Technische Universität Dresden

Computer Vision Approaches for Mapping Gene Expression onto Lineage Trees

Dissertation

zur Erlangung des akademischen Grades
Doktor rerum naturalium (Dr. rer. nat.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von
Manan Lalit

Gutachter:
Dr. Dagmar Kainmueller, Max Delbrück Center for Molecular Medicine, Berlin
Dr. Ivo F. Sbalzarini, Max Planck Institute of Molecular Cell Biology and Genetics, Dresden

Dresden, den 05. April 2022

Summary

This project concerns studying the early development of living organisms. This period is accompanied by dynamic morphogenetic events. There is an increase in the number of cells, changes in the shape of cells and specification of cell fate during this time. Typically, in order to capture the dynamic morphological changes, one can employ a form of microscopy imaging such as Selective Plane Illumination Microscopy (SPIM) which offers a single-cell resolution across time, and hence allows observing the positions, velocities and trajectories of most cells in a developing embryo. Unfortunately, the dynamic genetic activity which underlies these morphological changes and influences cellular fate decision, is captured *only as static snapshots* and often requires processing (sequencing or imaging) multiple distinct individuals. In order to set the stage for characterizing the factors which influence cellular fate, one must bring the data arising from the above-mentioned static snapshots of multiple individuals and the data arising from SPIM imaging of other distinct individual(s) which characterizes the changes in morphology, into the same frame of reference.

In this project, a computational pipeline is established, which achieves the aforementioned goal of mapping data from these various imaging modalities and specimens to a canonical frame of reference. This pipeline relies on the three core building blocks of *Instance Segmentation*, *Tracking* and *Registration*. In the Chapter 2, I introduce EmbedSeg which is my solution to performing instance segmentation of 2D and 3D (volume) image data. In the Chapter 3, I introduce LineageTracer which is my solution to performing tracking of a time-lapse (2d+t, 3d+t) recording. In the Chapter 4, I introduce PlatyMatch which is my solution to performing registration of volumes. Errors from the application of these building blocks accrue which produces a *noisy* observation estimate of gene expression for the digitized cells in the canonical frame of reference. These noisy estimates are processed to infer the underlying hidden state by using a Hidden Markov Model (HMM) formulation, and details surrounding this are provided in the Section 4.5. Lastly, for wider dissemination of these methods, one requires an effective visualization strategy. A few details about the employed approach are discussed in the Section 3.4.

The pipeline was designed keeping imaging volume data in mind, but can easily be extended to incorporate other data modalities, if available, such as Single cell RNA Sequencing (scRNA-Seq) (more details are provided in the Section 5.3). The methods elucidated in this dissertation would provide a fertile playground for several experiments and analyses in the future. Some of such potential experiments are detailed in the Section 5.4. Current weaknesses of the computational pipeline are discussed additionally in the Section 5.4.

Acknowledgements

It takes a village to raise a PhD. This thesis work reached its current state thanks to contributions from several people.

Firstly I would like to thank Tim-Oliver Buchholz. I found that Tim-Oliver's suggestions during group meetings seemed to be very on-point. I especially thank him for developing the *bdv ui* panel, for introducing me to the website `slides.com` and for presenting his notes on *github* tagging and releases to the group. General chats with Tim-Oliver, outside of work, were also very illuminating, and his approach to time-management and work-life balance often left me reflective and wishing to incorporate some of his thinking.

I learnt a lot through my interactions with Alexander Krull. Alex would patiently distill a concept into its bare minimum. There was never a feeling of being hurried in a meeting with him and I often found that post the discussion, I had several brain sparks, which I don't think I would have observed, just by reading literature. I especially thank Alex for his supervision on the *PPN2V* project and for guiding me into deep learning research.

I am indebted to Mette Handberg-Thorsager for sharing her *Platynereis* imaging data with me, around which this thesis work is structured. I would receive immediate replies from her, even if I would contact her on weekends. I especially thank her for explaining to me the missing gaps in *Platynereis* research, her diligent notes on the imaging data collected by her and for preparing and sharing with me, beautiful visualizations of the embryos.

I would like to thank Johannes Girstmair. We initially started collaborating on the registration of multi-channel images of several, individual *Macrostomum* and *Echinoplana* embryos. During some of these day-long sessions, I got a chance to try out new techniques which I had been aware of but until then had never tested, and Johannes' enthusiasm and drive helped us obtain initial results fairly quickly.

I would like to thank Matthias Arzt for several, excellent scientific conversations which often brought me out of an impassé and provided a direction on what to try out next. Often when I would have a new idea, my first impulse would be to run to Matthias' desk and vet the idea by him. I found it very inspiring to hear Matthias talk about his philosophy towards coding, how to present work to a group of people and his thoughts on raising children.

It was a privilege to work with Mangal Prakash. Discussing scientific ideas with Mangal and hearing his opinions on a variety of day-to-day issues, was a lot of

fun for me. I thank him for his collaboration on *PPN2V* and *GauSePP*.

Meeting Max Petzold over lunch at the *Mensa* used to be a highlight for me. It was very exciting to see Max's evolution as a bio-image analyst and scientist. I thank him for some very fun conversations and wish him luck for his future. I would also like to thank Aryaman Gupta, Archit Bhatnagar, Abhijeet Krishna, Alishan Sahu and Suhrid Ghosh for chats around a myriad of topics.

I thank Giulia Serafini, Anaïs Bailles and Vinca Yadav for their very kind reminders about the Tomancak Lab hangouts, outside of work. I also thank Anaïs for introducing me to the *Improv* Group and for offering very supportive counsel regarding job applications.

Nuno Pimpão Martins and Christopher Schmied provided very insightful feedback during practice sessions for upcoming talks. Joran Deschamps conducted the Jug Lab group meetings with incredible panache and I thank him especially for the one joint session, where we discovered how multi-threading should be used while training a model in a *napari* plugin. Past and present lab members - Ashesh, Heino Andreas, Charlène Brillard, Tom Burke, Marina Cuenca, Anna Goncharova, Yu-Wen Hsieh, Akanksha Jain, Sheida Rahnamai Kordasiabi, Anil Kumar, Pavel Mejstrik, Damian Dalle Nogare, Isabella Osei, Tobias Pietzsch, Anirban Ray, Deborah Schmidt, Gabriella Turek, Vladimir Ulman, Bruno Cossermelli Vellutini, Tomáš Vičar and Igor Zubarev - were amazing colleagues and I thank them for their excellent camaraderie.

It was fun to chat with Katia over the years as I would buy coffee from her. Brian von Rueden from the PhD office was extremely helpful while trying to figure my way around all degree-related requirements at TU Dresden.

I ended up following a research path, thanks to meeting some very inspirational mentors - Per-Olof Berglund, Dr. Lars Davidson, Dr. Shaun Forth, Dr. Mikael Karlsson and Dr. Debarka Sengupta. Words stated by them, at different stages, have continued to stay with me and have dictated life decisions, for which I thank them immensely.

That I got an opportunity to pursue fundamental research at one of the most well known institutes, is thanks to Pavel and Florian for accepting me in their groups and I whole-heartedly thank them for their mentorship during these last four years and for entrusting me with this interesting project.

I found it always an incredible experience to hear Pavel talk about science. While co-writing a paper alongside Pavel, I learnt several valuable lessons related to expanding text while explaining a concept and to caption a figure well. Questions raised by him in the group meetings enabled me to get an appreciation of developmental biology concerns. Additionally the Tomancak Lab retreats, especially the one to Szeged and Brno, were very enjoyable and I would like to thank him for making them possible.

I found it very inspiring that despite his busy schedule, Florian is able to meet all his group members individually, on a weekly basis. Observing how Florian leads his group, how he structures and presents his talks and how he prepares a write-up for a manuscript, was very educative. A lot of my intuition surrounding deep learning research emerged from discussions initiated by him during the weekly journal club meetings. It was also quite evident how everyone lightens up when Florian was around, and I thank him for being such a pleasant mentor and for his endless stream of puns.

Lastly, I would like to thank my parents, Renu Gupta and Lalit Kumar, and my sister, Ragini Lalit, for providing their steadfast support behind all my decisions and for encouraging a love for reading.

Contents

Summary **iii**

Acknowledgements **v**

1 Introduction **1**
1.1 Contributions 8

2 Instance Segmentation **11**
2.1 Introduction 12
2.2 Related Work 13
2.2.1 Shape-agnostic Approaches 14
2.2.2 Shape-augmented Approaches 15
2.2.3 Embedding-based Approaches 15
2.3 Proposed Method 17
2.3.1 Network Architecture and Inference Scheme 19
2.3.2 Using the Medoid as Embedding Location 20
2.3.3 Tiled Predictions via Label Propagation 20
2.4 EmbedSeg (Sliced) 21
2.5 EmbedSeg (ILP) 21
2.6 Alternate Inference Strategies 24
2.7 Baselines, Experiments, Results 25
2.7.1 Used 2D Data 26
2.7.2 Used and Contributed 3D Data 26
2.7.3 Baseline Methods 26
2.7.4 Data Handling in 2D 27
2.7.5 Data Handling in 3D 28
2.7.6 Training Details for EmbedSeg & Neven *et al* 29
2.7.7 Performance Evaluations 29
2.7.8 Ablation Studies 32
2.8 Open Code and Online Resources 32
2.8.1 EmbedSeg napari 32
2.8.2 EmbedSeg Demo 34
2.9 Discussion 35

3 Tracking **45**
3.1 Introduction 46
3.2 Related Work 47
3.3 Our Approach 49

3.3.1 PointTrack . . . 49
3.3.2 LineageTracer Training . . . 50
3.3.3 LineageTracer Inference . . . 51
3.4 Visualization . . . 55
3.5 Discussion . . . 56

4 Registration **63**
4.1 Introduction . . . 64
4.2 Related Work . . . 65
4.2.1 *Platynereis dumerilii* . . . 66
4.2.2 Non-Rigid Point Cloud Registration . . . 66
4.2.3 DL-based Medical Imaging . . . 67
4.2.4 DL-based Graph Matching . . . 67
4.3 Our Approach . . . 68
4.3.1 PlatyMatch . . . 69
4.4 Learning Node Embeddings . . . 73
4.5 Hidden Markov Models . . . 75
4.6 Discussion . . . 76

5 Discussion **83**
5.1 Introduction . . . 84
5.2 Pooling Information from Multiple Live Embryos . . . 85
5.3 Integrating scRNA-Seq Data . . . 86
5.4 Potential Experiments & Weaknesses . . . 87

List of Figures

1.1 **Mapping gene expression between developmental stages.** Maximum projection of confocal images of *in situ* hybridized *Platynereis dumerilii* specimens at developmental stages of 16 hours post fertilization (hpf) (left) and 20 hours post fertilization (hpf) (right), with *Pax3/7* in magenta and *DAPI* in cyan. A central question in developmental biology is to map spatial gene expression patterns between different developmental stages. When successful, one would be able to answer the question - whether the cells expressing *Pax3/7* on the right are the progeny of the cells expressing *Pax3/7* on the left, or instead are independent transcription events. Image credits: Mette Handberg-Thorsager. 1

1.2 **Single cell RNA Sequencing (scRNA-Seq) loses spatial and dynamic context.** (A) To apply scRNA-Seq, one has to disassociate a piece of tissue, which provides a single cell slurry (B) scRNA-Seq is comprehensive and allows one to identify different cell types and states (C) scRNA-Seq loses spatial context *i.e.* one loses a sense of the spatial neighborhood of cells and is not able to monitor cell-to-cell interactions. Also, scRNA-Seq loses dynamic information *i.e.* one loses a sense of the temporal behaviour of cells as they migrate and divide. Adapted from `https://www.youtube.com/watch?v=UwoSLWlyC74`. 2

1.3 **Schematic of *in situ* hybridization (ISH).** (A) mRNA molecules produced by a specific gene, are visualized as gray strands. (B & C) Complimentary probes (shown as a black strand) are designed for these target mRNA molecules and are hybridized to the target molecules. These probes are labeled with a fluorophore (shown as a red star) which enables visualizing each individual mRNA molecule as a dot, when inspected under the confocal microscope. 4

1.4 **Max z projections of ISH specimens at the developmental stage of** 24 **hpf.** These specimens were aligned and registered to a reference atlas using *ProSPr* (Vergara et al., 2017). Image credits: Mette Handberg-Thorsager. 5

1.5 **Schematic and building blocks of the computational pipeline.** By leveraging the computer vision strategies of *Instance Segmentation*, *Tracking* and *Registration* (right), one can transfer information about cellular gene expression from the ISH specimens onto the lineage tree (left).
Here *green* indicates cells which are expressing a given gene, while *white* indicates cells which are not expressing that gene. By leveraging the available lineage tree, one can even conjecture about the state of gene expression in between developmental stages (for which no ISH data is available) - here, *yellow* indicates cells expressing a given gene. This computational pipeline enables setting up a canonical space for monitoring dynamic changes in cell morphology and gene expression, during the early development of *Platynereis dumerilii*. 6

2.1 **Visualization of the inference procedure of EmbedSeg.** (top-left) An exemplary raw input image. During inference, we iteratively pick seed pixels greedily from the predicted seediness map (top-right) and cluster other foreground pixels w.r.t. their predicted embeddings as explained in Section 2.3. The image on the (bottom-left) shows the thresholded seediness map as the white region, 5 randomly sampled foreground pixels (marked with colored pluses), embedding location of these pixels (dots, mostly tightly clustered within ellipses), and the predicted clustering bandwidth thresholded at a likelihood of 0.5 (semi-transparent ellipses). The (bottom-right) image shows the final predicted instance segmentation result. 19

2.2 **Qualitative results in 2D. EmbedSeg and two baselines compared on representative images of the *BBBC010*, *Usiigaci*, *DSB* and *MoNuSeg* datasets.** Columns show: full input image, zoomed insets, ground truth labels (GT), and instance segmentation results by the 3-class U-Net baseline, the best performing competing baseline, and EmbedSeg. Each segmented instance is shown in a unique random color. The strong baseline methods shown are Neven et al. (2019), Cellpose, Neven et al. (2019) and StarDist for the *BBBC010*, *Usiigaci*, *DSB* and *MoNuSeg* datasets respectively 22

2.3 **Qualitative results in 3D. Qualitative results of EmbedSeg on the Mouse-Organoid-Cells-CBG, *C. elegans*-Cells-HK, *Arabidopsis*-Cells-CAM and *Platynereis*-Nuclei-CBG datasets.** Columns show orthogonal XY, YZ and XZ slices of one representative input image, ground truth labels (GT), and our instance segmentation results using EmbedSeg, respectively. Note that each segmented instance is shown in a random but unique color. 23

2.4 **The EmbedSeg training and inference workflow as available via open Jupyter notebooks and napari plugin.** (a) Schematic showing that training and inference are decoupled by the possibility to store trained EmbedSeg models for later use. (b, d) Screenshots of the EmbedSeg training and prediction notebook, respectively. Both being openly available on GitHub. (c, e) Screenshot of the EmbedSeg napari plugin during training and prediction, respectively. The four quadrants in (c) show the raw image tail currently used for training (top left), the corresponding ground truth labels (top right), the current instance predictions (bottom right). In the bottom left quadrant we additionally show, a few randomly-selected pixels, their embedding locations, and the clustering bandwidth (visualized as pluses, dots, and an ellipse, respectively). See Figure 2.1 for a larger view. 33

2.5 **EmbedSeg web-based demo for 2D and 3D inference.** The web-based tool allows selecting a demo image (top) or dragging and dropping in a 2D or a 3D image (bottom), choosing a previously trained model, inspecting the instance segmentation prediction, dynamically updating the model prediction by changing parameters such as seediness threshold (left panel) and finally downloading the instance segmentation prediction. Available at `https://share.streamlit.io/mlbyml/embedseg_demo/main/main.py`. 35

3.1 **Schematic explaining the LineageTracer flow of information, during training and inference.** During training, a minibatch is constructed from t tracklets by considering o instance segmentations per tracklet and p pixels per instance segmentation. In this schematic figure, the sampled pixels are shown as white dots, $t = 4$ and $o = 3$. Low-level features such as relative pixel position within the instance segmentation, normalized intensity and convolution response from a trained EmbedSeg model at that pixel location, in addition to the global position of the instance segmentation within the image (input to the LineageTracer network as the *positional encodings*) are mapped to an E-dimensional feature per instance segmentation, by using MLPs, batch-normalization, average pooling, max pooling and learnable point-weighting operations. 50

3.2 **Qualitative results in 2D on the *Fluo-N2DL-HeLa* dataset.** (*i*) A trained EmbedSeg model is used to predict instance segmentations on all time frames independently (second row). (*ii*) Next, each instance segmentation is mapped to a higher dimensional feature using a trained LineageTracer model, which allows extracting non-conflicting track segments (third row). (*iii*) Then an ILP is solved which allows connecting these track segments to form a lineage tree (fourth row). (*iv*) Since over-segmentations were not explicitly modeled in the previous ILP formulation, these are addressed in the disentangling step (final row) by identifying track segments which are below a certain duration and re-solving an ILP with a different set of constraints (see main text). The corresponding green tiles and red tiles in the second-last and last rows indicate that over-segmentations have been merged and highlight the benefit of disentangling the lineage tree. 54

3.3 **Displaying volumetric image data alongside instance segmentations.** Volumetric image data and corresponding instance segmentations for a given time frame are loaded simultaneously. By changing value of the parameters *Slice X*, *Slice Y* and *Slice Z* in the graphical user interface (top-right), one can selectively display specific YZ, ZX and XY planes in the volume. 56

3.4 **Visualization of instance segmentations of the nuclei in a *Platynereis dumerilii* embryo imaged with SPIM imaging from 5 hpf ($t = 0$) to 15 hpf ($t = 400$). Each of the initial 38 lineages at time frame $t = 0$ are assigned a unique, distinct color.** Instance segmentations are obtained by using a 3D EmbedSeg model, trained on manually-curated GT instance segmentation annotations. These predicted instance segmentations per time frame (each time frame is processed independently) are associated between time frames by using a trained LineageTracer model, followed by manual curation of the predicted lineage tree. Next, these instance segmentations are converted to triangular meshes by executing the *Marching Cubes* algorithm provided with PyVista python package (Sullivan and Kaszynski, 2019). Lastly, these meshes are imported for visualization in the browser, by using a javascript file which sets up the lighting and staging. Such a visualization enables identifying the ancestry of a cell at a given time frame. Additionally, by loading each of the cells (meshes) at a given time frame, with gene expression information, one can calculate the intra- and inter- lineage variance in transcriptional activity. 57

4.1 ***Intra* and *Inter*modal Registration.** (A, B) 2-D schematics illustrating the two use cases: (A) images of distinct, independent *in-situ* specimens, acquired through confocal microscopy are registered to each other, which enables formation of an average, virtual atlas. (B) images of *in-situ* specimens, acquired through confocal microscopy are registered to the appropriate frame (tp: time point) in a time-lapse movie acquired through SPIM imaging. Nuclei indicated in darker shades are the ones expressing the gene being investigated. In both cases, the information about gene expression is transferred from the source nucleus to the corresponding target nucleus. (Figure modified from Lalit et al. (2020)). 65

4.2 **Overview of the how the shape context feature is built for each nucleus detection.** (*A*) Nuclei instance segmentations are obtained by inputting a gray-scale volume to a trained EmbedSeg model. These instance segmentations are represented as a point cloud. (B, C, D, E) In order to ensure that the shape context geometric descriptor is rotationally covariant, we modify the original coordinate system to obtain a unique coordinate system for each nucleus detection. (*B*) First, we consider any nucleus detection, represented as a yellow ellipsoid. (*C*) Next, we identify the centroid of the complete point cloud (shown as a pink ball), and draw a vector from the centroid to the queried nucleus detection. This vector, shown as a yellow arrow, is the local Z axis for the queried nucleus detection. (*D*) Then we draw a vector pointing from the queried nucleus detection to the location of a landmark. For the *Platynereis dumerilii* data, we notice that detecting the location of one of the four internal macromere is possible using an automated, heuristic approach, and therefore we draw a vector pointing from the queried nucleus detection to this macromere. In the absence of such a landmark, one can use the vector that represents the direction of the largest variance of the point cloud detections *i.e.* the first PCA component corresponding to the x, y, z positions of the point cloud detections. This vector, shown as a red arrow, represents the local X axis. (*E*) The local Y-axis is evaluated as the cross product of the first two vectors, and is shown as a pink arrow. (*F*) Next, the neighbourhood around each nucleus detection is binned in order to compute the shape context signature for each detection. 70

4.3 **Using an HMM to decipher to underlying hidden gene expression states.** The results of the registration pipeline lead to normalized scores (between 0 and 1) per cell in the lineage tree (left). Using an HMM representation and the Viterbi Algorithm, one can infer the underlying binary, gene expression states per cell (right). 75

List of Tables

2.1 **Used 3D datasets.** We introduce four new volumetric microscopy datasets, covering various practically relevant imaging conditions and microscopy modalities. All datasets come with high quality ground truth labels for training. 25

2.2 **Quantitative evaluation on four 2D datasets.** For each dataset, we compare results of multiple baselines (rows) to results obtained with our proposed pipeline highlighted in gray. The columns show the Mean Average Precision (AP_{dsb}) for selected IoU thresholds. Best and second best performing methods per column are indicated in bold and underlined, respectively. 30

2.3 **Quantitative Evaluation on seven 3D datasets.** For each dataset, we compare results of multiple baselines (rows) to results obtained with our proposed pipeline (EmbedSeg) highlighted in gray. The columns show the Mean Average Precision (AP_{dsb}) for selected IoU thresholds. Best and second best performing methods per column are indicated in bold and underlined, respectively. . . 31

2.4 **Ablation studies.** For the BBBC010, Usiigaci and *Platynereis*-Nuclei-CBG datasets, we show how using the centroid instead of the medoid during training and/or removing test-time augmentation negatively impacts overall performance. Like before, columns show AP_{dsb} (first row) or ΔAP_{dsb} (rows 2-4) at selected IoU thresholds. Individual rows show: results obtained with EmbedSeg; using centroids instead of medoids during training; using medoids but without test-time augmentation; using centroids during training and no test-time augmentation. . 32

4.1 **Quantitative evaluation on a simulated and a real biological dataset.** The columns show the Matching Accuracy $(0 - 1)$ obtained by baseline methods. Best and second best performing methods per column are indicated in bold and underlined, respectively. 74

Acronyms

CNN Convolutional Neural Network.

DAPI 4'6-diamidino-2-phenylindole.

DL Deep Learning.

dpf days post fertilization.

FFD Free-Form Deformation.

GMM Gaussian Mixture Model.

GNN Graph Neural Network.

GT Ground truth.

HMM Hidden Markov Model.

hpf hours post fertilization.

ICP Iterative Closest Point.

ILP Integer Linear Program.

IoU Intersection over Union.

ISH *in situ* hybridization.

MAP maximum a-posteriori.

MLP Multi-Layer Perceptron.

MOTS multi object tracking and segmentation.

PCA Principal Component Analysis.

RANSAC Random Sample Consensus.

scRNA-Seq Single cell RNA Sequencing.

SOTA State-of-the-art.

SPIM Selective Plane Illumination Microscopy.

TI Trajectory Inference.

WMISH Whole Mount *in situ* hybridization.

Introduction

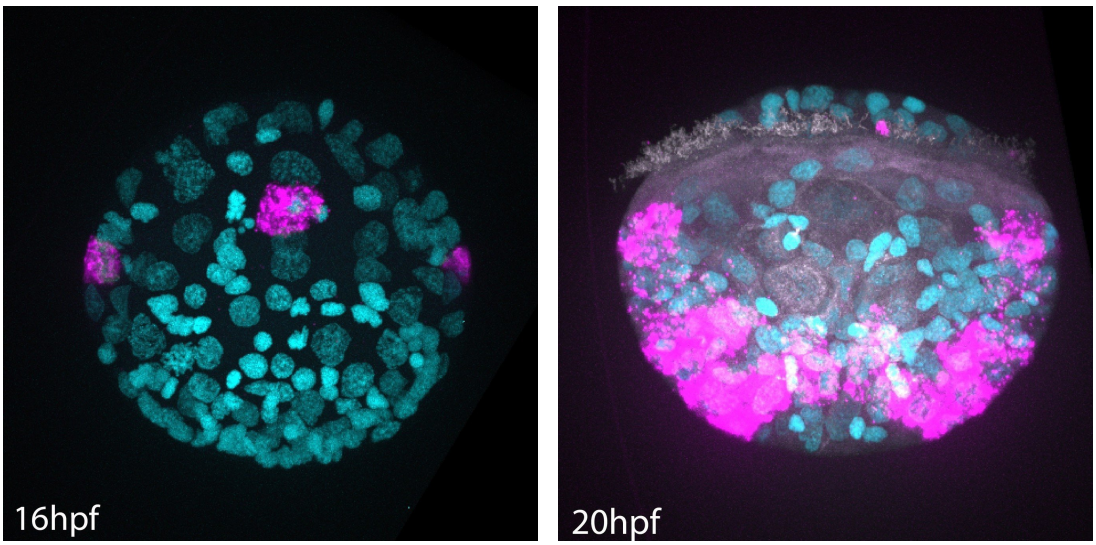

Figure 1.1: **Mapping gene expression between developmental stages.** Maximum projection of confocal images of *in situ* hybridized *Platynereis dumerilii* specimens at developmental stages of 16 hours post fertilization (hpf) (left) and 20 hours post fertilization (hpf) (right), with *Pax3/7* in magenta and *DAPI* in cyan. A central question in developmental biology is to map spatial gene expression patterns between different developmental stages. When successful, one would be able to answer the question - whether the cells expressing *Pax3/7* on the right are the progeny of the cells expressing *Pax3/7* on the left, or instead are independent transcription events. Image credits: Mette Handberg-Thorsager.

Two central questions which exist in developmental biology are (*i*) how gene expression unfolds over time and (*ii*) how dynamic changes in gene expression correlate with dynamic changes in cellular morphology, as an embryo grows from a state of a few cells to becoming a fully-differentiated organism.

With the recent advancements in Single cell RNA Sequencing (scRNA-Seq), it has now become possible to obtain transcriptional signatures of individual cells at discrete time points during the developmental life-cycle of an organism. Then by ordering the sequenced cells along a trajectory based on similarity in their expression patterns (a group of strategies referred to as Trajectory Inference (TI) or *pseudo-time* analysis), one can achieve a mapping between cells sequenced at different time points (Saelens et al., 2019), and thus in turn, address how gene expression unfolds over time, the first of the two questions.

