DISSERTATION
submitted to the
Combined Faculty of Natural Sciences and Mathematics
of Heidelberg University, Germany
for the degree of
Doctor of Natural Sciences

Put forward by

Steffen Wolf
Born in Munich, Germany

Oral examination: May 13, 2020

Machine Learning for Instance Segmentation

Referees: Prof. Dr. rer. nat. Fred A. Hamprecht
Prof. Dr. rer. nat. Carsten Rother

Abstract

Volumetric Electron Microscopy images can be used for connectomics, the study of brain connectivity at the cellular level. A prerequisite for this inquiry is the automatic identification of neural cells, which requires machine learning algorithms and in particular efficient image segmentation algorithms.

In this thesis, we develop new algorithms for this task. In the first part we provide, for the first time in this field, a method for training a neural network to predict optimal input data for a watershed algorithm. We demonstrate its superior performance compared to other segmentation methods of its category.

In the second part, we develop an efficient watershed-based algorithm for weighted graph partitioning, the *Mutex Watershed*, which uses negative edge-weights for the first time. We show that it is intimately related to the multicut and has a cutting edge performance on a connectomics challenge. Our algorithm is currently used by the leaders of two connectomics challenges [55, 90].

Finally, motivated by inpainting neural networks, we create a method to learn the graph weights without any supervision.

Zusammenfassung

3D-Elektronenmikroskopbilder können für die Konnektomik, dem Studium der Neuronverbindungen im Nervensystem, genutzt werden. Als Vorbereitung für diese Fragestellung nutzt man die automatische Identifizierung von Neuronen durch effiziente Bildsegmentierungsalgorithmen aus dem maschinellen Lernen.

In dieser Arbeit entwickeln wir neue Algorithmen für diese Aufgabe. Im ersten Teil stellen wir zum ersten Mal in diesem Forschungsgebiet eine Prozedur vor, die ein neuronales Netzwerk so trainiert, dass optimale Eingabedaten für einen Watershed Algorithmus bereitgestellt werden. Wir zeigen, dass diese Methode im Vergleich zu anderen Segmentierungsverfahren derselben Art bessere Resultate liefert.

Im zweiten Teil, entwickeln wir einen effizienten Algorithmus (den *Mutex Watershed*) für das Partitionieren eines gewichteten Graphen. Dieser erlaubt zum ersten Mal die Verwendung negativer Gewichte. Wir zeigen, dass der *Mutex Watershed* eng mit dem *multicut* verwandt ist und zum Zeitpunkt der Veröffentlichung eine Spitzenposition in einer Konnektomik-Challenge eingenommen hat. Unser Algoithmus is momentan Teil von zwei führenden Segmentierungsmethoden [55, 90].

Zum Abschluss stellen wir eine Methode unter Nutzung von bildvervollständigenden neuronalen Netzwerken bereit, welche den gewichteten Graph ganz ohne Überwachung lernt.

Acknowledgments

First of all, I would like to thank Professor Fred A. Hamprecht, for sparking my passion for Machine Learning through his lectures and accompanying projects. As my superviser, he provided me with the resources and support to develop and explore my ideas and develop myself as a scientist. I enjoyed our intense and productive discussions, where he always challenged me to dig deeper. I am incredibly grateful for the pleasant environment that he procures, full of incredibly friendly, supportive, and intelligent people that made collaborating a pure joy. I especially want to thank Constantin Pape and Alberto Bailoni, who played an indispensable role in conceiving and understanding the Mutex Watershed. In this spirit, I also want to thank Lukas Schott and Yuyan Li for working so closely together with me and push the Watershed algorithms to their limits. I would also like to thank the senior group members Ullrich Köthe, Anna Kreshuk, Carsten Hauboldt, and Martin Schiegg for the guidance and helping me to get up on my science feet. In particular, I like to thank Ullrich Köthe for our in-depth discussions about image segmentation algorithms and his late-night support during my first paper.

I would also like to thank Jan Funke, who invited me to visit his group in Janelia, explored new ideas with me, and encouraged me to think about the bigger picture. I fondly remember the friendly atmosphere and intelligent discussions in his group that immediately made me feel at home. I also want to thank my close friend Stefan Richter who was always reliable in planing our eventing, but most importantly, as a supporter during stressful times. A special thanks goes to Barbara Werner for fighting our bureaucratic battles. I would also like to thank my official supervisor Carsten Rother for his interest in my project and valuable comments during my presentations.

Finally, I would like to thank Luka Smalinskaite for her love, support, and simply making my life better. Special thanks, of course, also goes to my parents for their continuous, unconditional support throughout my life.

Contents

1 Introduction

Understanding the content of digital images is one of the fundamental tasks of Computer vision. Typically, "understanding" means to find a transformation which maps images to a condensed description appropriate for further analysis. One example is the task of image classification, where the image has to be assigned to a specific element of a given set of classes. If a set of images and their corresponding class assignments are available, one can learn this transformation, thus extrapolating the assignment to new images. This learning-based image analysis, a part of Machine Learning, has become a common approach over a wide range of tasks. Notably, (convolutional) neural networks, as a way to parametrize these transformations, show remarkable performance. In some applications, they are on par with medical experts in skin cancer diagnosis [44] or even surpass humans on challenges such as ImageNet[134] where more than 14 millions of images have to be assigned to thousand different classes.

A deeper understanding of the content of an image can be expressed by finding objects in an image. This can be done in different ways. In the following we will focus on image segmentation, which groups the image pixels into meaningful regions. While semantic segmentation is the assignment of each pixel to a discrete set of labels (*e.g.* {road, sky, tree, car,...}), instance segmentation groups pixel together and allows to distinguish between instances of the same class (*e.g.*, assigning each car in the image to its own cluster). One notable example that we will discuss later is the instance segmentation of neuronal tissue images. Here, every pixel belongs to a neuron cell (*i.e.*, we only have one class), which makes semantic segmentation not applicable. Instead, the image pixels have to be grouped such that each group contains only pixels of one neuron. In this work we introduce new algorithms for instance segmentation specifically tailored for neuron segmentation. The combination of both tasks, where pixels are to be grouped, and each group is assigned to a label is known as a semantic instance segmentation which we will address in Chapter 4.

1.1 Machine Learning for Segmentation

Neural networks have demonstrated an incredible performance in many fields. This success was made possible by the development of specialized network architectures. For image analysis tasks, neural networks commonly are arranged in layers (referred to as multilayer perceptions) where the output of each neuron only depends on the neurons of the previous layer. The structure of these connections determines the architecture of the neural network. The most important architecture class for segmentation tasks are convolutional neural networks (CNNs), which arrange their neurons spatially and only form connections in their local neighborhoods. This drastically reduces the number of parameters in the network and is also desirable since these layers can be efficiently implemented using convolutional kernels.

To solve the task of semantic segmentation, a specialized CNN, the fully convolutional network (FCN) was designed, that laid the groundwork for further modern methods [99]. The FCN maps each pixel to a class by analyzing a patch centered on this pixel. A full segmentation can then be efficiently generated in a sliding-window procedure. This architecture was improved by using an encoder-decoder structure, called U-Net [132], and feature pyramids FPN [95] which have recently been extended to semantic instance segmentation [71]. Especially the U-Net has been particularly successful in biological applications [132] and is used repeatedly throughout this thesis.

