INAUGURAL – DISSERTATION
zur
Erlangung der Doktorwürde
der
Gesamtfakultät für Mathematik, Ingenieur- und Naturwissenschaften
der
RUPRECHT – KARLS – UNIVERSITÄT
HEIDELBERG

vorgelegt von

Adrian Wolny, M.Sc.
geboren in Kraków, Polen

Tag der mündlichen Prüfung:
27.04.2023

Learning Instance Segmentation from Sparse Supervision

Advisor: Prof. Dr. Fred A. Hamprecht

Abstract

Instance segmentation is an important task in many domains of automatic image processing, such as self-driving cars, robotics and microscopy data analysis. Recently, deep learning-based algorithms have brought image segmentation close to human performance. However, most existing models rely on dense groundtruth labels for training, which are expensive, time consuming and often require experienced annotators to perform the labeling. Besides the annotation burden, training complex high-capacity neural networks depends upon non-trivial expertise in the choice and tuning of hyperparameters, making the adoption of these models challenging for researchers in other fields.

The aim of this work is twofold. The first is to make the deep learning segmentation methods accessible to non-specialist. The second is to address the dense annotation problem by developing instance segmentation methods trainable with limited groundtruth data. In the first part of this thesis, I bring state-of-the-art instance segmentation methods closer to non-experts by developing PlantSeg: a pipeline for volumetric segmentation of light microscopy images of biological tissues into cells. PlantSeg comes with a large repository of pre-trained models and delivers highly accurate results on a variety of samples and image modalities. We exemplify its usefulness to answer biological questions in several collaborative research projects. In the second part, I tackle the dense annotation bottleneck by introducing SPOCO, an instance segmentation method, which can be trained from just a few annotated objects. It demonstrates strong segmentation performance on challenging natural and biological benchmark datasets at a very reduced manual annotation cost and delivers state-of-the-art results on the CVPPP [100] benchmark.

In summary, my contributions enable training of instance segmentation models with limited amounts of labeled data and make these methods more accessible for non-experts, speeding up the process of quantitative data analysis.

Zusammenfassung

Die Instanzsegmentierung ist eine wichtige Aufgabe in vielen Bereichen der automatischen Bildanalyse, etwa bei selbstfahrenden Autos, in der Robotik und bei der Analyse von Mikroskopie Datensätzen. In jüngster Zeit haben Algorithmen, die auf Deep Learning basieren, die Bildsegmentierung in die Nähe menschlicher Leistung gebracht. Die meisten der bestehenden Modelle sind jedoch auf vollständig annotierte Grundwahrheit für das Training angewiesen, deren Erstellung teuer und zeitaufwendig ist und häufig nicht ohne Expert*innenwissen zu leisten ist. Abgesehen dieser hohen Annotationskosten erfordert das Training komplexer neuronaler Netze mit hoher Kapazität spezielle Fachkenntnisse bei der Wahl und Einstellung von Hyperparametern, was die Anwendung dieser Modelle für Forscher*innen aus anderen Bereichen zu einer Herausforderung macht.

Diese Arbeit hat zwei Ziele. Erstens sollen die Deep-Learning-Segmentierungsmethoden auch fachfremden Nutzern zugänglich gemacht werden. Zweitens soll das Problem der Erfordernis von vollständig annotierter Grundwahrheit durch die Entwicklung von Instanzsegmentierungsmethoden angegangen werden, die mit begrenzten Grunddaten trainiert werden können. Der erste Teil beschreibt die Entwicklung des Programms Plantseg, das Nicht-Experten modernste Instanzsegmentierungsmethoden zugänglich macht. Plantseg implementiert eine Pipeline zur volumetrischen Segmentierung von lichtmikroskopischen Bildern biologischer Gewebe in Zellen. Es ermöglicht die Anwendung einer Vielzahl an vortrainierten Modellen und liefert hochpräzise Ergebnisse bei einer Vielzahl von verschiedenen mikroskopischen Proben und Bildmodalitäten. Der Nutzen für die Beantwortung biologischer Fragen wird am Beispiel mehrerer gemeinschaftlicher Forschungsprojekte veranschaulicht. Im zweiten Teil wird eine Lösung zum Problem der vollständig annotierten Grundwahrheit vorgestellt: SPOCO, eine Methode zur Instanzsegmentierung, die mit nur wenigen annotierten Objekten trainiert werden kann. SPOCO zeigt eine starke Segmentierungsleistung auf anspruchsvollen natürlichen und biologischen Benchmark-Datensätzen bei sehr geringen manuellen Annotationskosten und liefert Spitzenergebnisse auf dem CVPPP [100] Benchmark.

Zusammenfassend ermöglichen die hier vorgestellten Beiträge das Training von Instanzsegmentierungsmodellen mit einer begrenzten annotierten Grundwahrheit und machen diese Methoden für Nicht-Experten zugänglich. Dadurch wird letztlich der Prozess der quantitativen Datenanalyse beschleunigt.

Acknowledgments

First and foremost, I would like to thank my supervisors, Doctor Anna Kreshuk and Professor Fred Hamprecht for their guidance during my PhD. I want to thank Anna for supervising me from the beginning of my PhD, first as a postdoc in Fred's lab and then as a group leader at EMBL. Her curiosity, passion and excellent mentoring is the driving force behind many successful scientific projects. I feel extremely fortunate to have joined her lab at EMBL as one of the first PhD students. She created an open and collaborative environment and gave me the chance to work on exciting, multi-disciplinary projects. I wish to thank Fred for the opportunity to conduct research in his lab. Talented people and a broad range of topics pursued in his group provided the perfect conditions to start my scientific journey.

I would also like to extend my thanks to other members of the Kreshuk Lab. In particular, Constantin Pape for his broad technical expertise and all the effort he has put into our joint research projects. Qin Yu for his help and commitment to finish SPOCO experiments. Dominik Kutra for his kindness, conversations in German and all the excellent vinyl records he sent me during the Covid lockdown. Fynn Beuttenmüller, Alex Matskevytch, Valentyna Zinchenko, Johannes Hugger and all other members of the lab with whom I shared interesting conversations and memorable moments during group retreats.

Many thanks to the members of the Image Analysis and Learning Lab at Heidelberg University. Especially, Lorenzo Cerrone for our joint work on PlantSeg. Manuel Haussmann, Roman Remme and Sebastian Damrich for many stimulating discussions. I also want to thank Professor Ullrich Köthe for agreeing to review this thesis.

I feel fortunate to have collaborated with many amazing researchers from Plant Morphodynamics consortium. Professor Alexis Maizel who introduced me to the fascinating world of plant morphogenesis. Kudos to Professor Kay Schneitz, Richard Smith, Athul Vijayan, Rachele Tofanelli, Tejasvinee Mody, Marion Louveaux, Christian Wenzl and Sören Strauss for our joint projects. I would also like to thank Vladyslav Bondarenko, Takafumi Ichikawa, Dimitri Fabrèges and Takashi Hiiragi for our fruitful collaboration at EMBL.

I would like to express my gratitude to my parents for their support and trust in my life's decisions.

My special thanks go to my wife, Agata, for her unconditional love, support and bringing happiness to every moment of my live. Her encouragement to pursue my dreams is the very reason this work came into being.

Contents

1 Introduction

The goal of computer vision is to interpret the content of digital images and videos in a way similar to the human visual system. One of the key task required to automate understanding of complex visual environment is instance segmentation. It is a type of image segmentation technique that delineates individual objects within an image and assigns them different categories as a function of their appearance. Instance segmentation is very challenging due to diverse shape patterns, obscured boundaries between objects and variable number of instances in an image. Also, in contrast to object detection, each instance cannot simply be described by a bounding box, but has to be represented by a pixel-wise mask.

Instance-level segmentation has a wide range of applications in various fields, from robotics and autonomous driving, where it can be used to identify objects within the environment for tasks such as grasping, navigation and obstacle avoidance [144, 149], to medical imaging, where segmentation and identification of different cell types is helpful in diagnosing a variety of diseases [91, 107]. Biological imaging provides a particularly large set of use cases for the instance segmentation task, with imaging modalities ranging from natural photographs for phenotyping [100] to electron microscopy for analysis of cellular ultrastructure [146]. For instance, modern microscopy captures terabytes of high-resolution volumetric images of plants and animals, and automatic segmentation of individual cells is needed to study the geometry and interactions of individual cells within a tissue or organism. It provides valuable insights into the organization and function of cells and tissues. By analyzing cell segmentation, we can better understand the processes that drive development of living systems and identify potential areas for further research. See Figure 1.1 for examples of instance segmentation tasks appearing in urban scene understanding, developmental biology and plant phenotyping.

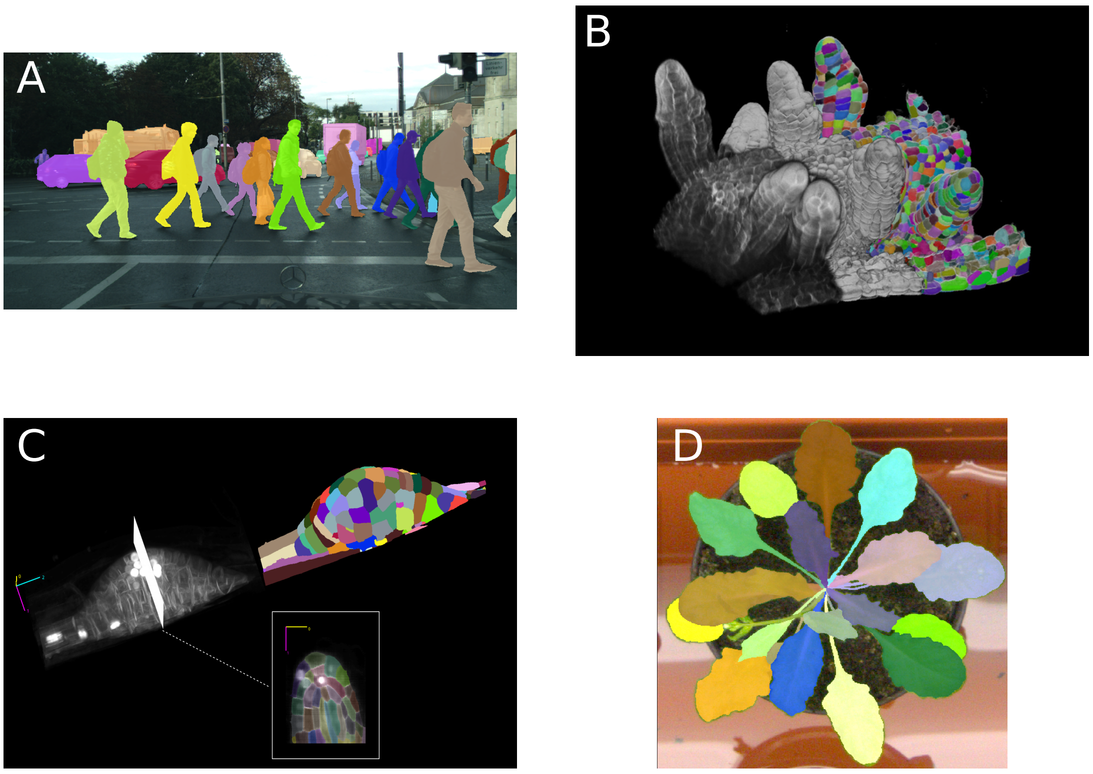


Figure 1.1: Examples of instance segmentation problems explored in this thesis. (**A**) segmentation of dynamic objects, such as people, cars, trucks, etc. in urban environment [30] (Section 3.4). (**B**) segmentation of cells in a confocal volume of a developing plant ovule (Section 2.3): (left) input signal, (middle) cell membranes predicted by a neural network, (right) segmented cells. (**C**) cells segmented from a 3D light sheet microscopy image of an emerging lateral root (Section 2.3): (left) 3D image of the primordium, (upper right) segmented primordial cells, (lower right) sample cross section with input signal and segmentation. (**D**) segmentation of individual leafs of a plant growing in a pot [100] (Section 3.4).

Within the last decade deep learning has achieved significant success in a variety of computer vision tasks, including instance segmentation [26, 57, 80]. However, one of the major bottlenecks of deep learning for segmentation is the need for large-scale densely annotated images for neural network training. Collecting and annotating such datasets is very costly, labor intensive and requires specialized software and tools. This task is especially difficult for biomedical images which vary in imaging modalities and acquisition settings and necessitate trained annotators who are capable of accurately identifying and labeling objects of interests.

As an example, the groundtruth segmentation for the lateral root dataset shown in Figure 1.1 C took an expert biologist around 5 months to create. In addition, no large public data collections (e.g. ImageNet [32], Cityscapes [30]) are available for pre-training. Another obstacle to widespread adoption of modern deep learning-based segmentation methods lies in their accessibility. They are complex multi-step procedures, which involve training neural networks and post-processing to convert the network outputs into final segmentation. Those techniques require a high level of expertise and only a handful of software packages strive to make them accessible to non-expert users in biology and medical sciences.

In this thesis, I address the problems of accessibility *and* annotation bottleneck of modern instance segmentation methods. To this end, I have developed a user-friendly segmentation tool for microscopy (Chapter 2). Initially developed for the electron microscopy connectomics, this techique is aimed to segment 3D confocal and light-sheet microscopy images. I have performed extensive ablation studies and took important design decisions, reducing the complexity for the end users. Our package comes with a large number of convolutional neural networks, pre-trained on multiple microscopy image domains, which can be used directly on novel segmentation problems, shortening the path to scientific discovery. I also make the training of 2D and 3D segmentation networks more accessible by creating a popular software package for this purpose (see Chapter 4).
I have also introduced a novel instance segmentation method, which does not require dense labeling and can be trained from a small subset of annotated objects in the image (Chapter 3). Our approach is general-purpose and can be applied to both natural and microscopy images. It is particularly useful in 3D microscopy images which commonly contain hundreds of objects of interests, such as cells, neurons or organelles or in medical imaging, where different patient populations render dense groundtruth creation prohibitive. Our method can be used in those challenging settings to produce high quality instance segmentation at a very reduced annotation cost. I demonstrate the versatility of methods developed in this thesis in multiple collaborative projects.

In the next section, I provide an more formal definition of the image segmentation problem and summarize two families of machine learning algorithms used to solve the instance segmentation task. I use both techniques in this work.

1.1 Machine Learning for Image Segmentation

Image segmentation aims to group the pixels in the image into meaningful regions. There are several types of this problem: *semantic segmentation* associates every pixel of an image with a class label, e.g. {*road, person, car, building, sky*}. Here multiple objects of the same class are considered a single region in the image. In contrast, *instance segmentation* distinguishes between instances of the same class and assigns each pixel to a specific instance label, e.g. {*person1, person2, person3, ...*} or background. *Panoptic segmentation* combines semantic and instance segmentation. In this approach pixels belonging to countable objects such as people, animals, cars, are assigned individual instance labels (e.g. {*person1, person2, car1, ...*}), whereas pixels belonging to regions of similar texture are given semantic labels (e.g. {*grass, sky, road*}). Our main task of instance segmentation often requires solving semantic segmentation first, therefore in the following, I provide a brief overview of modern techniques used to solve both tasks.

State-of-the-art image segmentation methods are based on deep learning: a subfield of machine learning which uses neural networks to learn representations of the input data, useful to solve a given task. Those models are organized into layers of non-linear functions stacked on top of each other. Each successive layer automatically learns increasingly meaningful representations by being exposed to training data. Each layer is parameterized by its weights and learning is the process of adjusting the weights, such that for a given input, the network's output is close to the expected target value. We measure the discrepancy between the network prediction and the target value using a loss function, e.g. cross entropy. The value of the loss function is being minimized by updating the weights using a combination of backpropagation algorithm [50] and stochastic gradient descent [114] or its variants [71, 34, 87]. The weights are being updated until a stop criterion is met.

Convolutional neural networks (CNNs) is a class of model architectures commonly used for computer vision applications. CNNs are composed of multiple convolutional layers, where each layer learns local patters found in a small region of the input, e.g. 3×3 pixels for 2D images. This spatially local processing reduces the number of learnable parameters and introduces shift invariance. By stacking convolutional layers on top of each other, we increase the spatial context of the network, also known as the receptive field. In contrast to image classification networks which predict a class label per image, semantic segmentation networks have to predict a label for each pixel in the image. For this purpose a fully convolutional network (FCN) [86] was proposed. Modern semantic segmentation models have improved the FCN, by introducing the encoder-decoder architecture with skip connections [113] and feature pyramids [82]. In particular, the U-Net architecture [113, 21] has been highly successful in solving semantic segmentation problems for microscopy and is used as a base model for methods presented in this thesis.

Semantic segmentation networks mentioned above require a fixed number of labels and use loss functions which are *not* permutation invariant. Therefore those models cannot be directly applied to instance segmentation, where instance labels are permutation invariant and vary in numbers. Various multi-step pipeline have been proposed instead. Proposal-based methods [57, 55] such as Mask R-CNN are a popular choice for instance segmentation in natural images. However, they have not been adopted for microscopy imaging due to difficulties segmenting complex non-convex shapes and lack of pre-trained backbones. A well-known paradigm is to start from the semantic segmentation of instance boundaries serving as a foreground class, followed by graph partitioning methods. This technique have been used for natural images [72] and is especially popular for microscopy [8, 46, 77, 106]. I present an overview of this method in Section 1.1.1 and further in Chapter 2, where I use it for 3D microscopy segmentation. Other approaches are based on embedding networks [38, 16] which learn pixel level representation that can easily be clustered into instances with a simple post-processing step. I review those methods in Section 1.1.2 and extend this technique in Chapter 3, where I introduce our weakly-supervised instance segmentation.

1.1.1 Graph-based Instance Segmentation

One way to represent images, is in terms of a graph $G = (V, E)$ with nodes V and edges E. One can choose a *grid graph*, where edges connect neighboring pixels, or a *region adjacency graph*, where nodes correspond to "superpixels", i.e. perceptually meaningful group of pixels, and edges connect adjacent superpixels. The edge weight can be interpreted as the probability of the adjacent nodes belonging to the same instance. Edge weights can be derived from the node properties or given by the edge classifier. The instance segmentation is given by grouping the nodes into instances based on the weights associated with the edges. Seeded watershed [31, 97, 96] and graph-based agglomeration [39, 45, 105] are two important examples of graph-based segmentation methods. An important limitation of both watershed and agglomerative clustering is that they can only work with positive (attractive) edge weights. As a consequence, an additional stopping criteria, such as seed points for watershed or weight thresholds for agglomerative methods are required to partition the graph. Finding these hyperparameters is challenging task in itself, particularly for large graphs with unknown number of instances. Other methods like Multicut [5, 72, 147, 1, 2] can use both positive and negative affinities and can be formulated as a constrained optimization problem, removing the need for an auxiliary stopping criterion. Formally, let $y_e = y_{u,v} \in \{0, 1\}$ be a binary indicator variable for each edge $e = \{u, v\} \in E$. This variable specifies which edges are "cut", i.e. $y_e = 1$ if edge $e = \{u, v\}$ runs between two clusters and $y_e = 0$ if the edge is within a single cluster. The vector $\mathbf{y} \in \{0, 1\}^{|E|}$ is called *a Multicut*. Let C be the set of all cycles of the graph G. The minimum Multicut is defined as

$$\mathbf{y}^* = \min_{\mathbf{y}\in\{0,1\}^{|E|}} \sum_{e\in E} w_e y_e \tag{1.1}$$

$$\text{such that} \quad \forall C\ \forall e \in C \colon y_e \leq \sum_{e'\in C\setminus\{e\}} y_{e'} \tag{1.2}$$

The constraints in Equation 1.2 force each cycle to have none or at least two cut edges. They ensure that the set of cut edges is a valid partitioning. The Multicut is NP-hard in general, due to the exponential number of constraints in Equation 1.2, however efficient approximate solvers exist [7, 68, 69].

The methods presented in Chapter 2 use the seeded watershed to generate superpixels for a region adjacency graph and a neural network to predict the edge weights. This graph is partitioned using Multicut solvers or agglomerative clustering methods.

1.1.2 Embedding-based Instance Segmentation

An image can also be represented as a set of vectors at the pixel level. This is the main idea behind embedding-based models for instance segmentation. These methods usually have two stages. First, a vector representation is learned for each pixel, then based on the learned representations, the pixels are grouped together using a clustering algorithm. Specifically, a fully convolutional network $f\colon \mathbb{R}^3 \to \mathbb{R}^D$ maps each pixel into a D-dimensional vector space, where pixels belonging to the same objects are close together in that space and pixels from different objects are separated by a wide margin. The separation in the embedding space can be achieved with a contrastive learning objective [38, 16]. Given the embedding vectors $\boldsymbol{e}_p$ for each pixel p, the authors of [38] use the following similarity measure between pixels p and q:

$$\sigma(p,q) = \frac{2}{1+\exp(\|\boldsymbol{e}_p - \boldsymbol{e}_q\|_2^2)} \tag{1.3}$$

and train the embedding network by minimizing the following loss:

$$\mathcal{L} = -\frac{1}{N} \sum_{p,q\in S} [1_{\{y_p=y_q\}} \log(\sigma(p,q)) + \ 1_{\{y_p\neq y_q\}} \log(1-\sigma(p,q))] \tag{1.4}$$

where N is the number of pixels in the image and y_p is the instance label of pixel p. Since the complexity of this loss is $\mathcal{O}(N^2)$ only a sub-set of pixels is used for computation.

A more tractable loss, given in [16], consists of two main components illustrated in Figure 1.2. The *pull force* pulls together embeddings of pixels belonging to the same objects. To counterbalance the pull force and prevent everything collapsing to a single cluster in the embedding space, the *push term* pushes apart the centers of the clusters in the embedding space. This can be express in the following way:

$$L_{pull} = \frac{1}{C}\sum_{k=1}^{C}\frac{1}{N_k}\sum_{i=1}^{N_k}[\|\boldsymbol{\mu}_k - \boldsymbol{e}_i\|_2 - \delta_v]_+^2 \tag{1.5}$$

$$L_{push} = \frac{1}{C(C-1)}\sum_{\substack{k=1\\k\neq l}}^{C}\sum_{l=1}^{C}[2\delta_d - \|\boldsymbol{\mu}_k - \boldsymbol{\mu}_l\|_2]_+^2 \tag{1.6}$$

$$\mathcal{L} = L_{pull} + \lambda L_{push} \tag{1.7}$$

where C is the number of objects in the image, N_k the size of object k, δ_v, δ_d are margin hyperparameters and $[x]_+ = max(0, x)$ is the rectifier function. After network convergence the instance segmentation of a given image is generated by clustering the network outputs. Mean-shift [28] and HDBSCAN [17] are commonly used for this task.

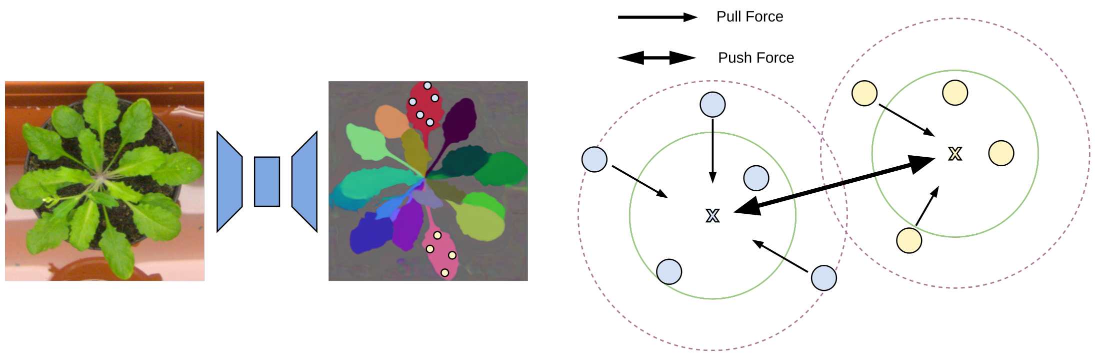


Figure 1.2: Schematic overview of the push and pull forces, the main components of the discriminative loss from [16]. (Left) Input image and corresponding pixel embeddings predicted by the FCN (embedding vectors are PCA-projected into RGB space for visualization). (Right) The pull force pulls embeddings (blue and yellow dots) towards their corresponding cluster centers, whereas the push force pushes cluster centers away from each other. Both forces are are hinged: they are only active up to a certain distance denoted by the circles around the cluster centers.

Pixels embeddings described above are non-spatial, i.e. no notion of pixel coordinates is used during training. Spatial awareness can be introduced by passing pixel coordinates to the network [85] or by explicitly predicting foreground areas and, for pixels in those, the direction to the object center [102, 26] or the distance to the object boundary [3]. Spatial embeddings require more complex post-processing and have difficulties segmenting occluded and variable-sized objects.

In this thesis, I use the formulation from [16] and show in Chapter 3 that it allows sampling individual instances in the image in a differentiable way. This differentiable instance sampling enables a loss function to be applied on individual instances and moves us closer towards single-stage, end-to-end learning for instance segmentation. I use this sampling technique to learn pixel embeddings with an additional self-supervised component and remove the dense annotation requirement.

1.2 Contributions

In this thesis, I develop deep learning-based instance segmentation methods, which are less demanding in terms of the amount of training data, and which are accessible to non-experts. I deliver highly accurate results for both natural and microscopy imaging domains.

In Chapter 2, I introduce PlantSeg [142]: an easy to use tool for volumetric instance segmentation of light microscopy data. Our pipeline is based on previous work done in electron microscopy images of neural tissue, where a combination of a strong boundary predictor and graph partitioning methods has been shown to outperform other methods. PlantSeg delivers highly accurate results on a variety of samples and image modalities. It comes with large number of pre-trained models which can be used out-of-the-box for solving novel segmentation problems.

In Chapter 3, I give a detailed overview of SPOCO [141]: label-efficient, embedding-based instance segmentation algorithm. Our formulation enables training the neural networks with *positive unlabeled* supervision, a form of sparse labelling which allows for a more diverse training set and is more natural for human annotator. Our method provides highly accurate segmentation on challenging natural and microscopy dataset at a very reduced manual annotation cost and delivers state-of-the-art results on the CVPPP [100] benchmark.

Techniques introduced in this work have been used to deliver high-quality segmentation in a number of collaborative research projects. In Section 2.4, I highlight applications of PlantSeg to problems in developmental biology and detection of virus specific antibodies. First, based on PlantSeg's pre-trained CNNs repository I developed a simple, iterative training procedure, which minimized expensive 3D proofreading and enabled high quality segmentation of mouse embryo cells [61]. Second, PlantSeg pipeline helped exploring the role of embryo-uterus interaction in mouse embryogenesis [15]. Finally, the tools I developed were useful for analysis of microscopy-based assay for detection of SARS-CoV-2 antibodies in human serum [107].

In Chapter 4, I briefly describe a number of software projects I have created and contributed to, which bring state-of-the-art algorithms closer to the end user. In particular, the *pytoch-3dunet* package I developed has become highly popular among practitioners of image analysis and has been used in (bio)medical research [119]. It comes with a simple to use config-based interface and reduces the complex task of network training and inference to preparing the training and test data in the right format. PlantSeg, SPOCO, pytorch-3dunet and other software projects related to this thesis have been open-sourced on GitHub.

As part of this work, I have also contributed to the creation of two 3D training datasets of confocal and light-sheet microscopy images (see Figure 1.1 B, C and Chapter 2 for more details). Both datasets are shared publicly to encourage further research in large-scale cell segmentation in developmental biology.

2 PlantSeg: Pipeline for Volumetric Instance Segmentation

In Chapter 1, I described the importance of instance segmentation in many application domains and the lack of easily accessible state-of-the-art segmentation pipelines to researchers in fields, such as developmental biology. Here, I present PlantSeg: a pipeline for volumetric segmentation of plant tissues into cells. It follows a graph-based image partitioning (see Section 1.1.1): an approach used to successfully segment neuroimages in the past [1, 77, 8, 45]. PlantSeg not only delivers accurate segmentation results, but given its large repository of models trained on fixed and live images coming from different microscope modalities, it can be used to segment novel 3D datasets directly, significantly reducing the time needed for quantitative analysis of volumetric images.

I present PlantSeg applications in the context of developmental biology, which requires accurate segmentation of individual cells in volumetric images as a basis for further analysis of cellular dynamics.

This chapter is based on the publication [142], where several authors have contributed. I have created the package for the network training, chosen the model architecture appropriate for different image modalities and run the experiments. Lorenzo Cerrone has equally contributed to the experiments presented in the paper and implemented the graphical user interface. Additionally, based on the segmentation results from PlantSeg: Athul Vijayan and Rachele Tofanelli performed the analysis of cell number in the ovule primordia, Amaya Vilches Barro and Marion Louveaux analysed the asymmetric division of the lateral root founder cell, Susanne Steigleder and Amaya Vilches Barro manually curated the ground truth segmentation for the later root dataset, Christian Wenzl performed cell size comparison between the mutant and the wild type of *Arabidopsis thaliana*, Sören Strauss, David Wilson-Sánchez and Rena Lymbouridou analyzed the leaf growth and differentiation.

2.1 Introduction

Large-scale quantitative study of morphogenesis in a multicellular organism entails an accurate estimation of the shape of all cells across multiple specimen. State-of-the-art light microscopes allow for such analysis by capturing the anatomy and development of plants and animals in terabytes of high-resolution volumetric images. With such microscopes now in routine use, segmentation of the resulting images has become a major bottleneck in the downstream analysis of large-scale imaging experiments. A few segmentation pipelines have been proposed [40, 122], but these either do not leverage recent developments in the field of computer vision or are difficult to use for non-experts.