However, a few difficulties exist with scRNA-Seq: (*i*) the low mRNA content-

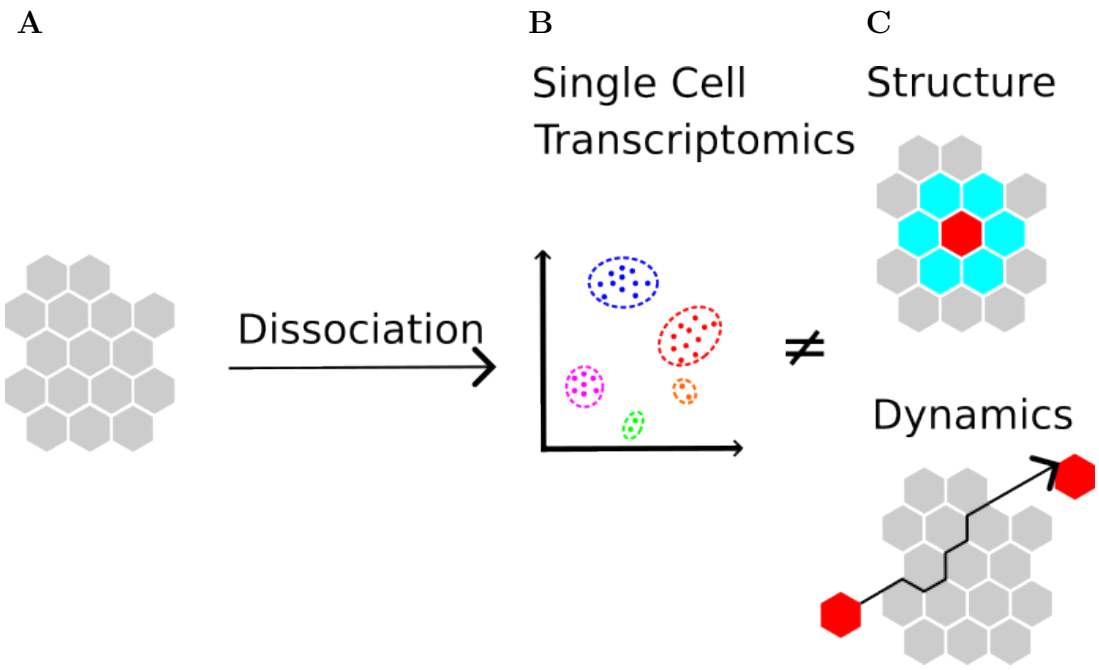

Figure 1.2: **Single cell RNA Sequencing (scRNA-Seq) loses spatial and dynamic context.** (A) To apply scRNA-Seq, one has to disassociate a piece of tissue, which provides a single cell slurry (B) scRNA-Seq is comprehensive and allows one to identify different cell types and states (C) scRNA-Seq loses spatial context *i.e.* one loses a sense of the spatial neighborhood of cells and is not able to monitor cell-to-cell interactions. Also, scRNA-Seq loses dynamic information *i.e.* one loses a sense of the temporal behaviour of cells as they migrate and divide. Adapted from `https://www.youtube.com/watch?v=UwoSLWlyC74`.

per cell results in sparse data, which makes it unclear if the gene was actually not expressed (*real* zero) or was dropped because of technical noise. This problem is alleviated by sequencing cells from several individuals synchronized at the developmental stage of interest, but has the unfortunate consequence of losing information about the lineage tree of any one individual (*ii*) since scRNA-Seq is reliant on physical disassociation of cells from their residing tissue, it introduces additional computational challenges in linking spatial information (in the form of x, y, z coordinates) to a library of sequenced cells (see Figure 1.2) (Teves and Won, 2020).

In this thesis work, I present an alternate solution to scRNA-Seq. The proposed strategy relies *completely* on imaging and jointly addresses both central questions introduced above. The computational pipeline integrates (through *registration*) *digitized* live embryos, imaged through Selective Plane Illumination Microscopy (SPIM) (*digitized* by using *instance segmentation* and *tracking*) with separate, digitized embryos, fixed and stained for the expression of specific genes, following the norms of Whole Mount *in situ* hybridization (WMISH). In the following paragraphs, I will elucidate the idea behind the underlying components.

A common strategy for performing *in vivo* imaging of living organisms is by generating transgenic organisms in which fluorescent proteins label individual cells and tissues. SPIM is a fluorescence microscopy technique where the transgenic specimen is illuminated with a focused light plane on the side (Huisken and Stainier, 2009). It offers a high spatial and temporal resolution which allows one to employ computer vision strategies such as *instance segmentation* to detect and identify the shape of each nucleus/cell from volumetric image snapshots, and *tracking* which *connects* these digitized segmentations (obtained per time-point) across time in order to build a lineage tree (this strategy is referred to as *tracking by detection* where in several detections/instance segmentations obtained per time point are connected across time or assigned the same id based on whether they show a high similarity in terms of a feature). The lineage tree enables keeping track of parent-child relationships and monitoring the positions, velocities and trajectories of most cells.

Since the resolution that can be achieved in live specimens is generally lower than in fixed specimens (owing to movement of the cells, light scattering *etc.*), often spatial gene expression maps are obtained through alternate imaging modalities. With WMISH, individual specimens are fixed and stained for the expression of key developmental genes and imaged through confocal microscopy. ISH utilizes fluorescently-labeled nucleotide probes to detect specific mRNAs, localized at different spatial coordinates in a cell (see Figure 1.3). The number of RNA

species which can be simultaneously measured by ISH is, however, limited. As a consequence, one must fix and stain *multiple* equally-aged specimens.

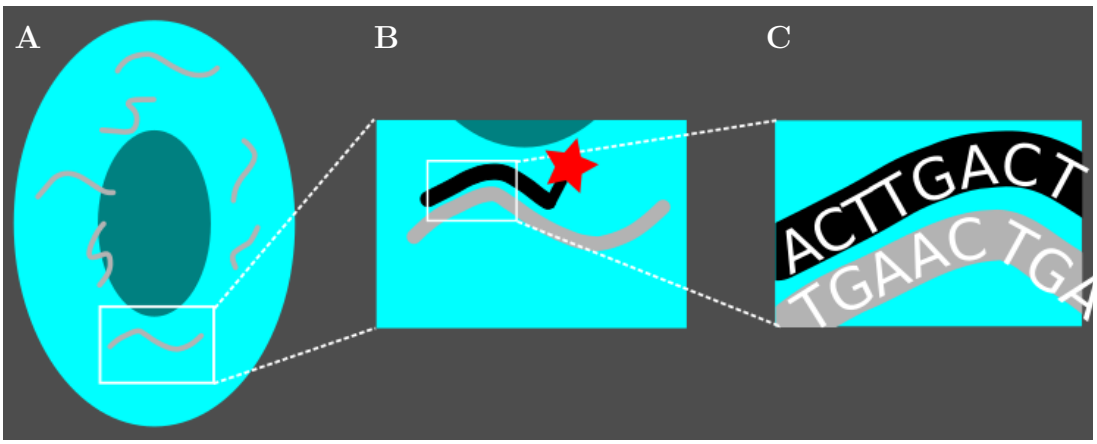

Figure 1.3: **Schematic of *in situ* hybridization (ISH).** (A) mRNA molecules produced by a specific gene, are visualized as gray strands. (B & C) Complimentary probes (shown as a black strand) are designed for these target mRNA molecules and are hybridized to the target molecules. These probes are labeled with a fluorophore (shown as a red star) which enables visualizing each individual mRNA molecule as a dot, when inspected under the confocal microscope.

Assuming that the above-mentioned digitization of the SPIM movie of the live embryo and the confocal images of the ISH specimens is successful, then one could estimate a bridging transform between the two imaging modalities by using a computer vision strategy called as *registration*. This is especially pertinent because the process of fixation causes non-uniform shrinking of an embryo. Additionally each individual *in situ* specimen possesses its own unique orientation and global position.

But to which time point in the SPIM movie of the live embryo should the in situ specimen register? For stereotypically-developing organisms where the variability in the positions, number and division patterns of cells, between equally aged individuals is low, one could use *cell count* as a proxy for staging the *in situ* specimen along the lineage tree arising from the live embryo.

I show the results of our computational pipeline on the marine annelid worm *Platynereis dumerilii*. *Platynereis dumerilii* is an excellent model system since embryos are known to develop stereotypically (Fischer et al., 2010). Further, embryos are large enough to micro-inject for fluorescent labeling of cells and nuclei (Özpolat et al., 2021). Additionally, nuclei are small and transparent enough for a long duration, which enables acquiring images with high cellular resolution. Lastly, it has been shown recently that linking cell lineages and individual cells between different image modalities is possible for early-stage *P. dumerilii* embryos (Vergara et al., 2021).

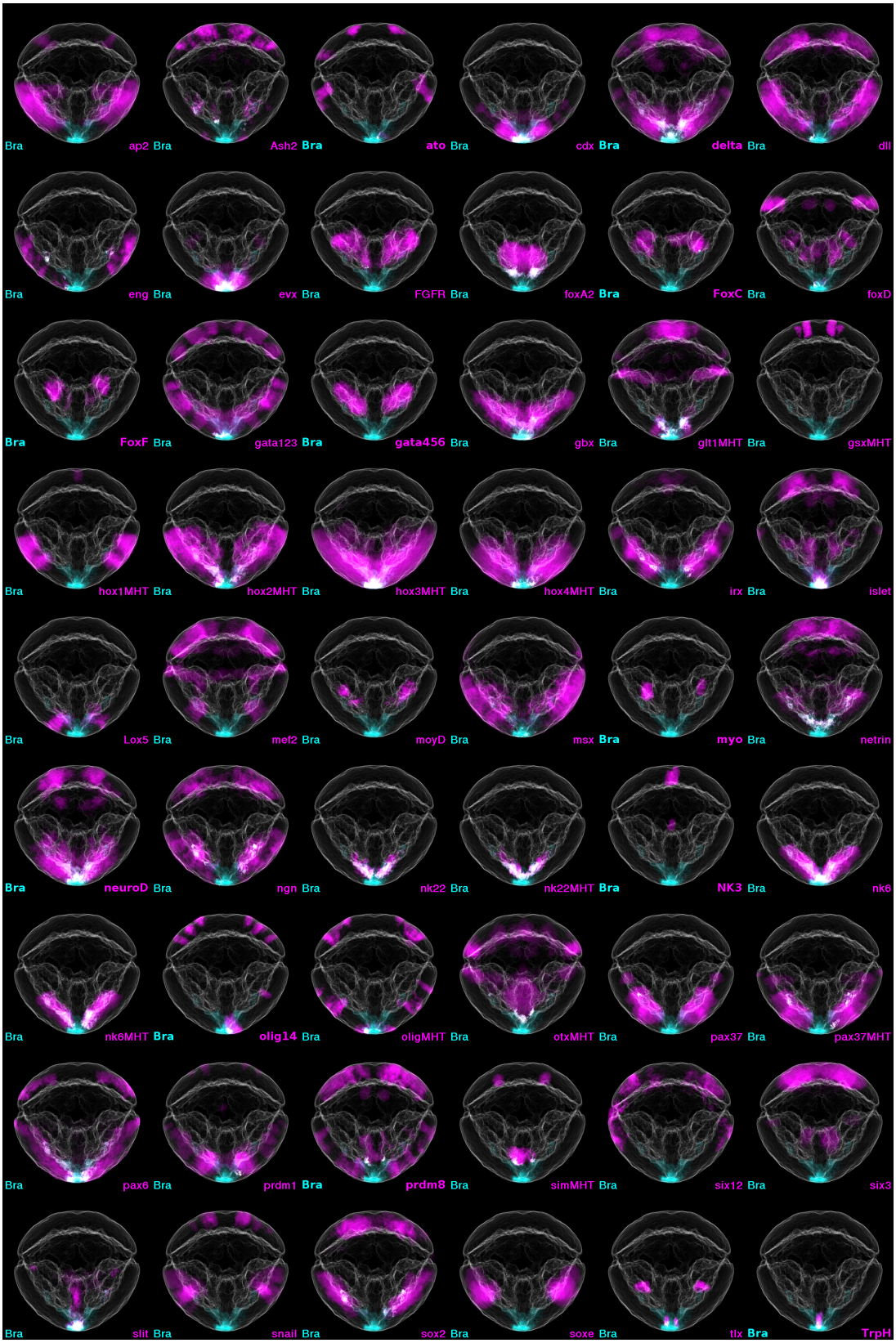

Figure 1.4: **Max z projections of ISH specimens at the developmental stage of** 24 **hpf.** These specimens were aligned and registered to a reference atlas using *ProSPr* (Vergara et al., 2017). Image credits: Mette Handberg-Thorsager.

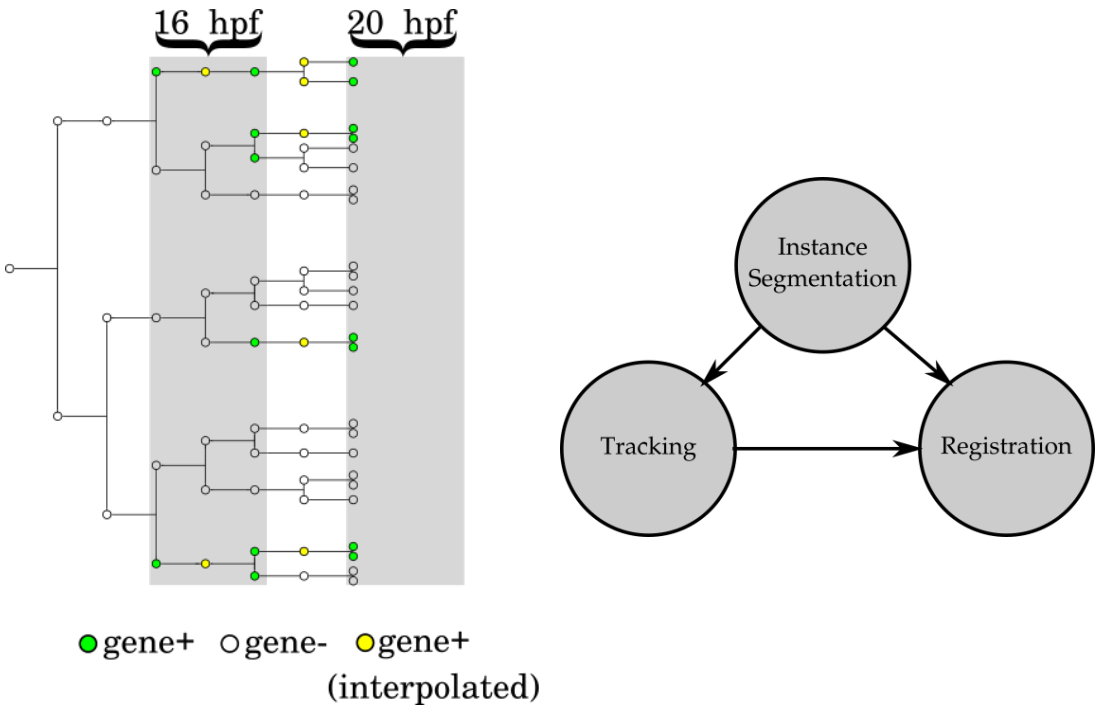

Figure 1.5: **Schematic and building blocks of the computational pipeline.** By leveraging the computer vision strategies of *Instance Segmentation*, *Tracking* and *Registration* (right), one can transfer information about cellular gene expression from the ISH specimens onto the lineage tree (left).
Here *green* indicates cells which are expressing a given gene, while *white* indicates cells which are not expressing that gene. By leveraging the available lineage tree, one can even conjecture about the state of gene expression in between developmental stages (for which no ISH data is available) - here, *yellow* indicates cells expressing a given gene. This computational pipeline enables setting up a canonical space for monitoring dynamic changes in cell morphology and gene expression, during the early development of *Platynereis dumerilii*.

To test my computational pipeline, I use two sets of real biological specimen. Firstly, representing the fixed biological specimen containing information about gene expression, we (in combination with our collaborators) collected whole-mount specimens of *Platynereis dumerilii* stained with ISH probes for several different, developmentally regulated transcription factors at three specific developmental stages of 16, 20 and 24 hpf (See Figure 1.4 for ISH specimens at the developmental stage of 24 hpf). These specimens were scanned in 3D by laser scanning confocal microscopy resulting in three-dimensional images containing the DAPI (nucleus) channel used in our registration as a common reference and the gene expression channel.

Secondly, representing the live imaging modality, we obtained access to a recording capturing the embryological development of the *Platynereis dumerilii* at cellular resolution *in toto* (Tomer et al., 2012) using a SimView light sheet

microscope. The embryos were injected with a fluorescent nuclear tracer prior to imaging and thus the time-lapse movie visualizes all the nuclei in the embryo throughout development. This movie which extends from 5 to 24 hpf stage of *Platynereis* development provides an appropriate inter-modal target to register the fixed specimen to on the basis of the common nuclear signal.

Since, digitization of the individual static snapshots of the ISH specimens and the movie of the live embryo, alongside the procedure for estimating a bridging transform are critical (see Figure 1.5), these will form a major chunk of this dissertation work. I shall firstly elucidate the method behind *Instance Segmentation* in Chapter 2. Next in Chapter 3, I shall delve into my approach to address *Tracking* and building lineage trees. Then in Chapter 4, I will attempt to explain the method behind *Registration*. Finally in Chapter 5, I will discuss the biological insights we can draw from using this pipeline in the context of *P. dumerilii* (also see Figure 1.1).

1.1 Contributions

Parts of this thesis have been published:

1. **Lalit, M.**, Tomancak, P., and Jug, F. "EmbedSeg: Embedding-based Instance Segmentation for Biomedical Microscopy Data". Accepted in: *Medical Image Analysis 2022.*

2. **Lalit, M.**, Tomancak, P., and Jug, F. "Embedding-based Instance Segmentation in Microscopy". In: *Medical Imaging with Deep Learning (MIDL) Conference 2021.*

3. **Lalit, M.**, Handberg-Thorsager, M., Hsieh, Y-W., Jug, F., Tomancak, P. "Registration of Multi-modal Volumetric Images by Establishing Cell Correspondence". In *ECCV Bio-Image Computing Workshop, Lecture Notes in Computer Science 2020.*

Separately, I also worked on other projects during the thesis work, a few of which were published:

1. Prakash, M., **Lalit, M.**, Tomancak, P., Jug, F., and Krull, A. "Fully Unsupervised Probabilistic Noise2Void". *In International Society of Biomedical Imaging 2020.*

2. Prakash, M., Buchholz, T-O., **Lalit, M.**, Tomancak, P., Krull, A., and Jug, F. "Leveraging Self-supervised Denoising for Image Segmentation". *In International Society of Biomedical Imaging 2020.*

3. Krull, A., Vičar, T., Prakash, M., **Lalit, M.** and Jug, F. "Probabilistic Noise2Void: Unsupervised Content-Aware Denoising". In *Frontiers in Computer Science 2020.*

Bibliography

Fischer, A. H., Henrich, T., and Arendt, D. (2010). The normal development of platynereis dumerilii (nereididae, annelida). *Frontiers in zoology*, 7(1):1–39.

Huisken, J. and Stainier, D. Y. (2009). Selective plane illumination microscopy techniques in developmental biology.

Özpolat, B. D., Randel, N., Williams, E. A., Bezares-Calderón, L. A., Andreatta, G., Balavoine, G., Bertucci, P. Y., Ferrier, D. E., Gambi, M. C., Gazave, E., et al. (2021). The nereid on the rise: Platynereis as a model system. *EvoDevo*, 12(1):1–22.

Saelens, W., Cannoodt, R., Todorov, H., and Saeys, Y. (2019). A comparison of single-cell trajectory inference methods. *Nature biotechnology*, 37(5):547–554.

Teves, J. M. and Won, K. J. (2020). Mapping cellular coordinates through advances in spatial transcriptomics technology. *Molecules and Cells*, 43(7):591.

Tomer, R., Khairy, K., Amat, F., and Keller, P. J. (2012). Quantitative high-speed imaging of entire developing embryos with simultaneous multiview light-sheet microscopy. *Nature methods*, 9(7):755–763.

Vergara, H. M., Bertucci, P. Y., Hantz, P., Tosches, M. A., Achim, K., Vopalensky, P., and Arendt, D. (2017). Whole-organism cellular gene-expression atlas reveals conserved cell types in the ventral nerve cord of *Platynereis dumerilii*. *Proceedings of the National Academy of Sciences*, 114(23):5878–5885.

Vergara, H. M., Pape, C., Meechan, K. I., Zinchenko, V., Genoud, C., Wanner, A. A., Mutemi, K. N., Titze, B., Templin, R. M., Bertucci, P. Y., et al. (2021). Whole-body integration of gene expression and single-cell morphology. *Cell*, 184(18):4819–4837.

Instance Segmentation

Contents

2.1 Introduction **12**
2.2 Related Work **13**
2.2.1 Shape-agnostic Approaches 14
2.2.2 Shape-augmented Approaches 15
2.2.3 Embedding-based Approaches 15
2.3 Proposed Method **17**
2.3.1 Network Architecture and Inference Scheme 19
2.3.2 Using the Medoid as Embedding Location 20
2.3.3 Tiled Predictions via Label Propagation 20
2.4 EmbedSeg (Sliced) **21**
2.5 EmbedSeg (ILP) **21**
2.6 Alternate Inference Strategies **24**
2.7 Baselines, Experiments, Results **25**
2.7.1 Used 2D Data 26
2.7.2 Used and Contributed 3D Data 26
2.7.3 Baseline Methods 26
2.7.4 Data Handling in 2D 27
2.7.5 Data Handling in 3D 28
2.7.6 Training Details for EmbedSeg & Neven *et al* 29
2.7.7 Performance Evaluations 29
2.7.8 Ablation Studies 32
2.8 Open Code and Online Resources **32**
2.8.1 EmbedSeg napari 32
2.8.2 EmbedSeg Demo 34
2.9 Discussion **35**

2.1 Introduction

The task of instance segmentation addresses the combined problem of identifying the semantic label (class label) for each pixel in an image, and at the same time determining a unique id for each object instance in the image.

Instance segmentation of cellular and sub-cellular structures in biomedical microscopy images is needed for a plethora of downstream analyses, such as (*i*) outlining tissues, cells, nuclei, or other organelles, for example to identify phenotypic changes that result from environmental or genetic modifications (Yang et al., 2017), (*ii*) feature-based image registration between time-points or imaging modalities (Kist et al., 2021; Lalit et al., 2020), or (*iii*) tracking-by-detection where segmented instances get associated over time to form object tracks, for example to study the interaction dynamics of biological objects (Stern et al., 2021; Gomez et al., 2021).

Owing to its importance, many methods have been proposed which all tackle the task of identifying unique object instances in biomedical image data (Meijering, 2012). More recently, Deep Learning (DL) has enabled new approaches to automated instance segmentation and has lead to substantial improvements in segmentation quality (He et al., 2018; Schmidt et al., 2018; Hirsch et al., 2020; Stringer et al., 2021). A shortcoming of many existing methods is that they do not directly optimize for the Intersection over Union (IoU) metric during training. Instead, simpler to use cross-entropy loss (for classification) or L1 or L2 losses (for regression) are typically used (Stringer et al., 2021; Schmidt et al., 2018). Optimizing for such 'auxiliary metrics' promotes sub-par results with respect to IoU, which is in many cases, the metric one would truly like to maximize for most useful automated results.

Another weakness of many existing segmentation approaches is that they only operate on 2D image data. While some methods are intrinsically not fit for full 3D data, *e.g.* Upschulte et al. (2022), other methods would generalize to full 3D operations, but lack respective implementations (Neven et al., 2019).

Still, 2D methods can of course be applied to axis-parallel 2D slices of a given 3D dataset. If such redundant 2D results are then averaged (or otherwise suitably merged) during inference, reasonable 3D instance segmentation results can often be achieved (Stringer et al., 2021). In general, we observe that methods to segment volumetric (3D image) data are less common, despite being desperately needed in countless biomedical applications.

Here I present a bottom-up instance segmentation method called EmbedSeg[1], a capable and very compact model for end-to-end instance segmentation which we initially presented at the MIDL conference (Lalit et al., 2021). In EmbedSeg, each pixel predicts its own *spatial embedding*, *i.e.* another unique pixel location (embedding pixel) that is meant to represent the entire object this particular pixel is part of. At the same time, EmbedSeg learns an instance-specific clustering band-width used to cluster nearby embedding pixels into object instances. The segmentation mask of an object is defined by all original image pixels that point to the same cluster of embedding pixels. An additional *seediness score* for each pixel is simultaneously predicted as well, indicating how likely it is for the respective pixel, and its associated clustering band-width, to indeed represent a true object instance.

Pixel-wise spatial embeddings, clustering bandwidth, and seediness score are learnt jointly and end-to-end, allowing the individual predictions to fit well together right away. As I describe later, EmbedSeg does this by directly optimizing to maximize the Intersection over Union (IoU) metric during training, which thereby leads to results that are close to ground truth in a practically sensible distance score.

EmbedSeg proposes several novelties over existing embedding based instance segmentation methods that help to improve the quality of segmentation results on biomedical image data. Maybe practically most relevant, EmbedSeg is not limited to 2D images but can directly be trained and applied on volumetric data in full 3D. I provide four possible modes (*2D*, *3D*, *3D-sliced* and *3D-ilp*), one could choose from, for training and inferring on one's data and respective annotations, which are elucidated in the following sections.

2.2 Related Work

The field of instance segmentation encompasses a vast body of literature. Here we focus on rather recent state-of-the-art methods that have found successful application on biomedical image data.

We distinguish three types of instance segmentation methods: (*i*) Shape-agnostic approaches, that map each pixel to a set of semantic classes and do not explicitly model or learn the shapes of objects to be segmented. (*ii*) Shape-augmented approaches, which map each pixel to an explicit shape representation of the underlying object. (*iii*) Embedding-based approaches, that learn latent embedding locations for each pixel and post-process these to obtain final object

[1] A memory-efficient open-source implementation of EmbedSeg is available at `https://github.com/juglab/EmbedSeg`.

instance masks.

2.2.1 Shape-agnostic Approaches

The vanilla U-Net (Ronneberger et al., 2015) followed by, for example, connected component analysis, is a member of this category of approaches. But several other methods for biomedical instance segmentation that belong to the category as well have themselves employed a U-Net backbone.

Dietler et al. (2020), for example, segmented non-convex shaped budding cells of *Saccharomyces cerevisiae* (yeast) by training a U-Net to learn the mapping between an image pixel and its class. During inference, they thresholded the foreground class probability map and performed a distance transform on this thresholded map in order to obtain interior seed locations and followed this up with a watershed-based transformation strategy for obtaining the final masks (Dietler et al., 2020).

Scherr et al. (2020) propose a different approach where a U-Net is trained to map each pixel in a given image to a normalized distance to the surrounding cell boundary, also introducing a novel neighbor distance. These learnt distance fields are then further evaluated during inference time to obtain instance masks.

In *DenoiSeg* (Buchholz et al., 2020), the authors train a U-Net jointly for the task of denoising and 3-Class pixel classification (foreground, background, membrane). Results show that segmentation can gain significantly in performance by co-learning the unsupervised denoising task, for which typically much more training data is readily available. During inference, these class probability maps are post-processed (connected components analysis) in order to obtain the final instance masks.

Also in this category of methods is *Cellpose* (Stringer et al., 2021). In Cellpose a U-Net is trained to predict a flow at each pixel. The ground truth vector flow field is pre-computed from the instance masks as solution to the steady-state heat diffusion equation, assuming a heat source placed at the center of the object instance and a Dirichlet boundary condition at the object boundary. During inference, the predicted flows are followed to group pixels which converge to the same location into object instances. Cellpose can also be applied to 3D image data, but owing to the fact that acquiring ground truth instance masks as training data is hard, the authors propose to train a 2D model and apply it in a suitable way to all axis parallel slices of the 3D data.