In this thesis we will approach the task of instance segmentation by learning to predict the input weights for a graph-based segmentation algorithm, which we will review in Section 1.2. An alternative approach is given by detection-based instance segmentation, which can be divided into two steps. First, bounding boxes are detected that define the instances and then pixels inside each bounding box are assigned to the instance if they are within a predicted mask [54]. However these bounding boxes may overlap and thus produce overlapping segments which is not always desirable [72].

We will also investigate methods for dense instance segmentation where all pixels have to be assigned to exactly one cluster. In particular we will be present graph-based segmentation algorithms which in particular dominate the field of connectomics. One key aspect of their success was the use of Machine Learning to predict meaningful graph weights as an input for sophisticated graph partitioning algorithms. Very accurate graph weights can be learned by training an edge classifier that predicts the transitions between objects [89] or the use of a structured loss function that directly optimizes the segmentation performance [151]. In the following section we will briefly discuss a selection of graph partitioning algorithms that are commonly used in combination with learned graph weight estimators.

Kruskal's Algorithm:

KA $\big(\mathcal{G}(V, E)$, **weights** $w : E \to \mathbb{R}^+$**:**

```
A ← ∅
for (i, j) = e ∈ E in ascending order of w_e do
    if not connected(i, j; A) then
        A ← A ∪ e
return A
```

Algorithm 1: Kruskal's Algorithm for constructing a minimal spanning tree A on the weighted graph $\mathcal{G}(V, E)$.

1.2 Graph-based Segmentation Algorithms

Here we explain the basic ideas and notation of graph-based segmentation algorithms, which will be frequently used in this thesis. In general, graph-based image segmentation methods represent the image as a graph $G = (V, E)$ where each node $u \in V$ corresponds to a pixel in the image. Nodes are connected by edges $(u, v) \in E$. A weight w_e is associated with each edge $e \in E$ based on some property of the pixels that it connects, such as their image intensities, gradients or the output of an edge classifier (*e.g.* obtained by a neural network). Depending on the method and application, the graph might be only sparsely connected, for example as a grid graph or a graph with limited local neighborhood connectivity. In this thesis we will build upon segmentation algorithms, including the watershed, that are closely related to minimum spanning trees (MST) [47, 177] on this graph and their construction algorithms. One example is Kruskal's algorithm that constructs a MST by considering all edges in order of their weight, adding them to the tree if the incident nodes are not already connected. Here, we introduce the notation that connected$(i, j; A)$ is true if there exists a path π from i to j which completely lies in a A.

A segmentation can be derived from the tree by breaking the tree at the edges with large weights [177] or using sets of seed nodes that may not be connected which is known as seeded watershed [38, 39, 107, 157]. This watershed segmentation can also be understood as a minimal energy solution to an energy minimization problem. Its objective function is a special case of the unifying power watershed energy minimization framework [37] which also encompasses the Random walker and Graph cuts [77].

These algorithms, however, can only ingest positive (attractive) weights and thus need an auxiliary input (*e.g.*, seed points and thresholds) to partition the graph into clusters. In contrast to the above class of algorithms, this thesis will provide watershed algorithms which can also

deal with repulsive weights. Other methods like Multi-label variants [75] and QPBO [133] and correlation clustering [14, 67, 70, 172, 173] can also use both attractive and repulsive interactions. However, only our new algorithm presented in Chapter 3 and correlation clustering (also known as multicut) can find the number of clusters implicitly and are therefore particularly suited for applications where the number of clusters is a-priori unknown. Since our algorithm and the multicut both minimize a similar objective function (see Section 3.4) we present a formal definition of the minimum multicut as

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

$$\text{subject to} \quad y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'} \quad \forall C \in \text{ cycles } (G) \quad \forall e \in C \tag{1.2}$$

When the image is to be partitioned into semantically similar objects the graph can be augmented with additional semantic nodes and edges [63]. If one wants to no internal boundaries inside a semantic class this problem can be modeled as a Multiway cut:

$$y^* = \arg\min_{y \in \{0,1\}^E} \sum_{e \in E} w_e y_e \tag{1.3}$$

$$\text{subject to} \quad y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'} \qquad \forall C \in \text{ cycles } (G) \quad \forall e \in C \tag{1.4}$$

$$\sum_{t \in T} y_{tv} = |T| - 1, \qquad \text{if } T \neq \emptyset, \forall v \in V \setminus T \tag{1.5}$$

$$y_{tt'} = 1, \qquad \forall t, t' \in T, t \neq t' c, f \tag{1.6}$$

$$y_{tu} + y_{tv} \geq y_{uv}, \qquad \forall uv \in E, t \in T \setminus \mathcal{A} \tag{1.7}$$

$$y_{tu} + y_{uv} \geq y_{tv}, \qquad \forall uv \in E, t \in T \tag{1.8}$$

$$y_{tv} + y_{uv} \geq y_{tu}, \qquad \forall uv \in E, t \in T \tag{1.9}$$

In case internal boundaries are desired, the constraints of eq. (1.7) can be removed [81]. Solving both the multicut and Multiway cut is in general NP-hard since the set of constraints in eq. (1.2) and eq. (1.4) are of exponential size. Therefore any exact solver will fail to scaling to large graphs [18]. In this work we will investigate a novel set of watershed algorithms that intimately relate to the multicut and Asymmetric Multiway Cut objective but can be efficiently solved on large graphs.

1.2.1 Segmentation for Connectomics

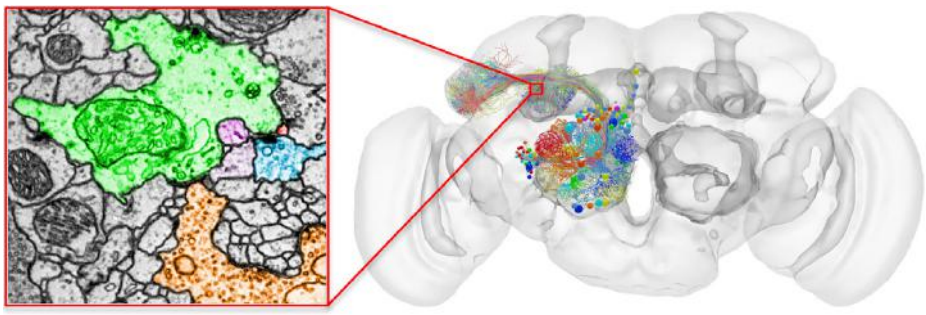


Figure 1.1: llustration of neuron reconstruction in the *Drosophila melanogaster* brain. The neurons are reconstructed from serial section transmission EM (TEM) volumes (represented on the left) of the fruit fly brain (ventral view on the right). Individual neuron instances are shown, represented by different colors, on the EM slice with corresponding colors in the brain volume. Image taken from [183].

Graph-based segmentation algorithms have been particularly successful in connectomics, a field of neuroscience that strives to reconstruct the complete central nervous systems of animals and studies their neural wiring diagram. A necessary step towards this goal is the segmentation of neural tissue delineating individual neuron cells and revealing their 3D shapes. This process is known as neuron reconstruction. The neural tissue is commonly imaged using electron microscopy techniques (e.g., serial section transmission EM) that yield 3D image volumes. For example, the brain of an adult *Drosophila melanogaster*, with a volume of $\sim 8 \cdot 10^7 \mu m^3$ and comprising $\sim$100,000 neurons has been imaged with nanometer resolution producing a dataset of 106 TB [183]. To study data-sets of this size, automated processing, especially automated segmentation, is paramount to not only reconstruct the complete neural wiring diagram but also study neuron morphology and ultra-structure.