With a few notable exceptions, such as the Brainbow experiments [138], imaging cell shape during morphogenesis relies on staining of the plasma membrane with a fluorescent marker. Segmentation of cells is then performed based on their boundary prediction. In the early days of computer vision, boundaries were usually found by edge detection algorithms [18]. More recently, a combination of edge detectors and other image filters was commonly used as input for a machine learning algorithm, trained to detect boundaries [90]. Currently, the most powerful boundary detectors are based on Convolutional Neural Networks (CNNs) [86, 73, 145]. In particular, the U-Net architecture [113] has demonstrated excellent performance on 2D biomedical images and has later been further extended to process volumetric data [21].

Once the boundaries are found, other pixels need to be grouped into objects delineated by the detected boundaries. For noisy, real-world microscopy data, this post-processing step still represents a challenge and has attracted a fair amount of attention from the computer vision community [130, 105, 8, 140, 45]. If centroids ("seeds") of the objects are known or can be learned, the problem can be solved by the watershed algorithm [31, 22]. For example, in [35] a 3D U-Net was trained to predict cell contours together with cell centroids as seeds for watershed in 3D confocal microscopy images. This method, however, suffers from the usual drawback of the watershed algorithm: misclassification of a single cell centroid results in sub-optimal seeding and leads to segmentation errors.

Recently an approach combining the output of two neural networks and watershed to detect individual cells showed promising results on segmentation of cells in 2D [136]. Although this method can in principle be generalized to 3D images, the necessity to train two separate networks poses additional difficulty for non-experts.

While deep learning-based methods define the state-of-the-art for all image segmentation problems, only a handful of software packages strive to make them accessible to non-expert users in biology (reviewed in [101]). Notably, the U-Net segmentation plugin for ImageJ [37] conveniently exposes U-Net predictions and computes the final segmentation from simple thresholding of the probability maps. CDeep3M [53] and DeepCell [132] enable, via the command-line, the thresholding of the probability maps given by the network, and DeepCell allows instance segmentation as described in [136]. Although not available at the time of PlantSeg development, CellPose [123], a general segmentation algorithm for cellular data, has been introduced recently. More advanced post-processing methods are provided by the ilastik Multicut workflow [13], which can be integrated with CNN-based prediction.

Here, I present PlantSeg, a deep learning-based pipeline for volumetric instance segmentation of dense plant tissues at single-cell resolution. PlantSeg processes the output from the microscope with a CNN to produce an accurate prediction of cell boundaries. Building on the insights from previous work on cell segmentation in electron microscopy volumes of neural tissue [8, 45, 129], the second step of the pipeline delivers an accurate segmentation by solving a graph partitioning problem. I trained PlantSeg on 3D confocal images of fixed *Arabidopsis thaliana* ovules and 3D+t light sheet microscope images of developing lateral roots, two standard imaging modalities in the studies of plant morphogenesis. I investigated a range of network architectures and graph partitioning algorithms and selected the ones which performed best with regard to manually annotated ground truth. PlantSeg was benchmarked on a variety of datasets covering a range of samples and image resolutions. Overall, PlantSeg delivers excellent results on unseen data and, as I show through quantitative and qualitative evaluation, even non-plant datasets do not necessarily require network retraining. Combining the repository of accurate neural networks trained on the two common microscope modalities and going beyond just thresholding or watershed with robust graph partitioning strategies is the main strength of the package. PlantSeg is an open-source tool which contains the complete pipeline for segmenting large volumes. Each step of the pipeline can be adjusted via a convenient graphical user interface while expert users can modify configuration files and run PlantSeg from the command line. Users can also provide their own pre-trained networks for the first step of the pipeline using a popular 3D U-Net implementation (https://github.com/wolny/pytorch-3dunet), which was developed as a part of this project. Although PlantSeg was designed to segment 3D images, 2D data can be segmented as well. Besides the tool itself, all training data[1] together with the pre-trained networks are publicly available.

2.2 Methods

The segmentation pipeline consists of two major steps (see Figure 2.1). In the first step (Section 2.2.1), a fully convolutional neural network is trained to predict cell boundaries. Afterwards, a region adjacency graph is constructed from the pixels with edge weights computed from the boundary predictions. In the second step (Section 2.2.2), the final segmentation is computed as a partitioning of this graph into an unknown number of objects. Graph-based segmentation methods have been outlined in Section 1.1.1. This design is based on a body of work on segmentation of cells in electron microscopy images of neural tissue, where similar methods have been shown to outperform more simple post-processing of the boundary maps [8, 45, 129]. My main contribution is a collection of accurate CNNs for cell membrane segmentation combined with a number of robust graph partitioning strategies in a single cell segmentation package. I trained the networks on two core datates (see below) of confocal

[1]All datasets used to support the findings of this study have been deposited in https://osf.io/uzq3w

and light sheet microscopy images, two common modalities used to study plant and animal morphogenesis. I open-sourced the two datasets to enable further development of efficient cell segmentation algorithms. I have also performed extensive ablation studies, which guided the choice of default hyperparameters, hiding complexity from the end users.

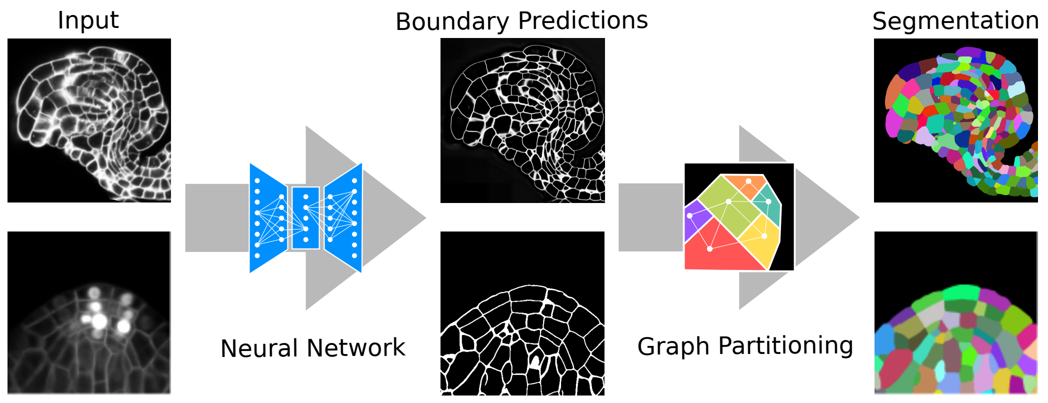


Figure 2.1: Segmentation of plant tissues into cells using PlantSeg. First, PlantSeg uses a 3D UNet neural network to predict the boundaries between cells. Second, a volume partitioning algorithm is applied to segment each cell based on the predicted boundaries. The neural networks were trained on ovules (top, confocal laser scanning microscopy) and lateral root primordia (bottom, light sheet microscopy) of *Arabidopsis thaliana*.

Datasets To make PlantSeg as generic as possible, I used both fixed and live samples for design, validation and testing of the models. The confocal and light sheet stacks, two microscope modalities common in studies of morphogenesis, were employed.

The first dataset consists of fixed *Arabidopsis thaliana* ovules at all developmental stages acquired by confocal laser scanning microscopy with a voxel size of $0.075 \times 0.075 \times 0.235$ μm. 48 volumetric stacks with hand-curated ground truth segmentation were used. A complete description of the image acquisition settings and the ground truth creation protocol is reported in [126].

The second dataset is composed of three time-lapse videos showing the development of *Arabidopsis thaliana* lateral root primordia (LRP). Each recording was obtained by imaging wild-type Arabidopsis plants expressing markers for the plasma membrane and the nuclei [135] using a light sheet fluorescence microscope (LSFM). Stacks of images were acquired every 30 minutes with constant settings across movies and time points, with a voxel size of $0.1625 \times 0.1625 \times 0.250$ μm. The first movie consists of 52 time points of size 2048×1050

$\times$ 486 voxels. The second movie consists of 90 time points of size 1940 $\times$ 1396 $\times$ 403 voxels and the third one of 92 time points of size 2048 $\times$ 1195 $\times$ 566 voxels. The ground truth was generated for 27 images depicting different developmental stages of LRP coming from the three movies (see below).

The two datasets were acquired on different types of microscopes and differ in image quality. To quantify the differences I used the peak signal-to-noise (PSNR) and the structural similarity index measure (SSIM) [59]. I computed both metrics using the input images and their ground truth boundary masks; higher values show better quality. The average PSNR measured on the light sheet dataset was 22.5 ± 6.5 dB (average ± SD), 3.4 dB lower than the average PSNR computed on the confocal dataset (25.9 ± 5.7). Similarly, the average SSIM is 0.53 ± 0.12 for the light sheet, 0.1 lower than 0.63 ± 0.13 value measured on the confocal images. Both datasets thus contain a significant amount of noise. LSFM images are noisier and more difficult to segment, not only because of the noise, but also due to part of nuclear labels being in the same channel as membrane staining.

Groundtruth Creation I briefly describe the groundtruth creation process for the challenging lateral root primorida dataset.

I started by segmenting the cell membranes using sparse user input (scribbles) with the Autocontext Workflow [128] of ilastik software [13]. Then, based on the membrane segmentation, I segmented individual cells using ilastik's Multicut Workflow [8]. The initial cell segmentation was refined iteratively as shown in Figure 2.2. First, the segmentation was manually proofread in selected regions by an expert biologists using Paintera [54] software. Second, a CNN was trained, using the PlantSeg package, to predict the cell boundaries on the manually corrected regions. Third, the full PlantSeg pipeline was applied to the entire dataset resulting in a more accurate segmentation of the image. From this point, the procedure of manual correction, CNN training and PlantSeg segmentation was repeated multiple times, until an accurate cell segmentation was reached. In the end, the ground truth segmentation was created for 27 out of 234 volumes across three movies. The whole process took around 5 months to complete.

2.2.1 Cell Boundary Segmentation

Being the current state of the art in bioimage segmentation, U-Net [113] was chosen as the base model architecture for predicting the boundaries between cells. Aiming for the best performance across different microscope modalities, I explored various aspects of neural network training such as: the network architecture, loss function, data augmentation, normalization layers and the size of patches used for training.
With regard to the network architecture, I compared the regular 3D U-Net as described in [21] with a Residual U-Net from [77]. I tested two loss functions commonly used for the semantic

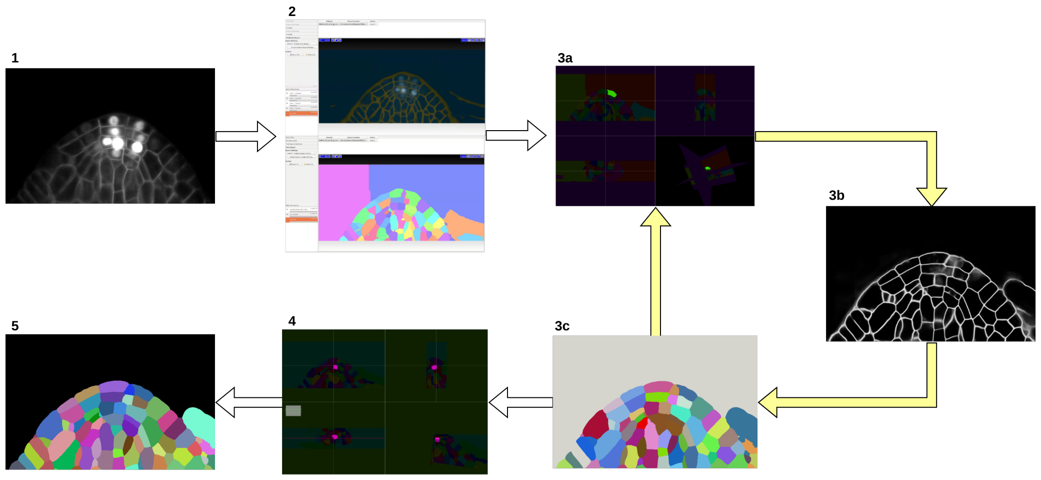


Figure 2.2: The groundtruth creation process. Starting from the input image (1), an initial segmentation is created with ilastik's Autocontext and Multicut Workflow (2). After a round of manual proofreading (3a) a 3D UNet is trained for boundary segmentation (3b). Then, PlantSeg is used to segment the volume (3c). Steps 3a, 3b and 3c are repeated until a final round of manual proofreading (4) which results in the groundtruth labels (5).

segmentation task: binary cross-entropy (BCE) ($\mathcal{L}_{BCE}$) [113] and Dice loss ($\mathcal{L}_{Dice}$) [124], as well as their linear combination termed BCE-Dice. The patch size and normalization layers were investigated jointly by comparing training on a single large patch, versus training on multiple smaller patches per network iteration. Group normalization [143] and standard batch normalization [62] were chosen in the single-patch and multi-patch settings respectively. I used random horizontal and vertical flips, random rotations in the XY-plane, elastic deformations [113] and noise augmentations (additive Gaussian, additive Poisson) of the input image during training in order to increase the network generalization on unseen data.

For the final PlantSeg package, I trained one set of CNNs on the ovules and the other on the lateral root, due to substantial difference in the two datasets.
In the ovule dataset, 39 stacks were randomly selected for training, two for validation and seven for testing. In the LRP dataset, 27 time points were randomly selected from the three videos for training, two time points were used for validation and four for testing.

In more detail, the 2D and 3D U-Nets were trained to predict the binary cell boundaries. Ground truth boundaries were generated from the ground truth cell labeling by the $find\_boundaries(\cdot)$ function from the Scikit-image package [131]. Resulting 2 voxels-thick boundaries between labeled regions were additionally processed with Gaussian smoothing to reduce the high frequency components in the boundary image. It helps to prevent over-fitting and makes the boundaries thicker, increasing the amount of foreground signal during training. Transforming the label image $\mathcal{S}_{\mathbf{x}}$ into the boundary image $\mathcal{I}_{\mathbf{x}}$ is given by Equation 2.1.

$$\mathcal{I}_{\mathbf{x}} = \begin{cases} 1 & \text{if } \Phi(\mathcal{S}_{\mathbf{x}}) * G_{\sigma} > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

Where $\Phi(\cdot)$ transforms the labeled volume into the boundary image, G_{σ} is the isotropic Gaussian kernel and $*$ denotes a convolution operator. I use $\sigma = 1.0$ in the experiments. Both standard and residual U-Net architectures were trained using Adam optimizer [71] with $\beta_1 = 0.9, \beta_2 = 0.999$, L2 penalty of 0.00001 and initial learning rate $\epsilon = 0.0002$. Networks were trained until convergence for 150K iterations, using the PyTorch framework [108]. For validation during training, I used the adjusted Rand (ARand) error computed between the ground truth segmentation and segmentation obtained by thresholding the probability maps predicted by the network and running the connected components algorithm. The learning rate was being reduced by a factor of 2 once the learning stagnated during training, i.e no improvements were observed on the validation set for a given number of iterations. Network with lowest ARand error was selected. For training with small patch sizes I used batch normalization and 4 patches of shape $100 \times 100 \times 80$ per network iteration. When training with a single large patch (size $170 \times 170 \times 80$) batch normalization statistics are noisy, so I replaced batchnorm with groupnorm layers. All networks use the same layer ordering where the normalization layer is followed by the 3D convolution and a rectified linear unit (ReLU) activation. This order of layers consistently performed better than alternative orderings. During training and inference, input images were standardized by subtracting mean intensity and dividing by the standard deviation.

The performance of CNNs is sensitive to changes in voxel size and object sizes between training and test images [104]. For that reason I also trained the networks using the original datasets downscaled by a factor of 2 and 3 in the XY dimension.

All released networks were trained according to the procedure described above using a combination of binary cross-entropy and Dice loss:

$$\mathcal{L} = \mathcal{L}_{BCE} + \lambda \mathcal{L}_{Dice} \tag{2.2}$$

($\lambda = 1$ in my experiments). 3D U-Nets trained at different scales of the two core datasets (light-sheet lateral root, confocal ovules) are made available as part of the PlantSeg package. For completeness, 2D U-Nets trained using the Z-slices from the original 3D stacks are also published, enabling segmentation of 2D images with PlantSeg.

During inference I parsed the volume patch-by-patch with a 50% overlap between consecutive tiles and average the probability maps. This strategy prevents checkerboard artifacts and reduces noise in the final prediction. The code used for training and inference can be found at https://github.com/wolny/pytorch-3dunet.

The best performing CNN architectures and training procedures is illustrated by the precision/recall curves evaluated at different threshold levels of the predicted boundary probability maps (see Figure 2.3). Training with a combination of binary cross-entropy and Dice loss performed best on average across the two datasets in question contributing to 3 out of 6 best performing network variants. BCE-Dice loss also generalized well on the out of sample data described in Section 2.3.1. Due to the regularity of cell shapes, the networks do not benefit from broader spatial context when only cell membrane signal is present in input images. Indeed, training the networks with bigger patch sizes does not noticeably increase the performance as compared to training with smaller patches. 4 out of 6 best performing networks use smaller patches and batch normalization (*Batch-Norm*) whereas only 2 out of 6 use bigger patches and group normalization (*Group-Norm*). Residual U-Net architecture (*3D-ResUnet*) performed best on the LRP dataset (Figure 2.3 (B)), whereas standard U-Net architecture (*3D-Unet*) was better on the ovule datasets (Figure 2.3 (A)). In conclusion, choosing the right loss function and normalization layers increased the final performance on the task of boundary prediction on both microscope modalities.

2.2.2 Segmentation Using Graph Partitioning

After the cell boundaries are predicted, segmentation of the cells can be formulated as a generic graph partitioning problem. The boundary predictions produced by the CNN are treated as a graph $G(V, E)$, where nodes V are represented by the image voxels, and the edges E connect adjacent voxels. The weight $w \in R^+$ of each edge is derived from the boundary probability maps. Solving the partitioning problem directly at voxel-level is computationally expensive for volumes of biologically relevant size. To make the computation tractable, the voxels are clustered into so-called supervoxels by running the DT watershed [31] on the distance transform of the boundary map. For this, I threshold the boundary probability maps at a given value δ to get a binary image ($\delta = 0.4$ was chosen empirically in my experiments). Then I compute the distance transform from the binary boundary image, apply a Gaussian smoothing ($sigma = 2.0$) and assign a seed to every local minimum in the resulting distance transform map. After supervoxels are generated with the watershed transform, the region adjacency graph (RAG) is constructed. In the region adjacency graph each node represents a supervoxel and edges connect adjacent supervoxels. The edge weights are computed by using the mean value of the probabilities maps along the boundary. Finally, the region adjacency graph is partitioned into an unknown number of segments to deliver a segmentation. I tested

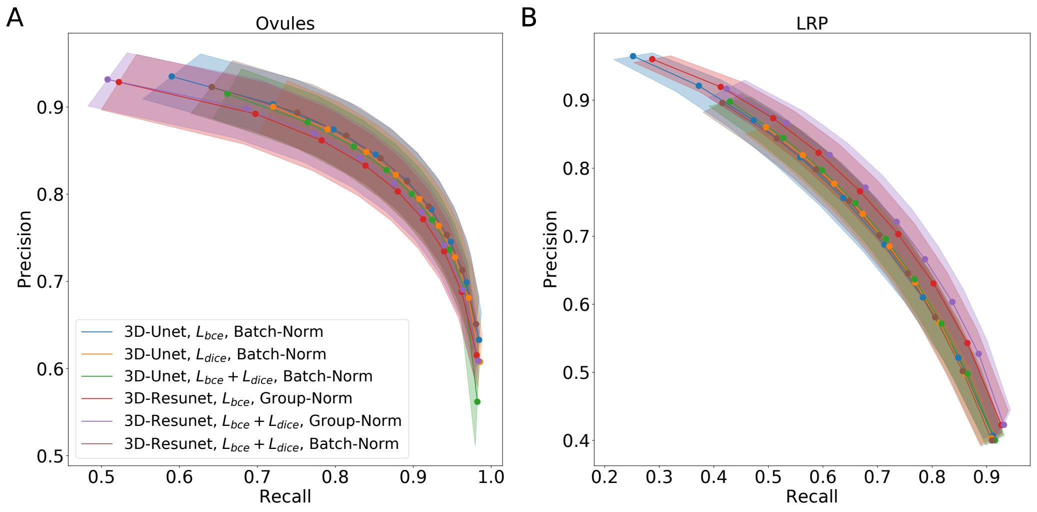


Figure 2.3: Precision-recall curves for different CNN variants on the ovule (A) and lateral root primordia (LRP) (B) datasets. Six training procedures that sample different type of architecture (3D U-Net *vs.* 3D Residual U-Net), loss function (BCE *vs.* Dice *vs.* BCE-Dice) and normalization (Group-Norm *vs.* Batch-Norm) are shown. Those variants were chosen based on the accuracy of boundary prediction task: 3 best performing models on the ovule and 3 best performing models on the lateral root datasets. Points correspond to averages of seven (ovules) and four (LRP) values and the shaded area represent the standard error.

four different partitioning strategies: Multicut [66], hierarchical agglomeration as implemented in GASP average (GASP) [4], Mutex watershed (Mutex) [140] as well as the distance transform watershed [31] as a baseline since similar methods have been proposed previously [35, 136].

To quantify the accuracy of the four segmentation strategies I use Adapted Rand error (ARand) for the overall segmentation quality and two other metrics based on the variation of information [95], measuring the tendency to over-split ($\mathrm{VOI}_{\mathrm{split}}$) or over-merge ($\mathrm{VOI}_{\mathrm{merge}}$). GASP, Multicut and Mutex watershed consistently produced accurate segmentation on both datasets with low ARand errors and low rates of merge and split errors (Figure 2.4 A-C). As expected the watershed tends to over-segment with higher split error and resulting higher ARand error. Multicut solves the graph partitioning problem in a globally optimal way and is therefore expected to perform better compared to greedy algorithms such as GASP and Mutex watershed. However, in this case the gain was marginal, probably due to the high quality of the boundary predictions.

The performance of PlantSeg was also assessed qualitatively by expert biologists. The segmentation quality for both datasets is very satisfactory. For example in the lateral root dataset, even in cases where the boundary appeared masked by the high brightness of the nuclear signal, the network correctly detected it and separated the two cells (Figure 2.4 D, green box). On the ovule dataset, the network is able to detect weak boundaries and correctly separate cells in regions where the human expert fails (Figure 2.4 E , green box). The main mode of error identified in the lateral root dataset is due to the ability of the network to remove the nuclear signal which can weaken or remove part of the adjacent boundary signal leading to missing or blurry cell contour. For example, the weak signal of a newly formed cell wall close to two nuclei was not detected by the network and the cells were merged (Figure 2.4 D, red box). For the ovule dataset, in rare cases of very weak boundary signal, failure to correctly separate cells could also be observed (Figure 2.4 E, red box).

Taken together, my analysis shows that segmentation of plant tissue using graph partitioning handles robustly boundary discontinuities present in plant tissue segmentation problems.

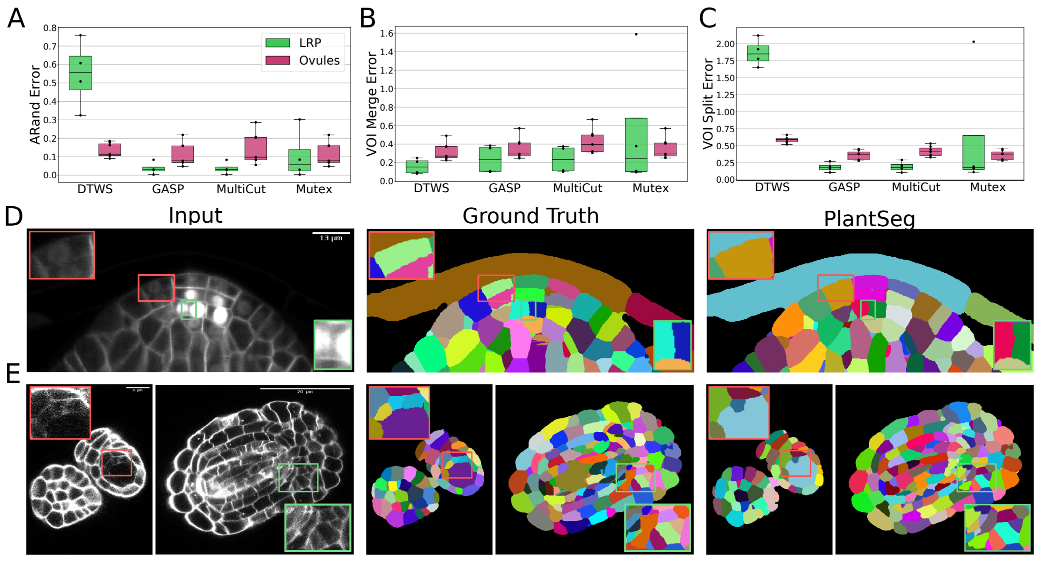


Figure 2.4: Segmentation using graph partitioning. (A-C) Quantification of segmentation produced by Multicut, GASP, Mutex watershed (Mutex) and DT watershed (DT WS) partitioning strategies. The Adapted Rand error (A) assesses the overall segmentation quality whereas $\mathrm{VOI}_{\mathrm{merge}}$ (B) and $\mathrm{VOI}_{\mathrm{split}}$ (C) assess erroneous merge and splitting events (lower is better). Box plots represent the distribution of values for seven (ovule, magenta) and four (LRP, green) samples. (D, E) Examples of segmentation obtained with PlantSeg on the lateral root (D) and ovule (E) datasets. Green boxes highlight cases where PlantSeg resolves difficult cases whereas red ones highlight errors. I obtained the boundary predictions using the *generic-confocal-3d-unet* for the ovules dataset and the *generic-lightsheet-3d-unet* for the root. All agglomerations have been performed with default parameters. 3D superpixels instead of 2D superpixels were used.

2.3 Results

Having trained the networks on the core datasets, I evaluated the effectiveness of the pipeline on external datasets of plant (Section 2.3.1) and animal tissues (Section 2.3.2) without retraining the CNNs.

2.3.1 Performance on External Plant Datasets

To test the generalization capacity of PlantSeg, I assessed its performance on data for which no network training was performed. To this end, I took advantage of the two publicly available

Dataset	PlantSeg (default parameters)			PlantSeg (tuned parameters)		
	ARand	VOI_{split}	VOI_{merge}	ARand	VOI_{split}	VOI_{merge}
Anther	0.328	0.778	0.688	0.167	0.787	0.399
Filament	0.576	1.001	1.378	0.171	0.687	0.487
Leaf	0.075	0.353	0.322	0.080	0.308	0.220
Pedicel	0.400	0.787	0.869	0.314	0.845	0.604
Root	0.248	0.634	0.882	0.101	0.356	0.412
Sepal	0.527	0.746	1.032	0.257	0.690	0.966
Valve	0.572	0.821	1.315	0.300	0.494	0.875

Table 2.1: Quantification of PlantSeg performance on the 3D Digital Tissue Atlas, using PlantSeg . The Adapted Rand error (ARand) assesses the overall segmentation quality whereas VOI_{merge} and VOI_{split} assess erroneous merge and splitting events. The petal images were not included in the analysis. They are very similar to the leaf and the ground truth is fragmented, making it difficult to evaluate the results in an objective way. Segmented images are computed using GASP partitioning with default parameters (left table) and fine-tuned parameters (right table).

datasets: Arabidopsis 3D Digital Tissue Atlas (`https://osf.io/fzr56`) composed of eight stacks of eight different *Arabidopsis thaliana* organs with hand-curated groundtruth, as well as the developing leaf of the Arabidopsis [42] with 3D segmentation given by the SimpleITK package [88]. The input images from the digital tissue atlas are confocal stacks of fixed tissue with stained cell contours and thus similar to the images of the Arabidopsis ovules, whereas the images of the leaf were acquired through the use of live confocal imaging. It's important to note that the latter image stacks contain highly lobed epidermal cells, which are difficult to segment with classical watershed-based algorithms. The confocal stacks were processed by PlantSeg and the resulting segmentation assessed qualitatively. Quantitative assessment was performed only for the digital tissue atlas, where the ground truth labels are available.

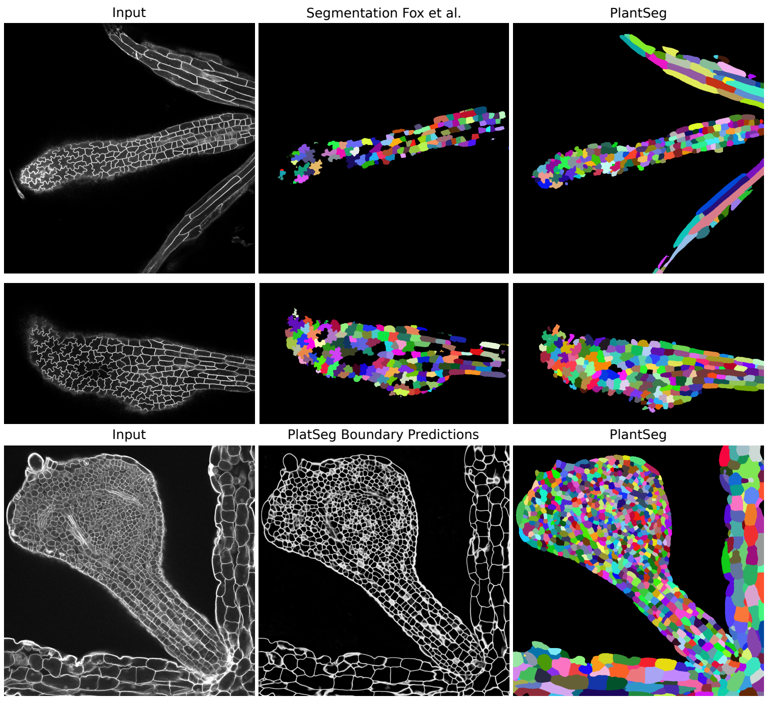


Figure 2.5: Qualitative results on the highly lobed epidermal cells from [42]. First two rows show the visual comparison between the SimpleITK (middle) and PlantSeg (right) segmentation on two different image stacks. PlantSeg's results on another sample is shown in the third row. In order to show pre-trained networks' ability to generalized to external data, I additionally depict PlantSeg's boundary predictions (third row, middle). I obtained the boundary predictions using the *generic-confocal-3d-unet* and segmented using GASP with default values. A value of 0.7 was choosen for the under/over segmentation factor.