Another recent and rather interesting member of this family of approaches

is called *PlantSeg* (Wolny et al., 2020). In a rather classical fashion, the authors propose to train a U-Net to learn the mapping between image voxels and the boundary probability. Most of the novelty is introduced during inference, where specialized graph partitioning methods are employed to obtain high quality instance segmentations on large 3D plant tissues datasets.

2.2.2 Shape-augmented Approaches

StarDist (Schmidt et al., 2018) and *StarDist-3D* (Weigert et al., 2020) are arguably the most widely applied methods in this category. StarDist jointly predicts, at each pixel (voxel), the distance to the boundary of the surrounding object and the pixel's foreground probability. The boundary prediction happens by predicting distances along a predefined number of directions (rays) projecting out of every given pixel. During inference, all candidate pixel predictions are post-processed by first sorting them in the decreasing order of their predicted foreground probability score and then suppressing overlapping proposals using a greedy non-maximum suppression scheme. By design, all predicted instance masks are star-convex, *i.e.* the entire interior of each instance can be seen from the pixel (voxel) that made the initial prediction.

In order to address also more complex, non star-convex shapes, *PatchPerPix* (Hirsch et al., 2020) proposes to predicts a dense binary mask per pixel. These learnt local per-pixel (per-voxel) shape descriptor masks are, during inference, assembled into complete object instance masks.

With *Contour Proposal Networks*, (Upschulte et al., 2022) introduce a different shape representation where a model learns the mapping between an image pixel and the corresponding Elliptical Fourier Descriptor representation of the underlying object. This is achieved by a loss formulation which encourages the predictions of the model to be accurate in the pixel and the frequency domains. While this is a very elegant approach, it is currently only applicable to 2D data.

2.2.3 Embedding-based Approaches

These methods map each pixel to an unique (latent) embedding location, such that pixels belonging to the same object map to nearby locations. During inference, embedding locations need to be clustered, *e.g.* by using a fast variant of the mean-shift algorithm (Fukunaga and Hostetler, 2006), and the final object instance masks are defined by all pixels that embedded themselves into one of those identified clusters.

Embedding-based segmentation methods initially emerged in the context of multi-person pose estimation in classical computer-vision. Newell et al. (2017)

initially suggested a framework where each pixel predicted a so called *tag*. The proposed objective encouraged pairs of tags to have similar values if and only if they belonged to the same object.

In the same year, Brabandere et al. (2017) suggested a specific hinge-loss which additionally forced embeddings arising from pixels belonging to different objects to have different values, which lead to improved clustering of embedding locations during inference time.

Payer et al. (2018) employed cosine similarity (instead of euclidean distance) between pixel embeddings in the loss formulation, and showed applicability to biomedical image segmentation and tracking. Their learnt pixel embeddings are clustered during inference into instance segmentations using the HDBSCAN algorithm (Campello et al., 2015).

Novotny et al. (2018) later showed that constructing dense pixel embeddings to separate objects is not possible with a fully convolutional setup and suggested using semi-convolutions to alleviate this problem.

Lee et al. (2021) instead showed that first calculating an affinity graph from learnt pixel or voxel embeddings (in their case arising from overlapping image patches) and then partitioning the (aggregated, complete) affinity graph using the Mutex Watershed algorithm (Wolf et al., 2020) even enables instance segmentations of intertwined branching neurons in electron microscopy images.

Another idea by Kulikov and Lempitsky (2020) proposes to calculate harmonic embeddings from ground truth instance masks and then learning the mapping between the input image and these auxiliary, ground truth pixel embeddings. During test-time, quite similar to the method by Brabandere et al. (2017), the mean-shift clustering algorithm was used to obtain the final instance masks.

In the wonderful work by Neven et al. (2019), the authors acknowledge that in most embedding-based methods either a fixed-size clustering bandwidth or no bandwidth is specified in the loss formulation, and that this leads to sub-optimal results during inference. Hence, they proposed to instead jointly learn an instance-specific clustering bandwidth and pixel-level embeddings. Furthermore, they proposed a modified loss formulation which allowed optimizing for the IoU metric over each ground truth instance mask during training, which leads to better quality instance segmentation masks.

2.3 Proposed Method

The goal of instance segmentation is to cluster a set of pixels $\vec{X} = \{\vec{x}_1 \dots \vec{x}_i \dots \vec{x}_N\}$, where $\vec{x} \in \mathcal{R}^D$, with D being the dimensionality of the given input images, into a set of segmented object instances $S = \{S_1 \dots S_k \dots S_K\}$.

This is achieved by learning an offset vector $\vec{o}_i$ for each pixel $\vec{x}_i$, so that the resulting (spatial) embedding $\vec{e}_i = \vec{x}_i + \vec{o}_i$ points to its corresponding object center (instance center) $\vec{C}_k$. Here, $\vec{o}_i$, $\vec{e}_i$ and $\vec{C}_k$ are in $\mathcal{R}^D$, with $D \in \{2, 3\}$, depending if the input is a 2D image or a 3D volume.

In order to do so, a Gaussian function ϕ_k for each object S_k is used, which converts the distance between a (spatial) pixel embedding $\vec{e}_i$ and the instance center $\vec{C}_k$ into the probability of belonging to that object.

$$\phi_k\left(\vec{e}_i\right) = \exp\left(-\left\|\frac{\left(\vec{e}_i - \vec{C}_k\right)^T \vec{\Sigma}_k^{-1} \left(\vec{e}_i - \vec{C}_k\right)}{2}\right\|\right). \tag{2.1}$$

A high probability is interpreted as the pixel embedding $\vec{e}_i$ being close to the instance center $\vec{C}_k$, meaning that the corresponding pixel is likely to belong to the object S_k. A low probability, in contrast, means that the pixel is more likely to belong to either the background or to another object. More specifically, if $\phi_k(\vec{e}_i) > 0.5$, the pixel at location $\vec{x}_i$ will be assigned to object S_k. Here, $\vec{\Sigma}_k \in \mathcal{R}^{D \times D}$ is the diagonal covariance matrix, representing the explicitly learned and predicted cluster bandwidth for object S_k. The corresponding standard deviation vector for object S_k is indicated as $\vec{\sigma}_k \in \mathcal{R}^D$ whose entries along the d^{th} dimension are denoted as $\sigma_{k,d}$. For example, for volumetric image data, where $D = 3$,

$$\vec{\Sigma}_k = \begin{bmatrix} \sigma_{k,1}^2 & 0 & 0 \\ 0 & \sigma_{k,2}^2 & 0 \\ 0 & 0 & \sigma_{k,3}^2 \end{bmatrix}.$$

In order to allow larger objects to predict a larger and similarly, smaller objects to predict a smaller $\vec{\Sigma_k}$, we let each pixel $\vec{x}_i$ of object k individually predict a $\vec{\sigma_i}$ and compute the corresponding $\vec{\sigma_k}$ for the constituting object as the mean of all predicted $\vec{\sigma_i}$ for that object, *i.e.*

$$\vec{\sigma_k} = \frac{1}{|S_k|} \sum_{\vec{\sigma_i} \in S_k} \vec{\sigma_i}.$$

By comparing the predicted ϕ_k of a given object to the corresponding ground truth foreground mask S_k, we compute the differentiable Lovász-Softmax loss L_{IoU} (Berman et al., 2018; Yu and Blaschko, 2015).

At inference time, the remaining computational task to be addressed is how to identify the best clusters that represent all object instances. More concretely, this problem is to look for a suitable cluster center and cluster *margin* around it, such that all pixel embeddings of a given object fall within this area, which no other pixels have embedded themselves in it.

For this purpose, we also let each pixel predict a *seediness* score that indicates how likely it is for this given pixel to be the designated embedding location for the object instance it is part of. This means, the seediness score should actually be close to the output of the gaussian function in Equation (2.1).

With this all in mind, we can now formulate a training loss function for the seediness score as

$$L_{\text{seed}} = \frac{1}{N} \sum_{i=1}^{N} w_{\text{fg}} \mathbb{1}_{\{s_i \in S_k\}} \|s_i - \phi_k(\vec{e_i})\|^2 + w_{\text{bg}} \mathbb{1}_{\{s_i \notin S_{\text{fg}}\}} \|s_i - 0\|^2,$$

which allows minimizing the distance between the output of the gaussian function corresponding to any pixel and the predicted seediness score, arising from that pixel. The seediness score for the background pixels are regressed to 0.

Furthermore, to ensure that at inference, while sampling highly seeded pixels, $\vec{\sigma}_k \approx \hat{\vec{\sigma}}_k$, we include a smoothness loss

$$L_{\text{var}} = \frac{1}{|S_k|} \sum_{\vec{\sigma_i} \in S_k} \|\vec{\sigma}_i - \vec{\sigma}_k\|^2.$$

Hence, the joint loss function for training EmbedSeg is then given by the weighted sum

$$L = w_{\text{seed}} L_{\text{seed}} + w_{\text{IoU}} L_{\text{IoU}} + w_{\text{var}} L_{\text{var}}. \tag{2.2}$$

2.3.1 Network Architecture and Inference Scheme

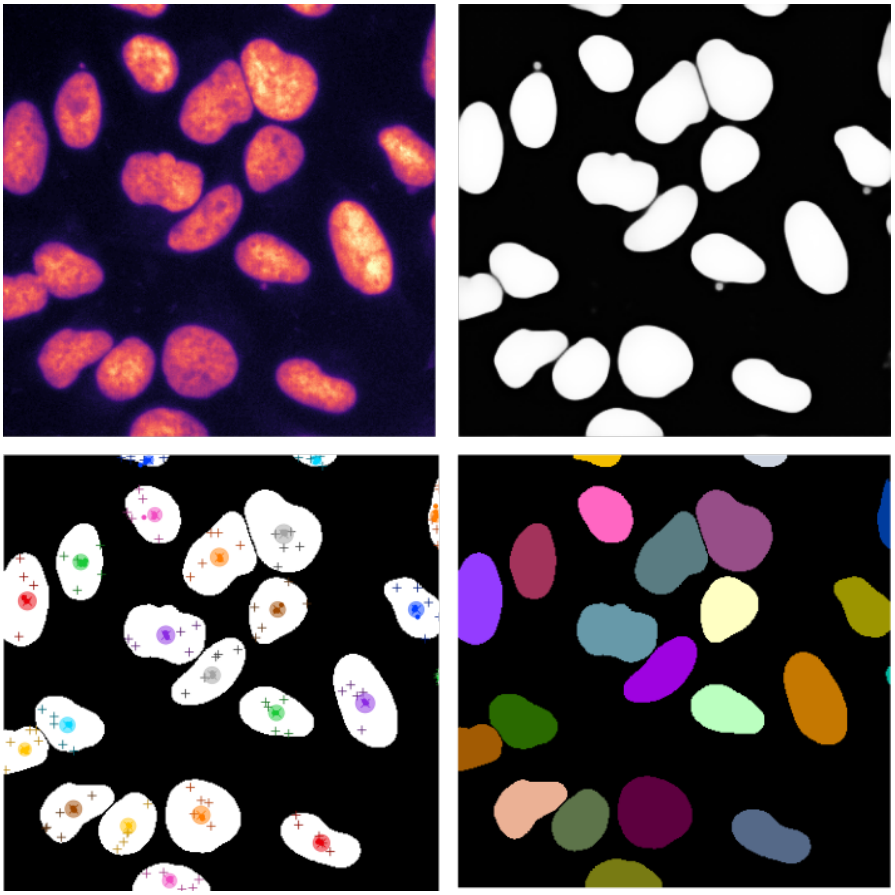

Figure 2.1: **Visualization of the inference procedure of EmbedSeg.** (top-left) An exemplary raw input image. During inference, we iteratively pick seed pixels greedily from the predicted seediness map (top-right) and cluster other foreground pixels w.r.t. their predicted embeddings as explained in Section 2.3. The image on the (bottom-left) shows the thresholded seediness map as the white region, 5 randomly sampled foreground pixels (marked with colored pluses), embedding location of these pixels (dots, mostly tightly clustered within ellipses), and the predicted clustering bandwidth thresholded at a likelihood of 0.5 (semi-transparent ellipses). The (bottom-right) image shows the final predicted instance segmentation result.

EmbedSeg uses a branched ERF-Net (Romera et al., 2018; Neven et al., 2019) for 2D images and branched ERF-Net 3D for volumetric image data. Once trained, and as also visualized in the bottom left panel of Figure 2.1, the following inference scheme is used to find object instances: (i) all pixels with a seediness score $s_i > s_{\text{fg}}$ are collected in a set of foreground pixels S_{fg}, (ii) from all pixels in S_{fg}, we pick $\vec{x}_{\text{seed}}$, the pixel with the highest seediness score $s_i > s_{\text{min}}$ for a suitably chosen s_{min}, (iii) if such an $\vec{x}_{\text{seed}}$ exists, we collect all foreground pixels in S_{fg} that embed themselves at a location where the embedding likelihood defined by $\vec{e}_{\text{seed}}$ and $\vec{\sigma}_{\text{seed}}$ is > 0.5. Together, these pixels define a segmented instance S_k.

Finally, (*iv*) we remove all pixels S_k from S_{fg} and jump to step two until no more valid seed pixels $\vec{x}_{\text{seed}}$ exist in S_{fg}. In all presented experiments we use $s_{\text{fg}} = 0.5$ and $s_{\text{min}} = 0.9$.

To additionally boost performance during prediction, we use 8-fold and 16-fold test-time augmentation for 2D and 3D applications, respectively (Zeng et al., 2017; Wang and Solomon, 2019). This means that we take each image to be segmented and transform it by axis-aligned rotations and flips, predict each augmented copy independently, and then back-transform these predictions, average them, and compute object instance just as we do without augmentation.

2.3.2 Using the Medoid as Embedding Location

While Neven *et al.* either learn the desired embedding location during training or simply use the centroid of known objects, we argue that this is not the optimal choice when object shapes are more complex (*i.e.* not star-convex).

Instead, we reason that it is desirable to choose a point that minimizes the average distance to all pixels $\vec{x}_i \in S_k$, *i.e.* the *geometric median* (GM). Like the centroid, also the GM has the unfortunate property that it can lie outside of its defining object. Such object-external points are bad embedding points for two reasons: (*i*) the seediness score of such points will likely not be consistently high, and (*ii*) multiple such points might fall very close to each other in crowded image regions, potentially making a clean segregation (clustering) impossible.

Hence, we propose to use the *medoid* as the embedding location of choice instead. The medoid pixel of the object instance S_k is defined as the pixel of that object that has the smallest average distance to all other object pixels *i.e.*

$$\vec{x}_{\text{medoid}}(S_k) = \arg\min_{\vec{y} \in S_k} \frac{1}{|S_k|} \sum_{\vec{x} \in S_k} \|\vec{x}, \vec{y}\|_2.$$

2.3.3 Tiled Predictions via Label Propagation

During inference, we process evaluation images one-by-one. For larger data, mainly in 3D, this might still lead to out-of-memory exceptions caused by too little GPU memory being available. In order to provide a solution for this problem, we devised and implemented a label propagation strategy that enables us to stitch results obtained on overlapping tiles that by themselves do fit in GPU memory. The required size of the overlap margins is a direct consequence of the used network architecture and set to minimize computational and memory overhead.

During label propagation, for each pair of overlapping tiles, we re-assign ids to all instances which EmbedSeg has segmented in the overlapping region. This is performed by computing a maximum bipartite matching of predicted instances, using IoU as matching energy. Ids between neighboring tiles in the overlap region, are only considered as potential matches if their IoU score is greater than 0.5. Matched pairs below this threshold are interpreted as different objects.

Overall, this strategy enables the processing of arbitrary sized 2D images and 3D volumes without introducing additional parameters.

2.4 EmbedSeg (Sliced)

We also investigate a *Sliced* training regime for 3D images, where a 2D model is trained and applied to all axis parallel 2D slices of a volumetric 3D image, and the three individual prediction per voxel suitably merged to enable 3D instance segmentations. We found this interesting in direct comparison to Cellpose (Stringer et al., 2021) where a similar principle is used to segment 3D data. In practical applications this special training regime can be advantageous due to much reduced memory requirements during training and for only requiring 2D ground truth labels which are much easier to manually generate.

In more detail, for sliced training we use a 2D Branched ERF-Net on XY, YZ and ZX slices of the given 3D training data. For anisotropic data (typically lower pixel-resolution along the z axis), we first upsample the images using nearest neighbor interpolation and then train the network on all slices for which ground truth labels exist. During inference, we will naturally obtain three full prediction stacks which have to be merged to obtain consensus offsets, margin bandwidths, and seediness scores. Once merged, we follow the same inference scheme previously described in Section 2.3 and Figure 2.1 for full 3D instance segmentation.

2.5 EmbedSeg (ILP)

In this section, we introduce an alternate strategy to generate 3D instance segmentations on evaluation images using a trained 2D model. An advantage of this strategy, similar to *Sliced* (see Section 2.4) is a lower GPU memory footprint during the training of the 2D model. This strategy essentially re-assigns labels to objects predicted by a trained model, independently in each slice of a 3D volume by solving a global, integer linear-optimization task.

Firstly, we apply the trained 2D model independently on each slice of a 3D

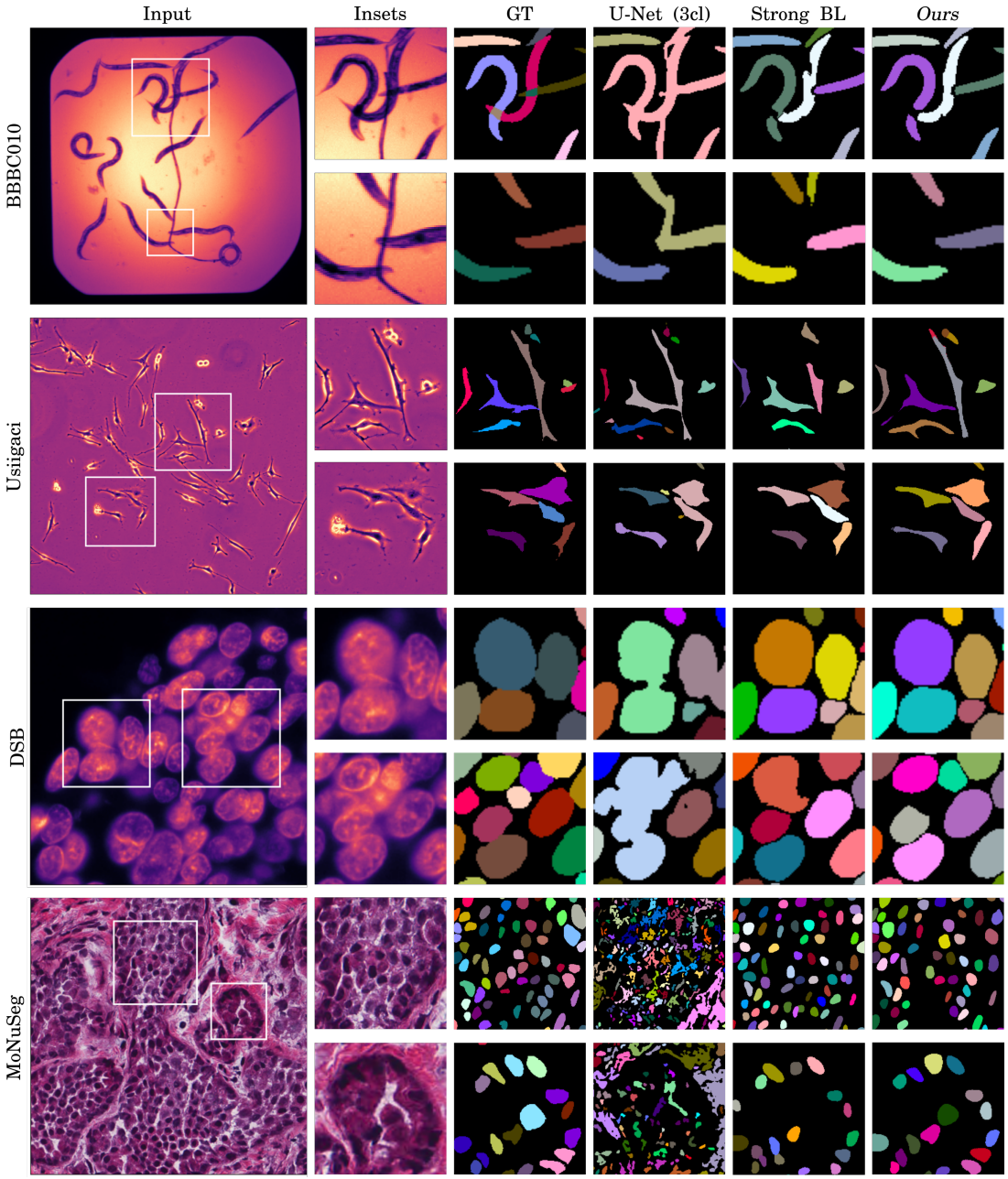

Figure 2.2: **Qualitative results in 2D. EmbedSeg and two baselines compared on representative images of the *BBBC010*, *Usiigaci*, *DSB* and *MoNuSeg* datasets.** Columns show: full input image, zoomed insets, ground truth labels (GT), and instance segmentation results by the 3-class U-Net baseline, the best performing competing baseline, and EmbedSeg. Each segmented instance is shown in a unique random color. The strong baseline methods shown are Neven et al. (2019), Cellpose, Neven et al. (2019) and StarDist for the *BBBC010*, *Usiigaci*, *DSB* and *MoNuSeg* datasets respectively

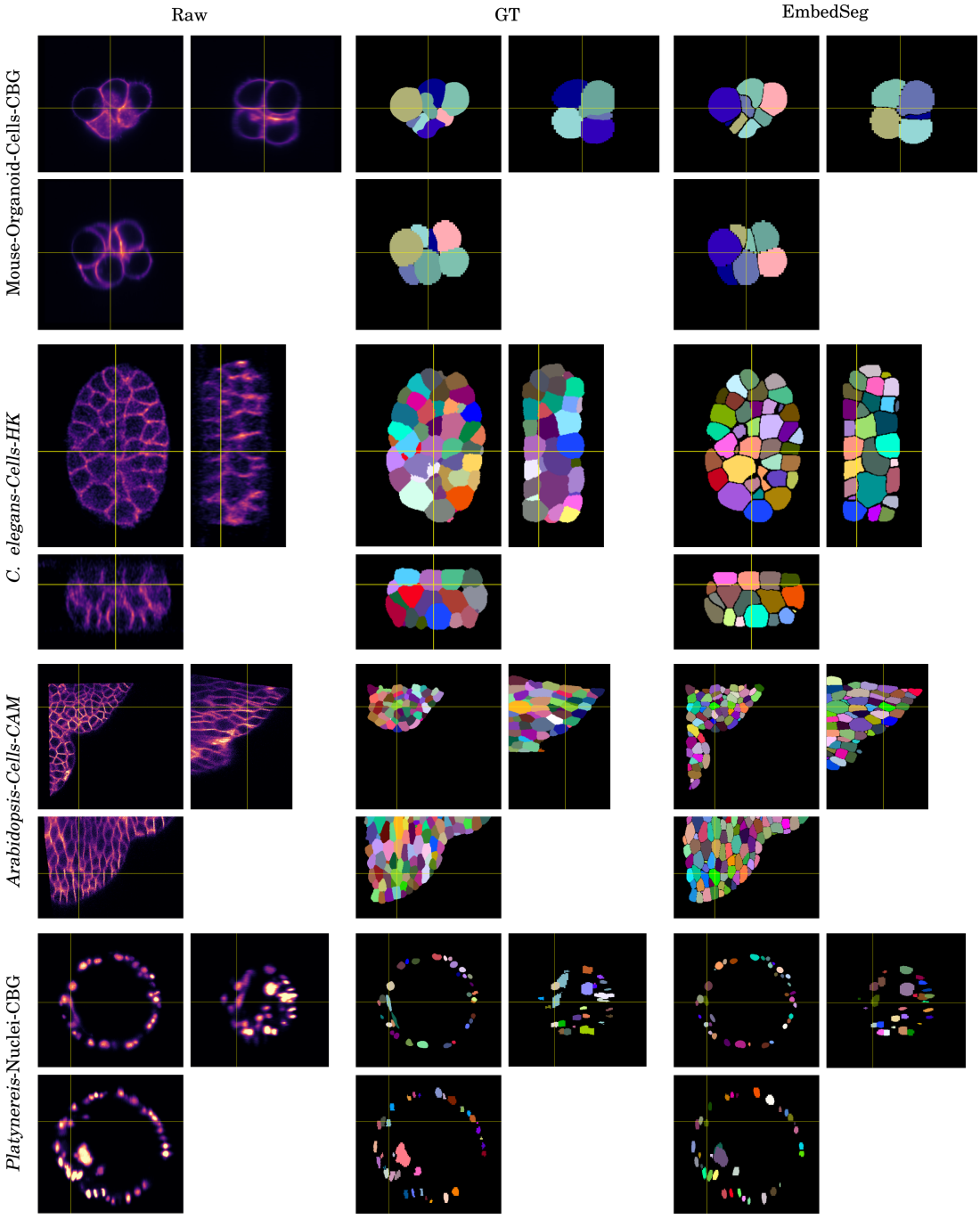

Figure 2.3: **Qualitative results in 3D. Qualitative results of EmbedSeg on the Mouse-Organoid-Cells-CBG, *C. elegans-Cells-HK*, *Arabidopsis-Cells-CAM* and *Platynereis*-Nuclei-CBG datasets.** Columns show orthogonal XY, YZ and XZ slices of one representative input image, ground truth labels (GT), and our instance segmentation results using EmbedSeg, respectively. Note that each segmented instance is shown in a random but unique color.

volume. This generates labels for object cross-sections which are not consistent across the slices. Hence, the next task is to re-associate them so that the labels for the same 3D object are equal.

We associate the generated instance segmentations between the slices by solving a global, integer linear program. A sparse, undirected graph is created by considering each instance segmentation as a node and linking it to instance segmentations in the next Z slices. Connections are allowed in the next Z slices instead of just the next slice in order to account for missing segmentations. Edges are built between consecutive slices by identifying nodes which have an IoU score greater than 0 (*i.e.* there is some overlap between the considered pair of nodes).

A binary appearance variable A_i^z is associated with the ith instance segmentation at depth z, and $\mathcal{A}$ denotes the set of all appearance indicator variables. A binary disappearance variable D_i^z is associated with the ith instance segmentation at depth z, and $\mathcal{D}$ denotes the set of all disappearance indicator variables. Binary assignment variables $E_{ij}^{z_e z_s}$ are associated with the ith instance segmentation at depth z_e and the jth instance segmentation at depth z_s ($z_e \in [z_s - Z, \ldots, z_s - 1]$), and $\mathcal{E}$ denotes the set of all assignment indicator variables.

The global optimization problem concerns solving for

$$\hat{\mathcal{E}}, \hat{\mathcal{A}}, \hat{\mathcal{D}} = \text{argmin } C_{\text{app}}\mathcal{A} + C_{\text{disapp}}\mathcal{D} + C_{\text{edge}}\mathcal{E}, \tag{2.3}$$

where the corresponding edge cost is set equal to $1 - \text{IoU}$ between the considered instance segmentations. The node costs for appearance and disappearance are both set equivalent to the normalized size (in number of pixels) of the instance segmentation.