One automatic method for neuron tracing, the flood-filling networks, uses a recurrent neural network to iteratively extend individual neurons [60]. Other approaches learn to predict affinity graph between voxels or supervoxels [89, 152] and determine the segmentation as optimal cuts of this graph [6, 7, 20, 49, 105, 119, 120]. Almost all of the top submissions of the CREMI [26] and SNEMI3D [135] segmentation challenge predict these affinities directly using convolutional networks [89, 163]. An alternative approach to the direct prediction of affinities was proposed by Lee et al. [90], who instead learn dense voxel embeddings via deep metric learning and derive affinities in the embedded space.

1.3 Contribution and Overview of this Thesis

The core of this thesis is a novel set of watershed algorithms, whose centerpiece, the *Mutex Watershed*, is presented in Chapter 3 and extended to semantic instance segmentation in Chapter 4. This set extends the classical family of watershed algorithms operating on purely attractive graphs by including repulsive interactions and effectively obviating the need for explicit seeds. We investigate machine learning approaches for predicting input weights for classical and Mutex Watersheds, focusing especially on supervised end-to-end learning in Chapter 2 and fully unsupervised learning in Chapter 5. The following list gives a brief overview of each chapter's content.

Chapter 2: We show how to train a the boundary map prediction jointly with the watershed computation. The estimator for the merging priorities is cast as a neural network that is convolutional (over space) and recurrent (over iterations). The latter allows the learning of complex shape priors and outperforms other seeded segmentation methods on the CREMI segmentation challenge.

Chapter 3: We propose a greedy algorithm for signed graph partitioning, the *Mutex Watershed*. Unlike seeded watershed, the algorithm can accommodate not only attractive but also repulsive interactions, allowing it to find a previously unspecified number of segments without the need for explicit seeds or a tunable threshold. We also prove that this simple algorithm finds a global optimum of an objective function that is intimately related to the multicut / correlation clustering integer linear programming formulation.

Chapter 4: The link between Mutex Watershed and correlation clustering suggests that a similar Watershed algorithm for joint graph partitioning and labeling exists whose objective function closely relates to the Asymmetric Multiway Cut objective. We prove its existence by extending the Mutex Watershed and demonstrate on 3D electron microscopy images that this joint formulation outperforms a procedure which separately optimizes of the partitioning and labeling problems.

Chapter 5: Deep neural networks trained to inpaint partially occluded images show a deep understanding of image composition. We investigate how this implicit knowledge of image composition can be leveraged for a fully self-supervised generation of Mutex Watershed inputs and thus self-supervised segmentation. We evaluate our method on two microscopy image datasets to show that it reaches comparable segmentation performance to supervised methods.

2 Learned Watershed

Common pipelines for segmentation and super-pixel generation consist of a learned boundary predictor and an inference step (e.g. seeded watershed). The following work was motivated by the observation that small holes in the boundary map estimation can drastically reduce the segmentation accuracy of the seeded watershed. This effect may be most prevalent when the boundary estimation (often a neuronal network) is trained with an unstructured loss that penalizes every pixel individually. Although heuristics (*e.g.* using the distance transform transform [20]) may mitigate this problem, it can ultimately only be addressed through structured loss functions that take the inference method into account. In this chapter[1], we present our approach to incorporate a neural network into the seeded watershed segmentation algorithm and train it *end-to-end*. Furthermore, this integration enables *adaptive* boundary prediction where predictions in the current iterations can be based on past decisions. Through a lesion study we show that *adaptive* prediction outperforms static baselines.

2.1 Introduction

The watershed algorithm is an important computational primitive in low-level computer vision. Since it does not penalize segment boundary length, it exhibits no shrinkage bias like multi-terminal cuts or (conditional) random fields and is especially suited to segment objects with high surface-to-volume ratio, e.g. neurons in biological images.

In its classic form, the watershed algorithm comprises three basic steps: altitude computation, seed definition, and region assignment. These steps are designed manually for each application of interest. In a typical setup, the altitude is the output of an edge detector (e.g. the gradient magnitude or the gPb detector [10]), the seeds are located at the local minima of the altitude image, and pixels are assigned to seeds according to the drop-of-water principle [39].

In light of the very successful trend towards learning-based image analysis, it is desirable to eliminate hand-crafted heuristics from the watershed algorithm as well. Existing work

[1] This chapter is based on our paper [162], which was published in 2017. The results in this chapter represent the current state at the time of publication. The implementation and training of the deep neural network for the experiments of this chapter have been carried out by Lukas Schott and myself with many enjoyable days of pair programming.

shows that learned edge detectors significantly improve segmentation quality, especially when convolutional neural networks (CNNs) are used [16, 34, 130, 169]. We take this idea one step further and propose to learn altitude estimation and region assignment *jointly*, in an end-to-end fashion: Our approach no longer employs an auxiliary objective (e.g. accurate boundary strength prediction), but trains the altitude function together with the subsequent region assignment decisions so that the final segmentation error is minimized directly. The resulting training algorithm is closely related to reinforcement learning.

Our method keeps the basic sructure of the watershed algorithm intact: Starting from given seeds[2], we maintain a priority queue storing the topographic distance of candidate pixels to their nearest seed. Each iteration assigns the currently best candidate to "its" region and updates the queue. The topographic distance is induced by an altitude function estimated with a CNN. Crucially, and deviating from prior work, we compute altitudes *on demand*, allowing their conditioning on prior decisions, i.e. partial segmentations. The CNN thus gets the opportunity to learn priors for likely region shapes in the present data. We show how these models can be trained end-to-end from given ground truth segmentations using *structured learning*. Our experiments show that the resulting segmentations are better than those from hand-crafted algorithms or unstructured learning.

2.2 Related Work

Various authors demonstrated that learned boundary probabilities (or, more generally, boundary strengths) are superior to designed ones. In the most common setting, these probabilities are defined on the pixel grid, i.e. on the nodes of a grid graph, and serve as input of a *node-based* watershed algorithm. Training minimizes a suitable loss (e.g. squared or cross-entropy loss) between the predicted probabilities and manually generated ground truth boundary maps in an *unstructured* manner, i.e. over all pixels independently. This approach works especially well with powerful models like CNNs. In the important application of connectomis (see section 2.6.3), this was first demonstrated by [59]. A much deeper network [34] was the winning entry of the ISBI 2012 Neuro-Segmentaion Challenge [12]. Results could be improved further by progress in CNN architectures and more sophisticated data augmentation, using e.g. U-Nets [130], FusionNets [126] or networks based on inception modules [20]. Clustering of the resulting watershed superpixels by means of the GALA algorithm [73, 117] (using altitudes from [12] resp. [130]) or the lifted multicut [20] (using altitudes from their own CNN) lead to additional performance gains.