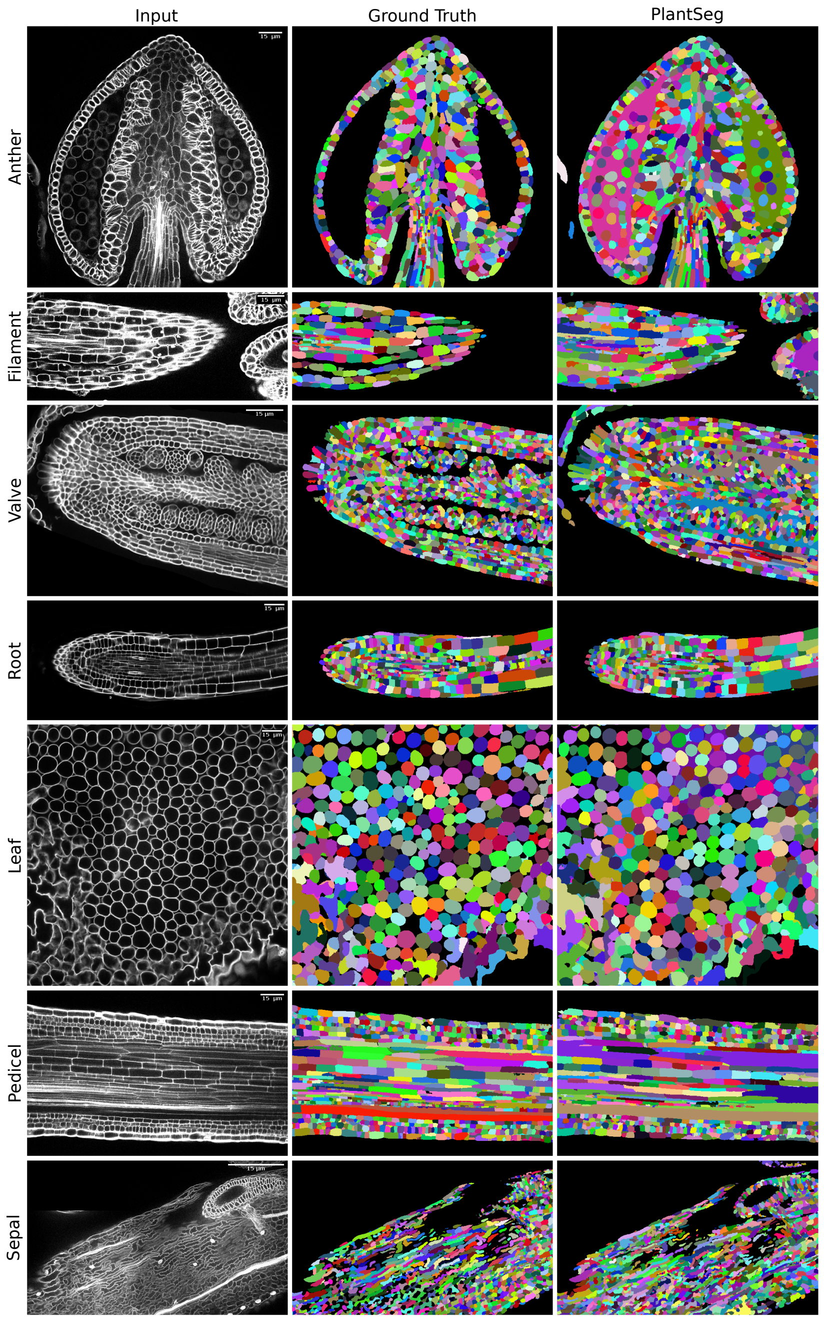


Figure 2.6: PlantSeg segmentation of different plant organs of the 3D Digital Tissue Atlas dataset, not seen in training. The input image, ground truth and segmentation results using PlantSeg are presented for each indicated organ.

Qualitatively, PlantSeg performed well on both datasets, giving satisfactory results on all organs from the 3D Digital Tissue Atlas, correctly segmenting even the oval non-touching cells of the anther and leaf: a cell shape not present in the training data (Figure 2.6). PlantSeg yielded especially good segmentation results when applied to the complex epidermal cells, visibly outperforming the results obtained using the SimpleITK framework (Figure 2.5).

Quantitatively, the performance of PlantSeg out of the box (default parameters) on the 3D Digital Tissue Atlas is on par with the scores reported on the LRP and ovules datasets on the anther, leaf, and the root, but lower for the other tissues (Table 2.1, left). Default parameters have been chosen to deliver good results on most type of data, however a substantial improvement can be obtained by parameter tuning. In case of the tissue 3D Digital Tissue Atlas (1) scaling the voxel size to reduce the resolution gap between the training data and the 3D Tissue Atlas and (2) increasing the over-segmentation factor to 0.7, improved segmentation by a factor of two as measured with the ARand error (Table 2.1, right). It should be noted that the ground truth included in the dataset was created for analysis of the cellular connectivity network, with portions of the volumes missing or having low quality ground truth (see e.g filament and sepal in Figure 2.6). For this reason, the performance of PlantSeg on these datasets may be underestimated.

Altogether, PlantSeg performed well qualitatively and quantitatively on datasets acquired by different groups, on different microscopes, and at different resolutions than the training data. This demonstrates the generalization capacity of the pre-trained models from the PlantSeg package.

2.3.2 Performance on a Non-plant Benchmark

For completeness, I compared PlantSeg performance with state-of-the-art methods on an open benchmark consisting of 2D+t videos of membrane-stained developing *Drosophila* epithelial cells [43]. Treating the movie sequence as 3D volumetric images not only resembles the plant cell images shown in this study, but also allows to pose the 2D+t segmentation as a standard 3D segmentation problem.

I compared the performance of PlantSeg on the 8 movies of this dataset to the four reported pipelines: MALA [46], Flood Filling Networks (FFN) [63], Moral Lineage Tracing (MLT) [64, 111] and Tissue Analyzer (TA) [43]. PlantSeg was evaluated in two settings. In the first one, I did not train the CNNs on the benchmark datasets, but used the CNNs provided with the PlantSeg package. This experiment gives an estimate of how well the pre-trained networks generalize to non-plant tissues. For the second evaluation, I retrained the network on the benchmark's training set to compare with state-of-the-art. Note that unlike other methods reported in the benchmark, I do not introduce additional constraints to account for the data being 2D+t rather than 3D, i.e. I do not enforce the lineages to be moral [43].

For the first experiment, peripodial cells were segmented using the 3D U-Net trained on the ovule dataset together with GASP segmentation, whereas proper disc cells were segmented with 2D U-Net trained on the ovule dataset in combination with Multicut algorithm. Both networks are part of the PlantSeg package. Qualitative results are shown in Figure 2.7: PlantSeg produces highly accurate segmentation on both the peripodial and proper imaginal disc cells. A few over-segmentation (peripodial cells) and under-segmentation (proper disc) errors are marked in the figure. This impression is confirmed by quantitative benchmark results in Table 2.2.

For the second experiment, I trained the network on the ground truth labels included in the benchmark (*PlantSeg (trained)*). Here, my pipeline is comparable to state-of-the-art. The difference in SEG metric between "vanilla" PlantSeg and *PlantSeg (trained)* is 6.9 percent points on average, which suggests that for datasets sufficiently different from the ones PlantSeg networks were trained on, re-training the models might be necessary. Looking at the average run-times of the methods reported in the benchmark shows that PlantSeg pipeline is clearly the fastest approach with the average run-time of 3 min per movie when run on a server with a modern GPU versus 35 min (MALA), 42 min (MLT) and 90 min (FFN).

I argue that the collection of pre-trained networks and graph partitioning algorithms make PlantSeg versatile enough to work well on wide variety of membrane stained tissues, beyond plant samples.

Method	PERIPODIAL	PROPER DISC
MALA	0.907 ± 0.029	0.817 ± 0.009
FFN	0.879 ± 0.035	0.796 ± 0.013
MLT-GLA	0.904 ± 0.026	0.818 ± 0.010
TA	-	0.758 ± 0.009
PlantSeg	0.787 ± 0.063	0.761 ± 0.035
PlantSeg (trained)	0.885 ± 0.056	0.800 ± 0.015

Table 2.2: Epithelial Cell Benchmark results. I compare PlantSeg to four other methods using the standard SEG metric [93] calculated as the mean of the Jaccard indices between the reference and the segmented cells in a given movie (higher is better). Mean and standard deviation of the SEG score are reported for peripodial (3 movies) and proper disc (5 movies) cells. Additionally, I report the scores of PlantSeg pipeline executed with a network trained explicitly on the epithelial cell dataset (last row).

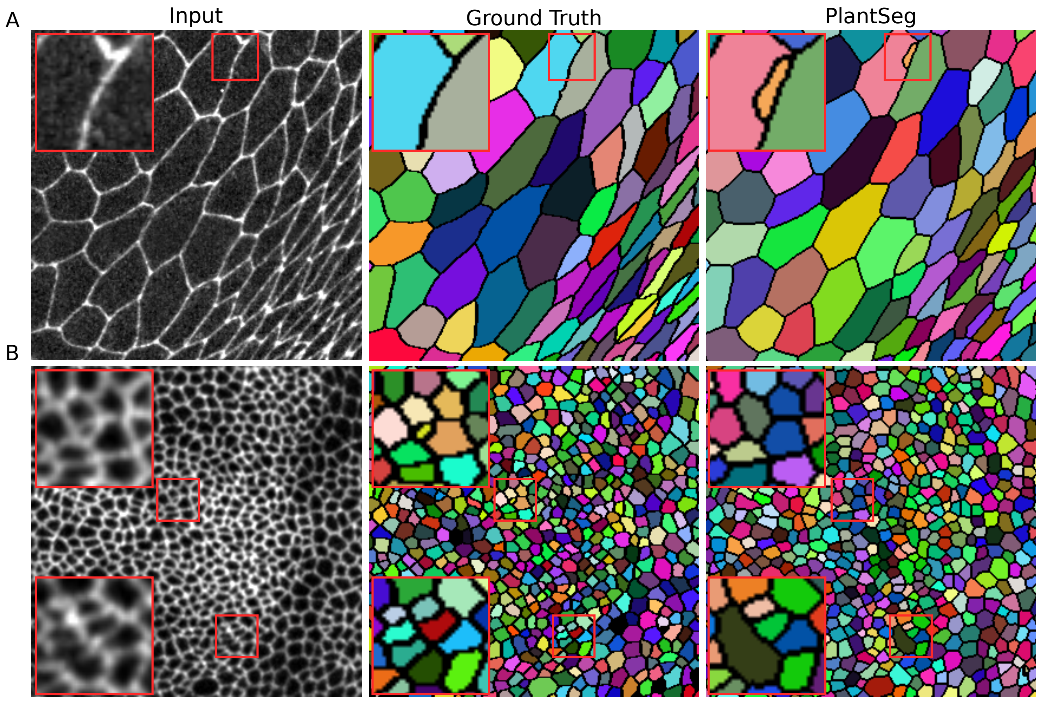


Figure 2.7: Qualitative results on the Epithelial Cell Benchmark. From top to bottom: Peripodial cells (A), Proper disc cells (B). From left to right: raw data, ground truth segmentation, PlantSeg segmentation results. PlantSeg provides accurate segmentation of both tissue types using only the networks pre-trained on the *Arabidopsis* ovules dataset. Red rectangles show sample over-segmentation (A) and under-segmentation (B) errors. Boundaries between segmented regions are introduced for clarity and they are not present in the pipeline output.

2.3.3 PlantSeg Applications in Developmental Biology

Together with my collaborators, I demonstrate the usefulness of PlantSeg on four concrete biological applications that require accurate extraction of cell geometries from complex, densely packed 3D tissues. First, PlantSeg allowed to sample the variability in the development of ovules in a given pistil and reveal that those develop in a relatively synchronous manner (Figure 2.8). Second, PlantSeg allowed the precise computation of the volumes of the daughter cells resulting from the asymmetric division of the lateral root founder cell. This division results in a large and a small daughter cells with volume ratio of $\sim \frac{2}{3}$ between them (Figure 2.10). Third, segmentation of the epidermal cells in the shoot apical meristem revealed that these cells are enlarged in the *bce* mutant compared to wild type (Figure 2.11). Finally, we showed that PlantSeg can be used to improve the automated surface segmentation of time-lapse leaf stacks which enables different downstream analyses such as growth tracking at cell resolution (Figure 2.9).

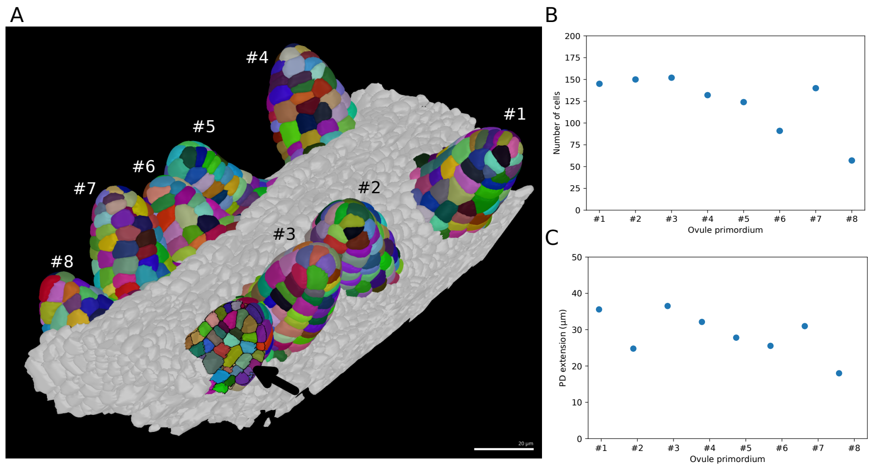


Figure 2.8: Ovule primordium formation in *Arabidopsis thaliana*. (A) 3D reconstructions of individually labelled stage 1 primordia of the same pistil are shown (stages according to [117]). Scale bar: $20\mu m$. The arrow indicates an optical mid-section through an unlabeled primordium revealing the internal cellular structure. The raw 3D image data were acquired by confocal laser scanning microscopy according to [126]. Using MorphographX [6], quantitative analysis was performed on the three-dimensional mesh obtained from the segmented image stack. Cells were manually labelled according to the ovule specimen (from $\#1$ to $\#8$). (B, C) Quantitative analysis of the ovule primordia shown in (A). (B) shows a graph depicting the total number of cells per primordium. (C) shows a graph depicting the proximal-distal (PD) extension of the individual primordia (distance from the base to the tip). Analysis indicates that ovule primordium formation within a pistil is relatively uniform, with some variability, i.e. primordia #6 and #8 exhibited a smaller number of cells.

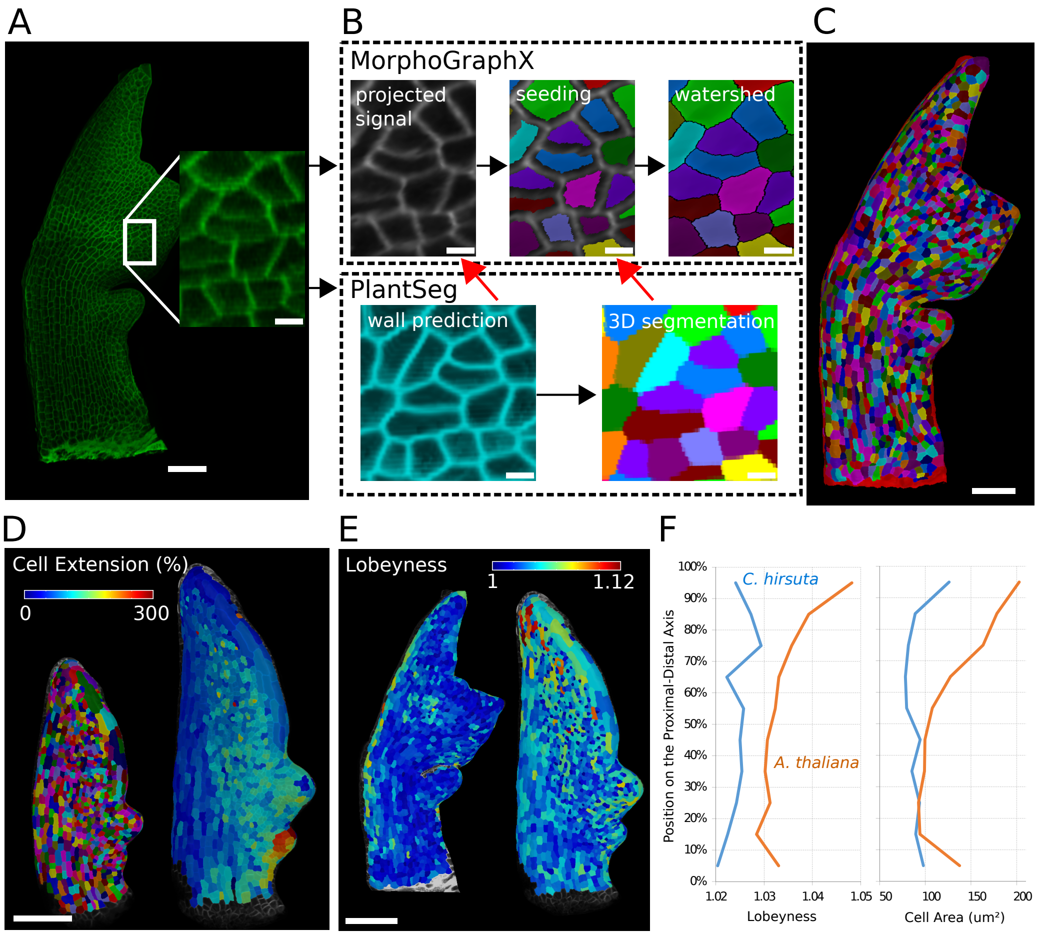


Figure 2.9: Creation of cellular segmentations of leaf surfaces and downstream quantitative analyses. (A-C) Generation of a surface segmentation of a *C. hirsuta* leaf in MorphoGraphX assisted by PlantSeg. (A) Confocal image of a 5-day-old *C. hirsuta* leaf (leaf number 5) with an enlarged region. (B) Top: Segmentation pipeline of MorphoGraphX: a surface mesh is extracted from the raw confocal data and used as a canvas to project the epidermis signal. A seed is placed in each cell on the surface for watershed segmentation. Bottom: PlantSeg facilitates the segmentation process in two different ways (red arrows): By creating clean wall signals which can be projected onto the mesh instead of the noisy raw data and by projecting the labels of the 3D segmentation onto the surface to obtain accurate seeds for the cells. Both methods reduce segmentation errors with the first method to do so more efficiently. (C) Fully segmented mesh in MorphoGraphX. (D-F) Quantification of cell parameters from segmented meshes. (D) Heatmap of cell growth in an *A. thaliana* 8th-leaf 4 to 5 days after emergence. (E) Comparison of cell lobeyness between *A. thaliana* and *C. hirsuta* 600-μm-long leaves. (F) Average cell lobeyness and area in *A. thaliana* and *C. hirsuta* binned by cell position along the leaf proximal-distal axis. Scale bars: $50\mu m$ (A, C), $100\mu m$ (D, E), $5\mu m$ (inset in A, B). Overall, *A. thaliana* leaves showed higher cell size and lobeyness than *C. hirsuta*

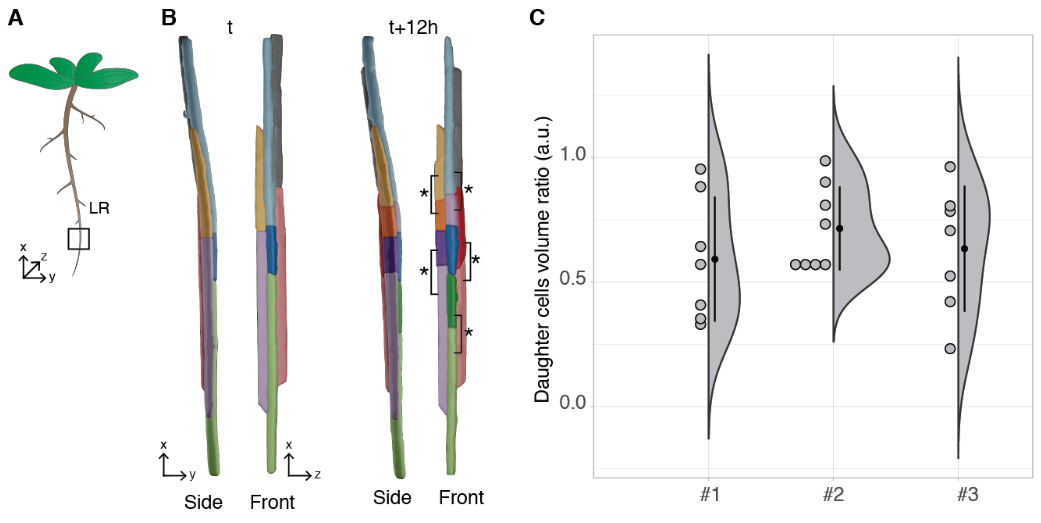


Figure 2.10: Asymmetric cell division of lateral root founder cells. (A) Schematic representation of *Arabidopsis thaliana* with lateral roots (LR). The box depict the region of the main root that initiates LRs. (B) 3D reconstructions of LR founder cells seen from the side and from the front at the beginning of recording (*t*) and after 12 hours (*t+12*). The star and brackets indicate the two daughter cells resulting from the asymmetric division of a LR founder cell. (C) Half-violin plot of the distribution of the volume ratio between the daughter cells for 3 different movies (#1, #2 and #3). The average ratio of 0.6 indicates that the cells divided asymmetrically.

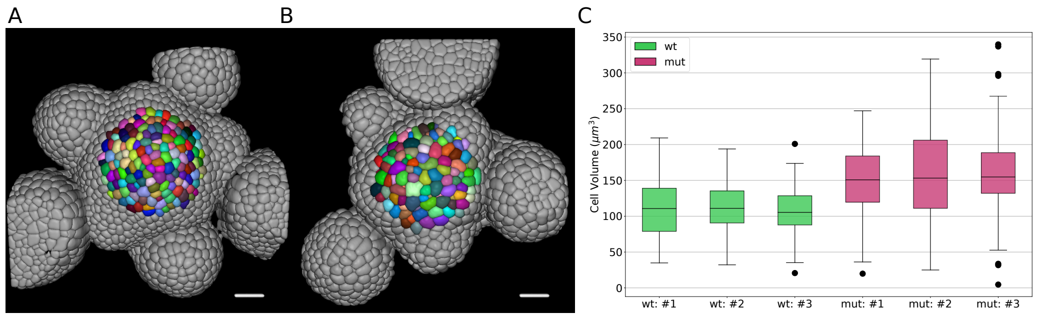


Figure 2.11: Volume of epidermal cell in the shoot apical meristem of Arabidopsis. Segmentation of epidermal cells in *wildtype* (A) and *bce* mutant (B). Cells located at the center of the meristem are colored. Scale bar: $20\mu m$. (C) Quantification of cell volume (μm^3) in 3 different *wildtype* and *bce* mutant specimens. The mean volume of epidermal cells in the *bce* mutant is increased by roughly 50%.

2.4 PlantSeg in Collaborative Research Projects

In this section, I highlight applications of PlantSeg to problems in developmental biology and detection of virus specific antibodies. In each of the three projects my method automates cell segmentation and thus enables further analysis such as the study of cellular dynamics and cell type classification. In each case, I briefly describe my contributions and the biological insights enabled by the quantitative analysis of the segmentation results.

First, in Section 2.4.1 I describe my contribution to the publication [61] by Takafumi Ichikawa et al. In this work, together with Dimitri Fabrèges and Rene Snajder, I have used PlantSeg for automated segmentation of 3D time-lapse images of mouse embryo at the early stage of development. Given the difficulties in creating high quality dense groundtruth segmentation of 3D microscopy images, I proposed and implemented an iterative, human-in-the-loop, network training scheme which minimizes the manual proofreading and still achieves high quality segmentation results. Dimitri Fabrèges was responsible for setting up a large scale grid search for finding optimal hyperparameters for final segmentation.

Second, in Section 2.4.2 I show how PlantSeg helped with segmentation of the challenging 3D light microscopy images of mouse embryo developing in artificial uterus [15] by Vladyslav Bondarenko et al. My contributions include segmentation of fixed 3D images of cell nuclei and live 3D images of cell membranes in an *ex vivo* system.

Finally, in Section 2.4.3 I describe my contribution to [107], a microscopy-based assay for SARS-CoV-2 antibodies detection. Here, PlantSeg routines were used in the image analysis pipeline developed to automatically score antibody response of human sera. This work was spearheaded by Constantin Pape, who implemented a batch processing workflow https://github.com/hci-unihd/batchlib for scalable analysis of the high-throughput screening data, Vibor Laketa, who designed the assay and acquired the images and Roman Remme, who implemented the infected cell classification. My contributions involve groundtruth creation, training the CNNs for cell segmentation, creation of a simple application for labeling the faulty images and setting up a central database for storing all intermediate results to ensure full reproducibility of the experiments.

Apart from the publications mentioned in this section, I also helped to generate a three-dimensional atlas of developing plant ovule in [134]. Here, together with Lorenzo Cerrone, I trained neural networks for cell membrane prediction and made them available via PlantSeg for large scale segmentation of 3D confocal stacks of fixed ovules. The resulting atlas enables quantitative spatio-temporal analysis of cellular and gene expression patterns at cellular and tissue resolution.

2.4.1 *Ex vivo* Development of Mammalian Embryo

In [61] the authors developed an *ex vivo* 3D culture in gel which allows the study of mammalian development around the time of implantation. The 3D system permits the live-imaging of the mouse embryo with light-sheet microscopy, which opens the possibility to study the cellular dynamics through automatic cell segmentation. Based on the quantitative analysis of this cellular dynamics the authors revealed the importance of the mechano-chemical interactions between embryonic and extra-embryonic tissues during early mammalian development.

Automatic Cell Segmentation The segmentation pipeline used to process the 3D images of the mTmG membrane signal consists of four steps. In the first step the 3D input images are down-sampled by a factor of 4 along the XY axes. The dimension of the resulting images is $512 \times 512 \times 400$ voxels with a physical voxel size of $0.832 \times 0.832 \times 1.000 \mu m^3$ (X, Y, Z). In the second step, a dedicated neural network trained with the PlantSeg framework is used to generate a probability map of cell membrane locations. In the third step the probability maps are segmented using a set of algorithms provided by PlantSeg. The best segmentation algorithm and its corresponding hyper-parameters were found by a custom-made pipeline which explored thousands of different configuration parameters simultaneously using EMBL's computing cluster. In the final step, the epiblast (EPI) cells are manually selected from the segmented stack through visual inspection, manually corrected when appropriate, and used for further analysis. An overview of the pipeline steps applied to a sample image is shown in Figure 2.12.

Since no ground truth segmentation was initially available, the high performance of our pipeline was achieved by the following iterative procedure. In the first iteration a pre-trained

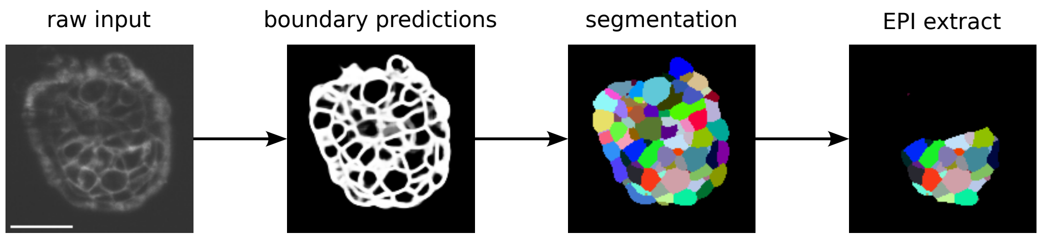


Figure 2.12: Sample image of a representative mouse embryo developing during the first 24h segmented with PlantSeg. A down-sampled cross-section from a sample light-sheet volume (left), boundary predicted by a dedicated PlantSeg's CNN (middle), segmentation produced by GASP image partitioning (right). Input image from Takafumi Ichikawa, Hiiragi Lab, EMBL. Scale bar: 50 μm.

neural network available in the PlantSeg package was used to generate the initial membrane probability maps. In particular, I used a CNN trained on the Arabidopsis ovules dataset named *confocal_unet_bce_dice_ds2*. Having the cell boundary prediction, the initial segmentation was produced with PlantSeg. The segmentation results were improved by visually choosing and cropping around the most correctly segmented regions and using those segments as pseudo labels to train a dedicated neural network for the membrane prediction task. The process of choosing the best segmentation results and re-training the network was performed four times. The resulting network was used together with the PlantSeg pipeline to produce the final segmentation results. This iterative self-training procedure of (1) predicting the boundaries, (2) segmenting with PlantSeg, (3) choosing the best outcomes as pseudo labels and (4) training the boundary network with the pseudo labels was partially motivated by [47]. In this work the authors use a form of knowledge distillation in which the *student* network, parameterized identically as the *teacher* network, is trained to match the output distribution of the teacher. Using this training procedure iteratively, the authors show that after a few iterations students outperform their teachers by significant margins. My scheme is similar in that it uses an iterative self-training with the teacher and the student networks of the same capacity. It differs in several aspects. In the first iteration, the student is trained from a "noisy" labels produced by a teacher network trained on a different data distribution, i.e. the teacher was trained on confocal images of plant tissue and used to predict the boundaries in the light-sheet images of animal tissue. The number of segmentation mistakes is reduced by a human who chooses the most plausible segmentation results, which are then used as a training data for future generations. This human-in-the-loop self-training procedure, shown in Figure 2.13, allowed the PlantSeg pipeline to achieve high quality segmentation results without the need of expensive manual correction of individual cells in 3D.