2.6 Alternate Inference Strategies

Currently, the seediness map predicted by the trained model is processed in a greedy fashion to carve out individual instances (see Section 2.3.1). However, one could adopt alternate ways of processing this seediness map. For example, in the case that single point locations per cell are additionally available by a process of manual curation, then these available seed points could be used instead, in order to cluster neighboring pixels at the embeddings predicted for these seed locations.

Another strategy is to use the Multi-Cut approach, employed in *PlantSeg* (Wolny et al., 2020). In order to include the Multi-Cut strategy, one must first

find the local maxima of the predicted seediness map, next run the seeded watershed algorithm from the locations of these local maxima, and doing so generates a number of *super-voxels*. These super-voxels are perceived as nodes of a graph, where the edges are placed between super-voxels which are adjacent. The corresponding edge weight is set as the negative of the average seediness score along the line segment joining any two nodes (super voxels) (if a ground truth cell is over-segmented into two or more super voxels, then the edge weight between these over-segmentations would be low. On the other hand, in between actual ground truth cells, the edge weight would be higher in magnitude). Finally, running the Multi-Cut algorithm would merge super-voxels and provide an improved instance segmentation.

A third strategy would be to let each foreground pixel (*i.e.* seediness score above a certain threshold s_{fg} - see Section 2.3.1) predict the corresponding instance it belongs to. In the next step, these predicted, overlapping instances can be resolved either by a similar inference scheme as in StarDist (Schmidt et al., 2018) where overlapping instances are processed greedily or as in PatchPerPix (Hirsch et al., 2020) where overlapping instances are processed by building a patch affinity graph and the final instance segmentation is obtained through signed graph partitioning.

2.7 Baselines, Experiments, Results

Table 2.1: **Used 3D datasets.** We introduce four new volumetric microscopy datasets, covering various practically relevant imaging conditions and microscopy modalities. All datasets come with high quality ground truth labels for training.

Name	**Description**	**Pixel Size** (Z,Y,X) [μm^3]	**Bit Depth**	**Used Microscope**
Mouse-Organoid-Cells-CBG (Lalit et al., 2021)	Mouse Embryonic Stem Cells, R1 cell line, labeled membrane	(1.0, 0.1733, 0.1733)	uint16	Selective Plane Illumination Microscopy
C. elegans-Cells-HK (Cao et al., 2020)	Cells of multiple, developing *Caenorhabditis elegans* embryos (4-350 cell stage), labeled membrane	(0.22, 0.22, 0.22)	uint8	Laser Scanning Confocal Microscopy
Arabidopsis-Cells-CAM (Willis et al., 2016; Refahi et al., 2021)	Cells in the shoot apical meristem of 6 NPA-treated Arabidopsis plants, labeled membrane	Variable resolution	uint8	Laser Scanning Confocal Microscopy
Platynereis-Nuclei-CBG (Lalit et al., 2021)	Nuclei of a developing *Platynereis dumerilii* embryo at stages between 0 to 16 hours post fertilization, injected with a fluorescent nuclear tracer	(2.031, 0.406, 0.406)	uint16	Simultaneous Multi-view Light-Sheet Microscopy
Paryhale-Nuclei-IGFL (Alwes et al., 2016; Weigert et al., 2020)	Nuclei from the regenerating legs of a developing *Paryhale hawaiensis* embryo expressing Histone-EGFP from the PhHS>H2B-EGFP transgene,	(2.1605, 0.2768, 0.2768)	uint8	cLSM Zeiss 780 inverted confocal microscope
Mouse-Skull-Nuclei-CBG (Lalit et al., 2021)	Nuclei of the skull region of developing mouse embryos, labeled with DAPI	(0.200, 0.073, 0.073)	uint16	Inverted Zeiss LSM 880 Microscope
Platynereis-ISH-Nuclei-CBG (Lalit et al., 2021)	Nuclei of whole-mount *Platynereis dumerilli* specimens at stage of 16 hours post fertilization, labeled with DAPI	(0.4501, 0.4499, 0.4499)	uint8	Laser Scanning Confocal Microscopy

We measure the performance of EmbedSeg on a total of four 2D and seven 3D datasets against several state-of-the-art baseline methods that have most been developed in the context of biomedical image segmentation.

2.7.1 Used 2D Data

For 2D images, we tested all baseline methods introduced below on four publicly available datasets, namely the *BBBC010 C. elegans* brightfield dataset (Ljosa et al., 2012)[2], the *Usiigaci* NIH/3T3 phase-contrast dataset (Tsai et al., 2019), the *DSB* data from the Kaggle Data Science Bowl challenge of 2018 (Caicedo et al., 2019)[3] and the *MoNuSeg* dataset (Kumar et al., 2020).

2.7.2 Used and Contributed 3D Data

We tested the baseline methods mentioned below on seven 3D datasets, *i.e.* the *Mouse-Organoid-Cells-CBG*, *Platynereis-Nuclei-CBG*, *Mouse-Skull-Nuclei-CBG*, and *Platynereis-ISH-Nuclei-CBG* datasets which we have prepared using Labkit (Arzt et al., 2021) and made available online (Lalit et al., 2021), and the publicly available *C. elegans-Cells-HK* dataset (Cao et al., 2020), *Arabidopsis-Cells-CAM* dataset (Willis et al., 2016; Refahi et al., 2021) and the *Paryhale*-Nuclei-IGFL dataset (Alwes et al., 2016) which is available upon request to the authors of the corresponding publication.

Additional details about the used 3D datasets can be found in Table 2.1.

2.7.3 Baseline Methods

For 2D images, we compare the performance of EmbedSeg against several baseline methods, *i.e.* the methods described in Ronneberger et al. (2015); Stringer et al. (2021); Hirsch et al. (2020); Schmidt et al. (2018); Kulikov and Lempitsky (2020) and He et al. (2017).

Cellpose, in addition to providing a 2D public model, can also be trained from scratch. Hence whenever possible, we report numbers from evaluating on both the Cellpose (public) and the Cellpose model specifically trained by ourselves on that respective dataset.

For the *BBBC010 C. elegans* brightfield dataset (Ljosa et al., 2012), we flatten the one-hot encoded instance masks in order to be compatible with the available Cellpose and StarDist training code. This means that areas where object instances overlap now need to commit to one of the overlapping labels. While training

[2] We used the *C. elegans* infection live/dead image set version 1 provided by Fred Ausubel and available from the Broad Bioimage Benchmark Collection

[3] We used a subset of the image set BBBC038v1 which is available from the Broad Bioimage Benchmark Collection. The frequently used subset is provided for download from the StarDist (Schmidt et al., 2018) github repository at `https://github.com/stardist/stardist/releases/download/0.1.0/dsb2018.zip`

Cellpose, we used the default settings (*diameter* equal to 30, *n_epochs* equal to 500, *batch_size* equal to 8). With StarDist (Schmidt et al., 2018), we used the default settings of *n_rays* equal to 32, *grid* equal to $(2, 2)$, *train_batch_size* equal to 4, *train_patch_size* equal to $(256, 256)$ and *train_epochs* equal to 400. With StarDist-3D (Weigert et al., 2020), we used *n_rays* equal to 96, *grid* equal to $(2, 2, 2)$, *train_batch_size* equal to 2, *train_patch_size* equal to $(48, 96, 96)$ and *train_epochs* equal to 400.

For 3D images, we compare the performance of EmbedSeg when directly applied to the volumetric data and EmbedSeg (*sliced*), where axis-parallel 2D predictions are merged (see Section 2.7.6 below for details), against CellPose (Stringer et al., 2021) and StarDist-3D (Weigert et al., 2020). We do this because Cellpose is itself using a sliced inference mode for segmenting 3D data, while StarDist-3D is directly applied to the 3D image data.

2.7.4 Data Handling in 2D

The *BBBC010 dataset* consist of only 100 images of size 696×520 pixels each. Like others before us, we randomly split these images in two equally sized sets, one used for training, the other to evaluate the achieved instance segmentation performance (testing). We cropped 256×256 patches that are centered around each ground truth object (worm) and have used 15% of all crops as validation set. Reported results are averages over 9 independent data-splits and training runs.

For the *Usiigaci dataset*, we split the available 50 images of size 1024×1022 pixels as suggested by Tsai et al. (2019) in 45 training and 5 test images. We cropped 512×512 patches that are centered on all ground truth objects.

The *DSB dataset* is the largest collection of images, of which we use the same subset as originally suggested in Schmidt et al. (2018). It contains a total of 497 images of variable size and is pre-split in 447 training and 50 test images. Training was performed on object-centered 256×256 image crops.

The *MoNuSeg dataset* consists of a total of 44 images of size 1000×1000 pixels, and is pre-split in 30 training images (containing 22000 nuclei) and 14 test images (containing 7000 nuclei). This dataset was originally obtained by annotating tissue images of several patients with tumors of different organs, diagnosed and imaged at multiple hospitals, making the dataset quite heterogeneous and therefore challenging. Also here, we train on object-centered 256×256 image crops.

For the DSB, Usiigaci and MoNuSeg datasets, we hold out 15% of all train-

ing images (prior to cropping training patches), chosen at random for validation purposes. Reported numbers are averages over 9 independent training runs. Results from these independent runs are available at `https://github.com/juglab/EmbedSeg/wiki`. Separately, we calculate the crop size from the available label images during training in the provided example notebooks and this leads to similar results as presented in Table 2.2.

2.7.5 Data Handling in 3D

The *Mouse-Organoid-Cells-CBG dataset* consists of 108 volumes of $70 \times 378 \times 401$ (Z, Y, X) voxels each. We randomly select 15 and 11 images for validation and testing, respectively. Training is performed on object-centered crops of size $32 \times 200 \times 200$.

The *C. elegans-Cells-HK dataset* (Cao et al., 2020) contains 54 training images and 21 test images of $131 \times 285 \times 205$ voxels each. We randomly select 8 images for validation from the training set, downsample both training and validation images by a factor of 2 along X, Y and Z axes and train on object-centered crops of size $64 \times 80 \times 80$. During inference, we predict on similarly downsampled evaluation images and then upsample the predicted instance segmentations using nearest neighbor interpolation.

The *Arabidopsis-Cells-CAM* dataset is the largest collection of 3D images, comprising of 125 training images provided by Willis et al. (2016) and 10 test images provided by Refahi et al. (2021). These images correspond to multiple meristems, and have variable sizes and voxel resolutions. We randomly select 19 images for validation from the training set. Training is performed using object-centered crops of size $80 \times 80 \times 80$.

The *Platynereis-Nuclei-CBG dataset* contains 9 images ($113 \times 660 \times 700$ voxels each), of which we randomly select 2 and 2 images for validation and testing, respectively. Training is performed on object-centered crops of size $32 \times 136 \times 136$.

The *Paryhale*-Nuclei-IGFL data (Alwes et al., 2016) was originally used to demonstrate the performance of Stardist-3D (Weigert et al., 2020). It contains a total of 6 images of $34 \times 512 \times 512$ (Z, Y, X) voxels each. We train on object-centered $24 \times 120 \times 120$ crops. We randomly put aside 1 image for evaluation, and then hold out 2 training images chosen at random for testing.

The *Mouse-Skull-Nuclei-CBG dataset* contains only 2 images of $209 \times 512 \times 512$ and $125 \times 512 \times 512$ voxels respectively. Due to this very limited amount of available data, we test on the sub-volume $(:, :, 256{:}512)$ of the second image.

Training is performed on the remaining data using object-centered crops of size $96 \times 128 \times 128$.

The *Platynereis-ISH-Nuclei-CBG dataset* also contains only 2 images of size $515 \times 648 \times 648$ voxels. We test the performance on the the sub-volume (300:405, : , :) of the second image and train on object-centered crops of size $80 \times 80 \times 80$ on the remaining data.

For all 3D datasets, we report the average results on the test data over 3 independent runs. Results from these independent runs are available at `https://github.com/juglab/EmbedSeg/wiki`. Separately, we calculate the crop size from the available label images during training in the provided example notebooks and this leads to similar results as presented in Table 2.3.

2.7.6 Training Details for EmbedSeg & Neven *et al*

All results obtained with EmbedSeg and the method by Neven *et al*. on 2D datasets use the Branched ERF-Net (Romera et al., 2018; Neven et al., 2019) architecture, the Adam optimizer (Kingma and Ba, 2014) with a decaying learning rate $\alpha_i = 5e^{-4}\left[1 - \frac{i}{200}\right]^{0.9}$, where i denotes the current epoch.

For all training runs, we set w_{IoU}, w_{var} and w_{seed} (Eq. 2.2) equal to 1, 10, and 1 respectively.

For training and inference on 3D datasets, we employed the Branched ERF-Net operating with 3D convolutions, as previously introduced in Lalit et al. (2021).

During training, axis-aligned rotations and flips were used for augmenting the available data.

Every training was run for 200 epochs, and the model with the best performance w.r.t. IoU on the validation data was later used for reporting results on the evaluation data (find all results in Tables 2.2 and 2.3).

2.7.7 Performance Evaluations

All results on 2D images are compared using the Mean Average Precision (AP_{dsb} score (Schmidt et al., 2018), at IoU thresholds ranging from 0.5 to 0.9 (see Table 2.2), while the results on volumetric images are evaluated on at IoU thresholds ranging from 0.1 to 0.9 (see Table 2.3).

For all EmbedSeg and Neven *et al*. results, we compute the minimum object size in terms of the number of interior pixels using the available training and

Table 2.2: **Quantitative evaluation on four 2D datasets.** For each dataset, we compare results of multiple baselines (rows) to results obtained with our proposed pipeline highlighted in gray. The columns show the Mean Average Precision (AP_{dsb}) for selected IoU thresholds. Best and second best performing methods per column are indicated in bold and underlined, respectively.

	$AP_{0.50}$	$AP_{0.55}$	$AP_{0.60}$	$AP_{0.65}$	$AP_{0.70}$	$AP_{0.75}$	$AP_{0.80}$	$AP_{0.85}$	$AP_{0.90}$
					BBBC010				
3-Class U-Net	0.521	0.466	0.451	0.440	0.427	0.407	0.377	0.332	0.243
Cellpose (*public*)	0.225	0.204	0.184	0.155	0.097	0.043	0.013	0.002	0.000
Cellpose (*BBBC010*)	0.874	0.859	0.842	0.822	0.787	0.744	0.674	0.552	0.310
Harmonic Emb.	0.900					0.723			
PatchPerPix	0.930		0.905		0.879		**0.792**		**0.386**
StarDist	0.518	0.408	0.258	0.145	0.044	0.004	0.001	0.000	0.000
Neven *et al.*	0.953	0.941	0.927	0.904	0.878	0.830	0.731	0.563	0.297
EmbedSeg	**0.965**	**0.954**	**0.934**	**0.917**	**0.896**	**0.854**	0.762	**0.596**	0.326
					Usiigaci				
3-Class U-Net	0.245	0.188	0.133	0.090	0.049	0.016	0.008	0.000	0.000
Cellpose (*public*)	0.291	0.237	0.169	0.128	0.066	0.031	0.010	0.000	0.000
Cellpose (*Usiigaci*)	**0.704**	0.600	0.499	0.370	0.258	0.138	0.040	0.005	0.000
Mask R-CNN	0.583	0.520	0.439	0.365	0.235	0.130	**0.045**	**0.008**	0.000
StarDist	0.510	0.427	0.337	0.235	0.143	0.076	0.019	0.002	0.000
Neven *et al.*	0.648	0.570	0.463	0.343	0.233	0.115	0.035	0.004	0.000
EmbedSeg	**0.704**	**0.643**	**0.535**	**0.414**	**0.273**	**0.140**	0.044	0.005	0.000
					DSB				
3-Class U-Net	0.806	0.775	0.743	0.701	0.654	0.578	0.491	0.374	0.226
Cellpose (*public*)	0.868	0.852	0.829	0.802	0.755	0.676	0.563	0.418	0.234
Cellpose (*DSB*)	0.853	0.826	0.812	0.792	**0.768**	**0.716**	**0.645**	**0.536**	**0.402**
Mask R-CNN	0.832	0.805	0.773	0.730	0.684	0.597	0.489	0.353	0.189
PatchPerPix	0.868		0.827		0.755		0.635		0.379
StarDist	0.864	0.836	0.804	0.755	0.685	0.586	0.450	0.287	0.119
Neven *et al.*	0.873	0.852	0.830	0.799	0.762	0.704	0.623	0.511	0.373
EmbedSeg	**0.876**	**0.858**	**0.834**	**0.806**	**0.768**	0.715	**0.645**	0.530	0.399
					MoNuSeg				
Cellpose (public)	**0.757**	**0.725**	0.678	0.610	0.523	0.390	0.222	0.079	0.008
Cellpose (*MoNuSeg*)	0.726	0.695	0.651	0.597	0.517	0.405	0.256	0.106	0.016
StarDist	0.745	0.709	0.658	0.590	0.491	0.376	0.225	0.088	0.013
Neven *et al.*	0.704	0.686	0.661	0.618	0.546	0.431	0.274	0.111	0.016
EmbedSeg	0.717	0.701	**0.679**	**0.636**	**0.567**	**0.453**	**0.294**	**0.119**	**0.019**

Table 2.3: **Quantitative Evaluation on seven 3D datasets.** For each dataset, we compare results of multiple baselines (rows) to results obtained with our proposed pipeline (EmbedSeg) highlighted in gray. The columns show the Mean Average Precision (AP_{dsb}) for selected IoU thresholds. Best and second best performing methods per column are indicated in bold and underlined, respectively.

	$AP_{0.1}$	$AP_{0.2}$	$AP_{0.3}$	$AP_{0.4}$	$AP_{0.5}$	$AP_{0.6}$	$AP_{0.7}$	$AP_{0.8}$	$AP_{0.9}$
Mouse-Organoid-Cells-CBG									
Cellpose	0.217	0.214	0.212	0.210	0.203	0.197	0.183	0.146	0.042
StarDist-3D	**0.988**	**0.982**	**0.982**	**0.982**	**0.973**	0.970	0.958	0.774	0.052
EmbedSeg (*Full 3D*)	**0.988**	**0.982**	**0.982**	**0.982**	**0.973**	**0.973**	**0.973**	**0.970**	**0.929**
C. elegans-Cells-HK									
Cellpose	0.745	0.742	0.740	0.730	0.702	0.673	0.609	0.422	0.013
StarDist-3D	0.959	0.959	0.959	0.959	0.954	0.939	0.881	0.447	0.000
EmbedSeg (*Full 3D*)	**0.981**	**0.981**	**0.981**	**0.980**	**0.965**	**0.958**	**0.905**	**0.606**	**0.017**
Arabidopsis-Cells-CAM									
Cellpose	0.266	0.256	0.248	0.241	0.229	0.214	0.196	0.157	0.060
EmbedSeg (*Full 3D*)	**0.685**	**0.672**	**0.661**	**0.646**	**0.617**	**0.584**	**0.535**	**0.398**	**0.100**
Platynereis-Nuclei-CBG									
Cellpose	0.971	0.971	0.966	0.957	0.931	0.872	0.700	0.299	**0.009**
StarDist-3D	0.973	0.969	0.966	0.966	0.937	0.910	0.736	0.246	0.002
EmbedSeg (*Full 3D*)	**0.982**	**0.982**	**0.982**	**0.975**	**0.964**	**0.932**	**0.804**	**0.361**	0.004
Paryhale-Nuclei-IGFL									
U-Net	0.592	0.552	0.481	0.372	0.280	0.198	0.097	0.010	0.000
Cellpose	0.545	0.498	0.456	0.384	0.285	0.154	0.040	0.006	0.000
StarDist-3D	**0.766**	**0.757**	**0.741**	**0.698**	**0.593**	**0.443**	**0.224**	**0.038**	0.000
EmbedSeg	0.581	0.581	0.579	0.543	0.472	0.359	0.185	**0.038**	0.000
Mouse-Skull-Nuclei-CBG									
Cellpose	0.613	0.587	0.587	0.563	0.515	0.471	0.389	0.316	0.064
StarDist-3D	0.468	0.468	0.400	0.358	0.264	0.138	0.034	0.000	0.000
EmbedSeg (*Sliced*)	0.649	0.649	0.649	0.649	0.424	0.424	0.237	0.146	**0.108**
EmbedSeg (*Full 3D*)	**0.844**	**0.844**	**0.844**	**0.844**	**0.750**	**0.662**	**0.573**	**0.341**	0.044
Platynereis-ISH-Nuclei-CBG									
Cellpose	0.731	0.674	0.629	0.554	0.493	0.390	0.247	0.038	0.000
StarDist-3D	0.599	0.587	0.545	0.442	0.280	0.114	0.010	0.000	0.000
EmbedSeg (*Sliced*)	**0.893**	0.872	0.831	0.821	0.745	0.634	**0.509**	**0.175**	0.000
EmbedSeg (*Full 3D*)	0.882	**0.879**	**0.864**	**0.850**	**0.779**	**0.659**	0.507	0.135	0.000

validation masks. We then use this value during inference to avoid spurious false positives.

2.7.8 Ablation Studies

Table 2.4: **Ablation studies.** For the BBBC010, Usiigaci and *Platynereis*-Nuclei-CBG datasets, we show how using the centroid instead of the medoid during training and/or removing test-time augmentation negatively impacts overall performance. Like before, columns show AP_{dsb} (first row) or ΔAP_{dsb} (rows 2-4) at selected IoU thresholds. Individual rows show: results obtained with EmbedSeg; using centroids instead of medoids during training; using medoids but without test-time augmentation; using centroids during training and no test-time augmentation.

AP_{dsb}	$AP_{0.50}$	$AP_{0.55}$	$AP_{0.60}$	$AP_{0.65}$	$AP_{0.70}$	$AP_{0.75}$	$AP_{0.80}$	$AP_{0.85}$	$AP_{0.90}$
BBBC010 (2D)									
EmbedSeg	0.965	0.954	0.934	0.917	0.896	0.854	0.762	0.596	0.326
↳ medoid ⇒ centroid	-0.002	-0.002	-0.000	-0.002	-0.001	-0.004	+0.004	+0.001	+0.003
↳ no test-time augm.	-0.007	-0.008	-0.003	-0.008	-0.014	-0.020	-0.028	-0.033	-0.025
↳ both (=Neven *et al.*)	-0.011	-0.013	-0.007	-0.012	-0.018	-0.024	-0.032	-0.033	-0.029
Usiigaci (2D)									
EmbedSeg	0.704	0.643	0.535	0.414	0.273	0.140	0.044	0.005	0.000
↳ medoid ⇒ centroid	-0.014	-0.013	-0.006	-0.005	+0.006	+0.009	+0.002	-0.001	0.000
↳ no test-time augm.	-0.028	-0.048	-0.050	-0.052	-0.040	-0.030	-0.008	-0.001	0.000
↳ both (=Neven *et al.*)	-0.038	-0.053	-0.053	-0.055	-0.028	-0.020	-0.006	0.000	0.000
AP_{dsb}	$AP_{0.10}$	$AP_{0.20}$	$AP_{0.30}$	$AP_{0.40}$	$AP_{0.50}$	$AP_{0.60}$	$AP_{0.70}$	$AP_{0.80}$	$AP_{0.90}$
Platynereis-Nuclei-CBG (3D)									
EmbedSeg	0.982	0.982	0.982	0.975	0.964	0.932	0.804	0.361	0.004
↳ medoid ⇒ centroid	-0.006	-0.006	-0.008	-0.005	-0.010	-0.007	+0.018	+0.026	0.000
↳ no test-time augm.	-0.012	-0.012	-0.012	-0.012	-0.013	-0.019	-0.023	-0.037	-0.003
↳ both	-0.013	-0.014	-0.016	-0.018	-0.022	-0.013	-0.033	-0.019	0.000

In order to evaluate the contribution of (i) using the medoid instead of the centroid in EmbedSeg, and (ii) employing test-time augmentation, we have performed respective ablation studies and report the results on two representative 2D and one representative 3D dataset in Table 2.4.

2.8 Open Code and Online Resources

2.8.1 EmbedSeg napari

To foster widespread use of EmbedSeg, we provide all sources publicly on GitHub. This contains the full source code as well as tutorial-style example training and prediction Jupyter notebooks (see Figure 2.4 (b) and (d), respectively) for use on 2D and 3D image data.

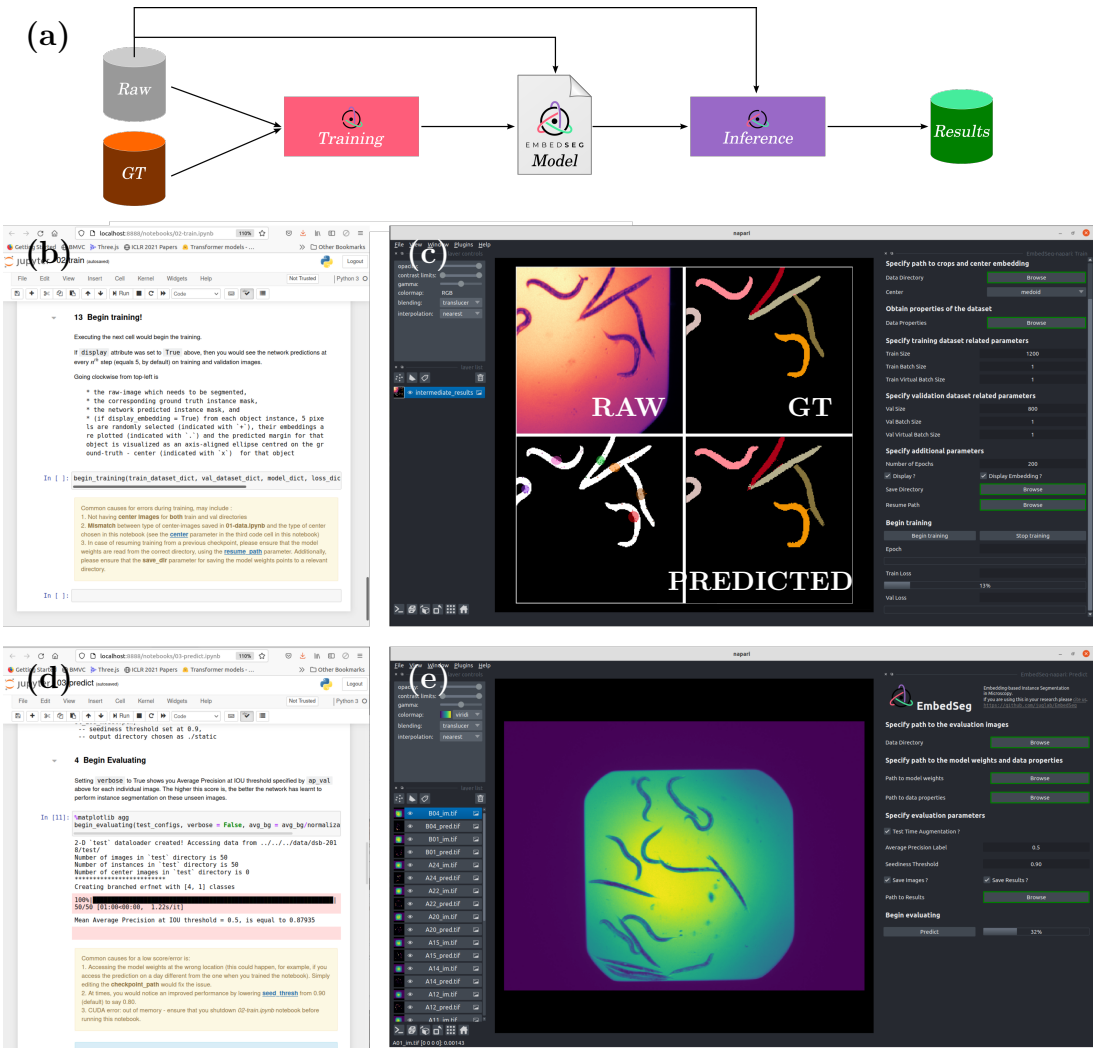

Figure 2.4: **The EmbedSeg training and inference workflow as available via open Jupyter notebooks and napari plugin.** (a) Schematic showing that training and inference are decoupled by the possibility to store trained EmbedSeg models for later use. (b, d) Screenshots of the EmbedSeg training and prediction notebook, respectively. Both being openly available on GitHub. (c, e) Screenshot of the EmbedSeg napari plugin during training and prediction, respectively. The four quadrants in (c) show the raw image tail currently used for training (top left), the corresponding ground truth labels (top right), the current instance predictions (bottom right). In the bottom left quadrant we additionally show, a few randomly-selected pixels, their embedding locations, and the clustering bandwidth (visualized as pluses, dots, and an ellipse, respectively). See Figure 2.1 for a larger view.