When ground truth is provided in terms of region labels rather than boundary maps, a suitable

[2] Incorporating seed definition into end-to-end learning is a future goal of our research, but beyond the scope of this paper.

boundary map must be created first. Simple morphological operations were found sufficient in [130], while [20] preferred smooth probabilities derived from a distance transform starting at the true boundaries. Outside connectomics, [16] achieved superior results by defining the ground truth altitude map in terms of the *vector* distance transform, which allows optimizing the prediction's gradient direction and height separately.

Alternatively, one can employ the *edge-based* watershed algorithm and learn boundary probabilities for the grid graph's edges. The corresponding ground truth simply indicates if the end points of each edge are supposed to be in different segments or not. From a theoretical perspective, the distinction between node- and edge-based watersheds is not very significant because both can be transformed into each other [109]. However, the algorithmic details differ considerably. Edge-based altitude learning was first proposed in [48], who used hand-crafted features and logistic regression. Subsequently, [152] employed a CNN to learn features and boundary probabilities simultaneously. Watershed superpixel generation and clustering on the basis of these altitudes was investigated in [185].

Learning with unstructured loss functions has the disadvantage that an error at a single point (node or edge) has little effect on the loss, but may lead to large segmentation errors: A single missed boundary pixel can cause a big false merger. Learning with *structured* loss functions, as advocated in this paper, avoids this by considering the boundaries in each image jointly, so that the loss can be defined in terms of segmentation accuracy rather than pointwise differences. Holistically-nested edge detection [76, 169] achieves a weak form of this by coupling the loss at multiple resolutions using deep supervision. Such a network was successfully used as a basis for watershed segmentation in [27]. The MALIS algorithm [151] computes shortest paths between pairs of nodes and applies a correction to the highest edge along paths affected by false splits or mergers. This is similar to our training, but we apply corrections to root error edges as defined below. Learned, sparse reconstruction methods such as MaskExtend [104] and Flood-filling networks [60] predict region membership for all nodes in a patch jointly, performing region growing for one seed at a time in a one-against-the-rest fashion. In contrast, our algorithm grows all seeds simultaneously and competitively.

2.3 Mathematical Framework

The watershed algorithm is especially suitable when regions are primarily defined by their boundaries, not by appearance differences. This is often the case when the goal is *instance* segmentation (one neuron vs. its neighbors) as opposed to semantic segmentation (neurons vs. blood vessels). In graphical model terms, pairwise potentials between adjacent nodes are crucial in this situation, whereas unary potentials are of lesser importance or missing altogether. Many real-world applications have these characteristics, see [37] and section 2.6 for examples.

We consider 4-connected grid graphs $G = (V, E)$. The input image $I : V \to \mathbb{R}^D$ maps all nodes to D-dimensional vectors of raw data. A segmentation is defined by a label image $S : V \to \{1, 2, \ldots, n\}$ specifying the region index or label of each node. The ground truth segmentation is called S^*. Pairwise potentials (i.e. edge weights) are defined by an altitude function over the graph's edges

$$f : E \to \mathbb{R} \tag{2.1}$$

where higher values indicate stronger boundary evidence. Since this paper focuses on how to learn f, we assume that a set of seed nodes $M = \{m_1, \ldots, m_n\} \subset V$ is provided by a suitable oracle (see section 2.6 for details). The watershed algorithm determines S by finding a mapping $\sigma : V \to M$ that assigns each node to the best seed so that

$$\sigma(w) = m_i \quad \Rightarrow \quad S(w) = i \tag{2.2}$$

Initially, node assignments are unknown (designated by λ) except at the seeds, where they are assumed to be correct:

$$\sigma_0(w) = \begin{cases} m_i & \text{if } w = m_i \text{ with } S^*(m_i) = i \\ \lambda & \text{otherwise} \end{cases} \tag{2.3}$$

In this paper, we build upon the *edge-based* variant of the watershed algorithm [39, 106]. This variant is also known as *watershed cuts* because segment boundaries are defined by cuts in the graph, i.e. by the set of edges whose incident nodes have different labels. We denote the cuts in our solution as ∂S and in the ground truth as ∂S^*.

Let $\Phi(m, w)$ denote the set of all paths from seed m to node w. Then the *max-arc topographic distance* between m and w is defined as [45]

$$T(m, w) = \min_{\phi \in \Phi(m,w)} \max_{e \in \phi} f(e) \tag{2.4}$$

In words, the highest edge in a path ϕ determines the path's altitude, and the path of lowest altitude determines the topographic distance. The watershed algorithm assigns each node to the topographically closest seed [129]:

$$\sigma(w) = \arg\min_{m \in M} \; T(m, w) \tag{2.5}$$

The minimum distance path from seed m to node w shall be denoted by $\phi_m(w)$. This path is not necessarily unique, but ties are rare and can be broken arbitrarily when $f(e)$ is a real-valued function of noisy input data.

It was shown in [39] that the resulting partitioning is equivalent to the minimum spanning forest (MSF) over seeds M and edge weights $f(e)$. Thus, we can compute the watershed segmentation incrementally using Prim's algorithm: Starting from initial seeds σ_0, each iteration k finds the lowest edge whose start point u_k is already assigned, but whose end point v_k is not

$$u_k, v_k = \underset{\substack{(u,v)\in E \\ \sigma_{k-1}(u)\neq\lambda,\ \sigma_{k-1}(v)=\lambda}}{\arg\min} f(e=(u,v)) \tag{2.6}$$

and propagates the seed assignment from u_k to v_k:

$$\sigma_k(w) = \begin{cases} \sigma_{k-1}(u_k) & \text{if } w = v_k \\ \sigma_{k-1}(w) & \text{otherwise} \end{cases} \tag{2.7}$$

In a traditional watershed implementation, the altitude $f(e)$ is a fixed, hand-designed function of the input data

$$f(e) = f_{\text{fixed}}(e|I) \tag{2.8}$$

for example, the image's Gaussian gradient magnitude or the "global Probability of boundary" (gPb) detector [10].

2.4 Joint Structured Learning of Altitude and Region Assignment

We propose to use structured learning to train an altitude regressor $f(e)$ *jointly* with the region assignment procedure defined by Prim's algorithm. We will discuss two types of learnable altitude functions: f_{static} comprises models that, once trained, only depend on the input image I, whereas f_{dyn} additionally incorporates dynamically changing information about the current state of Prim's algorithm.

2.4.1 Static Altitude Prediction

To find optimal parameters θ of a model $f_{\text{static}}(e|I;\theta)$, consider how Prim's algorithm proceeds: It builds a MSF which assigns each node w to the closest seed $\hat{m} = \sigma(w)$ by identifying the shortest path $\phi_{\hat{m}}(w)$ from $\hat{m}$ to w. Such a path can be wrong in two ways: it may cross ∂S^* and thus miss a ground truth cut edge, or it may end at a false cut edge, placing ∂S in the interior of a ground truth region. More formally, we have to distinguish two failure modes: (i)