To sum up, the segmentation results given by the iterative, PlantSeg-based procedure enabled further analysis of the interactions between embryonic and extra-embryonic tissues in early embryogenesis.

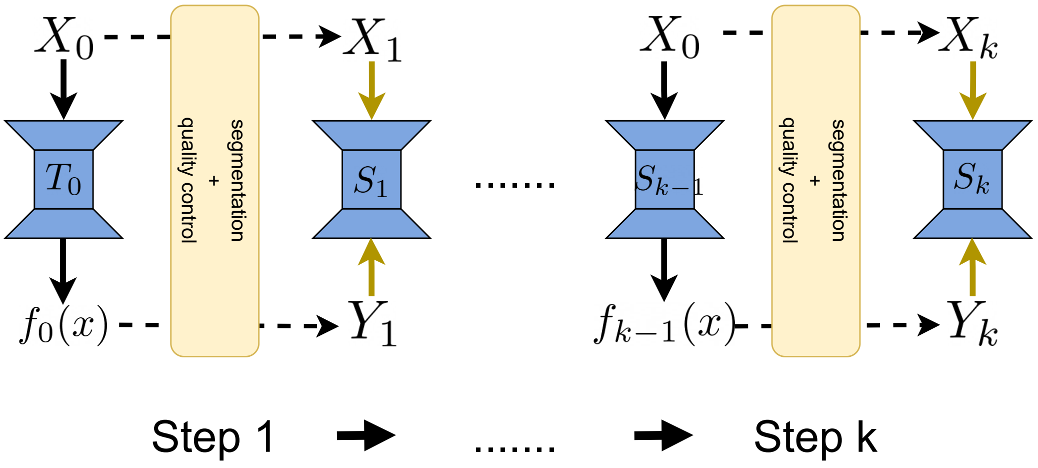


Figure 2.13: Human-in-the-loop self-training procedure used to train the boundary predictor without groundtruth labels. A teacher network T_0 is chosen from the PlantSeg package for initial cell membranes prediction $f_0(x)$ on the input images X_0. The resulting predictions are segmented with PlantSeg, followed by a manual quality control where a human annotator selects the most correctly segmented regions (X_1, Y_1), which are used to train the student network S_1. In the next generation the student becomes the teacher an the procedure is repeated. $k = 4$ iterations were performed to achieve a satisfactory predictor of cell membranes.

2.4.2 The Role of Embryo-Uterus Interactions in Mouse Embryogenesis

In [15], Vladyslav Bondarenko et al. introduced an engineered uterus-like environment which allows to study the embryo-uterus interaction *ex vivo*. My main contribution to this work was segmentation of cells and nuclei in live (light-sheet) and fixed (confocal) images. The segmentation problem was especially challenging, due to a significant amount of noise present in both image modalities. Representative images of the embryos together with the PlantSeg segmentation results are shown in Figure 2.14.
The segmentation results formed the basis for cell counting and cell tracking, which were used to explore the coordination between embryo growth and trophoblast migration. I have

made the neural networks used to segment the cells and nuclei available via the BioImage Model Zoo `https://bioimage.io`. The training data can be downloaded from `https://zenodo.org/record/6546550`.

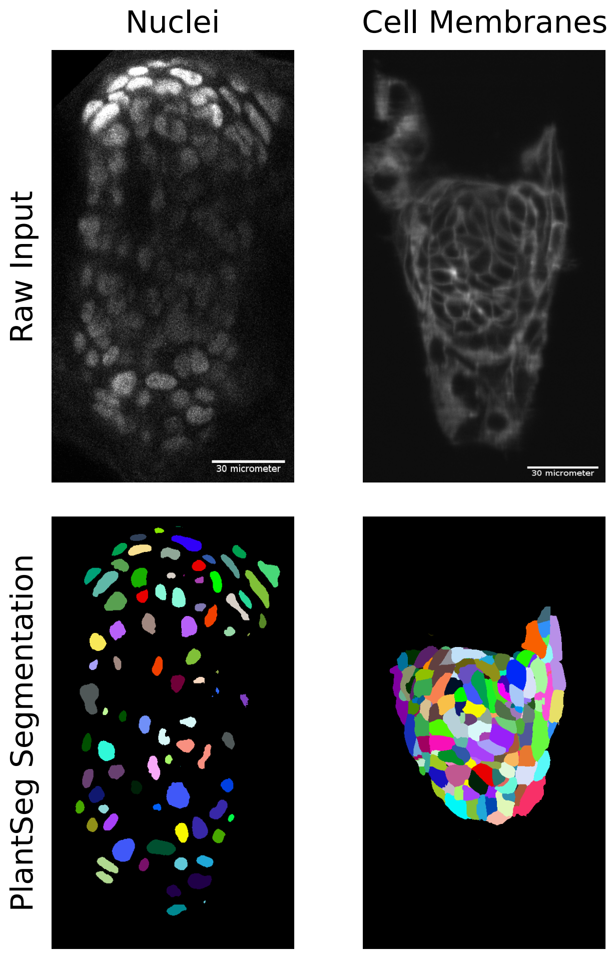


Figure 2.14: Sample segmentation of the nuclei and cells captured during the early stage of mouse embryo development in artificial uterus. **Left**: Sample cross-section from the 3D immunofluorescence image of cell nuclei (top) and the corresponding PlantSeg segmentation (bottom). **Right**: A cross-section from a sample light-sheet volume of the plasma-membrane (top) and the segmentation produced by PlantSeg (bottom). Input images from Vladyslav Bondarenko, Hiiragi Lab, EMBL.

Nuclei Segmentation The 3D confocal volumes were acquired with LSM780 and LSM880 in a confocal mode with a voxel size of $0.207 \times 0.207 \times 1 \mu m^3$ (X, Y, Z). A 3D U-Net [21] was trained with a multi-task objective of predicting the binary nuclei mask in the first output channel and the nuclei outlines in the second channel. The outline predictions were used to segment the individual nuclei using PlantSeg's *MutexWS* [140] partitioning algorithm. The nuclei mask predictions were used to remove the spurious instances in the background. Sample results in Figure 2.14 (left).

Model training was performed iteratively with an increasing amount of training data. Starting from four initial groundtruth volumes, in each iteration, I trained the network and performed the segmentation. The results were manually corrected and included in the training set for the next iteration. In total, 22 training and 13 validation data volumes were used for the final model training.

Membrane-based Cell Segmentation The 3D light-sheet images were acquired with MuVi-SPIM. A dedicated 3D U-Net was trained to predict the foreground membrane mask, which was used for the final cell segmentation with PlantSeg's *GASP* agglomeration algorithm (Figure 2.14 (right)). The groundtruth for the network training was bootstrapped by initially segmenting the stacks with pre-trained PlantSeg models (*confocal_unet_bce_dice_ds2x*), followed by manual correction of the erroneous cells. In total, four annotated stacks were used for training and one for validating the network. Both nuclei and membrane U-Nets were trained until convergence for 100K iterations, using the PyTorch framework [108]. The models with the best score on the validation set were selected.

In summary, my contributions to segmentation of challenging images of early stages of mouse embryo development captured *ex vivo* enabled accurate tracking and counting of cells of interest and further insights about the embryo-uterus interaction in periimplantation development.

2.4.3 Microscopy-based Detection of SARS-CoV-2 Antibodies

An outbreak of the novel pathogenic coronavirus SARS-CoV-2 and its rapid spread pose a global health emergency. Here, I briefly describe my contribution to [107] which introduces a microscopy-based assay for detection of SARS-CoV-2 specific antibodies in human samples. The possibility to detect antibodies against the viral proteins together with a robust image analysis workflow resulted in specific, sensitive assay. Such quantitative serological assays are needed for a better understanding of the immune response against the virus. The procedure described here provides a general framework for the application of quantitative high-throughput microscopy to rapidly develop serological assays for emerging virus infections.

Immunofluorescence Assay Analysis Workflow Cells infected with SARS-CoV-2 are used in the immunofluorescence (IF) assay as samples. The cells were seeded into 96-well plates and immunostained using anti-dsRNA antibody and patient serum. Images were acquired using an automated widefield microscope.

To get a measure for specific antibody binding we performed segmentation of cells and classified them into infected and non-infected categories based on the dsRNA staining. Then fluorescence intensities were measured in the serum channel per cell as a proxy for the amount of bound antibodies for both infected and non-infected cells. The ratio between intensity values in infected and non-infected cells is used to score the SARS-CoV-2 antibody response. My main contribution was to train the cell segmentation models for the later step of infected cell classification. In order to train the models, we manually labelled cells and annotated them as infected/non-infected in 10 images chosen from five positive and five control specimens.

In more detail the analysis workflow works as follows: First, images with obvious artefacts such as large dust particles or dirt and out-of-focus images were manually discarded. Then, images were processed to correct for the uneven illumination profile in each channel. Next, we segmented individual cells with a seeded watershed algorithm [14], using nuclei segmented via StarDist [116] as seeds and boundary predictions from a U-Net [113, 142] trained with the PlantSeg package. This approach was evaluated using leave-one-image-out cross-validation on the groundtruth images and resulted in an average precision [36] of 0.77 ± 0.08. Combined with extensive automatic quality control, which discards outliers in the results, the segmentation was found to be of sufficient quality for the analysis.

Then, the segmented cells were classified into infected and non-infected, by measuring the 95th percentile intensities in the dsRNA channel. Cells were marked as infected if this value exceeded 4.8 times the noise level, determined by the mean absolute deviation. This factor and the percentile were determined empirically using grid search on the manually annotated images. Using leave-one-out cross validation on the image level, we found that this approach yields an average F1-score of 84.3%. Figure 2.15 presents an overview of all the steps of the analysis.

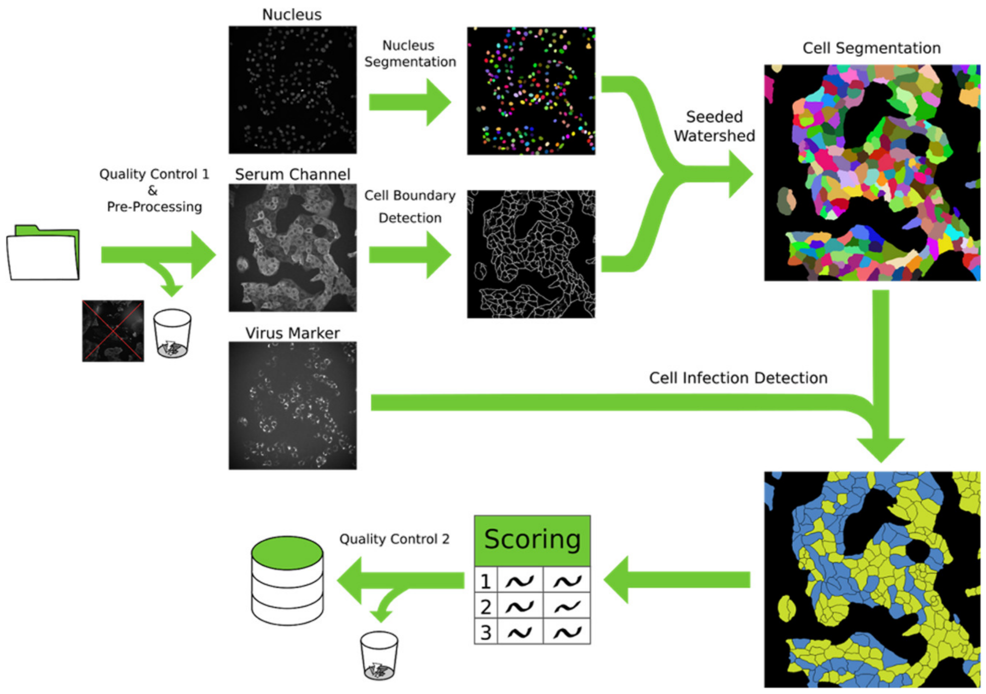


Figure 2.15: Image processing pipeline used to detect SARS-CoV-2 antibodies in immunofluorescence images of human sera. First, images with acquisition defects are discarded. Then, a pre-processing step corrects for barrel artifacts. Subsequently, cell segmentation is computed via seeded watershed, where seeds are generated by the StarDist [116] nuclei segmentation and cell contours are predicted using a neural network. Finally, using the virus marker channel each cell is classified as infected or not infected and we compute the scoring. A final automated quality control identifies and discards anomalous results. Figure from [107].

In summary, my contributions to the image analysis pipeline increased the throughput of the described IF-based assay for detection of SARS-CoV-2 specific antibodies in human serum. The strategy presented provides a general framework for serological testing based on quantitative high-throughput microscopy.

2.5 Conclusion

In this chapter, I presented PlantSeg, a simple, powerful, and versatile tool for cell segmentation. It implements a multi-step pipeline, where a fully convolutional neural network predicts cell boundaries, then the predicted boundary image is converted into a superpixel representation and finally the graph-based merging of superpixels delivers the final segmentation.

PlantSeg was trained on various 3D confocal and light sheet images and delivers high-quality segmentation on external datasets never seen during training as shown by both qualitative and quantitative benchmarks. I experimented with different U-Net designs and hyperparameters, as well as different graph partitioning algorithms, and equip PlantSeg with the ones that generalize the best. This is illustrated by the excellent performance of PlantSeg, without retraining of the CNNs, on a variety of plant tissues and organs imaged using confocal microscopy (3D Cell Atlas Dataset) including the highly lobed epidermal cells ([42]). The high accuracy of PlantSeg has also been shown empirically in an independent study [67]. Besides the plant data, I compared PlantSeg to the state-of-the-art on an open benchmark for the segmentation of epithelial cells in the *Drosophila* wing disc. Using only the pre-trained networks, PlantSeg performance was shown to be close to the benchmark leaders, while additional training on the benchmark's data has narrowed the gap even further.

Accurate and versatile extraction of cell outlines rendered possible by PlantSeg opens the door to rapid and robust quantitative morphometric analysis of plant cell geometry in complex tissues. This is particularly relevant given the central role plant cell shape plays in the control of cell growth and division [110].

I also highlighted three collaborative studies, demonstrating the versatility of the package in different areas of biological research. In two studies, PlantSeg was applied to segment volumetric time-lapse images of developing mouse embryo, which enabled the study of cellular dynamics. In the third publication, PlantSeg routines were utilized in the image analysis pipeline developed to automatically score antibody response in human sera.

Unlike intensity-based segmentation methods used, for example, to extract DAPI-stained cell nuclei, the approach presented in this chapter relies on boundary information derived from cell contour detection. While this approach grants access to the cell morphology and cell-cell interactions, it brings additional challenges to the segmentation problem. Blurry or barely detectable boundaries lead to discontinuities in the membrane structure predicted by the network, which in turn might cause cells to be under-segmented. The segmentation results produced by PlantSeg on new datasets are not fully perfect and still require proof-reading to reach 100% accuracy.

If nuclei are imaged along with cell contours, nuclear signal can be leveraged for improving the segmentation. One way to achieve this is based on lifted multicut formulation [60], which we have explored [106]. In there, additional repulsive edges are introduced in the region adjacency graph between nodes corresponding to the different nuclei. This modification is based on a rule that a single cell contains only one nuclei. In [106], we have demonstrated that this setup helps prevent false merge errors in cell segmentation. I added the lifted multicut segmentation scheme, together with the networks trained to predict nuclei in 3D light-sheet images to the

PlantSeg package.

During the development of PlantSeg, very few benchmark datasets were available to the community for plant cell segmentation tasks, a notable exception being the 3D Tissue Atlas. To address this gap, me and my collaborators publicly release the core datasets of Arabidopsis ovules and lateral root, together with the corresponding hand-curated groundtruth. We hope that the release of these datasets can catalyze future development and evaluation of cell instance segmentation algorithms.

3 SPOCO: Semi-Supervised Instance Segmentation

In Chapter 1, we recognized the high cost of dense, pixel-wise annotations used for CNNs training as a major obstacle for deep learning-based instance segmentation. Acquiring such training datasets is not only difficult and time consuming, but especially for biomedical images, it requires domain experts to perform the annotations. In this chapter, I propose to address the dense annotation bottleneck by introducing a semi-supervised segmentation approach. It does not require dense groundtruth and can be trained from just a handful of annotated objects. I consider the challenging case of *positive-unlabeled* supervision, where only a few objects of interests are labeled in a given image and everything else is unlabeled. Such annotations are much cheaper to create for human annotator and allow for greater variability in the training data. I extend an embedding-based approach described in Section 1.1.2 and introduce a novel self-supervised consistency loss for the unlabeled parts of the training data. I evaluate the proposed method on 2D and 3D segmentation problems in different microscopy modalities as well as on the Cityscapes and CVPPP instance segmentation benchmarks, achieving state-of-the-art results on the latter. This chapter is based on the publication [141].

3.1 Introduction

Instance segmentation is important for many application domains, forming the basis for the analysis of individual object appearance. Biological imaging provides a particularly large set of use cases for the instance segmentation task, with imaging modalities ranging from natural photographs for phenotyping to electron microscopy for detailed analysis of cellular ultrastructure. The segmentation task is often posed in crowded 3D environments or their 2D projections with multiple overlapping objects. Additional challenges – compared to segmentation in natural images – come from the lack of large, publicly accessible, annotated training datasets that could serve for general-purpose backbone training. Most microscopy segmentation networks are therefore trained from scratch, using annotations produced by domain experts in their limited time.

Over the recent years, several weakly supervised segmentation approaches have been introduced to lighten the necessary annotation burden. For natural images, image-level labels can serve as a surprisingly strong supervision thanks to the popular image classification datasets which include images of individual objects and can be used for pre-training [27]. There are no such collections in microscopy (see also Figure 3.5 for a typical instance segmentation problem example where image-level labels would be of no help). Semi-supervised instance segmentation methods [11, 10, 24] can create pseudo-labels in the unlabeled parts of the dataset. However, these methods require (weak) annotation of *all* the objects in at least a subset of images – a major obstacle for microscopy datasets which often contain hundreds of tightly clustered objects, in 3D.

The aim of my contribution is to address the dense annotation bottleneck by proposing a different kind of weak supervision for the instance segmentation problem: providing mask annotations only for a subset of instances in the image, leaving the rest of the pixels unlabeled. This "positive unlabeled" setting has been explored in image classification and semantic segmentation problems [83, 78], but - to the best of my knowledge - not for instance segmentation. Intrinsically, the instance segmentation problem is very well suited for positive unlabeled supervision: as I show empirically (Section 3.4.5), sampling a few objects in each image instead of labeling a few images densely exposes the network to a more varied training set with better generalization potential. This is particularly important for datasets with sub-domains in the raw data distribution, as it can ensure all sub-domains are sampled without increasing the annotation time. Furthermore, in crowded microscopy images which commonly contain hundreds of objects, and especially in 3D, dense annotation is significantly more difficult and time consuming than sparse annotation, for the same total number of objects annotated. The main obstacle for training an instance segmentation method on sparse object mask annotations lies in the assignment of pixels to instances that happens in a non-differentiable step which precludes the loss from providing supervision at the level of individual instances. To lift this restriction, I propose a differentiable instance selection step which allows us to incorporate any (differentiable) *instance-level* loss function into non-spatial pixel embedding network [16] training (Figure 3.1). I show that with dense object mask annotations and thus full supervision, application of the loss at the single instance level consistently improves the segmentation accuracy of pixel embedding networks across a variety of datasets. For my main use case of weak positive unlabeled (PU) supervision, I propose to stabilize the training from sparse object masks by an additional instance-level consistency loss in the unlabeled areas of the images. The conceptually simple unlabeled consistency loss, inspired by [56, 125], does not require the estimation of class prior distributions or the propagation of pseudo-labels, ubiquitously present in PU and other weakly supervised segmentation approaches [133, 81]. In addition to training from scratch, my approach can deliver efficient domain adaptation using a few object masks in the target domain as supervision.

In summary, I address the instance segmentation task with a CNN that learns pixel embeddings and propose the first approach to enable training with weak positive unlabeled supervision, where only a subset of the object masks are annotated and no labels are given for the background. To this end, I introduce: (1) a differentiable instance selection step which allows to apply the loss directly to individual instances; (2) a consistency loss term that allows for instance-level training on unlabeled image regions, (3) a fast and scalable algorithm to convert the pixel embeddings into final instances, which partitions the metric graph derived from the embeddings. I evaluate my approach on natural images (CVPPP [100] , Cityscapes [30]) and microscopy datasets (2D and 3D, light and electron microscopy), reaching the state-of-the-art on CVPPP and consistently outperforming strong baselines for microscopy. On all datasets, the bulk of CNN performance improvement happens after just a fraction of training objects are annotated.

3.2 Related Work

Proposal-based methods such as Mask R-CNN [57] are a popular choice for instance segmentation in natural images. These methods can be trained from weak bounding box labels [70, 79, 121, 103]. However, as they require a pre-trained backbone network and have difficulties segmenting complex non-convex shapes, they have not become the go-to segmentation technique for microscopy imaging. There, instance segmentation methods commonly start from the semantic segmentation [113], followed by a (non-differentiable) post-processing [8, 46, 77, 106].

Semantic instance segmentation with embedding networks was introduced by [38, 16]. The embeddings of [16] have no explicit spatial or semantic component. [38] predicts a seediness score for each pixel in addition to the embedding vector. The main advantage of pixel embedding-based segmentation methods lies in their superior performance for overlapping objects and crowded environments, delivering state-of-the-art results in many benchmarks, including those for biological data [92]. Furthermore, they achieve a significant simplification of the pipeline: the same method can now be trained for intensity-based and for boundary-based segmentation. my approach continues this line of work and employs non-spatial pixel embeddings.

Like the original proposal of [16], all modern embedding networks require fully segmented images for training and compute the loss for the whole image rather than for the individual instances. Even when the supervision annotations are weak, such as scribbles or saliency masks, they are commonly used to create full object proposals or pseudo-labels to allow the loss to be applied to the whole image [48, 133, 81]. Such methods exploit object priors learned by their components which have been pre-trained on large public datasets. At the moment, such datasets or pre-trained backbones are not available for microscopy images. Another popular approach to weak supervision is to replace mask annotations by bounding boxes [70] which are much faster to produce. Given a pre-trained backbone, bounding boxes can be reduced to single point annotations [75], but for training every object must be annotated, however weakly. The aim of my work is to lift this requirement and enable instance segmentation training with positive unlabeled supervision.

Positive unlabeled learning targets classification problems where negative labels are unavailable or unreliable [84, 9]. Three approaches are in common use: generation of negative pseudo-labels, biased learning with class label noise in unlabeled areas and class prior incorporation (see [9] for detailed review). PU learning has recently been extended to object detection [12] and semantic segmentation problems [78]. My approach enables PU learning for instance segmentation problems via an instance-level consistency loss applied to the unlabeled areas.

The core of my approach consists of the differentiable single instance selection step performed during training. Here, I have drawn inspiration from [102], where the clustering bandwidth is learned in the network training which allows to optimize the intersection-over-union loss for each instance. Still, as the network also needs to be trained to predict a seed map of cluster centers for inference, this method cannot be trained on partially labeled images. Differentiable single instance selection has also been proposed by AdaptIS [120]. However, this method does not use a learned pixel embedding space and thus requires an additional sub-network to perform instance selection. Importantly, AdaptIS does not introduce PU training and relies on a pre-trained backbone network which is not readily available for microscopy images.

3.3 Methods

3.3.1 Full Supervision

Given an image $I = \{I_1, ..., I_C\}$ composed of C objects (including background), N_k pixels in I_k, $N = \sum_{k=1}^{C} N_k$ pixels in the image and an embedding network $f\colon \mathbb{R}^3 \to \mathbb{R}^D$ which maps pixel i into a D-dimensional embedding vector $\boldsymbol{e}_i$, the discriminative loss [16] is defined by the *pull force* and the *push force* terms[1]:

$$L_{pull} = \frac{1}{C}\sum_{k=1}^{C}\frac{1}{N_k}\sum_{i=1}^{N_k}[\|\boldsymbol{\mu}_k - \boldsymbol{e}_i\| - \delta_v]_+^2 \tag{3.1}$$

$$L_{push} = \frac{1}{C(C-1)}\sum_{\substack{k=1 \\ k\neq l}}^{C}\sum_{l=1}^{C}[2\delta_d - \|\boldsymbol{\mu}_k - \boldsymbol{\mu}_l\|]_+^2 \tag{3.2}$$

[1]Similarly to [16] a regularization term ($\frac{1}{C}\sum_{k=1}^{C}\|\mu_k\|$) which keeps the embeddings bounded is added to the final loss with a small weight of 0.001. For clarity, I omit this term in the text

where $\|\cdot\|$ is the L2-norm and $[x]_+ = max(0,x)$ is the rectifier function. The pull force L_{pull} (Equation 3.1) brings the object's pixel embeddings closer to their mean embedding $\boldsymbol{\mu}_k$, while the push force L_{push} (Equation 3.2) pushes the objects away, by increasing the distance between mean instance embeddings. Note that both terms are hinged, i.e. embeddings within the δ_v-neighbourhood of the mean embedding $\boldsymbol{\mu}_k$ are no longer pulled to it. Similarly, mean embeddings which are further apart than $2\delta_d$ are no longer repulsed.

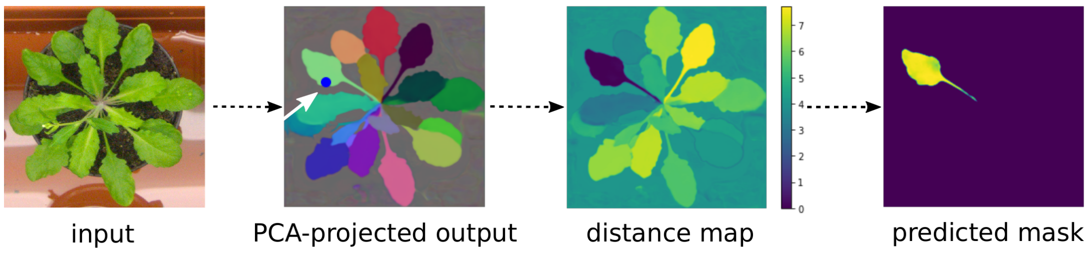


Figure 3.1: Differentiable instance selection for non-spatial embedding networks. First, I sample an anchor point randomly or guided by the groundtruth instances. Second, I compute a distance map in the embedding space from the anchor point to all image pixels. In the final step, a kernel function (Equation 3.3) transforms the distance map to the "soft" instance mask.

I exploit the clustering induced by this loss to select pixels belonging to a single instance and apply auxiliary losses at the instance level (Figure 3.2). Crucially, I find that given an instance I_k it is possible to extract a "soft" mask S_k for the current network prediction of the instance I_k in a *differentiable* way (Figure 3.1). I select an anchor point for I_k at random and project it into the learned embedding space to recover its embedding $\boldsymbol{a}_k$, which I term "anchor embedding". I compute the distance map from all image pixel embeddings to the anchor embedding and apply a Gaussian kernel function $\phi\colon \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ to "softly" select the pixels within the δ_v-neighborhood of $\boldsymbol{a}_k$ (δ_v is the pull term margin in Equation 3.1):

$$\begin{aligned} S_k &= \{\phi(\boldsymbol{e}_i, \boldsymbol{a}_k) \mid i = 1, ..., N\} \\ \phi(\boldsymbol{e}_i, \boldsymbol{a}_k) &= \exp\left(-\frac{\|\boldsymbol{e}_i - \boldsymbol{a}_k\|^2}{2\sigma^2}\right) \end{aligned} \tag{3.3}$$

I require the embeddings within the distance δ_v from the anchor embedding $\boldsymbol{a}_k$ have a kernel value greater than a predefined threshold $t \in (0,1)$, i.e. $\phi(\boldsymbol{e}_i, \boldsymbol{a}_k) \geq t \iff \|\boldsymbol{e}_i - \boldsymbol{a}_k\| \leq \delta_v$. I can thus determine σ^2: substituting $\|\boldsymbol{e}_i - \boldsymbol{a}_k\| = \delta_v$ in Equation 3.3, I get $\exp\left(-\frac{\delta_v^2}{2\sigma^2}\right) = t$, i.e. $\sigma^2 = \frac{-\delta_v^2}{2\ln t}$. I choose $t = 0.9$ in my experiments and refer to Section 3.5.6 for a detailed exploration of this hyperparameter.