Provided notebooks are designed to enable users to (*i*) learn on provided demo training data how an EmbedSeg model can be trained and later used to create instance segmentations on provided demo test data, and (*i*) modify the tutorial notebooks and train a suitable EmbedSeg model on their own data.

Additionally, we created an EmbedSeg plugin for napari (napari contributors, 2019), enabling even users without any programming experience to benefit from our method (see Figure 2.4 (c) and (e) for screenshots of this plugin during training and prediction, respectively). Among its functionality, the plugin offers the possibility to (*i*) stop training and re-starting it from saved checkpoints, (*ii*) live visualization of the model's prediction on training and validation data during training (visualization updates every fifth training step), (*iii*) simply dragging and dropping evaluation images into the napari viewer, (*iv*) fine-tuning pre-trained models (we provide all 11 models trained for this publication) with data provided by the users, and (*v*) training a an EmbedSeg model from scratch on data provided by the users. Taking all this into account, we strongly believe that this plugin is powerful and flexible enough that even seasoned bioimage analysts with extensive programming experience will fall back to the convenience of our napari integrated EmbedSeg UI.

In summary, we believe that the presented method, paired with the openly provided resources we described above, will enable many biomedical scientists to perform high-quality 2D and 3D instance segmentations and are looking forward to provide the necessary support when required.

2.8.2 EmbedSeg Demo

We also created a web-based plugin for providing an easier access to users to the capabilities of EmbedSeg (see Figure 2.5). Our web-based plugin hosts several models previously trained previously by us on diverse 2D and 3D datasets. We provide the capability for users to upload their own image data through a convenient drag-and-drop operation and later check out the performance on their uploaded data, using any of the available pretrained models.

This web-based plugin is intended to provide the user a glimpse of how well pretrained models can work on their data. In case, the instance segmentation results look somewhat convincing, then the user could either download the pretrained model weights and fine-tune those on manually prepared training labels; or instead choose to train a model completely from scratch, but with similar specifications as used for the pretrained model which performed the best in their opinion.

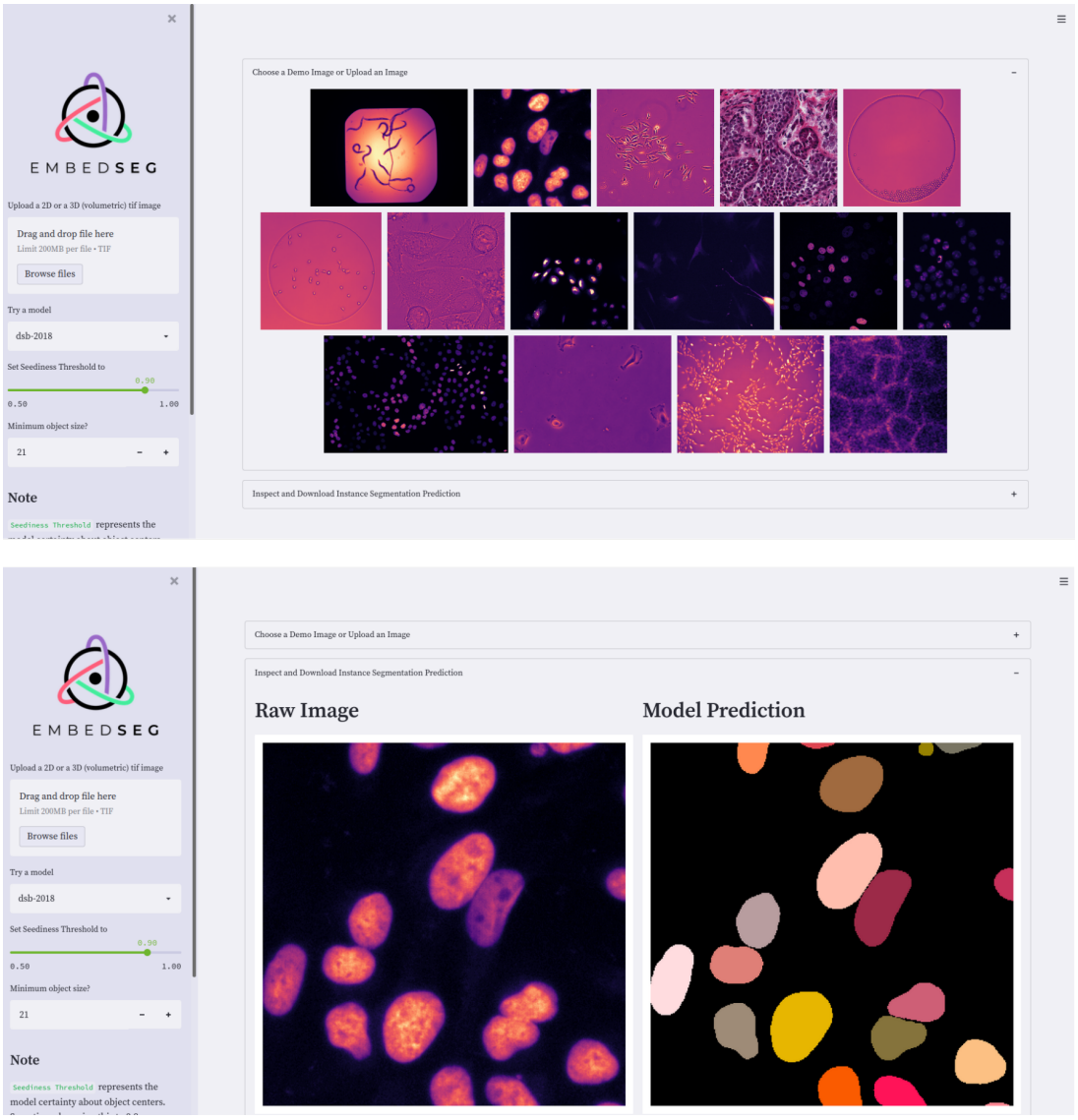

Figure 2.5: **EmbedSeg web-based demo for 2D and 3D inference.** The web-based tool allows selecting a demo image (top) or dragging and dropping in a 2D or a 3D image (bottom), choosing a previously trained model, inspecting the instance segmentation prediction, dynamically updating the model prediction by changing parameters such as seediness threshold (left panel) and finally downloading the instance segmentation prediction. Available at `https://share.streamlit.io/mlbyml/embedseg_demo/main/main.py`.

2.9 Discussion

EmbedSeg is an embedding-based instance segmentation method for 2D and 3D biomedical image data.

The unmodified[4] embedding-based method by Neven *et al.* shows promising

[4] Small adaptions of the code by Neven *et al.* (Neven et al., 2019) are required in order to

results on 2D microscopy data, but the modifications we propose (medoid embedding, test-time augmentation, extension to 3D, hyper-parameters deduced from training data, one-hot encoded masks, *etc.*) secure EmbedSeg's state-of-the-art results on many practically relevant biomedical microscopy datasets.

When comparing the results of EmbedSeg to all obtained baseline predictions, we noticed that Cellpose often runs into issues for 3D volumes which are downsampled along one of the axes. For example, Cellpose results on the Mouse-Organoid-Cells-CBG dataset produce spurious over-segmentations, which we believe are a side-effect of Cellpose's interpolation of individual 2D predictions of sliced 3D input. A similar trend is in fact observable for EmbedSeg (*Sliced*). While results are good for (near-)isotropic datasets such as *Platynereis*-ISH-Nuclei-CBG, sliced predictions lead to reduced quality results for anisotropic data like the Mouse-Skull-Nuclei-CBG dataset (where voxels in z-direction are downsampled by a factor of about 2.75).

StarDist-3D, by design, runs into trouble when the objects to be segmented are not star-convex (*e.g.* for the Mouse-Skull-Nuclei-CBG or *Platynereis*-ISH-Nuclei-CBG datasets) and EmbedSeg as well as Cellpose lead to favorable results. On datasets that do contain only star-convex objects, *e.g.* labeled cell nuclei in the *Paryhale*-Nuclei-IGFL dataset, StarDist-3D performs extraordinarily well.

Additionally, I noticed that StarDist-3D performance generally drops at higher IoU-thresholds $AP_{\geqslant 0.7}$ (see Tables 2.2 and 2.3). When I analyzed this, we found that the IoU score even for relatively well segmented instances can drop below 0.7, since ground truth object instances typically have smooth, rounded surfaces, while StarDist-3D instances are by definition the convex hull of a given number of vectors (rays) radiating out of a source pixel.

Embedding-based segmentation approaches can segment different parts of the same body even if those do not belong to the same connected component (for example, see qualitative predictions from EmbedSeg on the *BBBC010* dataset in Figure 2.2)

For pixels belonging to regions where multiple object instances overlap, one generally observes a low seediness score. This is caused by the trained model being uncertain about which of the overlapping instances to embed to. By modifying the learning task to map each pixel not to single location but instead to a distribution of spatial embedding locations per pixel, overlapping objects could be addressed in a more principled manner. I intend to implement this in a future version of

deal with non-RGB images, one-hot encoded instance masks *etc.*

EmbedSeg.

An additional and practically quite relevant feature of EmbedSeg is its small memory footprint on the GPU during training and our newly devised tiled predictions during inference. During training, I opt for small batch sizes and gradient accumulation. Additionally it is helpful to use the Branched ERF-Net architectures in 2D and 3D which both use factorized convolutions to reduce the number of trainable parameters (Romera et al., 2018). Together, this enables our users to benefit from EmbedSeg even on cheap laptop hardware.

The usability of EmbedSeg is further enabled by our openly available sources, tutorials, Jupyter notebooks, and the easy to use napari plugin. Hence, I strongly feel that EmbedSeg will lead to faster progress in many biomedical research projects that require high-quality 2D or 3D instance segmentations.

Bibliography

Alwes, F., Enjolras, C., and Averof, M. (2016). Live imaging reveals the progenitors and cell dynamics of limb regeneration. *Elife*.

Arzt, M., Deschamps, J., Schmied, C., Pietzsch, T., Schmidt, D., Haase, R., and Jug, F. (2021). Labkit: Labeling and segmentation toolkit for big image data. *bioRxiv*.

Berman, M., Triki, A. R., and Blaschko, M. B. (2018). The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks.

Brabandere, B. d., Neven, D., and Gool, L. V. (2017). Semantic Instance Segmentation with a Discriminative Loss Function. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Buchholz, T.-O., Prakash, M., Krull, A., and Jug, F. (2020). DenoiSeg: Joint Denoising and Segmentation.

Caicedo, J. C., Goodman, A., Karhohs, K. W., Cimini, B. A., Ackerman, J., Haghighi, M., Heng, C., Becker, T., Doan, M., McQuin, C., et al. (2019). Nucleus segmentation across imaging experiments: the 2018 data science bowl. *Nature methods*, 16(12):1247–1253.

Campello, R. J. G. B., Moulavi, D., Zimek, A., and Sander, J. (2015). Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection. *ACM Trans. Knowl. Discov. Data*, 10(1).

Cao, J., Guan, G., and Ho, V. W. S. e. a. (2020). Establishment of a morphological atlas of the *Caenorhabditis elegans* embryo using deep-learning-based 4D segmentation. *Nature Communications*, 11.

Dietler, N., Minder, M., and Gligorovski, V. e. a. (2020). A convolutional neural network segments yeast microscopy images with high accuracy. *Nature Communications*, 11.

Fukunaga, K. and Hostetler, L. (2006). The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition. *IEEE Trans. Inf. Theor.*, 21(1):32–40.

Gomez, H. F., Dumond, M. S., Hodel, L., Vetter, R., and Iber, D. (2021). 3d cell neighbour dynamics in growing pseudostratified epithelia. *Elife*, 10:e68135.

He, K., Gkioxari, G., Dollar, P., and Girschick, R. (2017). Mask R-CNN. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2018). Mask r-cnn.

Hirsch, P., Mais, L., and Kainmueller, D. (2020). Patchperpix for instance segmentation. *arXiv preprint arXiv:2001.07626*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kist, A. M., Zilker, J., Döllinger, M., and Semmler, M. (2021). Feature-based image registration in structured light endoscopy. In *Medical Imaging with Deep Learning*.

Kulikov, V. and Lempitsky, V. (2020). Instance Segmentation of Biological Images Using Harmonic Embeddings.

Kumar, N., Verma, R., Anand, D., Zhou, Y., Onder, O. F., Tsougenis, E., Chen, H., Heng, P.-A., Li, J., Hu, Z., Wang, Y., Koohbanani, N. A., Jahanifar, M., Tajeddin, N. Z., Gooya, A., Rajpoot, N., Ren, X., Zhou, S., Wang, Q., Shen, D., Yang, C.-K., Weng, C.-H., Yu, W.-H., Yeh, C.-Y., Yang, S., Xu, S., Yeung, P. H., Sun, P., Mahbod, A., Schaefer, G., Ellinger, I., Ecker, R., Smedby, O., Wang, C., Chidester, B., Ton, T.-V., Tran, M.-T., Ma, J., Do, M. N., Graham, S., Vu, Q. D., Kwak, J. T., Gunda, A., Chunduri, R., Hu, C., Zhou, X., Lotfi, D., Safdari, R., Kascenas, A., O'Neil, A., Eschweiler, D., Stegmaier, J., Cui, Y., Yin, B., Chen, K., Tian, X., Gruening, P., Barth, E., Arbel, E., Remer, I., Ben-Dor, A., Sirazitdinova, E., Kohl, M., Braunewell, S., Li, Y., Xie, X., Shen, L., Ma, J., Baksi, K. D., Khan, M. A., Choo, J., Colomer, A., Naranjo, V., Pei, L., Iftekharuddin, K. M., Roy, K., Bhattacharjee, D., Pedraza, A., Bueno, M. G., Devanathan, S., Radhakrishnan, S., Koduganty, P., Wu, Z., Cai, G., Liu, X., Wang, Y., and Sethi, A. (2020). A Multi-Organ Nucleus Segmentation Challenge. *IEEE Transactions on Medical Imaging*, 39(5):1380–1391.

Lalit, M., Handberg-Thorsager, M., Hsieh, Y.-W., Jug, F., and Tomancak, P. (2020). Registration of multi-modal volumetric images by establishing cell correspondence. In Bartoli, A. and Fusiello, A., editors, *Computer Vision – ECCV 2020 Workshops*, pages 458–473, Cham. Springer International Publishing.

Lalit, M., Tomancak, P., and Jug, F. (2021). Embedding-based instance segmen-

tation in microscopy. In Heinrich, M., Dou, Q., de Bruijne, M., Lellmann, J., Schläfer, A., and Ernst, F., editors, *Proceedings of the Fourth Conference on Medical Imaging with Deep Learning*, volume 143 of *Proceedings of Machine Learning Research*, pages 399–415. PMLR.

Lee, K., Lu, R., Luther, K., and Seung, H. S. (2021). Learning and Segmenting Dense Voxel Embeddings for 3D Neuron Reconstruction.

Ljosa, V., Sokolnicki, K. L., and Carpenter, A. E. (2012). Annotated high-throughput microscopy image sets for validation. *Nature Methods*, 9.

Meijering, E. (2012). Cell segmentation: 50 years down the road [life sciences]. *IEEE signal processing magazine*, 29(5):140–145.

napari contributors (2019). napari: a multi-dimensional image viewer for python.

Neven, D., Brabandere, B. D., Proesmans, M., and Gool, L. V. (2019). Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8837–8845.

Newell, A., Huang, Z., and Deng, J. (2017). Associative Embedding: End-to-End Learning for Joint Detection and Grouping. In *Proc. of the Neural Information Processing System (NeurIPS)*.

Novotny, D., Albanie, S., Larlus, D., and Vedaldi, A. (2018). Semi-convolutional Operators for Instance Segmentation. In *ECCV*.

Payer, C., Štern, D., Neff, T., Bischof, H., and Urschler, M. (2018). Instance Segmentation and Tracking with Cosine Embeddings and Recurrent Hourglass Networks. In Frangi, A. F., Schnabel, J. A., Davatzikos, C., Alberola-López, C., and Fichtinger, G., editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, pages 3–11, Cham. Springer International Publishing.

Refahi, Y., Zardilis, A., Michelin, G., Wightman, R., Leggio, B., Legrand, J., Faure, E., Vachez, L., Armezzani, A., Risson, A.-E., Zhao, F., Das, P., Prunet, N., Meyerowitz, E. M., Godin, C., Malandain, G., Jönsson, H., and Traas, J. (2021). A multiscale analysis of early flower development in Arabidopsis provides an integrated view of molecular regulation and growth control. *Developmental Cell*, 56(4):540–556.e8.

Romera, E., Alvarez, J., Bergasa, L., and Arroyo, R. (2018). ERFNet: Efficient Residual Factorized ConvNet for Real-time Semantic Segmentation. *IEEE*

Transactions on Intelligent Transportation Systems, 19.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation.

Scherr, T., Löffler, K., Böhland, M., and Mikut, R. (2020). Cell segmentation and tracking using CNN-based distance predictions and a graph-based matching strategy. *PLOS ONE*, 15(12):e0243219.

Schmidt, U., Weigert, M., Broaddus, C., and Myers, G. (2018). Cell detection with star-convex polygons. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 265–273. Springer.

Stern, T., Streichan, S. J., Shvartsman, S. Y., and Wieschaus, E. F. (2021). Deconstructing gastrulation at the single cell level. *Available at SSRN 3929008*.

Stringer, C., Wang, T., Michaelos, M., and Pachitariu, M. (2021). Cellpose: a generalist algorithm for cellular segmentation. *Nature methods*, 18(1):100–106.

Tsai, H.-F., Gajda, J., Sloan, T. F., Rares, A., and Shen, A. Q. (2019). Usiigaci: Instance-aware cell tracking in stain-free phase contrast microscopy enabled by machine learning. *SoftwareX*, 9:230–237.

Upschulte, E., Harmeling, S., Amunts, K., and Dickscheid, T. (2022). Contour proposal networks for biomedical instance segmentation. *Medical Image Analysis*, page 102371.

Wang, Y. and Solomon, J. M. (2019). Deep closest point: Learning representations for point cloud registration.

Weigert, M., Schmidt, U., Haase, R., Sugawara, K., and Myers, G. (2020). 3D object detection and segmentation in microscopy. In *Conference on Applications of Computer Vision, WACV*.

Willis, L., Refahi, Y., Wightman, R., Landrein, B., Teles, J., Huang, K. C., Meyerowitz, E. M., and Jönsson, H. (2016). Cell size and growth regulation in the Arabidopsis thaliana apical stem cell niche. *Proceedings of the National Academy of Sciences*, 113(51):E8238–E8246.

Wolf, S., Bailoni, A., Pape, C., Rahaman, N., Kreshuk, A., Kothe, U., and Hamprecht, F. A. (2020). The Mutex Watershed and its Objective: Efficient, Parameter-Free Graph Partitioning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–1.

Wolny, A., Cerrone, L., Vijayan, A., Tofanelli, R., Barro, A. V., Louveaux, M., Wenzl, C., Strauss, S., Wilson-Sánchez, D., Lymbouridou, R., Steigleder, S. S., Pape, C., Bailoni, A., Duran-Nebreda, S., Bassel, G. W., Lohmann, J. U., Tsiantis, M., Hamprecht, F. A., Schneitz, K., Maizel, A., and Kreshuk, A. (2020). Accurate and versatile 3D segmentation of plant tissues at cellular resolution. *eLife*, 9:e57613.

Yang, X., Bi, D., Czajkowski, M., Merkel, M., Manning, M. L., and Marchetti, M. C. (2017). Correlating cell shape and cellular stress in motile confluent tissues. *Proceedings of the National Academy of Sciences*, 114(48):12663–12668.

Yu, J. and Blaschko, M. (2015). Learning Submodular Losses with the Lovasz Hinge. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1623–1631, Lille, France. PMLR.

Zeng, T., Wu, B., and Ji, S. (2017). DeepEM3D: approaching human-level performance on 3D anisotropic EM image segmentation. *Bioinformatics*, 33(16):2555–2562.

Tracking

Contents

3.1 Introduction **46**

3.2 Related Work **47**

3.3 Our Approach **49**

3.3.1 PointTrack 49

3.3.2 LineageTracer Training 50

3.3.3 LineageTracer Inference 51

3.4 Visualization **55**

3.5 Discussion **56**

3.1 Introduction

Time-lapse imaging of an embryo allows studying how an organism grows from a few cell stage to become a multi-cellular being. In order to study the development of an embryo systematically, one must monitor the motion and division of all cells. In early works, cells were observed using transmission microscopy and lineage trees were sketched by hand (Sulston et al., 1983). Such manual analysis is time-consuming and hard to reproduce, and automated solutions for tracking are therefore essential. However, automated tracking of cells and nuclei continues to be a challenging problem.

Some of the requirements of such automated solutions include (*i*) detecting an unknown number of cells, (*ii*) handling over (*split*), under (*merged*) and missing-segmentations appropriately, (*iii*) accounting for cell divisions, (*iv*) minimizing incorrect assignment of segmentations to tracklets, since an incorrect assignment upstream leads to a higher cumulative error downstream, *etc.* Despite the presence of many, classical (non-learning based) solutions (Magnusson et al., 2015; Tinevez et al., 2017), results tend to be sometimes incorrect and the time to curate the outputs of these trackers is significant. Recently, new deep learning (DL)-based solutions (Sugawara et al., 2021; Ulicna et al., 2021; Moen et al., 2019; Wen et al., 2021; Malin-Mayor et al., 2021; Ershov et al., 2021) have become available. Due to promising improvements of cell segmentation algorithms, these methods follow the *tracking-by-detection* paradigm [1], where in the first step, nuclei/cells are detected or segmented in evaluation time-lapse data, by using a model trained on clean GT annotations. Next, by considering each detection as a potential target, these methods attempt to assign the generated segmentations the same or different label or id, if they are supposed to belong to the same or different tracklet respectively, while additionally also noting the parent-daughter relationships. This is either done by (*i*) generating partial track segments in a first step and then generating complete tracks from these track segments in a second step (Jaqaman et al., 2008; Tinevez et al., 2017; Ershov et al., 2021; Ulicna et al., 2021), or by (*ii*) performing tracking across a pair of frames (Magnusson et al., 2015; Sugawara et al., 2021).

Some of these methods additionally emphasize learning features between *a pair* of frames - for example, Sugawara et al. (2021); Malin-Mayor et al. (2021) learn the pixel-wise flow between adjacent time frames - and this extra information further benefits in improving tracking accuracy on evaluation data. In this chapter, I present LineageTracer, a 2D and 3D tracker for biological objects, which provides a *different* learning framework to the 2D and 3D tracking problem in the

[1] In tracking-by-detection, *detection* and *tracking* are decoupled, hence a reasonable quality of cell segmentation is needed for good tracking results.

biomedical domain. LineageTracer follows the tracking-by-detection paradigm where segmentations are firstly generated by a using a trained EmbedSeg model, next a Graph Neural Network (GNN) is trained - the task during training is formulated as learning a function (parameterized as the model weights) that maps the set of pixels (voxels) belonging to an object instance to a feature per object instance such that object instances that belong to the same tracklet are closer in this feature space and distant from object instances belonging to other tracklets. I show that learning features per instance segmentation *jointly* across a tracklet is more robust than learning pixel-wise features between a pair of frames. LineageTracer achieves SOTA results on a variety of 2D and 3D biological datasets.

3.2 Related Work

In this chapter, I highlight some of the recently published works addressing tracking of biological data. In *TrackMate* (Jaqaman et al., 2008; Tinevez et al., 2017; Ershov et al., 2021), nuclei are detected as local maxima of a Laplacian of Gaussian convolution response or inferred from a trained instance segmentation model. These detected nuclei are initially linked between consecutive frames in a temporally greedy manner. This produces track segments which are corrected for in a second step to close gaps, split merged nuclei detections and merge over segmented detections.

The first step of linking nuclei detections in *TrackMate* is modified in *Deep Tree* (Ulicna et al., 2021). A convolutional neural network is trained in order to classify each detection as belonging to one of five phases (Interphase, Prometaphase, Metaphase, Anaphase, Apoptosis). Next, a constant velocity Kalman filter model is employed, which uses the output of the appearance (classification) model and the nuclei positions to assemble track segments with no divisions. Similar to *TrackMate*, these track segments are then subject to a globally optimal step which produces the complete lineage tree.

In Kausler et al. (2012), tracking-by-detection is reformulated as a *chain* graph (a mixed directed-undirected probabilistic graphical model), which is capable of handling missing and false detections. The authors argue for including domain-specific knowledge, such as enforcing a biological lower-bound on the duration of the cell cycle, and this information is incorporated in the constraints to exclude biological impossible trackings, while solving an Integer Linear Program (ILP) to globally minimize an energy function for obtaining the maximum a-posteriori (MAP) configuration of indicator variables.

In *KTH-SE* (Magnusson et al., 2015), the authors use a dynamic programming

approach to create an entire lineage tree which is optimal with respect to a probabilistically-motivated scoring function. The authors include event variables associated with the number of cells in each detection, mitotic events, apoptotic events, migrations into the imaged area, and migrations out of the imaged area. They next find tracks that give the largest increase to the scoring function using the Viterbi Algorithm. In order to relax the Markov assumption implicit in the Viterbi Algorithm, the authors employ a strategy which they refer to as *swap operations* which allows editing previously created tracks, thus reducing error propagation.

In *DeepCell*, Moen et al. (2019) argue that constructing a cost function has downsides such as possessing limited accuracy and the time required for engineering *etc.*, and that it is rather better to *learn* the cost function by adopting a supervised approach. The authors approach learning the cost function as a classification task - the model produces unique signatures of detections which encodes for their individual appearance, morphology, motion and neighborhood, and for each pair of detections, these signatures are mapped through additional trainable weights to the final classification probabilities p_{same} (if two detections belong to same tracklet), p_{diff} (if they belong to different tracklets) and $p_{\text{parent-child}}$ (if they share a parent-daughter relationship).