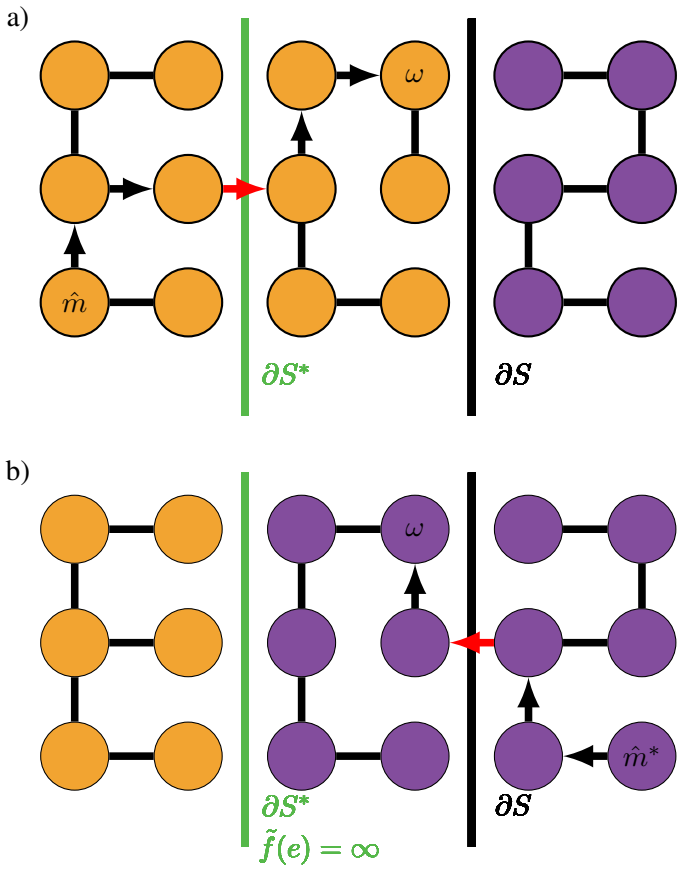


Figure 2.1: Example of root errors in the minimal spanning forest (a) and the constrained MSF (b) of a grid graph. Orange and purple indicate the segmentation S in (a) and S^* in (b). The root errors $\rho(w)$ (top) and $\rho^*(w)$ (bottom) of a wrongly labeled node w are marked red, with corresponding paths $\phi_{\hat{m}}(w)$ and $\psi_{m^*}(w)$ depicted by arrows.

A node was assigned to the wrong seed, i.e. $\hat{m} \neq m^* = m_{S^*(w)}$ or (ii) it was assigned to the correct seed via a non-admissible path, i.e. a path taking a detour across a different region. To treat both cases uniformly, we construct the corresponding ground truth paths $\psi_{m^*}(w)$.

These paths can be found by running Prim's algorithm with a modified altitude

$$\tilde{f}(e) = \begin{cases} \infty & \text{if } e \in \partial S^* \\ f_{\text{static}}(e|I;\theta) & \text{otherwise} \end{cases} \tag{2.9}$$

forcing cuts in the resulting constrained MSF to coincide with ∂S^* (see figure 2.1). We denote the topographic distances along $\phi_{\hat{m}}(w)$ and $\psi_{m^*}(w)$ as $T(\hat{m}, w)$ and $T^*(m^*, w)$ respectively. By construction of the MSF, ϕ and ψ are equal for all correct nodes. Conversely, they differ for incorrect nodes, causing distance T^* to exceed distance T. This property defines the set V_- of incorrect nodes:

$$V_- = \{w : T^*(m^*, w) > T(\hat{m}, w)\} \tag{2.10}$$

Every incorrect path $\phi_{\hat{m}}(w)$ contains at least one erroneous cut edge. The first such edge shall be called the path's *root* error edge $\rho(w)$ and is always a missing cut. Training should increase its altitude until it becomes part of the cut set ∂S. The root error edge $\rho^*(w)$ of a ground truth path $\psi_{m^*}(w)$ is the first false cut edge in ψ in failure mode (i) and the first edge where ψ deviates from ϕ in mode (ii). Here, the altitude should be decreased to make the edge part of the MSF, see figure 2.1. Accordingly, we denote the sets of root edges as $E_\uparrow := \{\rho(w) : w \in V_-\}$ and $E_\downarrow := \{\rho^*(w) : w \in V_-\}$.

Since all assignment decisions in Prim's algorithm are conditioned on decisions taken earlier, the errors in any path also depend on the path's root error. Structured learning must therefore consider these errors jointly, and we argue that training updates must be derived solely from the root edges: They are the only locations whose required correction direction is unambiguously known. In contrast, we cannot even tell if subsequent errors will simply disappear once the root error has been fixed, or need updates of their own. When the latter applies, however, these edges will eventually become root errors in later training epochs, and we delay updating them to that point.

Since we need a differentiable loss to perform gradient descent, we use the perceptron loss of distance differences:

$$\mathcal{L} = \sum_w T^*(m^*, w) - T(\hat{m}, w) \tag{2.11}$$

Correct nodes have zero contribution since $T^* = T$ holds for them. To serve as a basis for structured learning, we transform this into a loss over altitude differences at root edges. Since topographic distances equal the highest altitude along the shortest path, we have

$$T(\hat{m}, w) \geq f_{\text{static}}(\rho(w)) \tag{2.12}$$

To derive similar relations for T^*, consider how the constrained MSF is constructed from the unconstrained one: First, edges crossing ∂S^* are removed from the MSF. Each of the resulting orphaned subgraphs is then reconnected into the constrained MSF via the lowest edge not crossing ∂S^*. The newly inserted edges are the root edges ρ^* of all their child nodes, i.e. all nodes in the respective subgraph. Since these root edges did not belong to the original MSF,

their altitude cannot be less than the maximum altitude in the corresponding child subgraph. For $w \in V_-$, it follows that

$$T^*(m^*, w) = f_{\text{static}}(\rho^*(w)) \tag{2.13}$$

We can therefore upper-bound the perceptron loss by

$$\mathcal{L}_{\text{SL}} = \sum_{w \in V_-} f_{\text{static}}(\rho^*(w)) - f_{\text{static}}(\rho(w)) \geq \mathcal{L} \tag{2.14}$$

and minimize this upper bound. By rearranging the sum, the loss can be simplified into

$$\mathcal{L}_{\text{SL}}(\theta) = \sum_{e \in E} R(e) f_{\text{static}}(e|I;\theta) \tag{2.15}$$

where we introduced a weight function counting the children of each root edge

$$R(e) := \begin{cases} \sum_{w:\, e=\rho^*(w)} 1 & \text{if } e \in E_\downarrow \\ -\sum_{w:\, e=\rho(w)} 1 & \text{if } e \in E_\uparrow \\ 0 & \text{otherwise} \end{cases} \tag{2.16}$$

A training epoch of structured learning thus consists of the following steps:

1. Compute $f_{\text{static}}(e|I;\theta^{(t)})$ and $\tilde{f}^{(t)}(e)$ with current model parameters $\theta^{(t)}$ and determine the MSF and the constrained MSF.
2. Identify root edges and define the weights $R^{(t)}(e)$ and the loss $\mathcal{L}_{\text{SL}}^{(t)}(\theta)$.
3. Obtain an updated parameter vector $\theta^{(t+1)}$ via gradient descent on $\nabla_\theta \mathcal{L}^{(t)}(\theta)$ at $\theta = \theta^{(t)}$.

These steps are iterated until convergence, and the resulting final parameter vector is denoted as θ_{SL}.

2.4.2 Relation to Reinforcement Learning

In this section we compare the structured loss function $\mathcal{L}_{\text{SL}}$ with policy gradient reinforcement learning, which will serve as motivation for a refinement of the weighting function $R(e)$. To see the analogy, we refer to continuous control deep reinforcement learning as proposed by [94, 142, 147].