I can now formulate a loss on the instance level, where the objective is to minimize the discrepancy between the predicted masks S_k and the the corresponding groundtruth masks I_k. I choose the Dice loss [99] to compare the predicted mask to the groundtruth mask. The corresponding object level loss is given by:

$$L_{obj} = \frac{1}{C} \sum_{k=1}^{C} D(S_k, I_k) \tag{3.4}$$

where D is the Dice loss:

$$D(S_k, I_k) = 1 - \frac{2 \sum_i^N p_i q_i}{\sum_i^N p_i^2 + \sum_i^N q_i^2} \tag{3.5}$$

in which p_i and q_i represent pairs of corresponding pixel values of the predicted mask S_k and groundtruth mask I_k. Combining the losses in Equation 3.1, Equation 3.2 and Equation 3.4, I get:

$$L_{SO} = \alpha L_{pull} + \beta L_{push} + \lambda L_{obj} \tag{3.6}$$

which I refer to as the Single Object contrastive loss (L_{SO}). I use $\alpha = \beta = 1$ (similar to [16]) and $\lambda = 1$ in my experiments. I set the pull and push margin parameters to $\delta_v = 0.5, \delta_d = 2.0$.

While Equation 3.4 employs the Dice loss, my approach is not limited to Dice and can be used with any differentiable loss function at the single instance level, e.g. binary cross-entropy. Additionally, I explored the adversarial approach and trained a discriminator to distinguish the object masks coming from the differentiable instance selection method or from the groundtruth. The results of the adversarial regularization are shown in Table 3.3. Implementation details can be found in Section 3.5.3, the results are shown in Table 3.3.

3.3.2 Positive Unlabeled Supervision

To enable training from positive unlabeled supervision, I introduce two additional loss terms: one to push each cluster away from the pixels in the unlabeled region and the other to enforce embedding space consistency in the unlabeled region. For an unlabeled region U which can contain both background and unlabeled instances, I define an additional "push" term:

$$L_{U\_push} = \frac{1}{C} \sum_{k=1}^{C} \frac{1}{N_U} \sum_{i=1}^{N_U} [\delta_d - \|\boldsymbol{\mu}_k - \boldsymbol{e}_i\|]_+^2 \tag{3.7}$$

where C is the number of labeled clusters/instances and N_U is the number of pixels in the unlabeled region U.

Since there is no direct supervision applied onto the unlabeled part of the image, the fully convolutional embedding network propagates the high frequency patterns present in there into the feature space. This is especially apparent for natural images and microscopy images with complex background structures, e.g. electron microscopy (see Figure 3.3 top left and Figure 3.6 top, col 3). To overcome this issue, I introduce the embedding consistency term. Given two different embedding networks f and g, I perturb the input image x with two different random, location- and shape-preserving augmentations t and t' and pass it through f and g respectively. The resulting vector fields $f(t(x))$ and $g(t'(x))$ come from the same input geometry, hence they should result in consistent instance segmentation after clustering, also in the unlabeled part of the input. To enforce this consistency I randomly sample an anchor point from the unlabeled region, project it into the f- and g-embedding spaces, to get anchor embeddings $\boldsymbol{a}^f$ and $\boldsymbol{a}^g$ and compute two "soft" masks S^f and S^g according to Equation 3.3. Similarly to Equation 3.4 the embedding consistency is given by maximising the overlap of the two masks, using the Dice loss (D):

$$L_{U\_con} = \frac{1}{K} \sum_{k=1}^{K} D(S_k^f, S_k^g) \tag{3.8}$$

where K is the number of anchor points sampled from the unlabeled region U such that the whole region is covered by the union of extracted masks, i.e. $U \approx \bigcup_{k=1}^{K} S_k^f \cup S_k^g$. Having considered different variants of g-network including: weight sharing (with and without dropout) and independent training, I choose a momentum-based scheme [56, 51] where the network g (parameterized by θ_g) is implemented as an exponential moving average of the network f (parameterized by θ_f). The update rule for θ_g is given by: $\theta_g \leftarrow m\theta_g + (1-m)\theta_f$. f is trained by back-propagation. I refer to Section 3.5.2 for extensive ablations of the g-network types and Section 3.5.5 for the choice of a momentum coefficient $m \in [0, 1)$. Briefly, momentum variant provides the fastest convergence rate, improves training stability and is motivated by prior work [125, 20]. Significance of the embedding consistency term in weakly supervised setting is illustrated in Figure 3.3. Note how the complex patterns present in the background (e.g. the flower pot) are propagated into the embedding space of the network trained without the consistency term (top, column 2), leading to spurious objects in the background after clustering (middle, column 2). In contrast, the same network trained with the embedding consistency loss results in crisp embeddings, homogeneous background embedding and clear background separation with no false positives (column 3). I confirm this observation by PCA-projecting the embeddings of background pixels onto 2D subspace (bottom). Network trained sparsely with the consistency term implicitly pulls background pixels into a single cluster, similar to the fully supervised network where the background pull is enforced by the loss. In contrast, the network trained without the consistency loss does not form a tight background cluster in the feature space. In addition, with a limited annotation budget of a certain number of objects, I achieve (see Section 3.4.5) much better segmentation accuracy with objects distributed across many images than with a few images fully labeled. The latter is prone to over-fitting, whereas a more diverse training set and the presence of the strong consistency regularizer in the former enables it to train from just a few object mask annotations. My weakly supervised loss, termed Sparse Single Object loss (L_{SSO}), is given by:

$$L_{SSO} = \hat{L}_{SO} + \gamma \cdot L_{U\_push} + \delta \cdot L_{U\_con} \tag{3.9}$$

In my experiments I use $\gamma = \delta = 1$.
Table 3.2 shows that using the consistency term (Equation 3.8) in a fully-supervised setting, in addition to the instance-based term (Equation 3.4) improves the segmentation accuracy at the expense of longer training times. Figure 3.2 gives a graphical overview of the training procedure which I term *SPOCO* (SParse Object COnsistency loss). Extensive ablation study of the individual loss terms can be found in Section 3.5.1. In the experiments, I use the term SPOCO to refer to the fully-supervised training (taking all groundtruth objects including the background object). SPOCO@p refers to the weakly supervised positive unlabeled setting, in which a fraction $p \in (0, 1]$ of objects (excluding background) is taken for training. The background label is never selected in the weakly supervised setting, i.e. SPOCO@1.0 means that the network was trained with all labeled objects, excluding background.

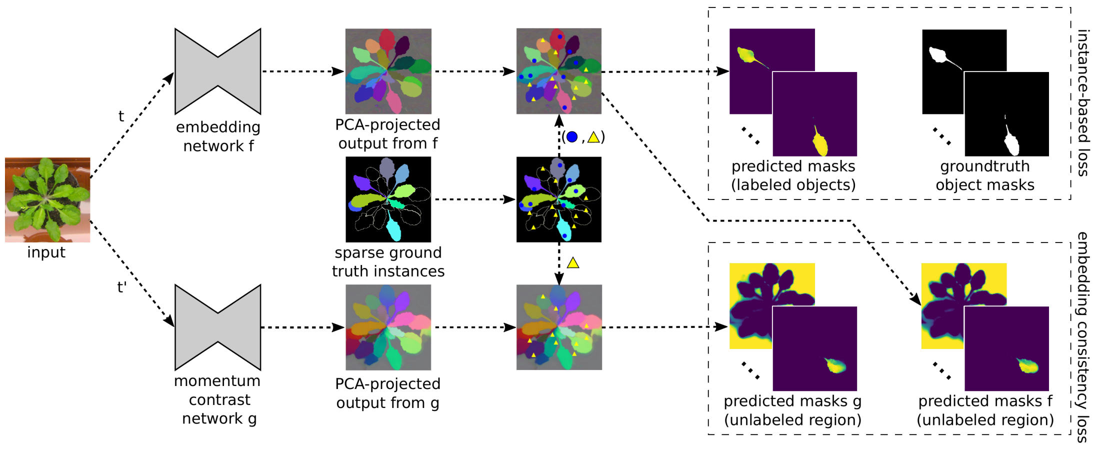

input
t
t'
embedding network f
momentum contrast network g
PCA-projected output from f
sparse ground truth instances
PCA-projected output from g
(●, △)
△
predicted masks (labeled objects)
groundtruth object masks
instance-based loss
predicted masks g (unlabeled region)
predicted masks f (unlabeled region)
embedding consistency loss

Figure 3.2: Overview of training procedure. Two augmented views of an input image are passed through two embedding networks $f(\cdot)$ and $g(\cdot)$ respectively. Anchor pixels inside labeled objects (blue dots) are sampled and their corresponding instances are extracted as shown in Figure 3.1. Discrepancy between extracted objects and groundtruth objects is minimized by the instance-based loss. Another set of anchors (yellow triangles) is sampled exhaustively from the unlabeled region and for each anchor two instances are selected based on the outputs from $f(\cdot)$ and $g(\cdot)$. Discrepancy between instances is minimized using the embedding consistency loss.

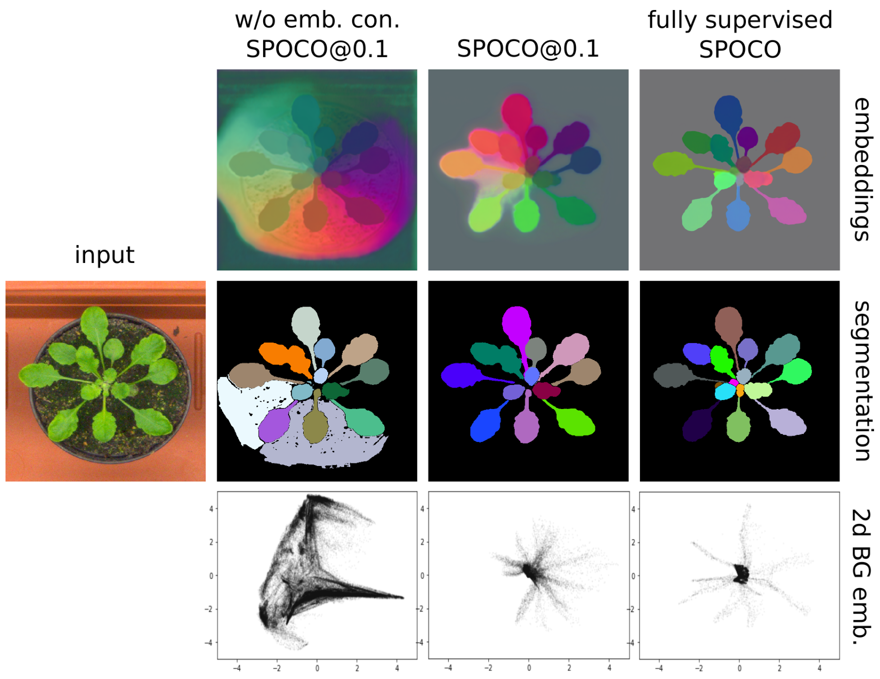


Figure 3.3: Different training schemes, left to right: SPOCO@0.1 trained without embedding consistency; SPOCO@0.1 trained with embedding consistency; SPOCO trained with full supervision (including the background label). **Top:** PCA-projected embeddings; **Middle:** corresponding clustering results; **Bottom:** background pixel embeddings PCA-projected onto 2D subspace.

3.3.3 Clustering

To create the final instance segmentation, the pixel-wise embeddings are clustered in a post-processing step. Mean-shift [28] and HDBSCAN [17] are commonly used for this task [16, 74, 109]. In this work, I experimented with two additional clustering schemes: (1) a hybrid approach called *consistency clustering*, where initial mean-shift clusters are refined to conform with the pull-push loss formulation (Sec 3.3.1) (2) partitioning [140] of a metric graph derived from pixel embeddings [76]. Embeddings from networks f and g are used together in (1), all other clustering methods use the f-embeddings only. See Section 3.4.6 for a detailed comparison of different clustering methods.

Consistency Clustering The procedure for Consistency Clustering is described in algorithm 1. It works by passing two augmented versions of the input through the networks f and g, producing embeddings $\mathcal{E}_f$ and $\mathcal{E}_g$ respectively. I cluster $\mathcal{E}_f$ using mean-shift with bandwidth set to the pull force margin δ_v. Then for each segmented object S_k, I randomly select M anchor points and for each anchor I extract a new object $\hat{S}_k^m$ by taking a δ_v-neighborhood around the anchor in the $\mathcal{E}_g$ space. If the median intersection-over-union (IoU) between S_k and each of the $\hat{S}_k^m$ objects is lower than a predefined threshold, I discard S_k from the final segmentation. This is based on the premise that clusters corresponding to the real objects should remain consistent between $\mathcal{E}_f$ and $\mathcal{E}_g$.

```
**Input:** Set of mean-shift segmented objects $\mathcal{S}$, embeddings from the g-network
        $\mathcal{E}_g = \{\boldsymbol{e}_0, \boldsymbol{e}_1, ..., \boldsymbol{e}_N\}$, IoU threshold $t_{IoU}$, number of anchors per object to
        sample $M$
**Output:** New set of segmented objects $\hat{\mathcal{S}}$
$\hat{\mathcal{S}} = \{\}$;
**for** $S_k \in \mathcal{S}$ **do**
    $\mathcal{A}_k = \{\boldsymbol{a}_k^1, ..., \boldsymbol{a}_k^M \mid \boldsymbol{a}_k^m \in \mathcal{E}_g\}$ - anchors of $S_k$;
    $I_{IoU} = \{\}$;
    **for** $\boldsymbol{a}_k^m \in \mathcal{A}_k$ **do**
        $\hat{S}_k^m = \{s_i \mid s_i = \|\boldsymbol{e}_i - \boldsymbol{a}_k^m\| < \delta_v\}$;
        $I_{IoU} \cup \text{IoU}(\hat{S}_k^m, S_k)$;
    **if** $med(I_{IoU}) > t_{IoU}$ **then**
        $\hat{\mathcal{S}} = \hat{\mathcal{S}} \cup \{S_k\}$;
**return** $\hat{S}$;
```

Algorithm 1: Consistency Clustering algorithm

Graph-based partitioning The affinity graph-based method proceeds similar to [76]: I convert the embedding space into a graph partitioning problem by introducing a grid-graph that contains a node for each pixel and connects all direct neighbor pixel via edges. Following [77] and [140] I introduce additional long-range edges that connect pixels that are not direct neighbors in a fixed offset pattern. Following [76] I derive the edge weight w_{ij}, or affinity, between pixel i and j from the embedding vector $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ via

$$w_{ij} = 1 - \max(\frac{2\delta_d - \|\boldsymbol{e}_i - \boldsymbol{e}_j\|}{2\delta_d}, 0)^2. \tag{3.10}$$

Here, δ_d is the hinge from Equation 3.2 and I use the L2 norm to measure the distance in the embedding space. This weight is derived from the distance term (Equation 3.2) and is maximally attractive (0) when the embedding distance is zero and becomes maximally repulsive (1) for embedding distances larger than $2\delta_d$. I obtain an instance segmentation with the Mutex Watershed algorithm [140], which operates on long-range affinity graphs. I introduce long-range edges between all pixel pairs with distance 3, 9 and 27 across all dimensions. This choice yields good segmentation results empirically.

3.4 Results

I evaluate my method using the following benchmark datasets:
CVPPP. I use the A1 subset of the popular CVPPP dataset [100] which is part of the LSC competition. The task is to segment individual leaf instances of a plant growing in a pot. The dataset consists of 128 training images with public groundtruth labels and 33 test images with no publicly available labels. Test images come with a foreground mask which can be used during inference.
Cityscapes. I use Cityscapes [30] to demonstrate the performance of my method on a large-scale instance-level segmentation of urban street scenes. There are 2975 training images, 500 validation images, and 1525 test images with fine annotations. I choose 8 semantic classes: *person, rider, car, truck, bus, train, motorcycle, bicycle* and train the embedding networks separately for each class using the training set in the full and weak supervision setting.
Light microscopy (LM) datasets. To evaluate the performance of my approach on a challenging boundary-based segmentation task I selected a 3D LM dataset of the ovules of *Arabidopsis thaliana* from [142], with 48 image stacks in total: 39 for training, 2 for validation and 7 for testing. Additionally, I use the 3D *A. thaliana* apical stem cells from [139] in a transfer learning setting. The images are from the same imaging modality as the ovules dataset (confocal, cell membrane stained), but differ in cell type and image acquisition settings. I choose the Ovules dataset as the source domain and Stem cells as the target (*plant1, plant2, plant4, plant13, plant15* are used for fine-tuning and *plant18* for testing).
Electron microscopy (EM) datasets. Here, I test my method in the transfer learning setting on the problem of mitochondria segmentation. An important difference between light and electron microscopy from the segmentation perspective lies in the appearance of the background which is simply dark and noisy for LM and highly structured for EM. The source domain (VNC dataset) [49] is a small annotated $20 \times 1024 \times 1024$ volume of the Drosophila larva acquired with voxel size of $50 \times 5 \times 5$nm. I use 13 consecutive slices for training and keep 7 slices for validation. As target domain I use the 3D MitoEM-R dataset from the MitoEM Challenge [137] a $500 \times 4096 \times 4096$ volume at $30 \times 8 \times 8$ nm resolution extracted from rat cortex. Slices

(0-399) are used for fine-tuning and (400-499) for testing.

I present the fully- and semi-supervised results on the CVPPP, Cityscapes and LM datasets in Section 3.4.1, Section 3.4.2 and Section 3.4.3 respectively. Transfer learning results on the LM and EM datasets are shown in Section 3.4.3 and Section 3.4.4. In Section 3.4.5 I demonstrate that given a limited annotation budget of a certain number of objects my method outperforms its fully-supervised counterpart.

Setups Any fully convolutional architecture with dense outputs could be used as an embedding network. I choose the U-Net [113, 21]. The depth of the U-Net is chosen such that the receptive field of features in the bottleneck layer is greater or equal to the input patch size. In all experiments, I train the networks from scratch without using any per-trained backbones. I use the Adam [71] optimizer with initial learning rate 0.0002 and weight decay 0.00001. Data augmentation consists of random crops, random flips, random scaling and random elastic deformations. For the momentum contrast embedding network, I additionally use additive Gaussian noise, Gaussian blur and color jitter as geometry preserving transformations.

In transfer learning experiments, the source network is always trained with the full groundtruth. On the target domain, I reduce the learning rate by a factor of 10 compared to the source network and use only a small fraction of the objects. VNC dataset is too small to train a 3D U-Net, so I perform EM segmentation in 2D, slice-by-slice. I also downsample VNC dataset by factor 1.6 in XY to match the voxel size of the target MitoEM data.

A detailed description of the network architecture, training procedure and hyperparameter selection can be found in the Section 3.6.

3.4.1 CVPPP Challenge

Table 3.1 shows the results on the test set. The challenge provides foreground masks for test set images and I assume they have been used by authors of [16, 112, 74] in test time inference. In this setting, SPOCO outperforms [74] and the current winner of the leaderboard on the A1 dataset, keeping the advantage even in the case when the foreground mask is not given, but learned by another network ("predicted FG"). Even without using the foreground mask in the final clustering, SPOCO is close to [74] in segmentation accuracy, achieving much better average difference in counting score ($|DiC|$). I evaluate weakly supervised predictions without the foreground mask as I cannot easily train a semantic network without background labels. Nevertheless, even when training with only 10% of the groundtruth instances (SPOCO@0.1), the Symmetric Best Dice (SBD) as compared with the fully supervised SPOCO (without FG) drops only by 10 percent points. Qualitative results from SPOCO@0.1 can be seen in Figure 3.3 (column 3), where the single under-segmentation error is present in the top left part of the image. HDBSCAN with $min\_size = 200$ is used for clustering in this case. Visual results and performance metrics for other clustering methods can be found in Section 3.4.6.

Method	SBD	$\|DiC\|$
Discriminative loss [16]	0.842	1.0
Recurrent attention [112]	0.849	**0.8**
Harmonic Emb. [74]	0.899	3.0
SPOCO (GT FG)	**0.932**	1.7
SPOCO (pred FG)	0.920	1.6
SPOCO (w/o FG)	0.886	1.3
SPOCO@0.1	0.788 ± 0.017	5.4 ± 0.3
SPOCO@0.4	0.824 ± 0.003	3.2 ± 0.5
SPOCO@0.8	0.828 ± 0.010	1.6 ± 0.2

Table 3.1: Results on the CVPPP test set. Segmentation (SBD) and counting ($|DiC|$) scores for fully supervised SPOCO are reported in 3 different clustering settings: (1) with the groundtruth foreground mask, (2) with the predicted foreground mask (3) without the foreground mask. Results for semi-supervised setting SPOCO@p (no foreground mask) are presented for 10%, 40% and 80% of randomly selected groundtruth instances.

3.4.2 Cityscapes Challenge

I train my method with sparse (SPOCO@0.4) and full supervision and compare it with the fully-supervised contrastive framework [16]. In [16] authors trained a single model with multiple classes, applying the loss only within a given semantic mask. Since groundtruth semantic masks are not available when training from sparsely labeled instances, I train one model (including my implementation of [16]) for each semantic class. For inference, I use pre-trained semantic segmentation model (DeepLabV3 [23]) to generate semantic masks and cluster the embeddings only within a given semantic mask. After initial mean-shift clustering I merge every pair of clusters if the mean cluster embeddings are closer than δ_d (push force hinge in Equation 3.2). Average Precision at 0.5 intersection-over-union computed on the validation set can be found in Table 3.2. My method outperforms [16] with only 40% of the groundtruth objects of each semantic class used for training. This is true for all classes apart from person, car and bicycle where the model requires larger number of annotated objects to reach high precision. Importantly, using consistency term in the fully-supervised setting improves the score by a large margin. The performance of SPOCO@0.4 is almost as good as the fully-supervised

Class	Discriminative loss [16]	SPOCO@0.4	SPOCO w/ con	SPOCO w/o con
person	**0.275**	0.230	0.260	0.270
rider	0.392	0.396	**0.451**	0.448
car	**0.416**	0.301	0.331	0.363
truck	0.486	0.558	**0.604**	0.527
bus	0.504	0.601	**0.637**	0.530
train	0.375	0.594	**0.656**	0.490
motorcycle	0.382	0.405	**0.464**	0.461
bicycle	**0.267**	0.214	0.266	0.255
average	0.387	0.412	**0.459**	0.418

Table 3.2: Segmentation results on the Cityscapes validation set. Average and per-class AP@0.5 scores are reported. *SPOCO w/ con* - fully-supervised SPOCO with the consistency term, *SPOCO w/o con* - fully-supervised SPOCO without the consistency term.

SPOCO without the consistency term. I hypothesize that strong regularization induced by the consistency term is crucial for classes with small number of instances. Figure 3.4 shows qualitative results on a few samples from the validation set. Network trained with discriminative loss frequently over-segments large instances (trucks, buses, trains). A common mistake in crowded scenes for both methods is the merging of neighboring instances. Segmentation scores at different sampling rates, comparison with a class-agnostic training setting as well as qualitative results can be found in the Section 3.5.4.

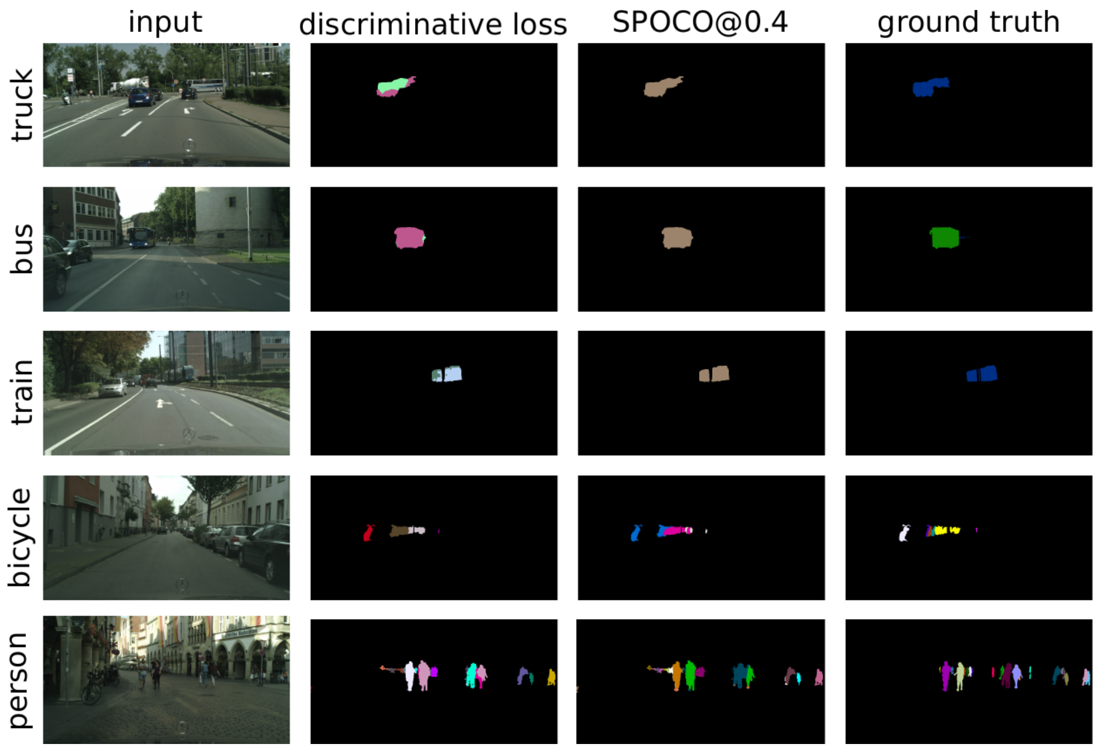


Figure 3.4: Segmentation results for randomly selected images of different semantic classes on the Cityscapes validation set.

3.4.3 3D Light Microscopy Datasets

I compare SPOCO to the method of [142]: a 3-step pipeline of boundary prediction, supervoxel generation and graph agglomeration. Following [142], Adapted Rand Error [44] is used for evaluating the segmentation accuracy. As shown in Table 3.3, the performance of SPOCO is close to that of the much more complex 3-step PlantSeg pipeline. An additional adversarial loss term (SPOCO with L_{wgan}, see Section 3.5.3) brings another performance boost and improves SPOCO accuracy beyond the [142] level.

Note that SPOCO trained with 10% of the groundtruth instances already outperforms the original embedding network with discriminative loss [16]. See Figure 3.5 (top row) for qualitative results on a randomly sampled test set patch.

Method	ARand error
PlantSeg [142]	0.046
Discriminative loss [16]	0.074
SPOCO	0.048
SPOCO with L_{wgan}	**0.042**
SPOCO@0.1	0.069
SPOCO@0.4	0.060
SPOCO@0.8	0.057

Table 3.3: Evaluation on a 3D light microscopy dataset of *Arabidopsis* ovules [142]. The Adapted Rand Error (ARand error) averaged over the 7 test set 3D stacks is reported. Bottom part of the table shows the scores achieved in the weakly supervised settings.

Table 3.4 shows SPOCO performance in a transfer learning setting, when fine-tuning a network trained on the Ovules dataset to segment the Stem dataset. The Ovules network trained only on source data does not perform very well, but just 5% of the target groundtruth annotations brings a two-fold improvement in segmentation accuracy. Results in Table 3.3 and Table 3.4 are based on HDBSCAN ($min\_size = 550$) clustering.

Method	ARand error
Stem	0.074
Ovules	0.227
Ovules+Stem@0.01	0.141 $\pm$ 0.002
Ovules+Stem@0.05	0.109 $\pm$ 0.002
Ovules+Stem@0.1	0.106 $\pm$ 0.004
Ovules+Stem@0.4	0.093 $\pm$ 0.003

Table 3.4: Evaluation on a 3D light microscopy dataset in a transfer learning setting. Ovules dataset acts as the source domain, Stem dataset as the target domain. Performance is measured by the Adapted Rand Error (lower is better). Mean $\pm$ SD are reported across 3 random samplings of the instances from the target dataset.

Qualitative results are shown in Figure 3.5 (bottom row). Note how the output embeddings from the Ovules network fine-tuned with just 1% of cells from the target dataset are less crisp due to the domain gap, but the clustering is still able to segment them correctly.

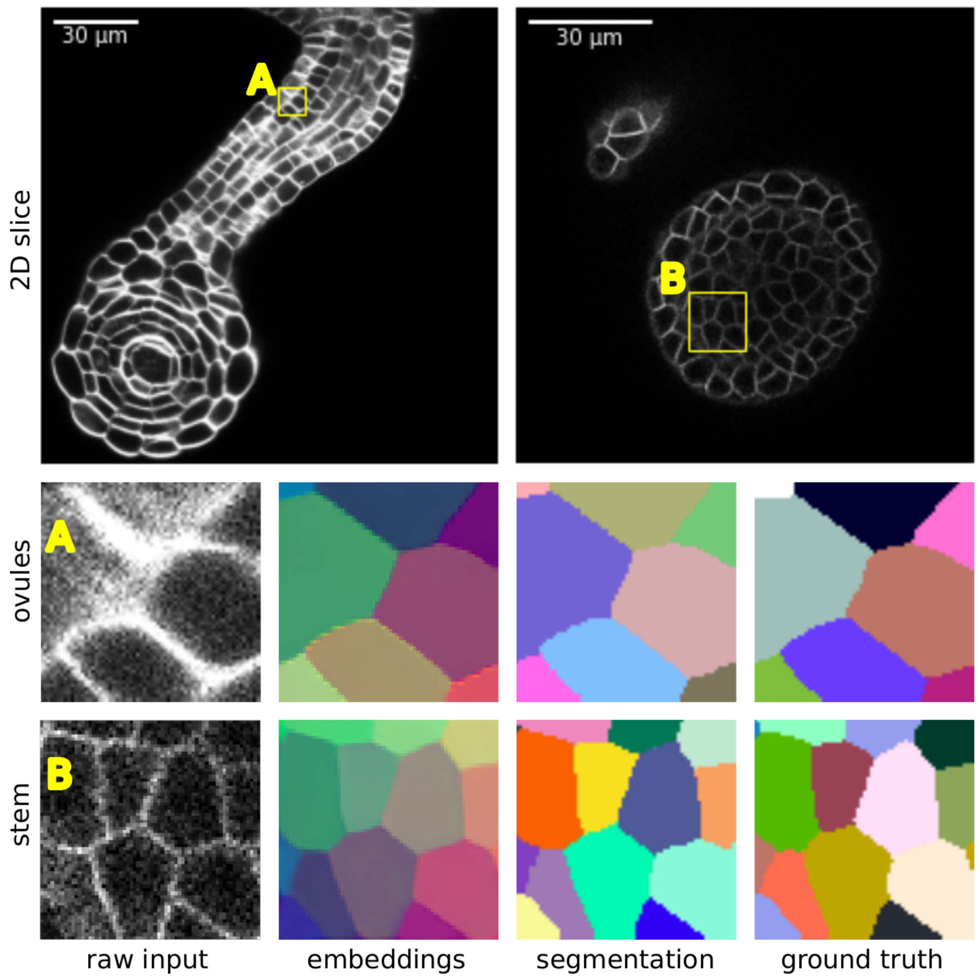


Figure 3.5: Light microscopy segmentation in standard and transfer learning settings. **TOP:** samples from the 3D Ovules (left) and Stem (right) datasets; **MIDDLE:** segmentation of a selected patch (A) from the source domain; **BOTTOM:** output of the source (Ovules) network fine-tuned with 1% of instances from the target (Stem) and the corresponding segmentation of a selected patch (B).