In *Elephant* (Sugawara et al., 2021) and *Linajea* (Malin-Mayor et al., 2021), a movement vector is learnt between a pair of consecutive frames from a time-lapse microscopy recording. Separately, cell positions are learnt by training a model on point annotations. During inference, lineages are built by combining the nuclei detections and movement vector, either through a pair-wise greedy association strategy (Sugawara et al., 2021) or through a globally-constrained optimization formulation (Malin-Mayor et al., 2021).

In *KIT-Sch-GE* (Löffler et al., 2021), the proposed tracking algorithm is able to fix segmentation errors while producing the tracking result. A sparse, directed graph is firstly built, where the segmented objects are represented as nodes. By estimating a motion vector between consecutive frames, edges are added between overlapping segmentations. Next, additional nodes for appearance, disappearance, movement, and mitosis, as well as segmentation errors: missing-, under- and over-segmentation are added at each time point, to the graph. By finding optimal paths through this graph, the segmented objects are linked over time, using *coupled minimum cost flow*. Since at this stage, segmentations can be assigned to more than one predecessor and more than two successors, an *untangling* optimization step is employed in order to obtain the final lineage tree.

LineageTracer is the most similar to *PointTrack* (Xu et al., 2020), where

by employing contrastive loss, the authors learn a high-dimensional feature vector per object state such that object states arising from the same tracklet are closer in this higher dimensional space and more distant from objects arising from other tracklets. Notable differences in our work from *PointTrack* include considering only the foreground pixels to build a unique signature per object instance, employing a globally constrained optimization procedure to infer the complete lineage tree, incorporating domain-specific knowledge such as minimum cell-cycle length, extension to volumetric images *etc.*

3.3 Our Approach

3.3.1 PointTrack

In *PointTrack* (Xu et al., 2020), the authors learn instance embeddings based on instance segmentations by converting the image representation to an unordered point cloud representation. This approach is inspired by the success of *PointNet* (Qi et al., 2017) where features are aggregated from point clouds. Results were shown on the task of multi object tracking and segmentation (MOTS) in the context of natural images. Such datasets comprise of RGB images containing several instances of multiple classes (pedestrians, cars *etc.*). Hence, each point is represented by data modalities such as RGB-color of the point, relative position of a 2D point or *offset* from the center (x, y) within the instance mask and class of the point, leading to each point arising from the foreground of the object having a 6-dimensional space.

Since above-mentioned data modalities emphasize extracting features from the instance segmentation regardless of the position of the instance segmentation within the image, the global position is encoded into a 64-dimensional positional embedding, following Vaswani et al. (2017), and fed into the GNN model in a separate branch. The GNN model is trained using the margin-based hard triplet loss (Yuan et al., 2019), which, for any anchor, pushes its hard-negative to an L_2 distance greater than the specified magnitude of margin from the hard-positive, in the instance-embedding space.

During inference, the authors compute an instance similarity between the latest embeddings of all active tracks and embeddings from instances in the current frame. Next, they employ the Hungarian Algorithm in order to associate instance to active tracks. Instances are only assigned to an active track if the matching similarity is above a certain threshold. Unassigned instances start new tracks, while active tracks which haven't been matched for a certain number of frames are switched off.

3.3.2 LineageTracer Training

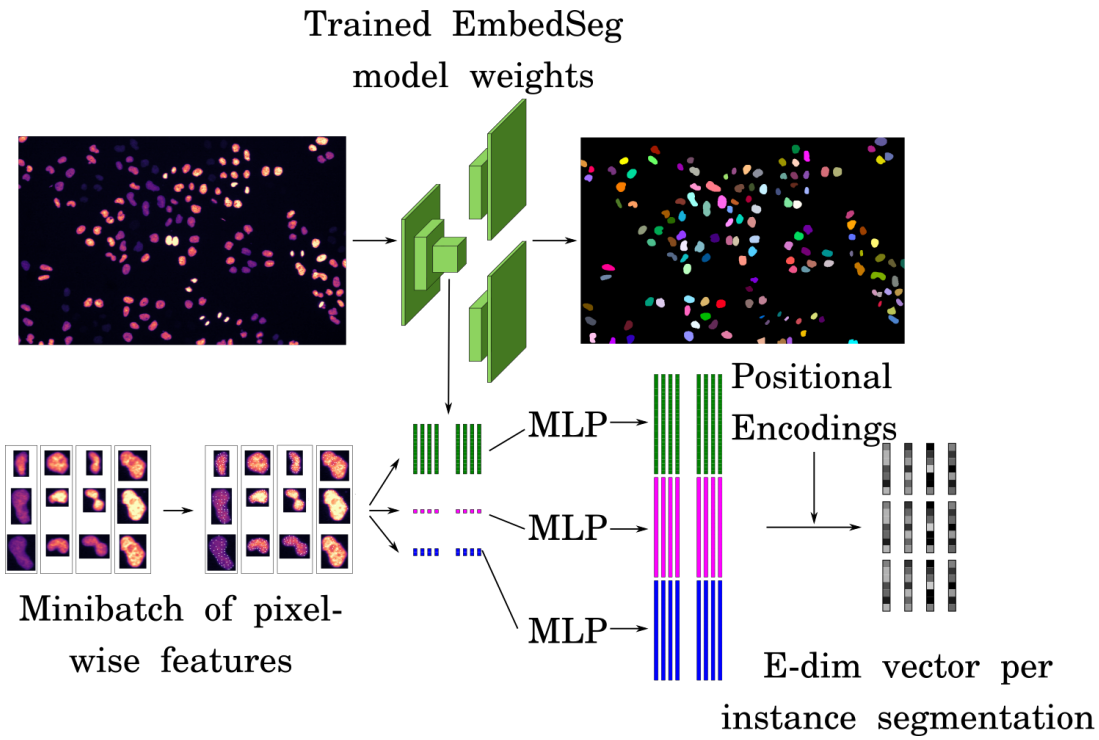

Figure 3.1: **Schematic explaining the LineageTracer flow of information, during training and inference.** During training, a minibatch is constructed from t tracklets by considering o instance segmentations per tracklet and p pixels per instance segmentation. In this schematic figure, the sampled pixels are shown as white dots, $t = 4$ and $o = 3$. Low-level features such as relative pixel position within the instance segmentation, normalized intensity and convolution response from a trained EmbedSeg model at that pixel location, in addition to the global position of the instance segmentation within the image (input to the LineageTracer network as the *positional encodings*) are mapped to an E-dimensional feature per instance segmentation, by using MLPs, batch-normalization, average pooling, max pooling and learnable point-weighting operations.

Images from the bio-medical domain are often gray-scale and not RGB, and show only two classes (foreground and background). Additionally, the features output from the bottleneck layer from a trained EmbedSeg model provide appearance information of the local neighborhood around an image pixel. Hence, to extend *PointTrack* to the bio-medical context, each point is now represented by a 2 (relative position) + 1 (normalized intensity) + 128 (convolutional response from EmbedSeg bottleneck layer) -dimensional feature vector for 2D images and $(3 + 1 + 128)$ -dimensional feature vector for 3D (volumetric) images, similarly.

Using a trained EmbedSeg (Lalit et al., 2021) model, instance segmenta-

tions are inferred in the frames of the time-lapse movie. During the training of the LineageTracer model, a mini-batch of pixel (voxel) features is obtained, by considering t tracklets, o instance segmentations per tracklet and p pixels (voxels) sampled without replacement per instance segmentation. These per-pixel features are concatenated with the global position of the instance segmentation and mapped to an E-dimensional vector per instance segmentation, by updating the model weights using the margin-based hard triplet loss with a margin m. In all our experiments, we use $t = 24$, $o = 3$, $p = 1024$, $E = 32$ and $m = 0.2$.

3.3.3 LineageTracer Inference

The *PointTrack* inference approach does not consider cell divisions as it is designed for natural images. There is also no enforcing of biological priors such as the presence of a minimum cell-cycle length [2].

In LineageTracer, the complete lineage tree in the evaluation time-lapse frames is determined by following three steps: (*i*) Generating track segments (*ii*) Obtaining a tentative lineage tree, and (*iii*) Disentangling the obtained lineage tree to eliminate *split* (over-) segmentations. These are explained in the following sections.

Generating track segments

The L_2 distance between the hard positive and hard negative instance embeddings for any anchor is equal to m, during the training phase. This provides a heuristic approach to extract partial track-segments greedily during inference. Track segments are initialized for all available instance segmentations at time $t = 0$. Next, for any instance segmentation at $t = 0$ (referred to as the *anchor*), the L_2 distance between its instance embedding and the instance embeddings from the segmentations at the next time frame, is computed. If the L_2 distance between the first and second nearest neighbor in the instance embedding space, is more than m, then the track segment is propagated by associating the anchor to its first nearest neighbor.

In situations, where the cell is about to divide, or when there is over-segmentation (such as *splits*) or under-segmentation (such as missing instance segmentations), this above-mentioned condition is no longer true for a given anchor. Additionally, if two separate anchors have the same first nearest neighbor, then their track segments are not propagated since any instance segmentation can only have one parent. In such cases, the active track segment is terminated, and

[2] Cells must pass through interphase, prophase, metaphase, anaphase and telophase. Thus, there is a biological lower bound on the duration of the cell cycle (Kausler et al., 2012).

new track segments are initialized for any unmatched instance segmentations at the subsequent time frame. This procedure is repeated similarly for the next frame by considering all active track segments. Once the entire evaluation time-lapse movie has been processed, the procedure is completed.

This heuristic approach gives us several short track segments. In order to specify the parent-daughter relationship between these track-segments, we proceed next to solving an Integer Linear Program (ILP).

Obtaining a tentative lineage tree

A sparse, directed graph is created by considering each track segment (obtained in the first step) as a node and linking the end of any track segment to track segments starting in the next T frames. Connections are allowed in the next T frames instead of just the next frame in order to handle cases of missing segmentations. The model contains three types of indicator variables: *appearance* variables, *disappearance* variables and *assignment* variables.

A binary appearance variable A_i^t is associated with the ith track segment starting at time frame t, and $\mathcal{A}$ denotes the set of all appearance indicator variables.

A binary disappearance variable D_i^t is associated with the ith track segment terminating at time frame t, and $\mathcal{D}$ denotes the set of all disappearance indicator variables.

Binary assignment variables $E_{ij}^{t_e t_s}$ are associated with the ith track segment terminating at time frame t_e and the jth track segment starting at t_s ($t_e \in [t_s - T, \ldots, t_s - 1]$), and $\mathcal{E}$ denotes the set of all assignment indicator variables.

A few consistency constraints can be imposed by our application towards development biology.

First of all, each track segment must either appear or it must be associated with one predecessor, which implies that the sum of the value of the appearance indicator variable and the sum of the values of the indicator variables for all incoming edges should equal to 1.

$$A_i^t + \sum_{k=1}^{T} \sum_{j} E_{ji}^{t-k,t} = 1. \tag{3.1}$$

Secondly, each track segment must either disappear or be associated with at

most two successors. This is enforced through the two equations below (the second equation is needed to prevent the value of the disappearance indicator variable and sum of the values of indicator variables for the outgoing edges, both being equal to zero).

$$D_i^t + 0.5 \sum_{k=1}^{T} \sum_j E_{ij}^{t,t+k} \leqslant 1, \tag{3.2}$$

$$2D_i^t + \sum_{k=1}^{T} \sum_j E_{ij}^{t,t+k} \geqslant 1. \tag{3.3}$$

.

The global optimization problem concerns solving for

$$\hat{\mathcal{E}}, \hat{\mathcal{A}}, \hat{\mathcal{D}} = \text{argmin } C_{\text{app}}\mathcal{A} + C_{\text{disapp}}\mathcal{D} + C_{\text{edge}}\mathcal{E}, \tag{3.4}$$

where C_{edge} is the scaled L_2 norm between respective instance embeddings in the E-dim space. C_{app} and C_{disapp} are set equivalent to 1.0.

Evidently, over-segmentations are not explicitly modeled in this formulation, and would lead to an undesired solution in certain spatio-temporal regions. Hence, in the third step, we identify tracklets which are shorter than a duration τ (τ is calculated from the available training data and represents the minimum cell cycle length). These identified tracklets are assumed to be faulty. We identify cases of touching instance segmentations in these tracklets. All such groups of instance segmentations which are touching, are assumed to be the erroneous group of segmentations, which should ideally be merged with each other.

But, which edges should be retained as a consequence of this post-processing?

In order to determine this, only the affected variables are re-solved for (while freezing the values of the remaining variables), using another ILP with a different constraints formulation. Details of this *disentangling* are provided in the next section.

Disentangling the obtained lineage tree

Firstly, tracklets which are shorter than a certain duration τ, are identified. Instance segmentations which currently are included in these *suspect* tracklets are checked if they are touching instance segmentations from other tracklets. This is done by dilating the instance segmentations by a pixel and seeing if there is

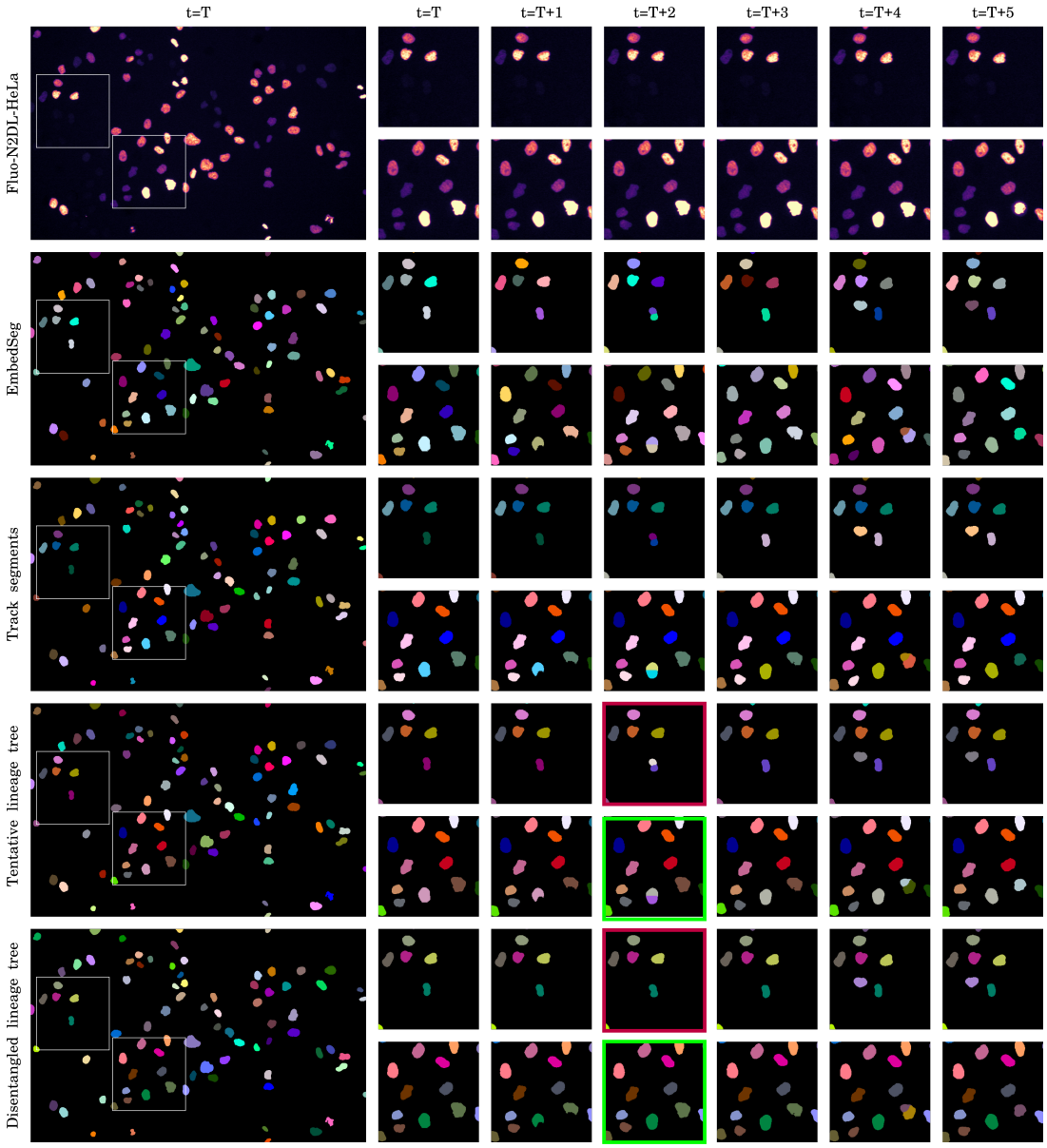

Figure 3.2: **Qualitative results in 2D on the *Fluo-N2DL-HeLa* dataset.** (*i*) A trained EmbedSeg model is used to predict instance segmentations on all time frames independently (second row). (*ii*) Next, each instance segmentation is mapped to a higher dimensional feature using a trained LineageTracer model, which allows extracting non-conflicting track segments (third row). (*iii*) Then an ILP is solved which allows connecting these track segments to form a lineage tree (fourth row). (*iv*) Since over-segmentations were not explicitly modeled in the previous ILP formulation, these are addressed in the disentangling step (final row) by identifying track segments which are below a certain duration and re-solving an ILP with a different set of constraints (see main text). The corresponding green tiles and red tiles in the second-last and last rows indicate that over-segmentations have been merged and highlight the benefit of disentangling the lineage tree.

any area overlap with instance segmentations belonging to other tracklets. If any such instance segmentations are identified, then these and their neighbors are considered to be merging and the indicator variables on these and their neighbors are re-opened for solving. The difference from the previous step is that now for each *set* of merged candidates, the sum of the number of incoming edges should be one or all the merged candidates should *appear*. Similarly, for each set of merged candidates, the sum of the outgoing edges should be at most two or all the merged candidates should *disappear*.

The benefit of this post-processing step can be seen in the second-last and last rows in Figure 3.2 (see the highlighted tiles), where some instance segmentations are detected as being over-segmented in the tentative lineage tree, these are then merged with their neighbors, which enables the underlying lineage tree to be disentangled.

3.4 Visualization

In order to display the results of my computational pipeline, I decided to use *threejs* (`https://threejs.org/`). *threejs* is a light-weight javascript library used to create and display animated 3D computer graphics on a web browser. Since existing visualization platforms such as Fiji (Schindelin et al., 2012), napari (napari contributors, 2019), Amira, Imaris *etc.* do not currently provide the capability to project genetic information onto morphological datasets (Leggio et al., 2019), I decided to benefit from a solution which provides an easy inroad to scripting and adapting based on user-specifications.

A desirable property of this framework is that it allows one to visualize both the meshes (obtained from the instance segmentations) and the raw image data as a texture, which permits the user the possibility to view the underlying gray-scale image data alongside the nuclei meshes, obtained from the live embryo (see Figure 3.3).

Another nice property of this framework is that it supports *blender* (`https://www.blender.org/`) and can import entire scenes prepared in *blender*, as well as export scenes which can be worked upon in *blender*. Lastly, it fosters collaboration between groups working remotely, as it is easy to host the javascript based visualization code and data on a server and share the hyperlink to the corresponding web-page.

In Figure 3.4, results from applying a trained EmbedSeg and LineageTracer model on a time-lapse recording of a live *Platynereis* embryo, are shown. Nuclei meshes are colored based on the lineages they are assigned to,

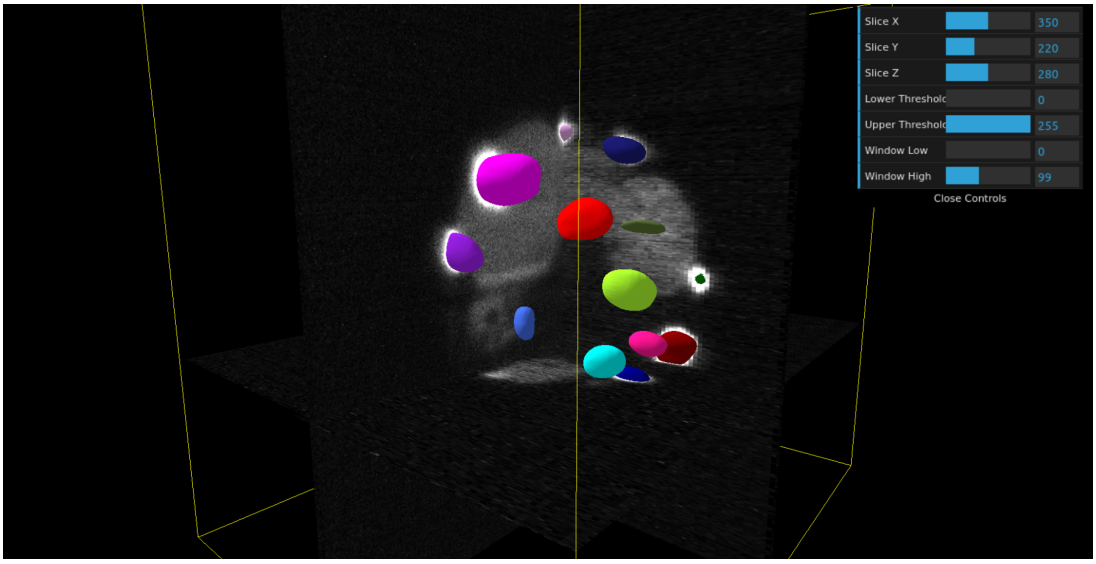

Figure 3.3: **Displaying volumetric image data alongside instance segmentations.** Volumetric image data and corresponding instance segmentations for a given time frame are loaded simultaneously. By changing value of the parameters *Slice X*, *Slice Y* and *Slice Z* in the graphical user interface (top-right), one can selectively display specific YZ, ZX and XY planes in the volume.

following the inference procedure using a trained LineageTracer model (as described in the Section 3.3.3).

3.5 Discussion

In this chapter, I described a new learning approach for the task of tracking, called LineageTracer, which allows learning a function that maps object-level properties (such as global position of an object in an image) and pixel-level properties (such as distance of pixels from their object center, intensity *etc.*) to a high-dimensional feature, suitable for associating instance segmentations across time and inferring a lineage tree. The advantages of such an approach is that the relative contribution of different input properties are implicitly learnt during the training procedure, and this allows a model to output a *context-specific* feature, which adapts to the properties of the dataset at hand. In comparison, other learning-based approaches for the task of tracking, often learn a flow-field between a pair of consecutive frames, while not considering the features of the states of an object constituting a tracklet, jointly. In the future, a more detailed quantitative comparison of these two broad schemes will be performed. Additionally, the relative impact of ignoring low-level, input features (such as intensity, distance to object center *etc.*) systematically will be determined through an ablation study.

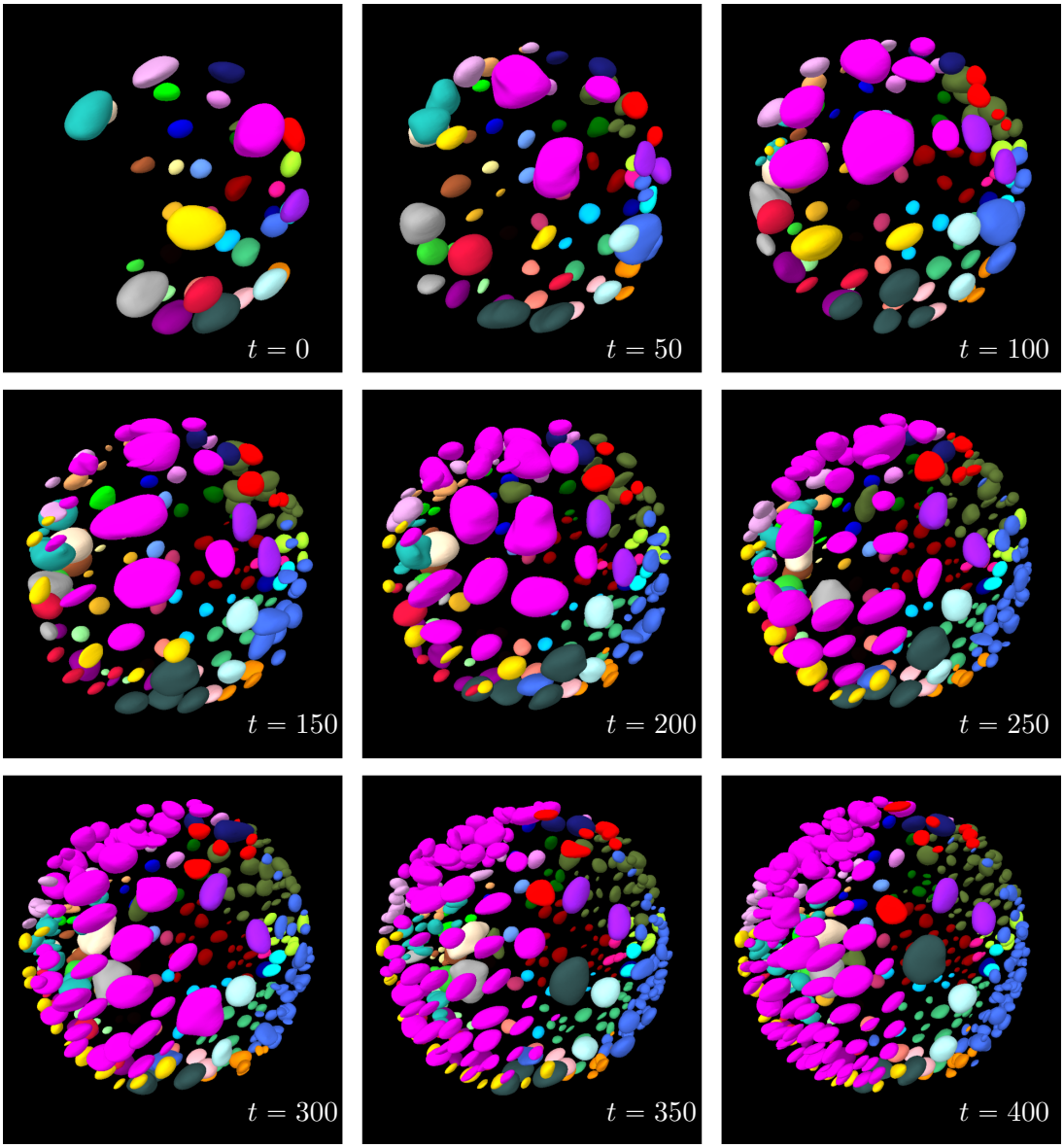

Figure 3.4: **Visualization of instance segmentations of the nuclei in a *Platynereis dumerilii* embryo imaged with SPIM imaging from 5 hpf ($t = 0$) to 15 hpf ($t = 400$). Each of the initial 38 lineages at time frame $t = 0$ are assigned a unique, distinct color.** Instance segmentations are obtained by using a 3D EmbedSeg model, trained on manually-curated GT instance segmentation annotations. These predicted instance segmentations per time frame (each time frame is processed independently) are associated between time frames by using a trained LineageTracer model, followed by manual curation of the predicted lineage tree. Next, these instance segmentations are converted to triangular meshes by executing the *Marching Cubes* algorithm provided with PyVista python package (Sullivan and Kaszynski, 2019). Lastly, these meshes are imported for visualization in the browser, by using a javascript file which sets up the lighting and staging. Such a visualization enables identifying the ancestry of a cell at a given time frame. Additionally, by loading each of the cells (meshes) at a given time frame, with gene expression information, one can calculate the intra- and inter- lineage variance in transcriptional activity.