Looking at the region growing procedure from a reinforcement learning perspective, we define states as tuples $s = (e, I)$ where $e \in E$ is the edge under consideration, and the action

space $\mathcal{A} := \mathbb{R}$ is the altitude to be predicted by $f_{\text{static}}(e|I;\theta)$. The *Policy Gradient Theorem* [147] defines the appropriate update direction of the parameter vector θ. In a continuous action space, it reads

$$\nabla_\theta J = \nabla_\theta \sum_s d^\pi(s) \int_{\mathcal{A}} \pi(a|s;\theta)\, Q^\pi(s,a)\ da \tag{2.17}$$

where J is the performance to be optimized, $d^\pi(s)$ the discounted state distribution, π the policy to be learned, and Q the action-value function estimating the discounted expected future reward

$$Q^\pi(s,a) = \mathbb{E}_\pi\Big[\sum_{t=0}^{T} \gamma^t r^t \Big| a_0 = a, s_0 = s; \pi\Big] \tag{2.18}$$

In our case, the state distribution reduces to $d^\pi(s) = \frac{1}{|V|}$ because Prim's algorithm reaches each edge exactly once. Inserting our deterministic altitude prediction

$$\pi(a|s) := f_{\text{static}}(s|I;\theta)\ \ \delta(a - f_{\text{static}}(s|I;\theta)), \tag{2.19}$$

where δ is the Dirac distribution, we get

$$\nabla_\theta J = \frac{1}{|V|} \nabla_\theta \sum_s f_{\text{static}}(s|I;\theta)\, Q^\pi(s,a). \tag{2.20}$$

Comparing equation (2.20) with equation (2.15), we observe that $\nabla_\theta J \sim \nabla_\theta \mathcal{L}_{\text{SL}}$, where our weights $R(e)$ essentially play the role of the action-value function Q. This suggests to introduce a *discount factor* in $R(e)$. To do so, we replace the temporal differences t between states in (2.18) with tree distances $\text{dist}(w, \rho(w))$ or $\text{dist}(w, \rho^*(w))$ counting the number of edges between node w and its root edge. This gives the discounted weights

$$R_{\text{RL}}(e) := \begin{cases} \sum\limits_{w:\, e=\rho^*(w)} \gamma^{\,\text{dist}(w,\,\rho^*(w))} & \text{if } e \in E_\downarrow \\ \sum\limits_{w:\, e=\rho(w)} -\gamma^{\,\text{dist}(w,\,\rho(w))} & \text{if } e \in E_\uparrow \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

with discount factor $0 \leq \gamma \leq 1$ to be chosen such that γ^{dist} decays roughly according to the size of the CNNs receptive field. Substituting $R_{RL}(e)$ for $R(e)$ in (2.15) significantly improves convergence in our experiments. This analogy further motivates the application of current deep reinforcement training methods as described in section 2.5.2.

2.4.3 Dynamic Altitude Prediction

In every iteration, region growing according to Prim's algorithm only considers edges with exactly one end node located in the already assigned set. This offers the possibility to delay altitude estimation to the time when the values are actually needed. On-demand altitude computations can take advantage of the partial segmentations already available to derive additional *shape* clues that help resolving difficult assignment decisions.

Relative Assignments: To incorporate partial segmentations, we remove their dependence on the incidental choice of label values by means of *label-independent projection*. Consider an edge $e = (u, v)$ where node u is assigned to seed m and node v is unassigned. We now construct a labeling relative to m, distinguishing nodes assigned to m ("me" region), to another seed ("them") and unassigned ("nobody"). Relative labelings are represented by a standard 1-of-3 coding:

$$\mathcal{P}(w \,|\, m, \sigma) = \begin{cases} (1,0,0) & \text{if } \sigma(w) = m \quad \text{(me)} \\ (0,1,0) & \text{if } \sigma(w) = \lambda \quad \text{(nobody)} \\ (0,0,1) & \text{otherwise} \qquad \text{(them)} \end{cases} \tag{2.22}$$

In practice, we process relative labelings by adding a new branch to our neural network that receives $\mathcal{P}$ as an input, see section 2.5.1 for details.

Non-Markovian modeling: Another potentially useful cue is afforded by the fact that Prim's algorithm propagates the assignments recursively. Thus during every evaluation of f the complete history from previous iterations along the growth paths ϕ_m can be incorporated. We encode the history $\mathcal{H} : V \to \mathbb{R}^r$ about past assignment decisions as an r-dimensional vector in each node. In practice, we incorporate history by adding a recurrent layer to our neural network.

We introduce the *dynamic* altitude predictions:

$$\begin{aligned} &f(e = (u,v)), \mathcal{H}(v) = \\ &\qquad f_{\text{dyn}}(e \,|\, I, \mathcal{P}(. \,|\, \sigma(u), \sigma), \mathcal{H}(u); \theta_{\text{dyn}}) \end{aligned} \tag{2.23}$$

that receives the relative assignments projection $\mathcal{P}$ and u's hidden state $\mathcal{H}(u)$ as an additional input and outputs both the edge's altitude $f(e)$ and v's hidden state $\mathcal{H}(v)$: This variant of the altitude estimator performs best in our experiments. The emergent behavior of our models suggests that the algorithm uses history to adjust local diffusion parameters such as preferred direction and "viscosity", similar to the viscous watershed transform described in [156].