3.4.4 Electron Microscopy Datasets

Table 3.5 continues the evaluation of SPOCO performance in a transfer learning setting. I report the average precision at 0.5 IoU threshold (AP@0.5) and the mean average precision (mAP). Similar to the LM case, just 1% of annotated objects in the target dataset bring a 1.5 fold improvement in the mean average precision compared to the network trained on source VNC domain only. A comparison to a network trained only on MitoEM (Table 3.5 bottom) shows that fine-tuning does significantly improve performance for low amounts of training data (1% of the target). With 10% of the annotated objects, fine-tuned VNC network does not reach the performance of the SPOCO@0.1 trained directly on MitoEM (target) due to reduced learning rate. Figure 3.6 illustrates the EM experiments. The VNC-net only partially recovers 4 out of 7 groundtruth instances and also produces a false positive. MitoEM@0.05 without consistency loss only recovers 2 instances, while the version with the consistency loss recovers the correct segmentation. Embeddings clustered with HDBSCAN ($min\_size = 600$).

Method	AP@0.5	mAP
MitoEM	0.560	0.429
VNC	0.234	0.137
VNC+MitoEM@0.01	0.368 ± 0.022	0.247 ± 0.022
VNC+MitoEM@0.05	0.398 ± 0.007	0.277 ± 0.006
VNC+MitoEM@0.10	0.389 ± 0.013	0.268 ± 0.007
MitoEM@0.01	0.088 ± 0.045	0.046 ± 0.025
MitoEM@0.05	0.403 ± 0.055	0.280 ± 0.046
MitoEM@0.10	0.481 ± 0.008	0.340 ± 0.007

Table 3.5: Evaluation on MitoEM dataset (target) fine-tuned from the VNC net (VNC+MitoEM@p) and trained from scratch (MitoEM@p) with different sampling ratios. The performance is measured with an average precision (AP@0.5, mAP). Mean ± SD are reported across 3 random samplings of the instances from the target dataset.

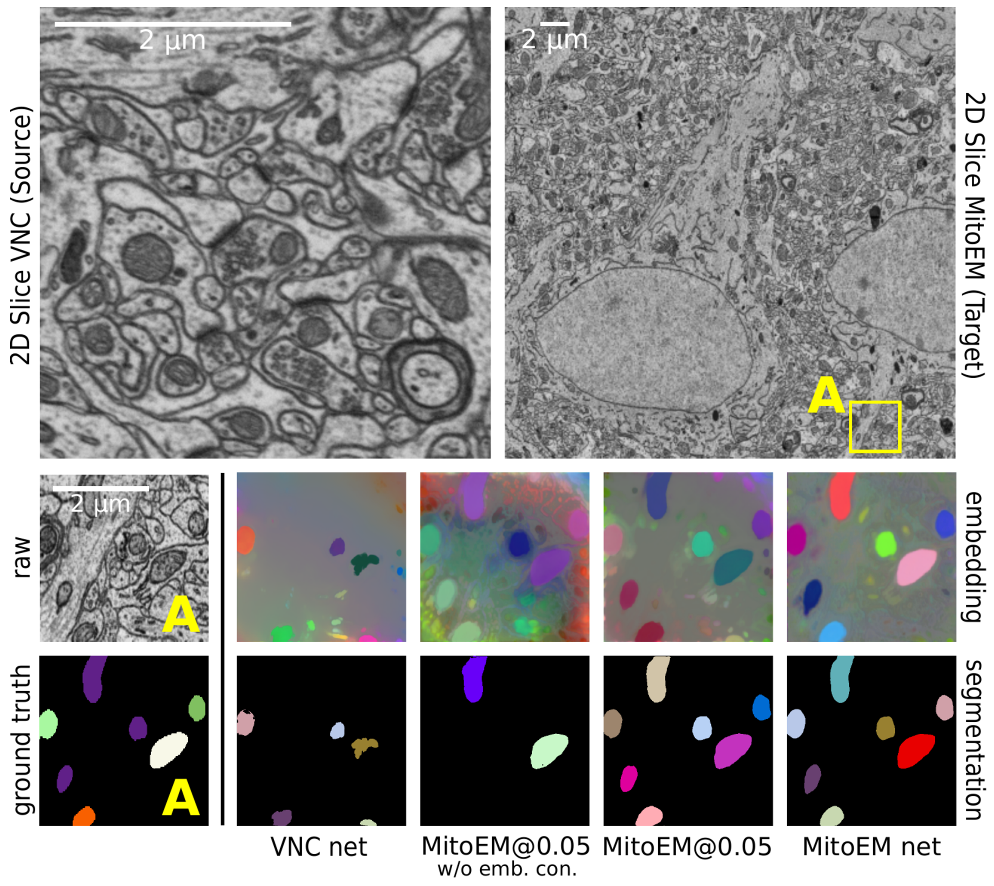


Figure 3.6: Electron microscopy segmentation in transfer learning setting. **Top:** samples from the source (VNC, left) and target (MitoEM, right) datasets; **Middle:** the input image and the RGB-projected embeddings: trained on VNC only, VNC-pretrained + MitoEM@0.05-finetuned without the embedding consistency, same but with the embedding consistency, trained on MitoEM only; **Bottom:** groundtruth and predicted segmentations.

3.4.5 Training with Limited Annotation Budget

Choosing a fixed annotation budget of N ground truth instances I can objectively compare the weakly supervised training with the dense, fully supervised one. I set $N = 16$, which corresponds to roughly 1% of the objects from the CVPPP training set containing 1683 objects spread across 103 files (I use train/val script described in Section 3.5.2). In the *dense* setup I randomly choose a single groundtruth file with 16 objects and dense labeling (including the background label), whereas in the *sparse* setting I randomly sample 16 objects from the whole training set, resulting in 16 files, each with only one object labeled. On the densely labeled image, I train using the fully supervised loss Equation 3.6, while on the sparsely annotated images I train using the weakly supervised loss Equation 3.9.

Segmentation metrics and embeddings emerging the two training schemes are shown in Table 3.6. The network trained from dense annotations is prone to over-fitting and result in visible artifacts in the embedding space. On the other hand, exposing the network to a much more varied training set in the sparse setting and the presence of a strong consistency regularizer results in a feature space of much better quality. Quantitative comparison confirms that the *sparse* significantly outperforms the *dense* setting in terms of segmentation and counting scores.

Training scheme	SBD	$\|DiC\|$
1% dense	0.380	9.8
1% sparse	**0.691**	**2.2**

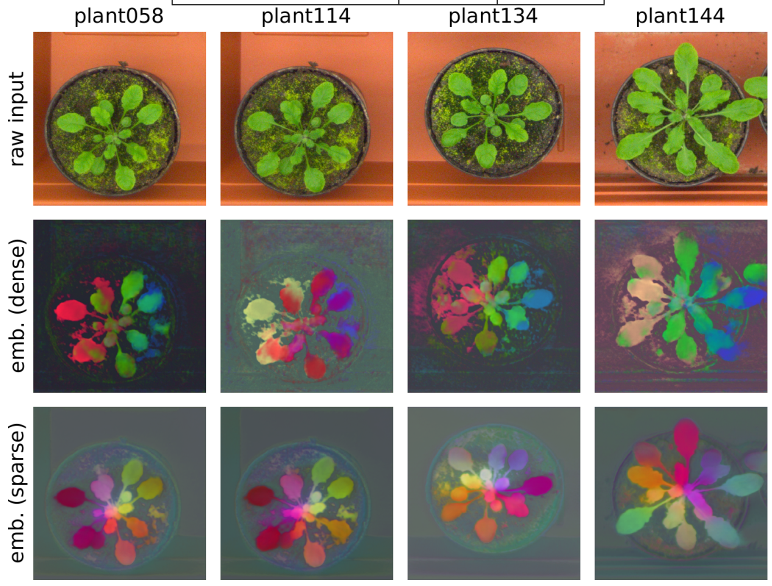


Table 3.6: **Top:** segmentation performance computed for the networks trained with limited annotation budget on the CVPPP validation set. Embeddings within the foreground semantic mask were clustered with mean-shift algorithm. **Bottom:** qualitative comparison of the embeddings trained in the *dense* and *sparse* setting. Four sample images where chosen from the CVPPP validation set.

3.4.6 Clustering Comparison

Quantitative comparison of 4 different clustering algorithms: HDBSCAN ($min\_size = 200$), Mean-shift (with bandwidth set to $\delta_v = 0.5$), Consistency Clustering ($t_{IoU} = 0.6$) and affinity-based clustering are shown in Table 3.7. I report the segmentation and counting scores as well as runtimes on the CVPPP validation set. I used the embedding networks trained using SPOCO@0.1 (i.e. 10% of randomly selected ground truth objects) and SPOCO (trained with full supervision).

Mean-shift has a high recall (correctly recovers most instances), but low precision (it tends to over-segment the image around the boundary of the objects, see Figure 3.7) resulting in high number of false positives and inferior counting scores. Consistency clustering significantly improves the initial mean-shift segmentation resulting in the best segmentation metric for the network trained with weak supervision (SPOCO@0.1). Affinity-based (Mutex Watershed) and density-based (HDBSCAN) methods have similar segmentation scores, with the former achieving much better counting performance in both full (SPOCO) and weak (SPOCO@0.1) supervision. The affinity-based approach has much lower runtimes compared to the other clustering methods.

Clustering	SBD	$\|DiC\|$	t [s]
	SPOCO@0.1		
Consistency	**0.729** $\pm$ 0.086	2.7 $\pm$ 1.7	252.3
HDBSCAN	0.653 $\pm$ 0.077	5.7 $\pm$ 1.7	82.3
Mean-shift	0.356 $\pm$ 0.048	20.7 $\pm$ 6.2	201.2
Affinity-based	0.615 $\pm$ 0.061	2.6 $\pm$ 2.3	**0.45**
	SPOCO		
HDBSCAN	**0.834**	1.6	164.7
Mean-shift	0.541	10.92	121.9
Affinity-based	0.833	0.88	**0.4**

Table 3.7: Performance and runtime comparison of the clustering methods on the CVPPP validation set. I compare the results for SPOCO@0.1, where mean $\pm$ SD are reported across 3 random samplings of the ground truth objects as well as fully supervised SPOCO (bottom), for which I report results from a single training.

Similarly, Table 3.8 shows comparison of 3 clustering algorithms: HDBSCAN ($min\_size = 600$), Mean-shift ($bandwidth = \delta_v$) and affinity-based on the Ovules test set. I skip the consistency clustering, since the embedding network is trained in the full supervision setting. I notice that for the dense tissue segmentation problems, HDBSCAN classifies the low density areas between the cells as noise, and additional post-processing is required in order to fill the empty space. Results reported in Table 3.3 and Figure 3.5 are based on the watershed post-processing. Here, for fair comparison with other methods I don't use the watershed post-processing on the HDBSCAN clustering results (see Figure 3.5 bottom). Overall, the parameter-free, affinity-based clustering is much faster than other methods under consideration and provides the best performance-runtime ratio. The downside of HDBSCAN is its sensitivity to the *min_size* hyperparameter, longer running times and the need for additional post-processing for dense tissue segmentation problems.

Method (Ovules)	Arand error	t [s]
HDBSCAN	0.133	95.0
Mean-shift	0.102	279.2
Affinity-based	**0.086**	**0.9**

Table 3.8: Performance (Adapted Rand Error) and runtime comparison of the clustering methods on the ovules test set. Embedding network trained with fully supervised SPOCO.

Qualitative results on samples from the CVPPP and Ovules datasets are illustrated in Figure 3.7.

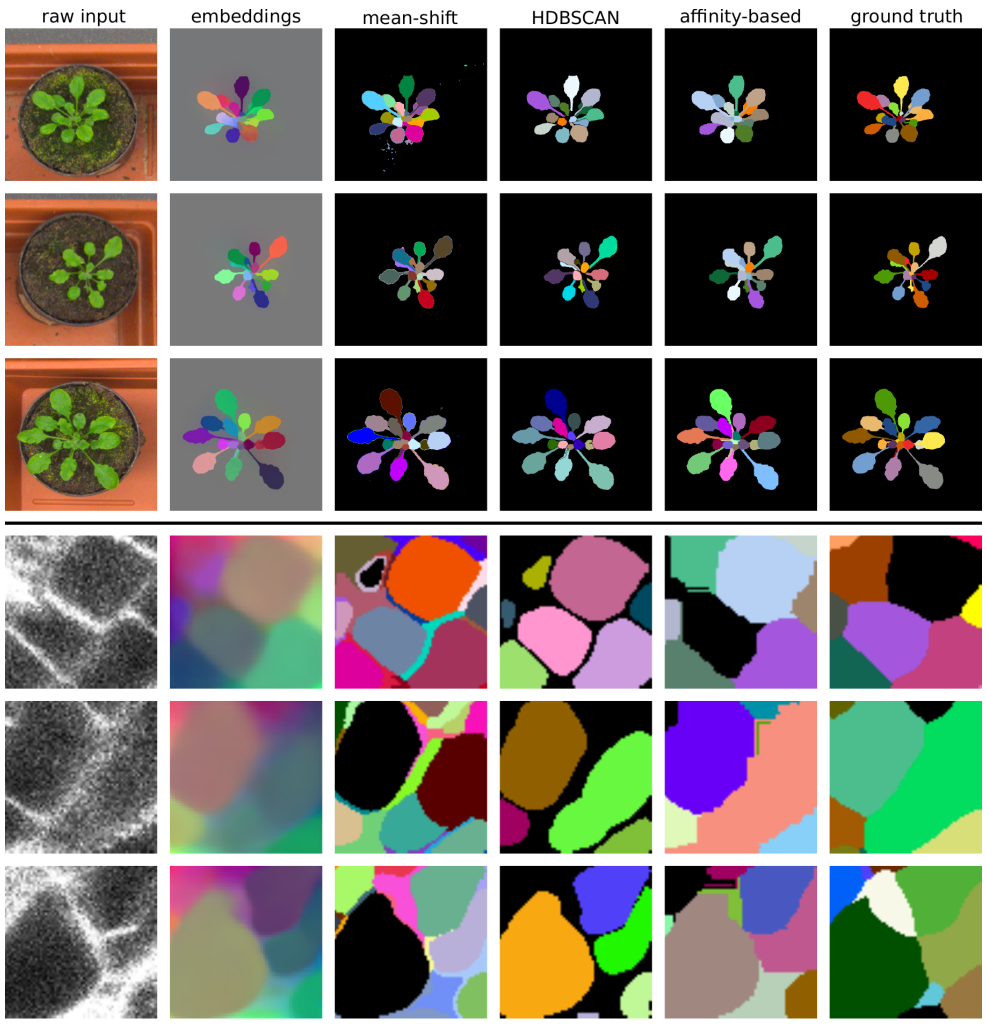


Figure 3.7: Qualitative comparison of different clustering schemes on the samples from the CVPPP validation set (top) and Ovules test set (bottom). Fully supervised SPOCO was used to train embeddings.

3.5 Ablation Study of SPOCO

In this section, I investigate the importance of different loss components as well as the choice of the g-network and their impact on the training dynamics and the final segmentation.

3.5.1 Ablation of Loss Functions

I study the impact of different loss variants and the choice of $g(\cdot)$ network on the final performance of my method. Table 3.9 presents segmentation and counting scores (CVPPP test set) with different loss variants. HDBSCAN with $min\_size = 200$ and no foreground mask was used for clustering the network outputs in all cases. I see that when only a few ground truth objects are used for training (10%, 40%) the consistency term L_{U_con} (Equation 3.8) has a much stronger impact on the final segmentation performance than the unlabeled "push" term L_{U_push} (Equation 3.7). The absence of L_{U_push} term worsens the segmentation and counting scores in all experiments.

Loss function	SBD	$\|DiC\|$
SPOCO@0.1	0.788 ± 0.017	5.4 ± 0.3
SPOCO@0.1 w/o L_{U_push}	0.734 ± 0.042	8.5 ± 0.4
SPOCO@0.1 w/o L_{U_con}	0.720 ± 0.037	6.3 ± 0.1
SPOCO@0.4	0.824 ± 0.003	3.2 ± 0.5
SPOCO@0.4 w/o L_{U_push}	0.779 ± 0.045	3.0 ± 0.7
SPOCO@0.4 w/o L_{U_con}	0.738 ± 0.019	2.1 ± 0.1
SPOCO@0.8	0.828 ± 0.010	1.6 ± 0.2
SPOCO@0.8 w/o L_{U_push}	0.797 ± 0.014	1.9 ± 0.4
SPOCO@0.8 w/o L_{U_con}	0.810 ± 0.010	2.1 ± 0.2

Table 3.9: Ablation study of different loss variants. I report segmentation (SBD) and counting ($|DiC|$) scores on the CVPPP test set. Ablation of the L_{U_con} L_{U_push} term in the semi-supervised setting is reported for 10%, 40% and 80% of randomly selected ground truth objects. Mean ± SD are reported across 3 random samplings of the ground truth objects.

In transfer learning setting, the embedding network trained on the source domain is fine-tuned on the target domain with just a few groundtruth objects. Results on the EM data, where VNC dataset [49] is the source and the MitoEM [137] is the target domain (Table 3.10) show significant drop in segmentation scores across all experiments if the embedding consistency is removed from the loss.

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

Table 3.10: Ablation of the consistency term L_{U_con} in the transfer learning setting with 1%, 5%, 10% of groundtruth objects (target domain). Average precision measured on the target task of MitoEM mitochondria segmentation is reported. VNC dataset serves as a source domain. Mean ± SD are reported across 3 random samplings of the instances from the target dataset.

To finalize the ablation study, I trained the network in a fully-supervised setting using the consistency regularization from the weakly-supervised setup. Table 3.11 shows the comparison between all four variants. Using the consistency term L_{U_con} together with the instance-based term L_{obj} on the Cityscapes validation set gives the highest mAP@0.5 score of 0.459 as compared to 0.387 (discriminative loss), 0.418 (discriminative loss + L_{obj}) and 0.429 (discriminative loss + L_{U_con}). For CVPPP the SBD metric on the validation set improves from 0.847 (full supervision as in [16]) to 0.852 (discriminative loss + L_{obj}), to 0.853 (discriminative loss + L_{obj}) and to 0.849 (discriminative loss + L_{obj} + L_{U_con}).

Loss function	CVPPP	Cityscapes
Discriminative Loss [16]	0.847	0.387
Discriminative Loss + L_{obj}	0.852	0.418
Discriminative Loss + L_{U_con}	**0.853**	0.429
Discriminative Loss + L_{obj} + L_{U_con}	0.849	**0.459**

Table 3.11: Weakly-supervised regularization in the fully-supervised setting. SBD (CVPPP datasets) and mAP@0.5 (Cityscapes dataset) computed on the validation sets.

3.5.2 G-network Ablations

I experiment with different types of the g network used in the consistency loss to better understand its effect. Apart from the momentum g described in Section 3.3.2 I consider three

other variants: (1) weights are *shared* between f and g, i.e. $\theta_g = \theta_f$, (2) g shares the weights with f, but uses spatial *dropout* [127] in the bottleneck layer of the U-Net architecture, (3) g uses independent set of weights *trained* by back-propagation.
For the purpose of this ablation, I split the CVPPP A1 training set into 103 randomly selected images used for training and report the results on the remaining 25 images.

Table 3.12 shows the segmentation and counting scores for the HDBSCAN-clustered embeddings together with training dynamics for different variants of g. "Dropout" variant shows a good initial convergence rate, but overfits quickly and has the worse final performance. "Trained" and "shared" variants show comparable average scores, however the variance is much larger for the "trained", which is prone to training instabilities. "Momentum" outperforms the others by a large margin and has the fastest convergence speed. Figure 3.8 shows the PCA-projected netork outputs for two randomly selected CVPPP images (test set) and five different settings. One can see that sparse training without the consistency loss (column 2) fails to separate the background. Using dropout g leads to artifacts in the unlabeled region. "Shared" and "trained" variants of g (columns 4 and 5) provide limited background separation, but fail to produce crisp embeddings. The "momentum" variant (column 6) is able to correctly separate the background.

g-network	SBD	$\lvert DiC \rvert$
Shared	0.602 ± 0.016	6.0 ± 0.6
Dropout	0.507 ± 0.060	7.9 ± 0.9
Trained	0.591 ± 0.131	5.7 ± 2.0
Momentum	$\mathbf{0.649} \pm \mathbf{0.045}$	$\mathbf{4.5} \pm \mathbf{1.6}$

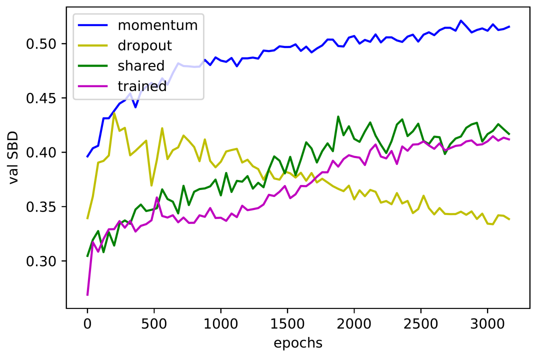


Table 3.12: **Top**: segmentation performance computed for different types of g-network on the CVPPP validation set. **Bottom**: comparison between g-network variants during training. SPOCO@0.1 used for training, mean $\pm$ SD across 3 random samplings of the groundtruth objects is shown.

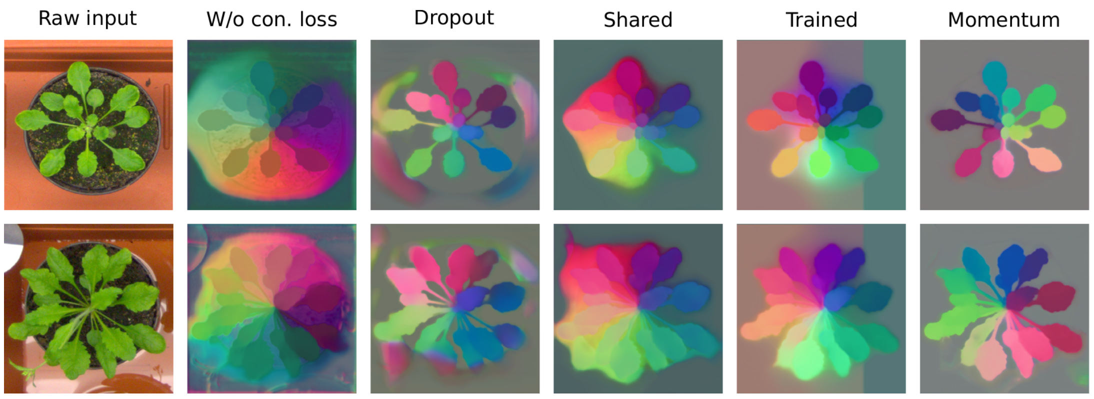


Figure 3.8: Qualitative comparison of the PCA-projected outputs from the network f for different training setups: (col 2) no consistency term, (col 3) dropout g, (col 4) shared g, (col 5) trained g, (col 6) momentum g. Two images from the CVPPP test set were randomly selected. SPOCO@0.1 was used for training.

3.5.3 Adversarial Training

As mentioned in Section 3.3.1 the differentiable object selection can be used as a basis for the the adversarial training. I this case, the pixel embedding network plays the role of the generator: it generates object masks for randomly selected anchor points, whereas a separate discriminator network learns to distinguish between the groundtruth masks and generated masks.

For adversarial training I use Wasserstein GAN with gradient penalty (WGAN-GP) [52] objective function given by:

$$V_D(G, D) = \mathbb{E}_{\tilde{\mathbf{x}}\sim\mathbb{P}_g}[D(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}[D(\mathbf{x})] + \lambda\mathbb{E}_{\hat{\mathbf{x}}\sim\mathbb{P}_{\hat{\mathbf{x}}}}[(\|\nabla_{\hat{\mathbf{x}}}D(\hat{\mathbf{x}})\|-1)^2] \tag{3.11}$$

for the critic D, and:

$$V_G(G, D) = -\mathbb{E}_{\tilde{\mathbf{x}}\sim\mathbb{P}_g}[D(\tilde{\mathbf{x}})] \tag{3.12}$$

for the generator G. In our case the final loss for the generator, i.e. the embedding network, is a linear combination of the embedding objective and the WGAN loss:

$$L_{adv} = L_{emb} + \zeta V_G(G, D) \tag{3.13}$$

with L_{emb} is either L_{SO} (full supervision) defined in Equation 3.6 or L_{SSO} (weak supervision) defined in Equation 3.9. In the equations above: $\mathbb{P}_r$ is the distribution of ground truth mask, $\mathbb{P}_g$ is the distribution of predicted "soft" masks and $\mathbb{P}_{\hat{\mathbf{x}}}$ is the sampling distribution (uniformly sampling along straight lines between pairs of points sampled from the data distribution $\mathbb{P}_r$ and the generator distribution $\mathbb{P}_g$). Table 3.15 (middle-bottom) shows the architecture of the critic used in the Ovules dataset experiments. Training of the embedding network and the critic is done using the Adam optimizer with $\beta_1 = 0.5$, $\beta_2 = 0.9$ and initial learning rate of 0.0001 for both networks. I use $n_{critic} = 5$ iterations per each iteration of the embedding network. I use $\lambda = 10$ (gradient penalty weight) and $\zeta = 0.1$ in my experiments. In order to prevent uninformative gradients from the critic at the beginning of the training process, L_{wgan} is enabled after the warm-up period of 50K iterations.

In my experiments, adversarial training does not by itself bring a significant performance improvement over the Dice-based loss. However using the combination of both can be beneficial as I show in Table 3.3: the combined loss outperforms a much more complex 3-step state-of-the-art segmentation pipeline. This finding is similar to [89] where authors use an adversarial approach to train a semantic segmentation model. My approach differs, because the discriminator focuses more on the individual object properties instead of the global statistics of the semantic mask predicted by the network.

3.5.4 Cityscapes Single-class vs Class-agnostic

In Table 3.13, I compare two different training setups at different object sampling ratios for the Cityscapes dataset: (1) *single-class* reported in the main text, where embedding network is trained on objects from a single semantic class and (2) *class-agnostic* where all objects from all classes are used to train a single embedding network. The class-agnostic training works better for riders, cars, motorcycles and bicycles at all sampling levels. The single-class training is better for trucks, buses, and trains. I hypothesize that the class-agnostic setup learns better representation of objects from correlated classes (e.g. person and rider, motorcycle and bicycle), but it is detrimental to trucks, buses and trains due to heavy class imbalance. A per-class weighting of the instance-based term could be beneficial in the class-agnostic setting, which I leave for future work.

Method	person	rider	car	truck	bus	train	motorcycle	bicycle	**average**
single-class@0.1	0.190	0.360	0.236	**0.438**	**0.481**	**0.490**	0.424	0.204	**0.353**
class-agnostic@0.1	**0.197**	**0.430**	**0.282**	0.243	0.276	0.167	**0.468**	**0.261**	0.291
single-class@0.4	**0.230**	0.396	0.301	**0.558**	**0.601**	**0.594**	0.405	0.214	**0.412**
class-agnostic@0.4	0.207	**0.459**	**0.332**	0.260	0.336	0.223	**0.471**	**0.266**	0.319
single-class@1.0	**0.260**	0.451	0.331	**0.604**	**0.637**	**0.656**	0.464	0.266	**0.459**
class-agnostic@1.0	0.259	**0.463**	**0.410**	0.370	0.395	0.378	**0.478**	**0.296**	0.381

Table 3.13: Comparison of SPOCO trained in a single-class vs class-agnostic settings at different sampling ratios. Shown are mAP@0.5 scores computed on the Cityscapes validation set.