Bibliography

Ershov, D., Phan, M.-S., Pylvänäinen, J. W., Rigaud, S. U., Le Blanc, L., Charles-Orszag, A., Conway, J. R. W., Laine, R. F., Roy, N. H., Bonazzi, D., Duménil, G., Jacquemet, G., and Tinevez, J.-Y. (2021). Bringing trackmate in the era of machine-learning and deep-learning. *bioRxiv*.

Jaqaman, K., Loerke, D., and Mettlen, M. e. a. (2008). Robust single-particle tracking in live-cell time-lapse sequences. *Nat Methods*, 5.

Kausler, B. X., Schiegg, M., Andres, B., Lindner, M., Koethe, U., Leitte, H., Wittbrodt, J., Hufnagel, L., and Hamprecht, F. A. (2012). A discrete chain graph model for 3d+t cell tracking with high misdetection robustness. In Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., and Schmid, C., editors, *Computer Vision – ECCV 2012*, pages 144–157, Berlin, Heidelberg. Springer Berlin Heidelberg.

Lalit, M., Tomancak, P., and Jug, F. (2021). Embedding-based instance segmentation in microscopy. In Heinrich, M., Dou, Q., de Bruijne, M., Lellmann, J., Schläfer, A., and Ernst, F., editors, *Proceedings of the Fourth Conference on Medical Imaging with Deep Learning*, volume 143 of *Proceedings of Machine Learning Research*, pages 399–415. PMLR.

Leggio, B., Laussu, J., Carlier, A., Godin, C., Lemaire, P., and Faure, E. (2019). Morphonet: an interactive online morphological browser to explore complex multi-scale data. *Nature communications*, 10(1):1–8.

Löffler, K., Scherr, T., and Mikut, R. (2021). A graph-based cell tracking algorithm with few manually tunable parameters and automated segmentation error correction. *PLOS ONE*, 16(9):1–28.

Magnusson, K. E. G., Jaldén, J., Gilbert, P. M., and Blau, H. M. (2015). Global linking of cell tracks using the viterbi algorithm. *IEEE Transactions on Medical Imaging*, 34(4):911–929.

Malin-Mayor, C., Hirsch, P., Guignard, L., McDole, K., Wan, Y., Lemon, W. C., Keller, P. J., Preibisch, S., and Funke, J. (2021). Automated reconstruction of whole-embryo cell lineages by learning from sparse annotations. *bioRxiv*.

Moen, E., Borba, E., Miller, G., Schwartz, M., Bannon, D., Koe, N., Camplisson, I., Kyme, D., Pavelchek, C., Price, T., Kudo, T., Pao, E., Graf, W., and Van Valen,

D. (2019). Accurate cell tracking and lineage construction in live-cell imaging experiments with deep learning. *bioRxiv*.

napari contributors (2019). napari: a multi-dimensional image viewer for python.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation.

Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., Preibisch, S., Rueden, C., Saalfeld, S., Schmid, B., et al. (2012). Fiji: an open-source platform for biological-image analysis. *Nature methods*, 9(7):676–682.

Sugawara, K., Cevrim, C., and Averof, M. (2021). Tracking cell lineages in 3d by incremental deep learning. *bioRxiv*.

Sullivan, C. and Kaszynski, A. (2019). Pyvista: 3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (vtk). *Journal of Open Source Software*, 4(37):1450.

Sulston, J. E., Schierenberg, E., White, J. G., and Thomson, J. N. (1983). The embryonic cell lineage of the nematode caenorhabditis elegans. *Developmental biology*, 100(1):64–119.

Tinevez, J.-Y., Perry, N., Schindelin, J., Hoopes, G. M., Reynolds, G. D., Laplantine, E., Bednarek, S. Y., Shorte, S. L., and Eliceiri, K. W. (2017). Trackmate: An open and extensible platform for single-particle tracking. *Methods*, 115:80–90. Image Processing for Biologists.

Ulicna, K., Vallardi, G., Charras, G., and Lowe, A. R. (2021). Automated deep lineage tree analysis using a bayesian single cell tracking approach. *Frontiers in Computer Science*, 3:92.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

Wen, C., Miura, T., Voleti, V., Yamaguchi, K., Tsutsumi, M., Yamamoto, K., Otomo, K., Fujie, Y., Teramoto, T., Ishihara, T., et al. (2021). 3deecelltracker, a deep learning-based pipeline for segmenting and tracking cells in 3d time lapse images. *Elife*, 10:e59187.

Xu, Z., Zhang, W., Tan, X., Yang, W., Huang, H., Wen, S., Ding, E., and Huang, L. (2020). Segment as points for efficient online multi-object tracking and segmentation. In *Proceedings of the European Conference on Computer Vision*

(ECCV).

Yuan, Y., Chen, W., Yang, Y., and Wang, Z. (2019). In defense of the triplet loss again: Learning robust person re-identification with fast approximated triplet loss and label distillation.

Registration

Contents

4.1 Introduction **64**
4.2 Related Work **65**
4.2.1 *Platynereis dumerilii* 66
4.2.2 Non-Rigid Point Cloud Registration 66
4.2.3 DL-based Medical Imaging 67
4.2.4 DL-based Graph Matching 67
4.3 Our Approach **68**
4.3.1 PlatyMatch 69
4.4 Learning Node Embeddings **73**
4.5 Hidden Markov Models **75**
4.6 Discussion **76**

4.1 Introduction

Obtaining gene expression patterns at the single-cell level enables correlating the gene expression maps with the control circuitry that determines changes in cell morphology and which leads to cell fate specification and differentiation (Luengo-Oroz et al., 2011). However, current techniques such as Fluoroscent *in situ* Hybridization (FISH) allows labeling only a few RNA species at a time. Therefore, in order to construct a spatial gene expression map, one has to pool information from multiple fixed specimens. This is done by obtaining an individual transform for each of these specimens to a canonical space, and this problem we refer to as the task of *Intramodal Registration* (See Figure 4.1 A).

Intramodal Registration provides us static snapshots of gene expression of the embryo at different developmental stages. In order to correlate these static snapshots with the dynamic changes in cell morphology, one has to combine information between these fixed specimens and a fully tracked and segmented developing embryo. This computational problem, we refer to as the task of *Intermodal Registration* (*Inter* as opposed to *Intra*, as often the developing embryo and fixed specimens are imaged through different imaging techniques such as Selective Plane Illumination Microscopy (SPIM) and Confocal Microscopy, respectively. See Figure 4.1 B).

Owing to the recent successes of DL-based instance segmentation techniques, it is possible to determine the locations and morphologies of most cells with high accuracy. This motivates the use of non-intensity based registration approaches as a first step. Such techniques can broadly be considered of two types (i) *point cloud matching* where instance segmentations of nuclei are considered as points carrying features such as x,y,z coordinates, size of the nucleus, average intensity of the nucleus etc and (ii) *graph matching*, where cells or nuclei are considered as nodes of a graph and are linked to each other through directed or undirected edges. Point cloud matching can therefore also be considered as a special case of graph matching, where the source and target graphs lack any edges.

One popular formulation of graph matching leverages the intuition that if a certain node a from graph 1 is matched to another node i from graph 2, then the neighbors of node a must also match to the neighbors of node i (Gold and Rangarajan, 1996). Despite the attractiveness of this formulation, often what constitutes the edges of a graph is not directly evident, say if one considers a volumetric image showing nuclei stained for *DAPI*. Usually this is averted by following a heuristic such as linking each node (nucleus) to its n nearest neighbors or by performing a Delaunay triangulation of the complete cloud of nuclei. However, in the presence of severe local deformation, this could cause difficulties for the

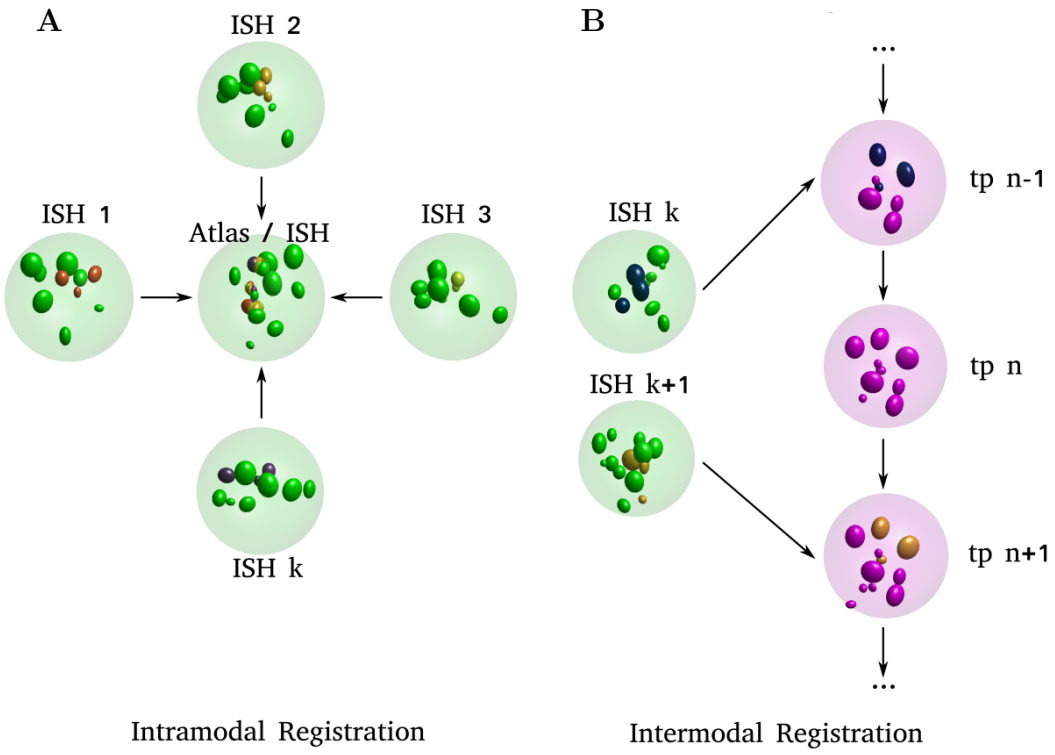

Figure 4.1: ***Intra* and *Inter*modal Registration.** (A, B) 2-D schematics illustrating the two use cases: (A) images of distinct, independent *in-situ* specimens, acquired through confocal microscopy are registered to each other, which enables formation of an average, virtual atlas. (B) images of *in-situ* specimens, acquired through confocal microscopy are registered to the appropriate frame (tp: time point) in a time-lapse movie acquired through SPIM imaging. Nuclei indicated in darker shades are the ones expressing the gene being investigated. In both cases, the information about gene expression is transferred from the source nucleus to the corresponding target nucleus. (Figure modified from Lalit et al. (2020)).

graph matching algorithm. An additional constraint arises due to the computational complexity of the graph matching algorithm. In presence of graphs with several nodes, which is common in microscopy images showing several hundred cells, obtaining the solution becomes intractable.

In this chapter, we will therefore turn our attention towards point-cloud affine registration and intensity-based non-rigid fine-tuning approaches.

4.2 Related Work

In this section, recent works related to registration of *Platynereis dumerilii* are introduced. Next, common baseline methods used in computer vision for point cloud registration are mentioned. Lastly, recent advancements in deep learning for

the tasks of (*i*) non-rigid image registration, from the domain of medical imaging and (*ii*) graph matching, are described.

4.2.1 *Platynereis dumerilii*

Tomer et al. (2010) developed a computational protocol called *PrImR* for registering gene expression images of 48 hours post fertilization (hpf)-old embryos. A global, affine transform is calculated, followed by the estimation of a local, non-rigid transform, by using the intensities on the channel carrying information about the axonal scaffold. The authors used the *Mutual Information* objective to guide the registration algorithm.

Vergara et al. (2017) developed an algorithm called *ProSPr* for generating a gene expression atlas for 6 days post fertilization (dpf)-old embryos. The gene expression atlas is produced by firstly registering the *DAPI* image channel of multiple individuals (using similarity in voxel intensities, like in *PrImR*).

In Lalit et al. (2020), I achieved registration of images of 16 hpf-old embryos. The registration algorithm leveraged the intuition that the nuclei which should be matched between the two queried images, should have similar local and global neighborhoods.

Vergara et al. (2021) registered the 6 dpf-old *ProSPr* gene expression atlas with an EM image stack, using the intensities in the average *ProSPr DAPI* image and a binary mask extracted from performing instance segmentation of the nuclei in the EM image. The authors used *Normalized Mutual Information* objective as a voxel-based similarity measure, with the local motion being described by Free-Form Deformation (FFD) based on cubic B-splines (Rueckert et al., 1999). This pipeline allows combining a static gene expression atlas with the morphology information of cells (with nuclei).

No related work, to my knowledge, has so far addressed fusing static gene expression snapshots with dynamic changes in cell morphology.

4.2.2 Non-Rigid Point Cloud Registration

In Coherent Point Drift (CPD) (Myronenko and Song, 2010), the authors consider the alignment of two point clouds as a probability density estimation problem. The moving point cloud is treated as a cloud of GMM centroids, which moves *coherently* as a group, in order to explain the *data* (the fixed point cloud).

Since it was unclear whether CPD always converges, and additionally to make the approach more robust to fixed point cloud rotations, Hirose (2021) re-

formulated it in a Bayesian setting with Bayesian Coherent Point Drift.

4.2.3 DL-based Medical Imaging

In *VoxelMorph* (Balakrishnan et al., 2019), the authors employ an unsupervised strategy to learn a function that maps a pair of source and target volumetric images (and optionally available respective binary segmentation masks) to a deformation field. This function is parametererized as the weights of a CNN and the training procedure is guided by a loss function set equal to the negative of the local cross correlation between the fixed and the registered moving image patches.

In Czolbe et al. (2021), the authors optimize the weights of the registration network by encouraging alignment of respective semantic features of the fixed and the registered moving image patches. These semantic features are produced using an additional network pretrained for an auxiliary task such as semantic segmentation.

Both *VoxelMorph* and Czolbe et al. (2021) showed best results during evaluation time, when the registration networks were trained with *both* - the pair of raw images, and the corresponding binary segmentation masks. Such methods require the raw image patches during training and evaluation time to be affinely-registered prior to feeding in the registration network, and hence are suitable (only) for estimating the non-rigid, local deformation.

4.2.4 DL-based Graph Matching

Current DL-based approaches map input point clouds to permutation, scale, rotation and translation-invariant embeddings by using different flavors of graph neural networks. Learning such node embeddings for each point (node of a graph) enables identifying matching pairs of points between the moving and fixed point clouds.

The intuition behind these approaches is to find a per-point embedding feature which quotients out the underlying motion between the moving and fixed point clouds, and which remains sensitive to local neighborhoods required for matching. In most of these approaches, in the case that graph edges are not known *a priori* in the datasets which need to be matched, then points are connected by directed edges to their k-nearest neighbors.

In Deep Closest Point (DCP) (Wang and Solomon, 2019), the node embeddings and the underlying transform are learnt jointly by providing ground truth rotation and translation matrices as targets during training. In Deep Graph

Matching Consensus (DGMC) (Fey et al., 2020), the learnt node embeddings are used to obtain a permutation matrix, which is compared with the ground truth permutation matrix, provided as a target during training.

4.3 Our Approach

Given two 3D point sets - the moving point set $\mathcal{X} = \{\mathbf{x}_i\}, i = 1, \ldots, M$ and the fixed point set $\mathcal{Y} = \{\mathbf{y}_j\}, j = 1, \ldots, N$, where $\mathbf{x}_i, \mathbf{y}_j \in \mathbb{R}^3$ are the point coordinates, one would like to estimate the 3×3 affine matrix $\mathbf{A}$ and the translation $\mathbf{t} \in \mathbb{R}^3$, which minimizes the L_2 error E,

$$E(\mathbf{A}, \mathbf{t}) = \sum_{i=1}^{M} e_i(\mathbf{A}, \mathbf{t}) = \sum_{i=1}^{M} \|\mathbf{A}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_{j*}\|^2, \tag{4.1}$$

where $\mathbf{y}_{j*} \in \mathcal{Y}$ is the optimal correspondence of $\mathbf{x}_i$. Provided the optimal correspondence matrix, a closed form solution to Equation 4.1 exists. A key challenge, thus, is to identify the corresponding point matches prior to estimating $\mathbf{A}$ and $\mathbf{t}$.

In Lalit et al. (2020), I leverage the intuition that nuclei segmentations which should be matched between two images of embryos at approximately equivalent developmental states, should have the same local neighborhoods. This, I consider as a valid assumption, especially for mosaically-developing embryos, where the variability in the positions of cells between similarly-aged individuals is low. The calculated local neighborhood signatures should be translation, rotation and scale invariant to allow for matching corresponding nuclei found in different poses in the moving and source images. One example of such geometric descriptors is *Shape Context*, which was introduced by Belongie et al. (2002) for measuring similarity between two-dimensional point clouds. I extend *Shape Context* features to handle a three-dimensional, sparse point cloud derived from nuclei instance segmentations. More details are discussed in the Section 4.3.1.

Another prominent example of geometric descriptor matching inspired by the computer vision work and applied biological image analysis is the bead-based registration of multi-view Selective Plane Illumination Microscopy (SPIM) data (Preibisch et al., 2010). Here, fluorescent beads embedded around a specimen are used as fiduciary markers to achieve registration of 3D scans of the same specimen from multiple imaging angles (referred to as views). This is achieved by building rotation, translation and scale-invariant bead descriptors in local bead neighbourhoods, which enables identification of corresponding beads in multiple views and thus allows image registration and subsequent fusion of the views. The

approach was extended to multi-view registration using nuclei segmented within the specimen instead of beads (Hörl et al., 2019), however the approach is not robust enough to enable registration across different specimen and/or imaging modalities.

4.3.1 PlatyMatch

The core of my method, which I refer to as PlatyMatch, is to match the nuclei in the various imaged specimens by means of building the shape context descriptors in a coordinate frame of reference that is unique to each nucleus. This makes the problem of matching rotationally invariant (see Figure 4.2). The descriptors are then matched in the descriptor space by finding the corresponding closest descriptor in the two specimens and these initial correspondences are pruned by Random Sample Consensus (RANSAC) to achieve an initial guess of the registration. This alignment is next refined by Iterative Closest Point (ICP). At this point, we diverge from the classical approach and evaluate the correspondences through a maximum bipartite matching to achieve the goal of matching every single nucleus from one specimen to a corresponding nucleus in the other. Optionally, after the maximum bipartite matching, the estimated correspondence can be used to non-linearly deform the actual images to achieve a visually more convincing overlap of corresponding nuclei. The individual steps of the pipeline are described in detail in the following subsections.

Finding Corresponding Nuclei between Two Point Clouds

Estimating a Global Affine Transform. In this section, we will provide the details of our implementation of the 3D shape context geometric descriptor, which is a signature obtained uniquely for all feature points in the source and target point clouds. This descriptor takes as input a point cloud P (which represents the nuclei detections described in the previous section) and a basis point p, and captures the regional shape of the scene at p using the distribution of points in a support region surrounding p. The support region is discretized into bins, and a histogram is formed by counting the number of point neighbours falling within each bin. As in Belongie et al. (2002), in order to be more sensitive to nearby points, we use a log-polar coordinate system (see Figure 4.2). In our experiments, we build a 3D histogram with 5 equally spaced log-radius bins and 6 and 12 equally spaced elevation (θ) and azimuth (ϕ) bins respectively.

For each basis point p, we define a unique right-handed coordinate system: the Z-axis is defined by the vector joining the centroid of the point cloud to the point of interest. Then we draw a vector pointing from the queried nucleus detection to the location of a landmark. For the *Platynereis dumerilii* data, we notice that

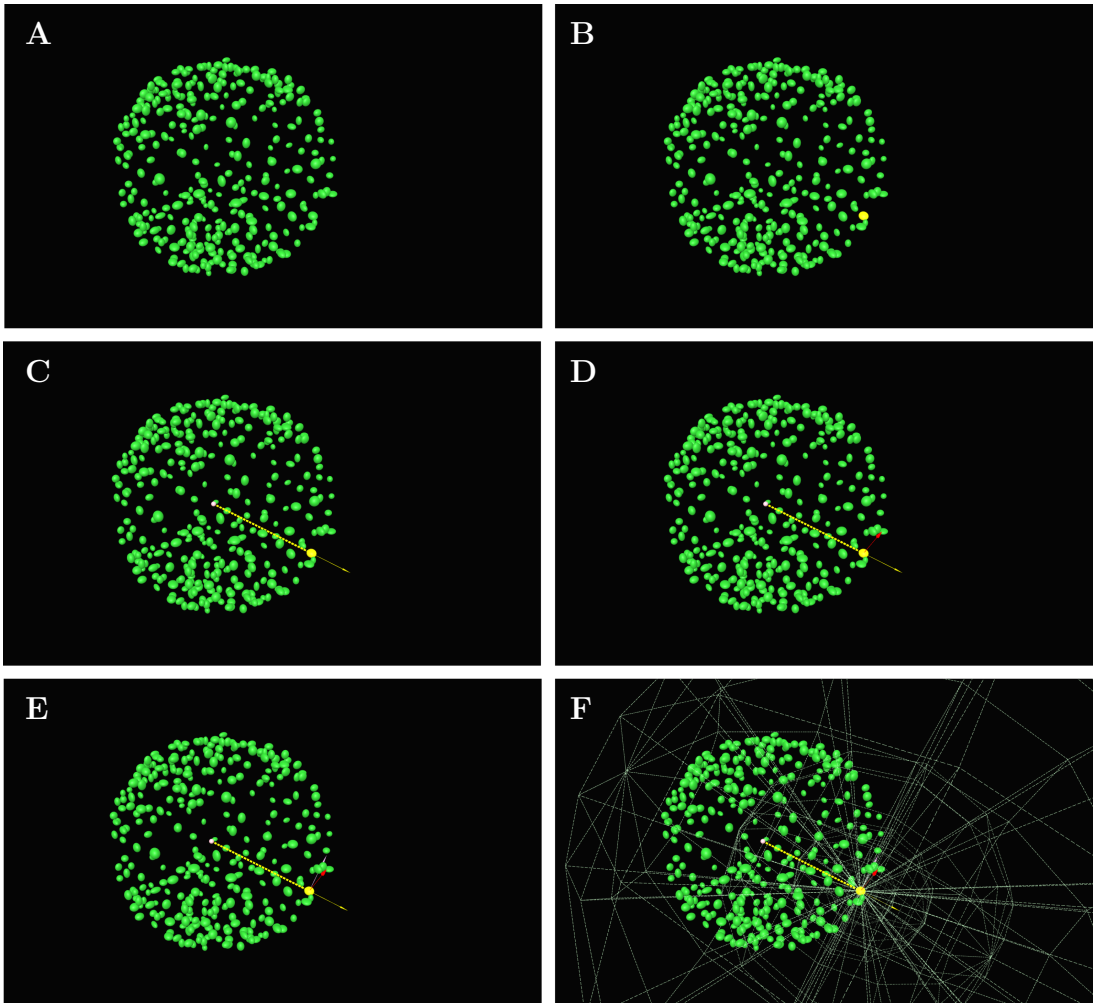

Figure 4.2: **Overview of the how the shape context feature is built for each nucleus detection.** (A) Nuclei instance segmentations are obtained by inputting a gray-scale volume to a trained EmbedSeg model. These instance segmentations are represented as a point cloud. (B, C, D, E) In order to ensure that the shape context geometric descriptor is rotationally covariant, we modify the original coordinate system to obtain a unique coordinate system for each nucleus detection. (B) First, we consider any nucleus detection, represented as a yellow ellipsoid. (C) Next, we identify the centroid of the complete point cloud (shown as a pink ball), and draw a vector from the centroid to the queried nucleus detection. This vector, shown as a yellow arrow, is the local Z axis for the queried nucleus detection. (D) Then we draw a vector pointing from the queried nucleus detection to the location of a landmark. For the *Platynereis dumerilii* data, we notice that detecting the location of one of the four internal macromere is possible using an automated, heuristic approach, and therefore we draw a vector pointing from the queried nucleus detection to this macromere. In the absence of such a landmark, one can use the vector that represents the direction of the largest variance of the point cloud detections *i.e.* the first PCA component corresponding to the x, y, z positions of the point cloud detections. This vector, shown as a red arrow, represents the local X axis. (E) The local Y-axis is evaluated as the cross product of the first two vectors, and is shown as a pink arrow. (F) Next, the neighbourhood around each nucleus detection is binned in order to compute the shape context signature for each detection.

detecting the location of one of the four internal macromere is possible using an automated, heuristic approach, and therefore we draw a vector pointing from the queried nucleus detection to this macromere. In the absence of such a landmark, one can use the vector that represents the direction of the largest variance of the point cloud detections *i.e.* the first PCA component corresponding to the x, y, z positions of the point cloud detections. The projection of this vector evaluated orthogonal to the Z-axis, constitutes the local X-axis. The Y-axis is evaluated as a cross product of the first two vectors (also see Section 4.3.1). Since the sign of the first principal component vector is a 'numerical accident' and thus not repeatable, we use both possibilities and evaluate two shape context descriptors for each feature point in the source cloud. Building such a unique coordinate system for each feature point ensures that the shape context descriptor is rotationally covariant. Additionally since the chemical fixation introduces shrinking of the embryo volume (the intermodal registration use case, see Figure 4.1 B) and since the embryo volume may considerably differ across a population (intramodal use case, see Figure 4.1 A), an additional normalization of the shape context descriptor is performed to achieve scale invariance. This is done by normalizing all the radial distances between p and its neighbours by the mean distance between all point pairs arising in the point cloud.

Different from Belongie et al. (2002) (where the authors use the χ^2 metric to identify the cost of matching two points p_i and q_j arising from two different point clouds i and j), we use the negative of cosine similarity to establish the cost of matching i.e.

$$C_{ij} \coloneqq C\left(p_i, q_j\right) = -\mathbf{s_{p,i}}.\mathbf{s_{q,j}}, \tag{4.2}$$

where $\mathbf{s_{p,i}}$ and $\mathbf{s_{q,j}}$ denote the L_2-normalized shape context vectors at p_i and q_j respectively. By comparing shape contexts resulting from the two clouds of cell nuclei detections, we obtain an initial temporary set of correspondences. These are filtered to obtain a set of inlier point correspondences using RANSAC (Fischer et al., 2010). In our experiments, we specified an affine transform model, which requires a sampling of 4 pairs of corresponding points. We executed RANSAC for 4000 trials, used the Moore-Penrose Pseudo-Inverse operation to estimate the affine transform between the two sets of corresponding locations, and calculate an inlier cutoff from the average sizes of the nuclei segmentations.

