *CVPR 2017 workshop*, Aug. 8, 2017. arXiv: 1708.02551 (cit. on pp. 64, 65).

[86] T. Darcet, M. Oquab, J. Mairal, and P. Bojanowski, *Vision transformers need registers*, Apr. 12, 2024. DOI: 10.48550/arXiv.2309.16588. arXiv: 2309.16588[cs] (cit. on p. 70).

[87] L. Ke, M. Ye, M. Danelljan, *et al.*, *Segment anything in high quality*, Oct. 23, 2023. DOI: 10.48550/arXiv.2306.01567. arXiv: 2306.01567[cs] (cit. on pp. 77, 79, 102, 113).

[88] M. K. Driscoll, E. S. Welf, A. R. Jamieson, *et al.*, "Robust and automated detection of subcellular morphological motifs in 3d microscopy images," *Nature Methods*, vol. 16, no. 10, pp. 1037–1044, Oct. 2019. DOI: 10.1038/s41592-019-0539-z (cit. on p. 92).

[89] H. Mazloom-Farsibaf, Q. Zou, R. Hsieh, G. Danuser, and M. K. Driscoll, "Cellular harmonics for the morphology-invariant analysis of molecular organization at the cell surface," *Nature Computational Science*, vol. 3, no. 9, pp. 777–788, Sep. 2023. DOI: 10.1038/s43588-023-00512-4 (cit. on p. 92).

[90] F. Y. Zhou, A. Weems, G. M. Gihana, *et al.*, *Surface-guided computing to analyze subcellular morphology and membrane-associated signals in 3d*, Apr. 20, 2023. DOI: 10.1101/2023.04.12.536640 (cit. on p. 92).

[91] J. Funke, B. Andres, F. A. Hamprecht, A. Cardona, and M. Cook, "Efficient automatic 3d-reconstruction of branching neurons from EM data," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 1004–1011. DOI: 10.1109/CVPR.2012.6247777 (cit. on p. 93).

[92] J. Bragantini, I. Theodoro, X. Zhao, *et al.*, *Ultrack: Pushing the limits of cell tracking across biological scales*, Sep. 3, 2024. DOI: 10.1101/2024.09.02.610652 (cit. on pp. 93, 98).

[93] N. Ravi, V. Gabeur, Y.-T. Hu, *et al.*, *SAM 2: Segment anything in images and videos*, Oct. 28, 2024. DOI: 10.48550/arXiv.2408.00714. arXiv: 2408.00714[cs] (cit. on pp. 101, 102).

[94] M. Caron, H. Touvron, I. Misra, *et al.*, *Emerging properties in self-supervised vision transformers*, May 24, 2021. arXiv: 2104.14294[cs] (cit. on p. 113).

[95] M. Oquab, T. Darcet, T. Moutakanni, *et al.*, *DINOv2: Learning robust visual features without supervision*, Feb. 2, 2024. DOI: 10.48550/arXiv.2304.07193. arXiv: 2304.07193[cs] (cit. on p. 113).

[96] DeepSeek-AI, A. Liu, B. Feng, *et al.*, *DeepSeek-v3 technical report*, Feb. 18, 2025. DOI: 10.48550/arXiv.2412.19437. arXiv: 2412.19437[cs] (cit. on p. 115).

[97] L. A. Royer, "Omega — harnessing the power of large language models for bioimage analysis," *Nature Methods*, vol. 21, no. 8, pp. 1371–1373, Aug. 2024. DOI: 10.1038/s41592-024-02310-w (cit. on p. 115).

[98] W. Lei, C. Fuster-Barceló, G. Reder, A. Muñoz-Barrutia, and W. Ouyang, "BioImage.IO chatbot: A community-driven AI assistant for integrative computational bioimaging," *Nature Methods*, vol. 21, no. 8, pp. 1368–1370, Aug. 2024. DOI: 10.1038/s41592-024-02370-y (cit. on p. 115).

[99] S. Zhang, G. Dai, T. Huang, and J. Chen, "Multimodal large language models for bioimage analysis," *Nature Methods*, vol. 21, no. 8, pp. 1390–1393, Aug. 2024. DOI: 10.1038/s41592-024-02334-2 (cit. on p. 115).

[100] R. Umeike, N. Getty, F. Xia, and R. Stevens, *Scaling large vision-language models for enhanced multimodal comprehension in biomedical image analysis*, Jan. 26, 2025. DOI: 10.48550/arXiv.2501.15370. arXiv: 2501.15370[cs] (cit. on p. 115).

[101] J. Burgess, J. J. Nirschl, L. Bravo-Sánchez, *et al.*, *MicroVQA: A multimodal reasoning benchmark for microscopy-based scientific research*, Mar. 17, 2025. DOI: 10.48550/arXiv.2503.13399. arXiv: 2503.13399[cs] (cit. on p. 116).