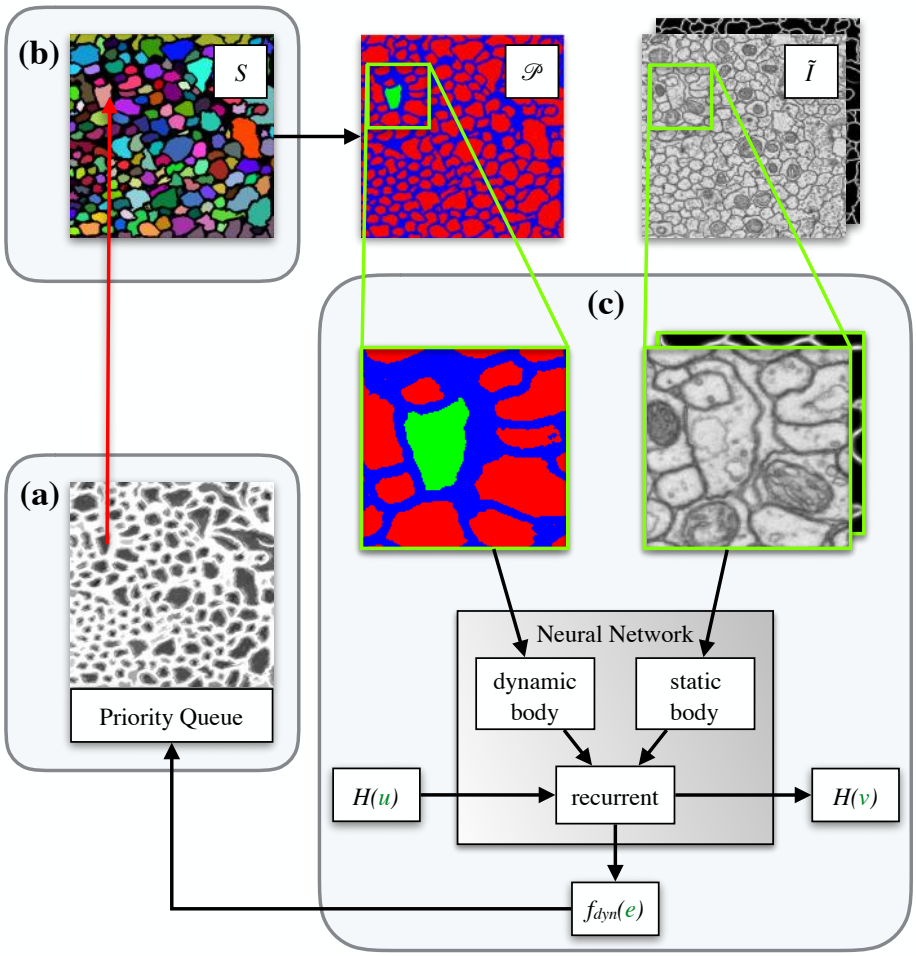


Figure 2.2: Overview implementation of learned watershed algorithm with neural network and priority queue. In each iteration the minimal edge according to equation (2.6) is found using a priority queue (a) and the region label is propagated (b), which updates the projection $\mathcal{P}$. For all unassigned edges that are not in the priority queue and need to be considered by Prim's algorithm in the next iteration, the altitude $f_{\text{dyn}}(e)$ is evaluated using the dynamic edge prediction network (c).

2.5 Methods

2.5.1 Neural Network Architecture

Our network architecture builds mainly on the work of Yu and Koltun [175] who introduced dilated convolutions to achieve dense segmentations and systematically aggregate multi-scale contextual information without pooling operations. We split our network into two convolutional branches (see Figure 2.3): The upper branch processes the static input I, and the lower one the dynamic input $\mathcal{P}(\cdot)$. Since the input of the upper branch doesn't change during prediction, its network activations can be precomputed for all edges, leading to a significant speed-up. We choose gated recurrent units (GRU) instead of long short-term memory (LSTM) in the recurrent network part, because GRUs have no internal state and get all history from the hidden state vector $\mathcal{H}(\cdot)$, saving on memory and bookkeeping.

2.5.2 Training Methods

Augmenting the Input Image: We noted above that structured learning is superior because it considers edges jointly. However, it can only rely on the sparse training sets $E_\downarrow \cup E_\uparrow$. In contrast, unstructured learning can make use of all edges and thus has a much bigger training set. This means that more powerful predictors, e.g. much deeper CNNs, can be trained, leading to more robust predictions and bigger receptive fields.

To combine the advantages of both approaches, we propose to augment the input image I with an additional channel holding node boundary probabilities predicted by an unstructured model $g(w|I;\theta_{\text{UL}})$:

$$\tilde{I} := [I \quad g] : V \to \mathbb{R}^{D+1} \tag{2.24}$$

We train the CNN g separately beforehand and replace I with the *augmented input* $\tilde{I}$ everywhere in f_{static} and f_{dyn}. This simplifies structured learning because the predictor only needs to learn a refinement of the already reasonable altitudes in g. In principle, one could even train f and g jointly, but the combined model is too big for reliable optimization.

Training Schedule: Taking advantage of the close relationship with reinforcement learning, we adopt the asynchronous update procedure proposed by [110]. Here, independent workers fetch the current CNN parameters θ from the master and compute loss gradients for randomly selected training images in parallel. The master then applies these updates to the parameters in an asynchronous fashion. We found experimentally, that this lead to faster and more stable convergence than sequential training.

In order to train the recurrent network part, we replace the standard temporal input ordering with the succession of edges defined by the paths ϕ and ψ. In a sense, backpropagation in time thus becomes backpropagation along the minimum spanning forest.

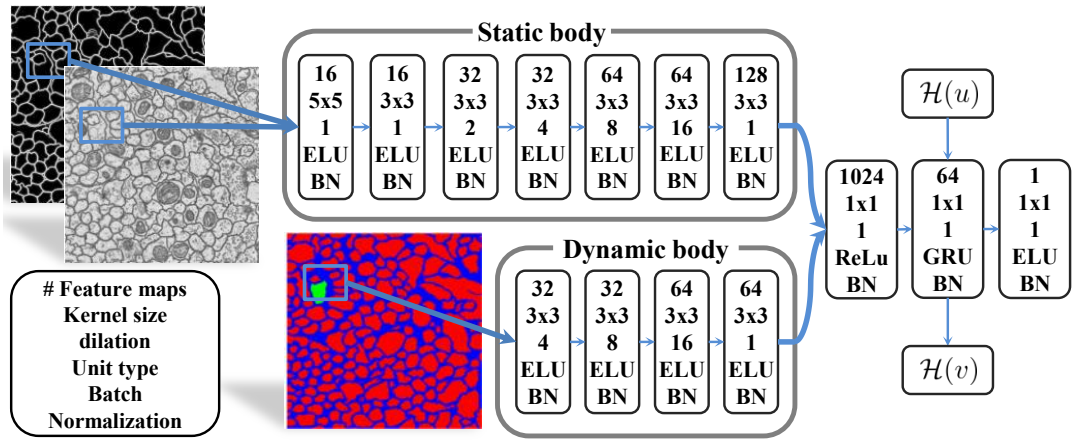


Figure 2.3: Neural Network architecture: The static body extracts features from the raw input and edge detector output. The more shallow dynamic body processes the interactions of different region projections $\mathcal{P}$. These informations are combined in a fully connected layer and set into a temporal context using a recurrent GRU layer. The network output is the priority of the edge towards the pixel at the center of the field of view.

2.6 Experiments and Results

Our experiments illustrate the performance of our proposed end-to-end trainable watershed in combination with static and dynamic altitude prediction. To this end, we compare with standard watershed and power watershed algorithms [37] on statically trained CNNs according to [20], see section 2.6.2. Furthermore we show in section 2.6.3 that the learned watershed surpasses the state-of-the-art segmentation in an adapted version of the CREMI Neuron Segmentation Challenge [26].

2.6.1 Experimental Setup and Evaluation Metrics

Seed Generation Oracle: All segmentation algorithms start at initial seeds M which are here provided by a "perfect" oracle. In our experiments, this oracle uses the ground truth segmentation to select one pixel with maximal L_2 distance to the region boundary per ground truth region.

Segmentation Metrics: In accordance with the CREMI challenge [26], we use the following segmentation metrics: The Rand score V^{Rand} measures the probability of agreement between segmentation S and ground truth S^* w.r.t. a randomly chosen node pair w, w'. Two segmentations agree if both assign w and w' to the same region or to different regions. The

Rand error ARAND $= 1 - V^{\text{Rand}}$ is the opposite, so that smaller values are better.