Inferring the underlying rotation matrices by fixing only two axes

In the discussion above, the local Z-axis vector $\mathbf{z}$ for each basis point in the *Platynereis dumerilii* data is drawn from the centroid of the point cloud (to which the basis point belongs) to the basis point itself. Then a vector is drawn from one (internal) macromere nucleus to the basis point and its perpendicular projection to the local Z-axis $\mathbf{z}$ is considered as the local X-axis vector $\mathbf{x}$. For an underlying rotation matrix $\mathbf{R}$, I noticed that:

$$\mathbf{R}(\mathbf{z} \times \mathbf{x}) = \pm \mathbf{R}\mathbf{z} \times \mathbf{R}\mathbf{x}.$$

This implies that both the possibilities of the local y-axis should be considered, in order to calculate the underlying rotation matrix between the moving and fixed point clouds composed from the instance segmentations. We determine which is the correct local y-axis with the help of RANSAC (the assumption is that the correct local y-axis would generate more inliers). An intuitive heuristic, for example, would be that if a certain vector provides 2 or more times the number of inliers than the minus of that vector, then the former is chosen to be the valid y-axis.

Obtaining a tighter fit with ICP.

The previous step provides us a good initial alignment. Next, we employ ICP which alternates between establishing correspondences via closest-point lookups and re-computing the optimal transform based on the current set of correspondences. Typically, one employs Horn's approach (Horn, 1987) to estimate strictly-rigid transform parameters. We see equivalently accurate results with iteratively estimating an affine transform, which we compute by employing the Moore-Penrose Inverse operation between the current set of correspondences.

Estimating the complete set of correspondences.

I build a $M \times N$-sized cost matrix C (here, I assume $M < N$ without loss of generality) where the entry C_{ij} is the euclidean distance between the i^{th} transformed source cell nucleus detection and the j^{th} target cell nucleus detection . Next, we employ the Hungarian Algorithm to perform a maximum bipartite matching and estimate correspondences $\hat{X}$:

$$\hat{X} = \arg\min_{X} \sum_{i=1}^{M} \sum_{j=1}^{N} C_{ij} X_{ij}, \text{ where } X_{ij} \in \{0,1\} \text{ s.t. } \sum_{k=1}^{k=N} X_{ik} = 1, \sum_{k=1}^{k=M} X_{kj} \leqslant 1.$$

Estimating a Non-Linear Transform

Since the two specimens being registered are distinct individuals, non-linear differences would persist despite the preceding, linear (affine) registration. I improve the quality of the image registration at this stage by implementing an optional non-linear transform (for example the Free-Form Deformation (FFD) (Rueckert et al., 1999), in which one tries to maximize the correlation of image intensities in the transformed moving and fixed images and interpolates by using Cubic B-Splines).

4.4 Learning Node Embeddings

In PlatyMatch, we partition the region around each nucleus detection in bins in order to calculate the per-nucleus shape context signature (see Section 4.3.1). The *size* and *number* of these bins are designed to be hyper-parameters. One wonders if the optimal local neighborhood signature (*i.e.* node embedding) can instead be *learnt*.

One difficulty in pursuing a learning route is that one needs access to multiple moving and fixed point clouds, with pairs of nuclei which have been manually annotated to be matching. These would serve as ground truth targets for updating the model weights during training. I reasoned that one could instead simulate the underlying motion, and thus obtain a simulated moving point cloud from a fixed point cloud (corresponding to real biological data) where one has access to all of the ground truth matches.

I use the Deep Graph Matching Consensus (DGMC) (Fey et al., 2020) framework to validate the hypothesis mentioned above. Nuclei detections (x, y, z coordinates of the centroid) from the live embryo imaged with SPIM imaging, are available over 598 time points. 60 of these time points are randomly chosen and reserved for testing. Out of the remaining, 547 time points are randomly chosen and used for training the model, while 81 time points are used for validation.

In order to prepare training data, moving point clouds are simulated from fixed point clouds. For each fixed point cloud, nuclei detections are centered, re-scaled to fit in a unit sphere and rotated by a randomly chosen rotation matrix. Additionally, I sample noise independently from $\mathcal{N}(0, 0.04)$, clip the noise to $[-0.2, 0.2]$ and add

Table 4.1: **Quantitative evaluation on a simulated and a real biological dataset.** The columns show the Matching Accuracy $(0-1)$ obtained by baseline methods. Best and second best performing methods per column are indicated in bold and underlined, respectively.

PlatyMatch	DGMC	DGMC + RANSAC
Simulated Dataset		
0.948	0.902	<u>0.942</u>
Real Biological Dataset		
0.347	0.215	<u>0.278</u>

it to these transformed and normalized coordinates, in order to obtain the moving point clouds for the training, validation and testing data subsets. Furthermore, nuclei detections from both the moving and fixed point clouds are connected to their $k = 8$ nearest neighbors using directed edges, in order to convert the point clouds into graphs.

For quantifying the performance of the trained model, I report *matching accuracy* from different baseline methods in Table 4.1. Matching accuracy is defined as the the number of times a point (nucleus detection) from a moving point cloud is correctly matched to the corresponding point in the fixed point cloud, divided by the total number of point detections in the moving point cloud.

For the baseline methods *DGMC* and *DGMC + RANSAC*, I employ a 5-layer Spline-CNN backbone (Fey et al., 2017). The model is trained for 200 epochs and the state of the model which performs the best in terms of matching accuracy on the validation dataset, is used for evaluation on the test dataset. The output of the *DGMC* model is a permutation matrix (each row and column sums to one). This permutation matrix is processed to compute the matching accuracy for the *DGMC* baseline method.

By following a similar strategy as in PlatyMatch, one can filter the produced set of correspondences from *DGMC* using RANSAC, which provides the numbers for *DGMC + RANSAC* baseline method.

The *DGMC* model which was trained on simulated data, was next tested on real biological data corresponding to pairs of different specimens. One notices a lowered performance in the case of the real biological data, indicating a failure of the model to generalize to these datasets.

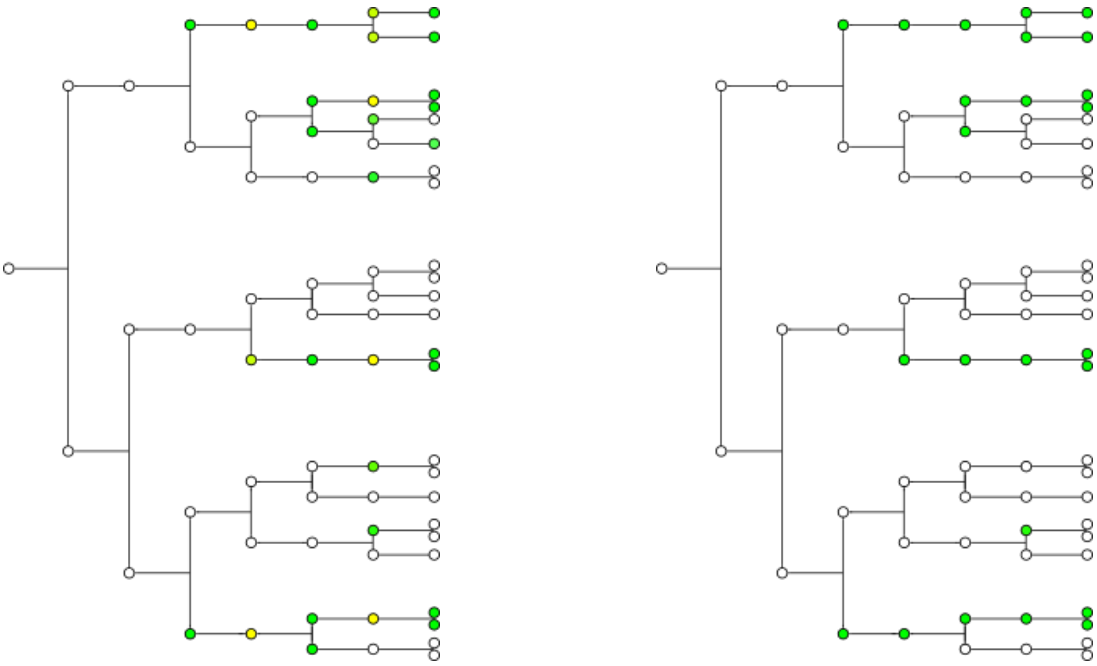

Figure 4.3: **Using an HMM to decipher to underlying hidden gene expression states.** The results of the registration pipeline lead to normalized scores (between 0 and 1) per cell in the lineage tree (left). Using an HMM representation and the Viterbi Algorithm, one can infer the underlying binary, gene expression states per cell (right).

4.5 Hidden Markov Models

After applying the registration approach elucidated so far, one can transfer the gene expression information onto an available lineage tree, which is the goal of this project. But the employed registration approach has matching errors, due to upstream errors in segmentation and also because of lack of *real* training data (*e.g.* pairs of matching nuclei detections for the moving and fixed images being registered), which would lead to noisy gene expression estimates for cells in the lineage tree. To counter this, I propose using an HMM-based strategy to infer the underlying gene expression states from the noisy estimates (obtained through registration).

A Hidden Markov Model (HMM) allows talking about the *observed* events (like the noisy normalized gene expression scores produced for each cell by the pipeline) and also the *hidden* events (like the underlying binary, gene expression states) that one can think of as causal factors in a probabilistic model (Jurafsky and Martin, 2018). An HMM is completely specified by:

$H = h_1 \dots h_N$	A set of N hidden states
$A = a_{11} \dots a_{NN}$	A transition probability matrix A, each a_{ij} representing the probability of moving from state i to state j
$O = o_1 \dots o_T$	A sequence of T observations
$B = b_i(o_t)$	Emission Probabilities, each expressing the probability of an observation o_t being generated from a state i
$\pi = \pi_1 \dots \pi_N$	An initial probability distribution over states.

The problem of specific interest for us is the *Decoding* Problem (Rabiner, 1989), which is described as - given an observation sequence O and an HMM specified by transition probabilities A and emission probabilities B, one would like to discover the best, hidden state sequence H. In the current context, $N = 2$, since each cell can either be in a hidden state $h = 0$ (*i.e.* not expressing a given gene) or $h = 1$ (*i.e.* expressing a given gene). Determining the sequence of hidden states which provides the highest likelihood in a brute-force manner, would require $N^T \simeq 2^{300\times30}$ evaluations at the developmental stage of 16 hpf (since that stage in the lineage tree corresponds roughly to 300 nuclei over 30 time points), which is not possible to compute. Hence, I address this by using the *Viterbi* Algorithm to decipher the sequence with the highest likelihood.

For a given state q_j at time t, the value $v_t(j)$ is computed recursively as

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t), \tag{4.3}$$

where $v_t(j)$ represents the probability that the HMM is in state j after seeing the first t noisy observations.

I use $A = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}$, $b_0(o_t) = 1 - o_t$, $b_1(o_t) = o_t$, $\pi_0 = 0.5$ and $\pi_1 = 0.5$ for my experiments.

4.6 Discussion

An issue that arises in the elucidated approach for PlatyMatch is related to the specification of the local X-axis vector (see Section 4.3.1). If the first vector as suggested by the PCA analysis is used, then in cases when the *in situ* specimens are somewhat squished perhaps because of the imaging protocol, would lead to a different vector than in the fixed (target) point cloud, which would make calculating the underlying transform unreliable.

An alternate strategy would be to use a landmark location (or a landmark instance segmentation) which can be reliably predicted in both the images, say based on its relative position within the point cloud or morphology *etc.*, and draw the local X-axis vector from the basis point to the location of the chosen landmark. A good candidate for such a landmark is one of the four, internal macromeres. However, since these cells lie deeper within the embryo, they often have poor resolution and especially in the confocal images of *in situ* specimens, the instance segmentation model could at times miss them, again making the process of calculating the underlying transform unreliable.

In the future, deep learning strategies for graph matching such as *Deep Graph Matching Consensus* (DGMC) (Fey et al., 2020) shall be further explored. There are a couple of reasons why DGMC did not exceed the performance of PlatyMatch in Table 4.1 (*i*) Graph-based approaches require graphs *with edges*. For an embryo, where only the nuclei are tagged with a histone marker and where one lacks information about the cell membrane, there is no one unique way to represent connections between the nodes of the graph (instance segmentations of the detected nuclei), which renders the tasks of training and generalization of the model to unseen datasets, difficult. (*ii*) Such deep learning models are often trained in a supervised or semi-supervised fashion, where it is crucial to have source (moving) and target (fixed) datasets with a few matching key-points for updating the model weights. In order to counter this shortcoming, I employed simulated datasets which resemble the spatial arrangement of a live embryo (fixed specimen) and the *in situ* specimen (moving specimen) at the developmental stage of interest, in order to train such models. The underlying motion was assumed to be a *similar* transform (*i.e.* scaling, rotation and translation were considered for simulating the motion). In the future, I hope to use non-linear, spline-based transforms for preparing the training data, which would perhaps enable the model to generalize to unseen datasets better.

Bibliography

Balakrishnan, G., Zhao, A., Sabuncu, M. R., Guttag, J., and Dalca, A. V. (2019). Voxelmorph: A learning framework for deformable medical image registration. *IEEE Transactions on Medical Imaging*, 38(8):1788–1800.

Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence*, 24(4):509–522.

Czolbe, S., Krause, O., and Feragen, A. (2021). Semantic similarity metrics for learned image registration. In Heinrich, M., Dou, Q., de Bruijne, M., Lellmann, J., Schläfer, A., and Ernst, F., editors, *Proceedings of the Fourth Conference on Medical Imaging with Deep Learning*, volume 143 of *Proceedings of Machine Learning Research*, pages 105–118. PMLR.

Fey, M., Lenssen, J. E., Morris, C., Masci, J., and Kriege, N. M. (2020). Deep graph matching consensus.

Fey, M., Lenssen, J. E., Weichert, F., and Müller, H. (2017). Splinecnn: Fast geometric deep learning with continuous b-spline kernels.

Fischer, A. H., Henrich, T., and Arendt, D. (2010). The normal development of platynereis dumerilii (nereididae, annelida). *Frontiers in zoology*, 7(1):1–39.

Gold, S. and Rangarajan, A. (1996). A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):377–388.

Hirose, O. (2021). A bayesian formulation of coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(7):2269–2286.

Hörl, D., Rojas Rusak, F., Preusser, F., Tillberg, P., Randel, N., Chhetri, R. K., Cardona, A., Keller, P. J., Harz, H., Leonhardt, H., et al. (2019). Bigstitcher: reconstructing high-resolution image datasets of cleared and expanded samples. *Nature Methods*, 16(9):870–874.

Horn, B. K. (1987). Closed-form solution of absolute orientation using unit quaternions. *Josa a*, 4(4):629–642.

Jurafsky, D. and Martin, J. H. (2018). Speech and language processing (draft). *preparation [cited 2020 June 1] Available from: https://web. stanford. edu/˜ jurafsky/slp3*.

Lalit, M., Handberg-Thorsager, M., Hsieh, Y.-W., Jug, F., and Tomancak, P. (2020). Registration of multi-modal volumetric images by establishing cell correspondence. In Bartoli, A. and Fusiello, A., editors, *Computer Vision – ECCV 2020 Workshops*, pages 458–473, Cham. Springer International Publishing.

Luengo-Oroz, M., Ledesma-Carbayo, M., Peyriéras, N., and Santos, A. (2011). Image analysis for understanding embryo development: a bridge from microscopy to biological insights. *Current Opinion in Genetics & Development*, 21(5):630–637. Developmental mechanisms, patterning and evolution.

Myronenko, A. and Song, X. (2010). Point set registration: Coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2262–2275.

Preibisch, S., Saalfeld, S., Schindelin, J., and Tomancak, P. (2010). Software for bead-based registration of selective plane illumination microscopy data. *Nature methods*, 7(6):418–419.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rueckert, D., Sonoda, L. I., Hayes, C., Hill, D. L., Leach, M. O., and Hawkes, D. J. (1999). Nonrigid registration using free-form deformations: application to breast mr images. *IEEE transactions on medical imaging*, 18(8):712–721.

Tomer, R., Denes, A. S., Tessmar-Raible, K., and Arendt, D. (2010). Profiling by image registration reveals common origin of annelid mushroom bodies and vertebrate pallium. *Cell*, 142(5):800–809.

Vergara, H. M., Bertucci, P. Y., Hantz, P., Tosches, M. A., Achim, K., Vopalensky, P., and Arendt, D. (2017). Whole-organism cellular gene-expression atlas reveals conserved cell types in the ventral nerve cord of *Platynereis dumerilii*. *Proceedings of the National Academy of Sciences*, 114(23):5878–5885.

Vergara, H. M., Pape, C., Meechan, K. I., Zinchenko, V., Genoud, C., Wanner, A. A., Mutemi, K. N., Titze, B., Templin, R. M., Bertucci, P. Y., et al. (2021). Whole-body integration of gene expression and single-cell morphology. *Cell*, 184(18):4819–4837.

Wang, Y. and Solomon, J. M. (2019). Deep closest point: Learning representations

for point cloud registration.

Discussion

Contents

5.1 Introduction **84**

5.2 Pooling Information from Multiple Live Embryos **85**

5.3 Integrating scRNA-Seq Data **86**

5.4 Potential Experiments & Weaknesses **87**

5.1 Introduction

In this dissertation, I explained the various components of the computational pipeline which are necessary for segmenting the nuclei imaged through SPIM and confocal imaging; associating across time, the predicted, instance segmentations of nuclei of the live embryo imaged with SPIM imaging; and then finally estimating a transform between the confocal images of the *in situ* specimen and the corresponding volume of the live embryo by leveraging *local neighborhood* as a feature for matching.

Although the transform matrix which enables aligning the gene expression image channel of the *in situ* specimen with the corresponding volume of the live embryo can be calculated, the question regarding assigning binary expression states to each cell of the live embryo still remains.

In an ideal setting, one could sum up the intensity of the gene expression channel within each cell segmentation, but since the cell membrane has not been tagged for the live embryos which were imaged, I instead use the nuclei instance segmentation as a proxy for quantifying gene expression per cell. I assume therefore that the intensity of the gene expression channel accumulated within each nucleus instance segmentation correlates with the number of mRNA molecules per cell. Furthermore, in order to normalize the effect of cell (nucleus) size, the accumulated intensity per nucleus is divided by the nucleus size (in terms of pixels). This provides a score for each cell, where a higher magnitude indicates a higher probability of gene expression per cell.

These calculated scores would be *noisy*, due to the upstream errors in the instance segmentation and registration methods. In order to address this, I consider the predictions so far as noisy observations and infer the underlying binary gene expression states per cell in the lineage, using an HMM model.

In the following sections, I discuss (i) how one could pool information to a common frame of reference, say if multiple embryos were imaged with SPIM, (ii) how other data modalities such as scRNA-Seq can additionally be included in this computational framework, and lastly (iii) which are the weaknesses of this computational pipeline, and upon completion, which potential experiments can additionally be explored.

5.2 Pooling Information from Multiple Live Embryos

In order to compare lineage trees arising from several live embryos, firstly the developmental speed must be normalized w.r.t. temperature, in order to eliminate the influence of temperature. In Figure 29 of Fischer et al. (2010), the authors show a uniform increase in the developmental speed of *Platynereis dumerilii*, by elevating the temperature above 18° Celsius. Either the suggested scaling factor s as stated by the authors can be used or a factor equal to the ratio of the difference in number of cells at any two arbitrarily chosen time points (say time points 0 and 50), between the embryos (say embryos 1 and 2) being compared

$$s = \frac{(N^2_{t=50} - N^2_{t=0})}{(N^1_{t=50} - N^1_{t=0})}, \tag{5.1}$$

can be calculated and uniformally applied to expand (or contract) the developmental time axis.

Time-lapse movies of *Platynereis dumerilii* recorded by my collaborators, usually begin at 5 hpf, at which stage the embryo has 38 cells. Identifying the identity of each of these 38 cells is manually achievable quickly. Once the lineage labels ($2d112$, $4d$ *etc.*) are known for the available lineage trees, one needs to only compare the number, morphology and position of cells at any given (temperature-normalized) time point between *corresponding* lineages while comparing multiple embryos.

These steps elucidated above allow investigating whether there is a high variance observed for certain, specific lineages at any given (temperature-normalized) time point when comparing between individuals and thus quantifying the level of stereotypicity in the early development of *Platynereis dumerilii*. It further allows building a canonical lineage tree where each cell is positioned at a location, which is set equal to the mean (x, y, z) location as determined from the positions of the corresponding matching cells and which additionally carries a measure of the uncertainty regarding its position and presence.

Lastly, in the case that a live embryo has been imaged with additional image channels such as a membrane marker, by establishing a matching between cells detected at corresponding time points, one can transfer the corresponding cell size information to the canonical lineage tree. Having cell instance segmentations in addition to the corresponding nuclei instance segmentations in the canonical lineage tree, allows accumulating the mRNA content per cell more accurately (also

see Section 5.4). It additionally allows representing an embryo at any given time point, as a spatial graph where cells are represented as nodes and are connected by edges to other cells which are touching (this is not straight-forward to do if one has access only to nuclei instance segmentations), which in turn opens up the application of graph-based approaches such as graph matching and sub-graph isomorphism to these biological datasets.

5.3 Integrating scRNA-Seq Data

One drawback of employing confocal images of specimens which were fixed and stained following the norms of WMISH protocol, is that it is limited in throughput - *multiple* embryos need to be fixed, and each fixed embryo is stained for the expression of *one (or a few)* transcription factors only. This further implies that for practical purposes, one can have access to an M-dimensional, gene expression signature for each cell, where M is significantly smaller than N, N being the number of transcription factors available in reality.

This problem can partially be alleviated by following strategies such as *osm-FISH* (Codeluppi et al., 2018) which allows one to quantify the expression of a large number of genes in tissue sections, but these still do not provide access to the complete set of genes.

I describe here an alternate approach, inspired by *NovoSpaRc* (Nitzan et al., 2019; Moriel et al., 2021) and Peyré et al. (2016), which allows mapping the gene expression signature (an M-dimensional vector) to an N-dimensional vector, in case scRNA-Seq data at a certain developmental stage of the embryo is available in addition to having a few, fixed and stained specimens.

The main objective of *NovoSpaRc* is to map c single cells (for which a count of RNA molecules for multiple genes *i.e.* a gene expression signature obtained through scRNA-Seq for example, is available) onto l coordinates of a physical space. This problem is formulated as an optimal transport problem in which a transport matrix T is calculated, which allows probabilistically mapping the available c cells to l locations, which in turn allows deciphering spatial information from the single-cell data.

I reason that instead of mapping single cells to locations (which is implicitly handled by my pipeline), one can use a similar strategy as *NovoSpaRc* to infer a mapping between single cells digitized through imaging (for which my pipeline calculates a binary, M-dimensional vector) to single cells obtained from other individuals at roughly the same developmental stage and sequenced (each cell corresponds to an N-dimensional vector). Once a transport matrix is inferred,

one has access to the $(N - M)$ genes additionally for each of the single cells of the live embryo imaged by SPIM.

Typically, in order to amplify the mRNA content per single cell, one sequences cells from multiple individuals (also see Chapter 1). In order to have a scRNA-Seq gene expression signature for an *average* individual, one could run a k-means clustering in the gene expression space, where k or the number of desired clusters is equal to the average number of cells per individual. Doing so would convert the transcriptome data from multiple individuals to that from an average individual, and then the optimal-transport mapping as described above can potentially be executed.

5.4 Potential Experiments & Weaknesses

This computational pipeline opens up possibilities for new experiments. It essentially enables deciphering the gene expression state of cells, digitized from the SPIM image volumes, at the stages for which the confocal images of *in situ* specimens is available. In the context of the data available from our collaborators, we have *in situ* data at the stages of 16, 20 and 24 hpf.

But how does one determine the gene expression state at the intermediate time points (from 16 *to* 20*, from* 20 *to* 24 *hpf)?* Due to the presence of the lineage tree, one is able to assign a prior on the likely gene expression scores for the detected cells at the intermediate time points. One potential experiment would be to use a reinforcement learning scheme which would suggest which intermediate time point should be fixed and stained. This time point should ideally be one which reduces the maximum uncertainty and following this scheme provides an iterative approach for doing fixation and staining.

How does one verify the results of the pipeline? To address this, for each gene at a given developmental stage, we have data for several replicates. These replicates are processed independently, and their corresponding predictions should ideally have a consensus. If a consensus is achieved, the data for this gene is retained, else not.

Currently the transition and emission probabilities of the HMM are hard-coded. Can these be learnt from the data? This is indeed the case currently (see Section 4.5). But these probabilities can be learnt through a potential experiment wherein the spatially noisy alignment and the subsequent assignment of the gene expression scores are corrected for, by a biologist. This enables learning the transition and emission probability matrices, which would enable the most optimal mapping from the lineage tree containing noisy scores per cell (node) to

the lineage tree containing binarized gene expression scores per cell, using the HMM formulation.

What are the major weaknesses of this pipeline? Post the alignment of the gene expression channel of the *in situ* specimen along the SPIM volume, there is still the open question of how many mRNA molecules are assigned to each cell. Since we lack the cell membrane information, we use the nuclei instance segmentations as a proxy for accumulating the gene expression information. This would introduce some errors for gene expression quantification per cell.

Another weakness is more general in nature, and exposes a pitfall of this pipeline. In order to spatially pool information from different individuals, and to align their corresponding nuclei channels, we currently use the intuition that the corresponding cell in both images should have the same local neighbourhood. This assumption would not work for non-mosaically developing organisms and in that context, a different approach to aligning and estimating transform between volumes arising from different individuals would be needed. One intuition is to benefit from more features than just using local neighborhood descriptors for finding corresponding matches and thus in turn, estimating the underlying transform matrix.

Bibliography

Codeluppi, S., Borm, L. E., Zeisel, A., La Manno, G., van Lunteren, J. A., Svensson, C. I., and Linnarsson, S. (2018). Spatial organization of the somatosensory cortex revealed by osmfish. *Nature methods*, 15(11):932–935.

Fischer, A. H., Henrich, T., and Arendt, D. (2010). The normal development of platynereis dumerilii (nereididae, annelida). *Frontiers in zoology*, 7(1):1–39.

Moriel, N., Senel, E., Friedman, N., Rajewsky, N., Karaiskos, N., and Nitzan, M. (2021). Novosparc: flexible spatial reconstruction of single-cell gene expression with optimal transport. *Nature Protocols*, 16(9):4177–4200.

Nitzan, M., Karaiskos, N., Friedman, N., and Rajewsky, N. (2019). Gene expression cartography. *Nature*, 576(7785):132–137.

Peyré, G., Cuturi, M., and Solomon, J. (2016). Gromov-wasserstein averaging of kernel and distance matrices. In *International Conference on Machine Learning*, pages 2664–2672. PMLR.

Selbstständigkeitserklärung

1. Hiermit versichere ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

2. Bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskripts habe ich Unterstützungsleistungen von folgenden Personen erhalten: (keine)

3. Weitere Personen waren an der geistigen Herstellung der vorliegenden Arbeit nicht beteiligt. Insbesondere habe ich nicht die Hilfe eines kommerziellen Promotionsberaters in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

4. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und ist auch noch nicht veröffentlicht worden.

5. Ich bestätige, dass ich die geltende Promotionsordnung der Fakultät Informatik der Technischen Universität Dresden anerkenne.

	Dresden, 05. April 2022
Unterschrift des Doktoranden	Ort, Datum