[102] R. Haase, C. Tischer, and N. Scherf, *Benchmarking large language models for bio-image analysis code generation*, Apr. 21, 2024. DOI: 10.1101/2024.04.19.590278 (cit. on p. 116).

[104] G. Ravelomanana, C. Kervrann, and T. Pécot, "Joined segmentation of nuclei and cells," presented at the I2K 2024 - Conference From Images to Knowledge, Milan, Italy, Oct. 2024 (cit. on p. 118).

[105] C. Pape, R. Remme, A. Wolny, *et al.*, "Microscopy-based assay for semi-quantitative detection of SARS-CoV-2 specific antibodies in human sera," *BioEssays*, vol. 43, no. 3, p. 2 000 257, 2021. DOI: 10.1002/bies.202000257 (cit. on pp. 119, 120).

[106] S. Gupta, X. Hu, J. Kaan, *et al.*, "Learning topological interactions for multi-class medical image segmentation," in *Computer Vision – ECCV 2022*, vol. 13689, Cham: Springer Nature Switzerland, 2022, pp. 701–718. DOI: 10.1007/978-3-031-19818-2_40 (cit. on p. 120).

Software

[27] Q. Yu, *GoNuclear: Deep learning toolkit for 3d nuclear instance segmentation in microscopy images*, Feb. 2024. https://kreshuklab.github.io/go-nuclear/ (cit. on pp. 15, 21, 27).

[28] Q. Yu, L. Cerrone, and A. Wolny, *PlantSeg 2.0: Powerful and user-friendly tissue segmentation*, Oct. 2024. https://kreshuklab.github.io/plant-seg/ (cit. on p. 15).

[56] L. Biewald, *Experiment tracking with weights and biases*, Software available from wandb.com, 2020. https://www.wandb.com/ (cit. on p. 21).

[73] F. Beuttenmüller, *bioimageio.core: Python libraries for loading, running and packaging bioimage.io models*, 2019. https://github.com/bioimage-io/core-bioimage-io-python (cit. on p. 45).

[76] T. Yep, *Torchinfo: View model summaries in PyTorch!* Mar. 16, 2020. https://github.com/TylerYep/torchinfo (cit. on p. 52).

[77] K. Shi, *pytorch_memlab: Profiling and inspecting memory in pytorch*, May 24, 2019. https://github.com/Stonesjtu/pytorch_memlab (cit. on p. 52).

[78] J. Kimmel, *pytorch_modelsize: Estimates the size of a PyTorch model in memory*, Nov. 18, 2017. https://github.com/jacobkimmel/pytorch_modelsize (cit. on p. 52).

[79] L. Cerrone, *PlantSeg Tools: Simple python tools for plantseg*, Oct. 23, 2020. https://github.com/hci-unihd/plant-seg-tools (cit. on p. 54).

[81] Executable Books Community, *Jupyter book: Build beautiful, publication-quality books and documents from computational content*, Feb. 12, 2020. DOI: 10.5281/ZENODO.2561065. https://jupyterbook.org/ (cit. on p. 58).

[82] T. Christie, *Mkdocs: Project documentation with markdown*, Jan. 11, 2014. https://www.mkdocs.org/ (cit. on p. 58).

[83] M. Donath, *Material for MkDocs: Documentation that simply works*, Jan. 28, 2016. https://squidfunk.github.io/mkdocs-material/ (cit. on p. 58).

[103] A. Wolny, *pytorch-3dunet: 3d u-net model for volumetric semantic segmentation written in pytorch*, 2018. https://github.com/wolny/pytorch-3dunet (cit. on pp. 118, 119).

Spielschiff WIKI | Feb 2023 | Köln, Deutschland.

My Work

[1] A. Wolny, Q. Yu, C. Pape, and A. Kreshuk, "Sparse object-level supervision for instance segmentation with pixel embeddings," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA: IEEE, Jun. 2022, pp. 4392–4401. DOI: 10.1109/CVPR52688.2022.00436 (cit. on pp. 1, 7–9, 63, 66–68, 70, 76).

[2] A. Vijayan, T. A. Mody, Q. Yu, *et al.*, "A deep learning-based toolkit for 3d nuclei segmentation and quantitative analysis in cellular and tissue context," *Development*, vol. 151, no. 14, dev202800, Jul. 18, 2024. DOI: 10.1242/dev.202800 (cit. on pp. 1, 3, 7, 15, 21, 23, 36, 37, 113, 119).

[27] Q. Yu, *GoNuclear: Deep learning toolkit for 3d nuclear instance segmentation in microscopy images*, Feb. 2024. https://kreshuklab.github.io/go-nuclear/ (cit. on pp. 15, 21, 27).

[28] Q. Yu, L. Cerrone, and A. Wolny, *PlantSeg 2.0: Powerful and user-friendly tissue segmentation*, Oct. 2024. https://kreshuklab.github.io/plant-seg/ (cit. on p. 15).