3.5.5 Momentum Coefficient Exploration

In this experiment, I explore the effect of the momentum coefficient m used in the momentum update of the g-network parameters (see Section 3.3.2). I use the train/val split of the CVPPP training set described in Section 3.5.2. Similar to [56] I show in Table 3.14 that the large momentum ($m = 0.999$) performs best. I hypothesize that using slowly moving g moving acts as a strong regularizer which prevents the embedding network f to adapt too quickly to the spare ground truth signal.

m	SBD	$\lvert DiC \rvert$
0.99	0.615 $\pm$ 0.042	5.2 $\pm$ 0.8
0.995	0.622 $\pm$ 0.116	5.4 $\pm$ 0.7
0.999	**0.649** $\pm$ **0.045**	**4.5** $\pm$ **1.6**

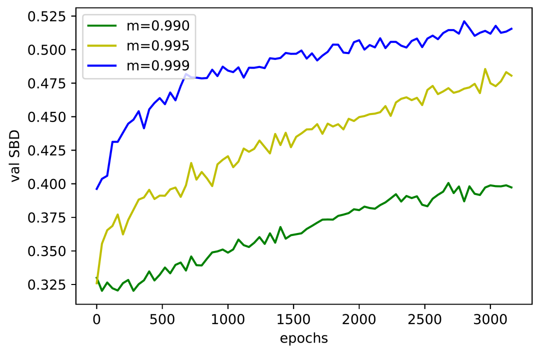


Table 3.14: The effect of the momentum coefficient value m on the SPOCO performance. (**top**) Segmentation and counting scores. (**bottom**) Evolution of the validation score during training. SPOCO@01 was used for training. Mean $\pm$ SD across 3 random samplings of the ground truth objects is shown.

3.5.6 Kernel Threshold Exploration

Figure 3.9 illustrates the effect of the kernel threshold parameter t (Equation 3.3) on the SPOCO model performance. Choosing a large value (e.g. $t = 0.9$) leads to a crisper, more separable embeddings than smaller values (e.g. $t \in \{0.25, 0.5, 0.75\}$). The difference in the final segmentation performance between a small and a large value of t is especially apparent in the sparse annotation regime. Indeed, when training with only 10% (SPOCO@0.1) or 40% (SPOCO@0.4) of ground truth objects, the mean SBD score improvement between $t = 0.5$ and $t = 0.9$ is 0.044 and 0.054 respectively. Although the performance gain is less pronounced when more supervision is provided (for SPOCO@0.8 the mean SBD reaches a plateau for $t \geq 0.5$), models trained with higher values of t are more robust as shown by the low variance of the SBD score. In my experiments, values of t greater than 0.95 lead to training instabilities.

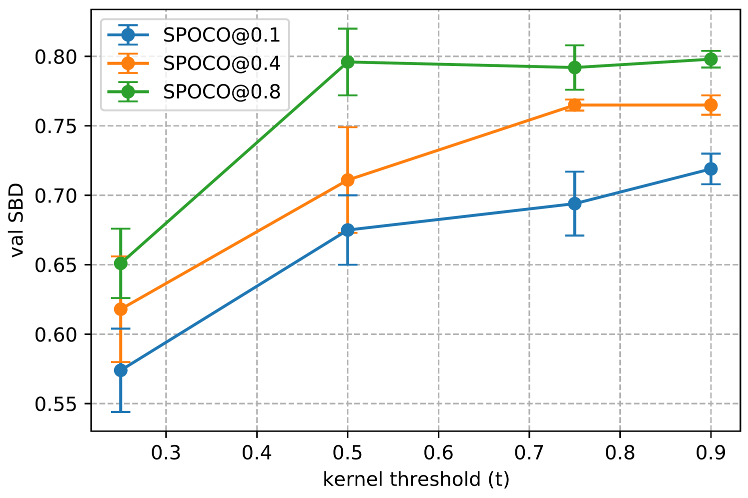


Figure 3.9: Effect of the kernel threshold t on the segmentation performance at different ground truth objects sampling rates $(0.1, 0.4, 0.8)$. SBD scores measured on the CVPPP validation set are shown for models trained with four values of t: $0.25, 0.5, 0.75, 0.9$. Mean $\pm$ SD are reported across 3 training runs for each (sampling rate, kernel threshold) pair. HDBSCAN $(min\_size = 200)$ is used for clustering.

3.6 Network Architecture and Training Parameters

The structure of the U-Net embedding network used for each dataset is described using the convolutional building block shown in Figure 3.10. The number of convolutional blocks in the encoder/decoder part of U-Net is chosen such that the receptive field of features in the last encoder layer is equal to or slightly bigger than the input size. I use group normalization [143] for 3D and electron microscopy experiments and batch normalization for CVPPP and Cityscapes datasets [62].

Details of the architectures used in experiments on different benchmark datasets as well as the architecture of the WGAN disriminator used in the adversarial setting (Section 3.5.3) are shown in Table 3.15.

Unless otherwise specified, Adam optimizer [71] with an initial learning rate of 0.0002, weight decay 10^{-5}, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ was used for training. Learning rate was reduced by a factor of 0.2 when the validation loss stopped improving after a dataset-dependent number of iterations. Training was stopped when the learning rate dropped below 10^{-6} or maximum number of iterations was reached.

I use 16-dimensional embedding space in all my experiments except Cityscapes, where 8-dimensional embeddings are used. Input images were globally normalized to zero mean and a unit standard deviation unless stated otherwise.

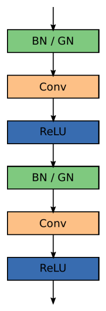


Figure 3.10: The convolutional block consist of two convolutional layers, optionally preceded by the normalization layer (*BN* for batchnorm, *GN* for groupnorm), and followed by the ReLU activation function. I use *ConvBN*, *ConvGN* or *Conv* to refer to the convolutional block with batchnorm [62], groupnorm [143] or no normalization layers respectively.

$x \in \mathbb{R}^{H\times W\times 3}$
ConvBN, 16, MP 2×2
ConvBN, 32, MP 2×2
ConvBN, 64, MP 2×2
ConvBN, 128, MP 2×2
ConvBN, 256, MP 2×2
ConvBN, 512, Up 2×2
Concat, 256 + 512
ConvBN, 256, Up 2×2
Concat, 128 + 256
ConvBN, 128, Up 2×2
Concat, 64 + 128
ConvBN, 64, Up 2×2
Concat, 32 + 64
ConvBN, 32, Up 2×2
Concat, 16 + 32
ConvBN, 16
1×1 conv, d

$x \in \mathbb{R}^{D\times H\times W\times 1}$
ConvGN, 64, MP 2×2
ConvGN, 128, MP 2×2
ConvGN, 256, MP 2×2
ConvGN, 512, Up 2×2
Concat, 256 + 512
ConvGN, 256, Up 2×2
Concat, 128 + 256
ConvGN, 128, Up 2×2
Concat, 64 + 128
ConvGN, 64
1×1 conv, $d = 16$

$x \in [0,1]^{D\times H\times W\times 1}$
Conv, 64, MaxPool 2×2
Conv, 128, MaxPool 2×2
Conv, 256, MaxPool 2×2
Conv, 512, Upsample 2×2
dense layer, 1

$x \in \mathbb{R}^{H\times W\times 1}$
ConvGN, 16, MP 2×2
ConvGN, 32, MP 2×2
ConvGN, 64, MP 2×2
ConvGN, 128, MP 2×2
ConvGN, 256, MP 2×2
ConvGN, 512, MP 2×2
ConvGN, 1024, Up 2×2
Concat, 512 + 1024
ConvGN, 512, Up 2×2
Concat, 256 + 512
ConvGN, 256, Up 2×2
Concat, 128 + 256
ConvGN, 128, Up 2×2
Concat, 64 + 128
ConvGN, 64, Up 2×2
Concat, 32 + 64
ConvGN, 32, Up 2×2
Concat, 16 + 32
ConvGN, 16
1×1 conv, $d = 16$

Table 3.15: U-Net architectures used for different benchmark datasets. In each table: first row corresponds to the input image dimension, subsequent rows show the operation name and the number of output channels after the operation, *Up* denotes nearest-neighbor upsampling, *MP* denotes max pooling operation and *Concat* denotes channel-wise concatenation of the output for a given decoder layer with the output from the corresponding encoder layer. d refers to the dimensionality of the output embeddings. **LEFT**: U-Net architecture for CVPPP and Cityscapes datasets. $(H, W) = (448, 448)$, $d = 16$ for CVPPP and $(H, W) = (384, 768)$, $d = 8$ for Cityscapes. **MIDDLE-TOP**: U-Net architecture for Ovules and Stem datasets, $(D, H, W) = (40, 64, 64)$. **MIDDLE-BOTTOM**: WGAN-GP critic architecture on Ovules dataset, $(D, H, W) = (40, 64, 64)$. Here, ReLU was replaced by leaky ReLU activation function with $\alpha = 10^{-2}$. **RIGHT**: U-Net architecture for VNC and MitoEM datasets, $(H, W) = (448, 448)$.

CVPPP Table 3.15 (left) shows the 2D U-Net architecture used in the experiment. All networks were trained for up to 80K iterations (unless the stopping criteria was not satisfied before) with a minibatch size of 4. Input images were randomly scaled, flipped horizontally and vertically and cropped to 448×448 pixels. Before passing to $f(\cdot)$ and $g(\cdot)$ networks, random color jitter and Gaussian blur were applied.

Cityscapes See Table 3.15 (left) for an overview of 2D U-Net architecture for the Cityscapes semantic instance segmentation task. All networks were trained for up to 90K iterations with a minibatch size of 16. The network output dimension was set to 8. Input images were randomly cropped to 358×768 patches. Random flipping and scaling (ratio in [0.5, 2.0]), Gaussian blurring, color jitter and random conversion to grayscale was applied to the input before passing it to $f(\cdot)$ and $g(\cdot)$ networks.

Light microscopy datasets 3D U-Net architecture used for the light microscopy datasets is shown in Table 3.15 (middle-top). Ovules networks were trained for up to 200K iterations (or until the stopping criteria was satisfied) with a minibatch size of 8. Stem networks were fine-tuned with a fixed, reduced learning rate of 0.00002 for 100K iterations. 3D patches of shape 40×64×64 (*ZYX* axes ordering) were used. Patches were augmented with random rotations, flips and elastic deformations. Gaussian noise was added to the input before passing through $f(\cdot)$ and $g(\cdot)$ networks.
For a fair comparison with other methods I do not stitch the patches to recover the whole volume, but evaluate on the patch-by-patch basis.

Electron microscopy datasets 2D U-Net architecture for the VNC and MitoEM datasets is shown in Table 3.15 (right). The source VNC network was trained for up to 100K iterations with a minibatch size of 4. MitoEM networks were fine-tuned with a fixed, reduced learning rate of 0.00002 for 100K iterations. 2D patches of shape 448×448 were used. Patches were augmented with random rotations, flips and elastic deformations. Gaussian noise was added to the input before passing through $f(\cdot)$ and $g(\cdot)$ networks.

3.7 Conclusion

I presented a novel approach to weak supervision for instance segmentation tasks which enables training in a positive unlabeled setting. Here, only a subset of object masks are annotated with no annotations in the background and the loss is applied directly to the annotated objects via a differentiable instance selection step. The unlabeled areas of the images contribute to the training through a self-supervised instance-level consistency loss. This setup, which allows to do instance segmentation using just a small number of positive labeled instances, shows great potential in the fields of perception, bio and medical image analysis where accessing high-quality training data is always a bottleneck.

I demonstrate the advantage of single-instance losses in a fully supervised setting, reaching state-of-the-art performance on the CVPPP benchmark and improving on strong baselines in several microscopy datasets. Weak positive unlabeled supervision is evaluated on the Cityscapes instance segmentation task and on biological datasets from light and electron microscopy, 2D and 3D, in direct training and in transfer learning. In all cases, the network demonstrates strong segmentation performance at a very reduced manual annotation cost. To the best of my knowledge, this is the first work to consider the positive-unlabelled supervision for the instance segmentation task.

In the future, I plan to explore the possibility of fully self-supervised pre-training using the consistency loss and an extended set of augmentations. This would open up the possibility for efficient fine-tuning of the learned feature space with point supervision for both semantic and instance segmentation tasks.

Limitations. The main drawback of the proposed approach is the lack of a universal clustering method to assign instance labels to pixels based on their embeddings. The existing methods all have benefits and drawbacks; there is no consistent winner that would work robustly across all segmentation benchmarks.

4 Software

As part of this thesis, I created or contributed to multiple open-source software projects that help researchers and practitioners segment, analyze and visualize 2D and 3D biological datasets. In this chapter, I briefly describe the most popular of these projects and their main functionalities.

4.1 pytorch-3dunet

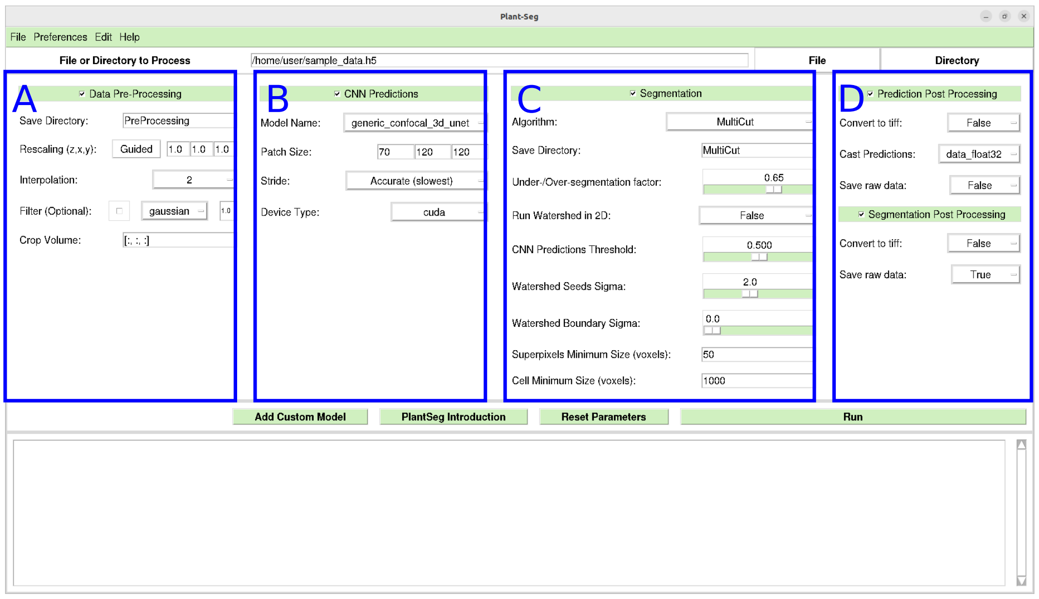
I created *pytorch-3dunet* a popular library for training dense segmentation networks. It provides a Pytorch [108] implementation of the various CNN architectures (e.g. 3D U-Net [21], Residual 3D U-Net [77]) and training routines for semantic segmentation and regression problems in 2D and 3D. Using the software, users can train complex networks and run predictions on their data without writing a single line of code. It provides common functionalities for neural network training and inference accessible via a convenient config-based interface. A sample YAML configuration file is shown in Listing 1. The user can specify the most common aspects of the training and prediction process, such as choosing the network architecture, optimization algorithm, loss function, data loader and data augmentation. The package supports parallel execution across multiple GPUs. At the time of writing this thesis the project has more than 1.3k stars on GitHub and has been extensively used by researchers (e.g. [119]) and machine learning practitioners. The source code together with a detailed documentation is available on GitHub: `https://github.com/wolny/pytorch-3dunet`.

```
1  model:
2    # model class
3    name: UNet3D
4    # number of input channels to the model
5    in_channels: 1
6    # number of output channels
7    out_channels: 1
8    # number of initial feature maps
9    f_maps: 32
10   # apply element-wise sigmoid after the final 1x1x1 conv
11   final_sigmoid: true
12 # loss function configuration
13 loss:
14   name: BCELoss
15 # optimizer configuration
16 optimizer:
17   learning_rate: 0.0002
18   weight_decay: 0.00001
19 # learning rate scheduler configuration
20 lr_scheduler:
21   name: ReduceLROnPlateau
22 # training directory and stopping criteria
23 trainer:
24   checkpoint_dir: CHECKPOINT_DIR
25   max_num_epochs: 1000
26   max_num_iterations: 150000
27 # data loaders configuration
28 loaders:
29   ...
```

Listing 1: An example of a YAML configuration file used for training a 3D segmentation network. User can specify various hyperparameters such as: the model architecture (l. 1-11), loss function (l. 13-14), parameters of the Adam [71] optimizer (l. 16-18), learning rate decay (l. 20-21) and stopping criteria (l. 25-26). Configuration of data loaders and augmentation pipeline has been omitted for clarity.

4.2 PlantSeg

Together with Lorenzo Cerrone I created *PlantSeg* a pipeline for volumetric segmentation of biological tissues into cells described in Chapter 2. The package comes with a large number of neural networks pre-trained on confocal and light-sheet microscopy datasets of fixed and live plant tissues. It relies on pytorch-3dunet for neural network predictions (Section 4.1) and elf library (`https://github.com/constantinpape/elf`), developed by Constantin Pape, for graph-based image partitioning. Figure 4.1 illustrates the main components of the graphical user interface (GUI). It allows the user to easily configure all steps of the segmentation pipeline, such as: selecting the neural network model and specifying hyperparameters of the partitioning algorithm. The source code, descriptions of the hyperparameters and detailed documentation of the software can be found on GitHub: `https://github.com/hci-unihd/plant-seg`. PlantSeg package is being actively developed and in the future we plan to extend it with a basic segmentation proofreading functionality and the SPOCO weakly supervised training scheme (Chapter 3).



File or Directory to Process: /home/user/sample_data.h5

☑ Data Pre-Processing
Save Directory: PreProcessing
Rescaling (z,x,y): Guided 1.0 1.0 1.0
Interpolation: 2
Filter (Optional): ☐ gaussian 1.0
Crop Volume: [:, :, :]

☑ CNN Predictions
Model Name: generic_confocal_3d_unet
Patch Size: 70 120 120
Stride: Accurate (slowest)
Device Type: cuda

☑ Segmentation
Algorithm: MultiCut
Save Directory: MultiCut
Under-/Over-segmentation factor: 0.65
Run Watershed in 2D: False
CNN Predictions Threshold: 0.500
Watershed Seeds Sigma: 2.0
Watershed Boundary Sigma: 0.0
Superpixels Minimum Size (voxels): 50
Cell Minimum Size (voxels): 1000

☑ Prediction Post Processing
Convert to tiff: False
Cast Predictions: data_float32
Save raw data: False

☑ Segmentation Post Processing
Convert to tiff: False
Save raw data: True

Figure 4.1: Graphical user interface of PlantSeg allows to configure all steps of the segmentation pipeline, i.e. pre-processing (A), model selection (B), partitioning algorithm (C) and post-processing (D).

4.3 Open Source Contributions

Apart from developing pytorch-3dunet and Plantseg, I have also implemented new functionalities in several well-known, open sources projects, in particular:

- I contributed to *ilastik*[1] a highly popular interactive learning and segmentation toolkit [13]. It leverages both shallow and deep machine learning algorithms to solve segmentation, tracking and and counting of cells and other objects of interests. In this project, I have made several improvements to the backend engine and the graphical user interface.
- I co-authored *ilastik4ij*[2] a plugin for Fiji[3], a well known biological-image analysis software [115]. The plugin is a wrapper around ilastik workflows (pixel classification, object classification, tracking), which allows them to be executed in Fiji. It also provides the functionality to read and modify the HDF5 file format [41].
- I also contributed to *MorphoGraphX*[4] a feature-rich application for the visualization and analysis of biological datasets [6]. Here, I helped to develop a plugin for the HDF5 data storage.

[1] `https://www.ilastik.org`
[2] `https://github.com/ilastik/ilastik4ij`
[3] `https://imagej.net/software/fiji`
[4] `https://morphographx.org/`

5 Conclusion

In this work, I have developed a tool for segmenting tissues into cells accessible to both novice and expert users (Chapter 2). This method, based on the CNN boundary predictors and graph partitioning algorithms, delivers accurate segmentation across different light microscopy datasets. Its robustness has been confirmed in an independent study [67] and I have demonstrated its utility to answer biological research questions in several collaborative studies [134, 118, 107, 61, 15]. I have also introduced SPOCO (Chapter 3), a semi-supervised instance segmentation, where a combination of contrastive learning and novel self-supervised objective allows to train embedding networks from sparse object annotations. Apart from state-of-the-art results on the CVPPP benchmark [100], my method uses *positive unlabeled* learning, a kind of weak supervision previously unexplored for the instance segmentation problems. It can be trained with sparse annotations directly, without relying on pre-trained backbones or saliency detectors and also demonstrates strong performance in transfer learning problems. Overall, I have developed instance segmentation tools and techniques, which are easily accessible *and* can be trained from sparse groundtruth data.

Future Work Due to their limited generalizability, the current machine learning systems need a very large amount of manually annotated data for training. Because of the difficulty in collecting large, expert-annotated datasets in areas of modern microscopy and medical imaging, I expect weakly-supervised methods such as SPOCO to play an important role for segmentation in these challenging domains. In particular, datasets with sub-domains, such as microscopy data with various fluorescent staining and acquisition settings, or medical imaging with different patient populations and health conditions, make an interesting use cases for SPOCO. In these settings, positive unlabeled supervision in each sub-domain can improve generalization without increasing the annotation time.

I believe that self-supervised learning (SSL) is one of the most promising ways to reduce the need for labeled examples. This technique obtains supervisory signals from the data itself, which allows the models to learn from orders of magnitude more data. It had great success in advancing the field of natural language processing [98, 33, 29], computer vision [56, 51, 25, 19, 20, 148] and protein structure prediction [65]. Recently, these methods have been adapted to semantic segmentation [48], but so far has been under-explored for instance segmentation. Here, SPOCO with its differentiable object sampling and self-supervised consistency objective offers interesting perspectives. One research direction is to explore more complex data augmentations, crucial for learning good representations [25], together with SPOCO self-supervised consistency objective in order to learn good pixel representations in an unsupervised fashion. Such pixel-level representations can be fine-tuned for the downstream visual tasks, such as semantic, instance or panoptic segmentation. Another research direction is using my differentiable instance sampling procedure (Section 3.3) to mine object mask proposals for self-training. More specifically, one could sample a number of anchor pixels and their corresponding instance masks from the image, then choose the most confident objects as *pseudo-labels* for label propagation [81, 133, 150] or teacher-student knowledge distillation [58].

Another technique to reduce the cost of dense pixel-wise labelling is combining shallow classifiers trained with scribble annotations, with the performance of deep CNNs. I have contributed to this research direction in [94], where a CNN is trained to correct the errors of a shallow model in a transfer learning setting. Based on these findings, one could incorporate other types of weak training signal, e.g. scribbles or point annotations, into SPOCO formulation to deliver stronger constraints in the unlabeled regions.

On the practical side, given the extensive use of the PlantSeg package for microscopy data analysis I envision its further development, such as including the proofreading functionality and integration with SPOCO for fast segmentation of biological data with minimal supervision.

In summary, approaches like SPOCO, combining differentiable instance selection and self-supervised learning, should be explored as a viable solution for instance segmentation in challenging domains with limited data.

Publications

As part of this work I have contributed to the following peer-reviewed publications:

[1] Stuart Berg et al. "ilastik: interactive machine learning for (bio)image analysis." In: *Nature Methods* 16.12 (Dec. 2019), pp. 1226–1232. URL: https://doi.org/10.1038/s41592-019-0582-9.

[2] V. Bondarenko et al. "Coordination Between Embryo Growth and Trophoblast Migration Upon Implantation Delineates Mouse Embryogenesis." In: *bioRxiv* (2022). eprint: https://www.biorxiv.org/content/early/2022/06/16/2022.06.13.495767.full.pdf. URL: https://www.biorxiv.org/content/early/2022/06/16/2022.06.13.495767.

[3] Takafumi Ichikawa et al. "An ex vivo system to study cellular dynamics underlying mouse peri-implantation development." In: *Developmental Cell* 57.3 (2022), 373–386.e9. URL: https://www.sciencedirect.com/science/article/pii/S1534580721010431.

[4] Alex Matskevych, Adrian Wolny, Constantin Pape, and Anna Kreshuk. "From Shallow to Deep: Exploiting Feature-Based Classifiers for Domain Adaptation in Semantic Segmentation." In: *Frontiers in Computer Science* 4 (2022). URL: https://www.frontiersin.org/article/10.3389/fcomp.2022.805166.

[5] Constantin Pape, Alex Matskevych, Adrian Wolny, Julian Hennies, Giulia Mizzon, Marion Louveaux, Jacob Musser, Alexis Maizel, Detlev Arendt, and Anna Kreshuk. "Leveraging domain knowledge to improve microscopy image segmentation with lifted multicuts." In: *Frontiers in Computer Science* 1 (2019), p. 6.

[6] Constantin Pape, Roman Remme, Adrian Wolny, Sylvia Olberg, Steffen Wolf, Lorenzo Cerrone, Mirko Cortese, Severina Klaus, Bojana Lucic, Stephanie Ullrich, et al. "Microscopy-based assay for semi-quantitative detection of SARS-CoV-2 specific antibodies in human sera: A semi-quantitative, high throughput, microscopy-based assay expands existing approaches to measure SARS-CoV-2 specific antibody levels in human sera." In: *BioEssays* 43.3 (2021), p. 2000257.

[7] Lilli Marie Schütz, Marion Louveaux, Amaya Vilches Barro, Sami Bouziri, Lorenzo Cerrone, Adrian Wolny, Anna Kreshuk, Fred A Hamprecht, and Alexis Maizel. "Integration of Cell Growth and Asymmetric Division during Lateral Root Initiation in Arabidopsis thaliana." In: *Plant and Cell Physiology* 62.8 (Mar. 2021), pp. 1269–1279. eprint: https://academic.oup.com/pcp/article-pdf/62/8/1269/41119202/pcab038.pdf. URL: https://doi.org/10.1093/pcp/pcab038.

[8] Athul Vijayan, Rachele Tofanelli, Sören Strauss, Lorenzo Cerrone, Adrian Wolny, Joanna Strohmeier, Anna Kreshuk, Fred A Hamprecht, Richard S Smith, and Kay Schneitz. "A digital 3D reference atlas reveals cellular growth patterns shaping the *Arabidopsis* ovule." In: *eLife* 10 (Jan. 2021). Ed. by Sheila McCormick, Christian S Hardtke, Sheila McCormick, and Dolf Weijers, e63262. URL: https://doi.org/10.7554/eLife.63262.

[9] Adrian Wolny, Qin Yu, Constantin Pape, and Anna Kreshuk. "Sparse Object-Level Supervision for Instance Segmentation With Pixel Embeddings." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 4402–4411.

[10] Adrian Wolny et al. "Accurate and versatile 3D segmentation of plant tissues at cellular resolution." In: *eLife* 9 (July 2020). Ed. by Christian S Hardtke, Dominique C Bergmann, Dominique C Bergmann, and Moritz Graeff, e57613. URL: https://doi.org/10.7554/eLife.57613.

Bibliography

[1] Bjoern Andres, Jorg H Kappes, Thorsten Beier, Ullrich Kothe, and Fred A Hamprecht. "Probabilistic Image Segmentation with Closedness Constraints." In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2611–2618.

[2] Bjoern Andres, Thorben Kroeger, Kevin L Briggman, Winfried Denk, Graham Knott, Ullrich Koethe, and Fred A Hamprecht. "Globally Optimal Closed-surface Segmentation for Connectomics." In: *Proc. Europ. Conf. Comp. Vision* (2012), pp. 1–14.

[3] M. Bai and R. Urtasun. "Deep Watershed Transform for Instance Segmentation." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2858–2866.

[4] Alberto Bailoni, Constantin Pape, Steffen Wolf, Thorsten Beier, Anna Kreshuk, and Fred A Hamprecht. "A generalized framework for agglomerative clustering of signed graphs applied to instance segmentation." In: *arXiv preprint arXiv:1906.11713* (2019).

[5] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. "Correlation clustering." In: *Mach. Learn.* 56.1-3 (2004), pp. 89–113.

[6] Pierre Barbier de Reuille et al. "MorphoGraphX: A platform for quantifying morphogenesis in 4D." In: *eLife* 4 (May 2015). Ed. by Dominique C Bergmann, e05864. URL: https://doi.org/10.7554/eLife.05864.

[7] Thorsten Beier, Thorben Kroeger, Jorg H Kappes, Ullrich Kothe, and Fred A Hamprecht. "Cut, glue & cut: A fast, approximate solver for multicut partitioning." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 73–80.

[8] Thorsten Beier, Constantin Pape, Nasim Rahaman, and Timo et al. Prange. "Multicut brings automated neurite segmentation closer to human performance." In: *Nature Methods* 14.2 (2017), pp. 101–102.

[9] Jessa Bekker and Jesse Davis. "Learning from positive and unlabeled data: A survey." In: *Machine Learning* 109.4 (2020), pp. 719–760.

[10] Miriam Bellver, Amaia Salvador, Jordi Torres, and Xavier Giro-i-Nieto. "Mask-guided sample selection for semi-supervised instance segmentation." In: *Multimedia Tools and Applications* 79.35 (2020), pp. 25551–25569.

[11] Miriam Bellver, Amaia Salvador, Jordi Torres, and Xavier Giró i Nieto. "Budget-aware semi-supervised semantic and instance segmentation." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2019*. 2019, pp. 93–102.

[12] Tristan Bepler, Andrew Morin, Micah Rapp, Julia Brasch, Lawrence Shapiro, Alex J Noble, and Bonnie Berger. "Positive-unlabeled convolutional neural networks for particle picking in cryo-electron micrographs." In: *Nature methods* 16.11 (2019), pp. 1153–1160.

[13] Stuart Berg et al. "ilastik: interactive machine learning for (bio)image analysis." In: *Nature Methods* 16.12 (Dec. 2019), pp. 1226–1232. URL: https://doi.org/10.1038/s41592-019-0582-9.

[14] Serge Beucher and Fernand Meyer. "The morphological approach to segmentation: the watershed transformation." In: *Optical Engineering-New York-Marcel Dekker Incorporated* 34 (1992), pp. 433–433.

[15] V. Bondarenko et al. "Coordination Between Embryo Growth and Trophoblast Migration Upon Implantation Delineates Mouse Embryogenesis." In: *bioRxiv* (2022). eprint: https://www.biorxiv.org/content/early/2022/06/16/2022.06.13.495767.full.pdf. URL: https://www.biorxiv.org/content/early/2022/06/16/2022.06.13.495767.

[16] Bert De Brabandere, Davy Neven, and Luc Van Gool. *Semantic Instance Segmentation with a Discriminative Loss Function*. 2017. arXiv: 1708.02551 [cs.CV].

[17] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. "Density-based clustering based on hierarchical density estimates." In: *Pacific-Asia conference on knowledge discovery and data mining*. Springer. 2013, pp. 160–172.

[18] J. Canny. "A Computational Approach to Edge Detection." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (Nov. 1986), pp. 679–698.

[19] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 9912–9924. URL: https://proceedings.neurips.cc/paper/2020/file/70feb62b69f16e0238f741fab228fec2-Paper.pdf.

[20] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. "Emerging Properties in Self-Supervised Vision Transformers." In: *CoRR* abs/2104.14294 (2021). arXiv: 2104.14294. URL: https://arxiv.org/abs/2104.14294.

[21] Ozgun Cciccek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. “3D U-Net: learning dense volumetric segmentation from sparse annotation.” In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2016, pp. 424–432.

[22] Lorenzo Cerrone, Alexander Zeilmann, and Fred A. Hamprecht. “End-To-End Learned Random Walker for Seeded Image Segmentation.” In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 12551–12560.

[23] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[24] Long Chen, Weiwen Zhang, Yuli Wu, Martin Strauch, and Dorit Merhof. “Semi-supervised Instance Segmentation with a Learned Shape Prior.” In: *Interpretable and Annotation-Efficient Learning for Medical Image Computing*. Springer, 2020, pp. 94–102.

[25] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. “A simple framework for contrastive learning of visual representations.” In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[26] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. “Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation.” In: *CVPR*. 2020.

[27] Hisham Cholakkal, Guolei Sun, Fahad Shahbaz Khan, and Ling Shao. “Object Counting and Instance Segmentation With Image-Level Supervision.” In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 12389–12397.

[28] Dorin Comaniciu and Peter Meer. “Mean shift: A robust approach toward feature space analysis.” In: *IEEE Transactions on pattern analysis and machine intelligence* 24.5 (2002), pp. 603–619.

[29] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. “Unsupervised Cross-lingual Representation Learning at Scale.” In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 8440–8451. URL: https://aclanthology.org/2020.acl-main.747.

[30] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. "The Cityscapes Dataset for Semantic Urban Scene Understanding." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[31] Jean Cousty, Gilles Bertrand, Laurent Najman, and Michel Couprie. "Watershed cuts: Minimum spanning forests and the drop of water principle." In: *IEEE transactions on pattern analysis and machine intelligence* 31.8 (2008), pp. 1362–1374.

[32] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[33] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. URL: https://aclanthology.org/N19-1423.

[34] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of machine learning research* 12.7 (2011).

[35] Dennis Eschweiler, Thiago Vallin Spina, Rohan C. Choudhury, Elliot Meyerowitz, Alexandre Cunha, and Johannes Stegmaier. "CNN-based Preprocessing to Optimize Watershed-based Cell Segmentation in 3D Confocal Microscopy Images." In: *CoRR* abs/1810.06933 (2018). arXiv: 1810.06933. URL: http://arxiv.org/abs/1810.06933.

[36] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The pascal visual object classes (voc) challenge." In: *International journal of computer vision* 88.2 (2010), pp. 303–338.

[37] Thorsten Falk, Dominic Mai, Robert Bensch, Özgün Çiçek, Ahmed Abdulkadir, Yassine Marrakchi, Anton Böhm, Jan Deubner, Zoe Jäckel, Katharina Seiwald, et al. "U-Net: deep learning for cell counting, detection, and morphometry." In: *Nature methods* 16.1 (2019), p. 67.

[38] Alireza Fathi, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama, and Kevin P. Murphy. *Semantic Instance Segmentation via Deep Metric Learning*. 2017. arXiv: 1703.10277 [cs.CV].

[39] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. “Efficient Graph-Based Image Segmentation.” In: *Int. J. Comput. Vision* 59.2 (2004), pp. 167–181.

[40] Romain Fernandez, Pradeep Das, Vincent Mirabet, Eric Moscardi, Jan Traas, Jean-Luc Verdeil, Grégoire Malandain, and Christophe Godin. “Imaging plant growth in 4D: robust tissue reconstruction and lineaging at cell resolution.” In: *Nature Methods* 7.7 (2010), pp. 547–553. URL: https://doi.org/10.1038/nmeth.1472.

[41] Mike Folk, Gerd Heber, Quincey Koziol, Elena Pourmal, and Dana Robinson. “An overview of the HDF5 technology suite and its applications.” In: *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*. 2011, pp. 36–47.

[42] Samantha Fox et al. “Spatiotemporal coordination of cell division and growth during organ morphogenesis.” In: *PLOS Biology* 16.11 (Nov. 2018), pp. 1–48. URL: https://doi.org/10.1371/journal.pbio.2005952.

[43] J Funke, L Mais, A Champion, N Dye, and D Kainmueller. “A Benchmark for Epithelial Cell Tracking.” In: *Computer Vision – ECCV 2018 Workshops* (2019).

[44] Jan Funke, Stephan Saalfeld, Davi Bock, Srini Turaga, and Eric Perlman. *Circuit Reconstruction from Electron Microscopy Images*. URL: https://cremi.org/ (visited on 08/27/2020).

[45] Jan Funke, Fabian Tschopp, William Grisaitis, Arlo Sheridan, Chandan Singh, Stephan Saalfeld, and Srinivas C Turaga. “Large scale image segmentation with structured loss based deep learning for connectome reconstruction.” In: *IEEE transactions on pattern analysis and machine intelligence* 41.7 (2018), pp. 1669–1680.

[46] Jan Funke, Fabian David Tschopp, William Grisaitis, Arlo Sheridan, Chandan Singh, Stephan Saalfeld, and Srinivas C Turaga. “A deep structured learning approach towards automating connectome reconstruction from 3d electron micrographs.” In: *arXiv preprint arXiv:1709.02974* (2017).

[47] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. “Born Again Neural Networks.” In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 1607–1616. URL: https://proceedings.mlr.press/v80/furlanello18a.html.

[48] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. *Unsupervised Semantic Segmentation by Contrasting Object Mask Proposals*. 2021. arXiv: 2102.06191 [cs.CV].

[49] Stephan Gerhard, Jan Funke, Julien Martel, Albert Cardona, and Richard Fetter. *Segmented anisotropic ssTEM dataset of neural tissue*. London, UK, 2013. URL: https://www.zora.uzh.ch/id/eprint/91121/.

[50] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

[51] Jean-Bastien Grill et al. "Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 21271–21284. URL: https://proceedings.neurips.cc/paper/2020/file/f3ada80d5c4ee70142b17b8192b2958e-Paper.pdf.

[52] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. "Improved Training of Wasserstein GANs." In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-Paper.pdf.

[53] Matthias G. Haberl et al. "CDeep3M–Plug-and-Play cloud-based deep learning for image segmentation." In: *Nature Methods* 15.9 (2018), pp. 677–680. URL: https://doi.org/10.1038/s41592-018-0106-z.

[54] Philipp Hanslovsky, Vanessa Leite, Stephan Saalfeld, Igor Pisarev, Jan Funke, Tobias Pietzsch, Ulrik Günther, John Bogovic, Uwe Schmidt, and Juan Nunez-Iglesias. *saalfeldlab/paintera paintera-0.20.1*. Sept. 2019. URL: https://doi.org/10.5281/zenodo.3458575.

[55] Zeeshan Hayder, Xuming He, and Mathieu Salzmann. "Boundary-Aware Instance Segmentation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[56] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. "Momentum Contrast for Unsupervised Visual Representation Learning." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[57] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask r-cnn." In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[58] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. "Distilling the Knowledge in a Neural Network." In: *NIPS Deep Learning and Representation Learning Workshop*. 2015. URL: http://arxiv.org/abs/1503.02531.

[59] A. Horé and D. Ziou. "Image Quality Metrics: PSNR vs. SSIM." In: *2010 20th International Conference on Pattern Recognition*. Aug. 2010, pp. 2366–2369.

[60] Andrea Horňáková, Jan-Hendrik Lange, and Bjoern Andres. "Analysis and optimization of graph decompositions by lifted multicuts." In: *International Conference on Machine Learning*. 2017, pp. 1539–1548.

[61] Takafumi Ichikawa et al. "An ex vivo system to study cellular dynamics underlying mouse peri-implantation development." In: *Developmental Cell* 57.3 (2022), 373–386.e9. URL: https://www.sciencedirect.com/science/article/pii/S1534580721010431.

[62] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 448–456. URL: https://proceedings.mlr.press/v37/ioffe15.html.

[63] Michal Januszewski, Jeremy Maitin-Shepard, Peter Li, Jörgen Kornfeld, Winfried Denk, and Viren Jain. "Flood-Filling Networks." In: *CoRR* abs/1611.00421 (2016). arXiv: 1611.00421. URL: http://arxiv.org/abs/1611.00421.

[64] Florian Jug, Evgeny Levinkov, Corinna Blasse, Eugene W. Myers, and Bjoern Andres. "Moral Lineage Tracing." In: *CoRR* abs/1511.05512 (2015). arXiv: 1511.05512. URL: http://arxiv.org/abs/1511.05512.

[65] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold." In: *Nature* 596.7873 (Aug. 2021), pp. 583–589. URL: https://doi.org/10.1038/s41586-021-03819-2.

[66] Jörg Kappes, Markus Speth, Björn Andres, Gerhard Reinelt, and Christoph Schn. "Globally optimal image partitioning by multicuts." In: *Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer. 2011, pp. 31–44.

[67] Anuradha Kar, Manuel Petit, Yassin Refahi, Guillaume Cerutti, Christophe Godin, and Jan Traas. "Assessment of deep learning algorithms for 3D instance segmentation of confocal image datasets." In: *bioRxiv* (2021). eprint: https://www.biorxiv.org/content/early/2021/06/10/2021.06.09.447748.full.pdf.

[68] Brian W Kernighan and Shen Lin. "An efficient heuristic procedure for partitioning graphs." In: *The Bell system technical journal* 49.2 (1970), pp. 291–307.

[69] Margret Keuper, Evgeny Levinkov, Nicolas Bonneel, Guillaume Lavoué, Thomas Brox, and Bjorn Andres. “Efficient decomposition of image and mesh graphs by lifted multicuts.” In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1751–1759.

[70] Anna Khoreva, Rodrigo Benenson, Jan Hosang, Matthias Hein, and Bernt Schiele. “Simple does it: Weakly supervised instance and semantic segmentation.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 876–885.

[71] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization.” In: *Proc. ICLR* (2014).

[72] Alexander Kirillov, Evgeny Levinkov, Bjoern Andres, Bogdan Savchynskyy, and Carsten Rother. “InstanceCut: From Edges to Instances With MultiCut.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[73] Iasonas Kokkinos. “Pushing the Boundaries of Boundary Detection using Deep Learning.” In: *ICLR 2016*. 2015.

[74] Victor Kulikov and Victor Lempitsky. *Instance Segmentation of Biological Images Using Harmonic Embeddings*. 2020. arXiv: 1904.05257 [cs.CV].

[75] Issam H. Laradji, Negar Rostamzadeh, Pedro O. Pinheiro, David Vazquez, and Mark Schmidt. “Proposal-Based Instance Segmentation With Point Supervision.” In: *2020 IEEE International Conference on Image Processing (ICIP)*. 2020, pp. 2126–2130.

[76] Kisuk Lee, Ran Lu, Kyle Luther, and H. Sebastian Seung. “Learning and Segmenting Dense Voxel Embeddings for 3D Neuron Reconstruction.” In: *IEEE Transactions on Medical Imaging* (2021), pp. 1–1.

[77] Kisuk Lee, Jonathan Zung, Peter Li, Viren Jain, and H Sebastian Seung. “Superhuman accuracy on the SNEMI3D connectomics challenge.” In: *arXiv preprint arXiv:1706.00120* (2017).

[78] Laurent Lejeune and Raphael Sznitman. *A Positive/Unlabeled Approach for the Segmentation of Medical Sequences using Point-Wise Supervision*. 2021. arXiv: 2107.08394 [cs.CV].

[79] Qizhu Li, Anurag Arnab, and Philip HS Torr. “Weakly-and semi-supervised panoptic segmentation.” In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 102–118.

[80] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. “Fully Convolutional Instance-Aware Semantic Segmentation.” In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4438–4446.

[81] Di Lin, Jifeng Dai, Jiaya Jia, Kaiming He, and Jian Sun. “ScribbleSup: Scribble-Supervised Convolutional Networks for Semantic Segmentation.” In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 3159–3167.

[82] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. “Feature pyramid networks for object detection.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[83] B. Liu, Y. Dai, X. Li, W.S. Lee, and P.S. Yu. “Building text classifiers using positive and unlabeled examples.” In: *Third IEEE International Conference on Data Mining*. 2003, pp. 179–186.

[84] Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S Yu. “Building text classifiers using positive and unlabeled examples.” In: *Third IEEE International Conference on Data Mining*. IEEE. 2003, pp. 179–186.

[85] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. “An intriguing failing of convolutional neural networks and the coordconv solution.” In: *Advances in Neural Information Processing Systems*. 2018, pp. 9605–9616.

[86] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[87] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization.” In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

[88] Bradley Lowekamp, David Chen, Luis Ibanez, and Daniel Blezek. “The Design of SimpleITK.” In: *Frontiers in Neuroinformatics* 7 (2013), p. 45. URL: https://www.frontiersin.org/article/10.3389/fninf.2013.00045.

[89] Pauline Luc, Camille Couprie, Soumith Chintala, and Jakob Verbeek. *Semantic Segmentation using Adversarial Networks*. 2016. arXiv: 1611.08408 [cs.CV].

[90] A. Lucchi, K. Smith, R. Achanta, G. Knott, and P. Fua. “Supervoxel-Based Segmentation of Mitochondria in EM Image Stacks With Learned Shape Features.” In: *IEEE Transactions on Medical Imaging* 31.2 (Feb. 2012), pp. 474–486.

[91] Amirreza Mahbod, Gerald Schaefer, Benjamin Bancher, Christine Loew, Georg Dorffner, Rupert Ecker, and Isabella Ellinger. “CryoNuSeg: A dataset for nuclei instance segmentation of cryosectioned H&E-stained histological images.” In: *Computers in Biology and Medicine* 132 (2021), p. 104349. URL: https://www.sciencedirect.com/science/article/pii/S0010482521001438.

[92] L Mais, P Hirsch, and D Kainmueller. “PatchPerPix for instance segmentation.” In: *Lecture Notes in Computer Science* 12370 (2020), pp. 288–304.

[93] Martin Maška et al. “A benchmark for comparison of cell tracking algorithms.” In: *Bioinformatics* 30.11 (Feb. 2014), pp. 1609–1617. eprint: http://oup.prod.sis.lan/bioinformatics/article-pdf/30/11/1609/17143058/btu080.pdf. URL: https://doi.org/10.1093/bioinformatics/btu080.

[94] Alex Matskevych, Adrian Wolny, Constantin Pape, and Anna Kreshuk. “From Shallow to Deep: Exploiting Feature-Based Classifiers for Domain Adaptation in Semantic Segmentation.” In: *Frontiers in Computer Science* 4 (2022). URL: https://www.frontiersin.org/article/10.3389/fcomp.2022.805166.

[95] Marina Meilua. “Comparing clusterings by the variation of information.” In: *Learning theory and kernel machines*. Springer, 2003, pp. 173–187.

[96] Fernand Meyer. “Minimum spanning forests for morphological segmentation.” In: *Mathematical morphology and its applications to image processing*. 1994, pp. 77–84.

[97] Fernand Meyer. “Topographic distance and watershed lines.” In: *Signal processing* 38.1 (1994), pp. 113–125.

[98] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. “Distributed Representations of Words and Phrases and their Compositionality.” In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[99] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. *V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation*. 2016. arXiv: 1606.04797 [cs.CV].

[100] Massimo Minervini, Andreas Fischbach, Hanno Scharr, and Sotirios A. Tsaftaris. “Finely-grained annotated datasets for image-based plant phenotyping.” In: *Pattern Recognition Letters* (2015), pp. -. URL: http://www.sciencedirect.com/science/article/pii/S0167865515003645.

[101] Erick Moen, Dylan Bannon, Takamasa Kudo, William Graf, Markus Covert, and David Van Valen. "Deep learning for cellular image analysis." In: *Nature Methods* 16.12 (2019), pp. 1233–1246. URL: https://doi.org/10.1038/s41592-019-0403-1.

[102] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. "Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8837–8845.

[103] Kazuya Nishimura, Ryoma Bise, et al. "Weakly supervised cell instance segmentation by propagating from detection response." In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2019, pp. 649–657.

[104] Nanne van Noord and Eric Postma. "Learning scale-variant and scale-invariant features for deep image classification." In: *Pattern Recognition* 61 (2017), pp. 583–592. URL: http://www.sciencedirect.com/science/article/pii/S0031320316301224.

[105] Juan Nunez-Iglesias, Ryan Kennedy, Stephen M Plaza, Anirban Chakraborty, and William T Katz. "Graph-based active learning of agglomeration (GALA): a Python library to segment 2D and 3D neuroimages." In: *Frontiers in neuroinformatics* 8 (2014), p. 34.

[106] Constantin Pape, Alex Matskevych, Adrian Wolny, Julian Hennies, Giulia Mizzon, Marion Louveaux, Jacob Musser, Alexis Maizel, Detlev Arendt, and Anna Kreshuk. "Leveraging domain knowledge to improve microscopy image segmentation with lifted multicuts." In: *Frontiers in Computer Science* 1 (2019), p. 6.

[107] Constantin Pape, Roman Remme, Adrian Wolny, Sylvia Olberg, Steffen Wolf, Lorenzo Cerrone, Mirko Cortese, Severina Klaus, Bojana Lucic, Stephanie Ullrich, et al. "Microscopy-based assay for semi-quantitative detection of SARS-CoV-2 specific antibodies in human sera: A semi-quantitative, high throughput, microscopy-based assay expands existing approaches to measure SARS-CoV-2 specific antibody levels in human sera." In: *BioEssays* 43.3 (2021), p. 2000257.

[108] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. "Pytorch: An imperative style, high-performance deep learning library." In: *arXiv preprint arXiv:1912.01703* (2019).

[109] Christian Payer, Darko Štern, Marlies Feiner, Horst Bischof, and Martin Urschler. “Segmenting and Tracking Cell Instances with Cosine Embeddings and Recurrent Hourglass Networks.” In: *Medical Image Analysis* 57 (Oct. 2019), pp. 106–119.

[110] Carolyn G. Rasmussen and Marschal Bellinger. “An overview of plant division-plane orientation.” en. In: *New Phytologist* 219.2 (July 2018), pp. 505–512. URL: http://doi.wiley.com/10.1111/nph.15183 (visited on 01/23/2019).

[111] Markus Rempfler, Jan-Hendrik Lange, Florian Jug, Corinna Blasse, Eugene W. Myers, Bjoern H. Menze, and Bjoern Andres. “Efficient Algorithms for Moral Lineage Tracing.” In: *CoRR* abs/1702.04111 (2017). arXiv: 1702.04111. URL: http://arxiv.org/abs/1702.04111.

[112] Mengye Ren and Richard S. Zemel. *End-to-End Instance Segmentation with Recurrent Attention*. 2017. arXiv: 1605.09410 [cs.LG].

[113] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation.” In: *Proc. MICCAI* (2015), pp. 234–241.

[114] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors.” In: *nature* 323.6088 (1986), pp. 533–536.

[115] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, et al. “Fiji: an open-source platform for biological-image analysis.” In: *Nature methods* 9.7 (2012), pp. 676–682.

[116] Uwe Schmidt, Martin Weigert, Coleman Broaddus, and Gene Myers. “Cell detection with star-convex polygons.” In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2018, pp. 265–273.

[117] Kay Schneitz, Martin Hülskamp, and Robert E. Pruitt. “Wild-type ovule development in Arabidopsis thaliana: a light microscope study of cleared whole-mount tissue.” In: *The Plant Journal* 7.5 (1995), pp. 731–749. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1046/j.1365-313X.1995.07050731.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1046/j.1365-313X.1995.07050731.x.

[118] Lilli Marie Schütz, Marion Louveaux, Amaya Vilches Barro, Sami Bouziri, Lorenzo Cerrone, Adrian Wolny, Anna Kreshuk, Fred A Hamprecht, and Alexis Maizel. “Integration of Cell Growth and Asymmetric Division during Lateral Root Initiation in Arabidopsis thaliana.” In: *Plant and Cell Physiology* 62.8 (Mar. 2021), pp. 1269–1279. eprint: https://academic.oup.com/pcp/article-pdf/62/8/1269/

41119202/pcab038.pdf. URL: https://doi.org/10.1093/pcp/pcab038.

[119] Zhao Shi et al. “A clinically applicable deep-learning model for detecting intracranial aneurysm in computed tomography angiography images.” In: *Nature Communications* 11.1 (Nov. 2020), p. 6090. URL: https://doi.org/10.1038/s41467-020-19527-w.

[120] Konstantin Sofiiuk, Olga Barinova, and Anton Konushin. “Adaptis: Adaptive instance selection network.” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7355–7363.

[121] Chunfeng Song, Yan Huang, Wanli Ouyang, and Liang Wang. “Box-driven class-wise region masking and filling rate guided loss for weakly supervised semantic segmentation.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3136–3145.

[122] Johannes Stegmaier, Fernando Amat, William C Lemon, Katie McDole, Yinan Wan, George Teodoro, Ralf Mikut, and Philipp J Keller. “Real-time three-dimensional cell segmentation in large-scale microscopy data of developing embryos.” In: *Developmental cell* 36.2 (2016), pp. 225–240.

[123] Carsen Stringer, Tim Wang, Michalis Michaelos, and Marius Pachitariu. “Cellpose: a generalist algorithm for cellular segmentation.” In: *Nature Methods* 18.1 (2021), pp. 100–106.

[124] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sébastien Ourselin, and M. Jorge Cardoso. “Generalised Dice overlap as a deep learning loss function for highly unbalanced segmentations.” In: *CoRR* abs/1707.03237 (2017). arXiv: 1707.03237. URL: http://arxiv.org/abs/1707.03237.

[125] Antti Tarvainen and Harri Valpola. “Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results.” In: *arXiv preprint arXiv:1703.01780* (2017).

[126] Rachele Tofanelli, Athul Vijayan, Sebastian Scholz, and Kay Schneitz. “Protocol for rapid clearing and staining of fixed Arabidopsis ovules for improved imaging by confocal laser scanning microscopy.” In: *Plant Methods* 15.1 (2019), p. 120. URL: https://doi.org/10.1186/s13007-019-0505-x.

[127] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. “Efficient Object Localization Using Convolutional Networks.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[128] Z. Tu and X. Bai. "Auto-Context and Its Application to High-Level Vision Tasks and 3D Brain Image Segmentation." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.10 (Oct. 2010), pp. 1744–1757.

[129] Srinivas C Turaga, Kevin L Briggman, Moritz Helmstaedter, Winfried Denk, and H Sebastian Seung. "Maximin affinity learning of image segmentation." In: *Proc. NIPS* (2009), pp. 1865–1873.

[130] Srinivas C Turaga, Joseph F Murray, Viren Jain, Fabian Roth, Moritz Helmstaedter, Kevin Briggman, Winfried Denk, and H Sebastian Seung. "Convolutional networks can learn to generate affinity graphs for image segmentation." In: *Neural computation* 22.2 (2010), pp. 511–538.

[131] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. "scikit-image: image processing in Python." In: *PeerJ* 2 (2014), e453.

[132] David A. Van Valen, Takamasa Kudo, Keara M. Lane, Derek N. Macklin, Nicolas T. Quach, Mialy M. DeFelice, Inbal Maayan, Yu Tanouchi, Euan A. Ashley, and Markus W. Covert. "Deep Learning Automates the Quantitative Analysis of Individual Cells in Live-Cell Imaging Experiments." In: *PLOS Computational Biology* 12.11 (Nov. 2016), pp. 1–24. URL: https://doi.org/10.1371/journal.pcbi.1005177.

[133] Paul Vernaza and Manmohan Chandraker. "Learning Random-Walk Label Propagation for Weakly-Supervised Semantic Segmentation." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017). URL: http://dx.doi.org/10.1109/CVPR.2017.315.

[134] Athul Vijayan, Rachele Tofanelli, Sören Strauss, Lorenzo Cerrone, Adrian Wolny, Joanna Strohmeier, Anna Kreshuk, Fred A Hamprecht, Richard S Smith, and Kay Schneitz. "A digital 3D reference atlas reveals cellular growth patterns shaping the *Arabidopsis* ovule." In: *eLife* 10 (Jan. 2021). Ed. by Sheila McCormick, Christian S Hardtke, Sheila McCormick, and Dolf Weijers, e63262. URL: https://doi.org/10.7554/eLife.63262.

[135] Amaya Vilches Barro et al. "Cytoskeleton Dynamics Are Necessary for Early Events of Lateral Root Initiation in Arabidopsis." In: *Current Biology* 29.15 (2019). 00000, 2443–2454.e5. URL: http://www.sciencedirect.com/science/article/pii/S0960982219307663 (visited on 10/10/2019).

[136] Weikang Wang, David A. Taft, Yi-Jiun Chen, Jingyu Zhang, Callen T. Wallace, Min Xu, Simon C. Watkins, and Jianhua Xing. “Learn to segment single cells with deep distance estimator and deep cell detector.” In: *Computers in Biology and Medicine* 108 (2019), pp. 133–141. URL: http://www.sciencedirect.com/science/article/pii/S0010482519301143.

[137] D. Wei et al. “MitoEM Dataset: Large-scale 3D Mitochondria Instance Segmentation from EM Images.” In: *International Conference on Medical Image Computing and Computer Assisted Intervention*. 2020.

[138] Tamily A. Weissman and Y. Albert Pan. “Brainbow: New Resources and Emerging Biological Applications for Multicolor Genetic Labeling and Analysis.” en. In: *Genetics* 199.2 (Feb. 2015). 00056, pp. 293–306. URL: https://www.genetics.org/content/199/2/293 (visited on 01/13/2020).

[139] Lisa Willis, Yassin Refahi, Raymond Wightman, Benoit Landrein, José Teles, Kerwyn Casey Huang, Elliot M. Meyerowitz, and Henrik Jönsson. “Cell size and growth regulation in the Arabidopsis thaliana apical stem cell niche.” In: *Proceedings of the National Academy of Sciences* 113.51 (2016), E8238–E8246. eprint: https://www.pnas.org/content/113/51/E8238.full.pdf. URL: https://www.pnas.org/content/113/51/E8238.

[140] Steffen Wolf, Constantin Pape, Alberto Bailoni, Nasim Rahaman, Anna Kreshuk, Ullrich Köthe, and Fred Hamprecht. “The Mutex Watershed: Efficient, Parameter-Free Image Partitioning.” In: *Proc. ECCV’18* (2018).

[141] Adrian Wolny, Qin Yu, Constantin Pape, and Anna Kreshuk. “Sparse Object-Level Supervision for Instance Segmentation With Pixel Embeddings.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 4402–4411.

[142] Adrian Wolny et al. “Accurate and versatile 3D segmentation of plant tissues at cellular resolution.” In: *eLife* 9 (July 2020). Ed. by Christian S Hardtke, Dominique C Bergmann, Dominique C Bergmann, and Moritz Graeff, e57613. URL: https://doi.org/10.7554/eLife.57613.

[143] Yuxin Wu and Kaiming He. “Group Normalization.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[144] Christopher Xie, Yu Xiang, Arsalan Mousavian, and Dieter Fox. “Unseen object instance segmentation for robotic environments.” In: *IEEE Transactions on Robotics* 37.5 (2021), pp. 1343–1359.

[145] Saining Xie and Zhuowen Tu. “Holistically-Nested Edge Detection.” In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1395–1403.

[146] C. Shan Xu et al. “An open-access volume electron microscopy atlas of whole cells and tissues.” In: *Nature* 599.7883 (Nov. 2021), pp. 147–151. URL: https://doi.org/10.1038/s41586-021-03992-4.

[147] Julian Yarkony, Alexander Ihler, and Charless C Fowlkes. “Fast planar correlation clustering for image segmentation.” In: *Proc. ECCV'12*. 2012, pp. 568–581.

[148] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. “Barlow Twins: Self-Supervised Learning via Redundancy Reduction.” In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 12310–12320. URL: http://proceedings.mlr.press/v139/zbontar21a.html.

[149] Ziyu Zhang, Sanja Fidler, and Raquel Urtasun. “Instance-Level Segmentation for Autonomous Driving With Deep Densely Connected MRFs.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[150] Yi Zhu, Karan Sapra, Fitsum A. Reda, Kevin J. Shih, Shawn Newsam, Andrew Tao, and Bryan Catanzaro. *Improving Semantic Segmentation via Video Propagation and Label Relaxation*. 2018. URL: https://arxiv.org/abs/1812.01593.