The *Variation of Information(VOI)* between S and S^* is defined as $VOI(S; S^*) = H(S|S^*) + H(S^*|S)$, where H is the conditional entropy [103]. To distinguish split errors from merge errors, we report the summands separately as $\text{VOI}^{\text{SPLIT}} = H(S|S^*)$ and $\text{VOI}^{\text{MERGE}} = H(S^*|S)$

2.6.2 Artificial Data

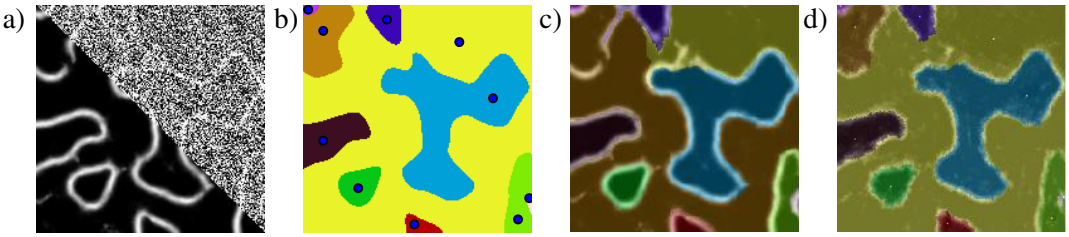


Figure 2.4: Artificial data example. a) Raw with $\sigma_{\text{noise}} = 0.6$ and prediction of baseline CNN. b) Ground truth. c) The brown region leaks out when standard watershed runs on top of baseline CNN. d) Our algorithm uses learned shape priors to close boundary gaps.

Dataset: In order to compare our models f_{static} and f_{dyn} with solutions based on unstructured learning, we create an artificial segmentation benchmark dataset with variable difficulty. First, we generate an edge image via the zero crossing of a 2D Gaussian process. This image is then smoothed with a Gaussian filter and corrupted with Gaussian noise at $\sigma_{\text{noise}} \in \{0.3, 0.6, 0.9\}$. For each σ_{noise}, we generate 1900 training images and 100 test images of size 252x252. One test image with corresponding ground truth and results is shown in figure 2.4.

Baseline: We choose a recent edge detection network from [175] to predict boundaries between different instances in combination with standard watershed (WS) and Power Watershed (PWS) [37] to generate an instance segmentation. Since these algorithms work best on slightly smoothed inputs, we apply Gaussian smoothing to the CNN output. The optimal smoothing parameters are determined by grid search on the training dataset. Additionally, we apply all watershed methods directly to smoothed raw image and report their overall best result as *RAW + WS*.

Performance: The measured segmentation errors of all algorithms are shown in table 2.1. Observed differences in performance mainly indicate how well each method handles low-contrast edges and narrow gaps between regions. The structurally trained watersheds outperform the unstructured baselines, because our loss function $\mathcal{L}_{\text{SL}}$ heavily penalizes the resulting segmentation errors. In all experiments, the *dynamic prediction* function f_{dyn} has the best performance, due to its superior modeling power. It can identify holes and close most contours correctly because it learns to derive shape and contingency clues from monitoring

ARAND	$\sigma_{\text{noise}} = 0.3$	$\sigma_{\text{noise}} = 0.6$	$\sigma_{\text{noise}} = 0.9$
$\mathcal{L}_{\text{SL}} + f_{\text{dyn}}$	**5.8 ± 0.8**	**12.5 ± 1.7**	**32.2 ± 1.8**
$\mathcal{L}_{\text{SL}} + f_{\text{static}}$	6.4 ± 0.9	13.8 ± 1.6	32.4 ± 2.2
NN + WS	6.5 ± 0.8	14.9 ± 3.6	33.4 ± 1.7
NN + PWS	6.5 ± 0.8	14.9 ± 1.7	33.2 ± 1.7
RAW + WS	24.0 ± 1.6	41.9 ± 1.8	55.0 ± 1.8

Table 2.1: Quality of the segmentation results on the artificial dataset. Reported lowest error for all parameters of baseline watersheds based on the rand error and a two pixel boundary distance tolerance.

intermediate results during the flooding process. A representative example of this effect is shown in figure 2.4.

2.6.3 Neurite Segmentation

Dataset: The MICCAI Challenge on Circuit Reconstruction from Electron Microscopy Images [26] contains 375 fully annotated slices of electron microscopy images I (of resolution 1250x1250 pixels). Part of a data slice is displayed in figure 2.6 top. Since the test ground truth segmentation has not been disclosed, we generate a new train/test split from the 3 original challenge training datasets by spltnting them into 3x75 z-continuous training- and 3x50 z-continuous test blocks.

Ideally, we would compare with [20] whose results define the state-of-the-art on the CREMI Challenge at time of submission. However, their pipeline, as described in their supplementary material, optimizes 2D segmentations jointly across multiple slices with a complex graphical model, which is beyond the scope of this paper.

Instead, we isolate the 2D segmentation aspect by adapting the challenge in the following manner: We run each segmentation algorithm with fixed ground truth seeds (see section 2.6.1) and evaluate their results on each z-slice separately. The restriction to 2D evaluation requires a slight manual correction of the ground truth: The ground truth accuracy in z-direction is just ± 1 slice. The official 3D evaluation scores compensate for this by ignoring deviations of ± 1 pixels in z-direction. Since this trick doesn't work in 2D, we remove 4 regions with no visual evidence in the image and all segments smaller than 5 pixels. Boundary tolerances in the x-y plane are treated as in the official CREMI scores where deviations from the true boundary are ignored if they do not exceed 6.25 pixels.

Baseline: We compare the Learned Watershed performance against the Power Watershed

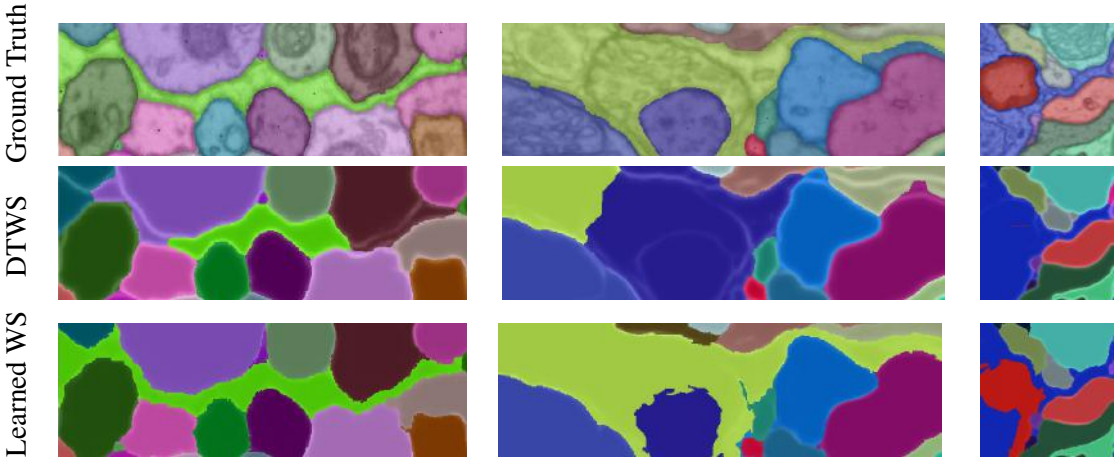


Figure 2.5: Detailed success and failure cases of our method. Success case: long thin neurites (left), weak boundaries (middle). Failure case (right).

[37], Viscous Watershed [156], RandomWalker [50], Stochastic Watershed [9] and Distance Transform Watershed [20]. The boundary probability prediction g (the same g as in equation (2.24)) was provided by a deep CNN trained with an unstructured loss-function. In particular, the Distance Transform Watershed (DTWS) and the prediction g were used to produce the current state-of-the-art on the CREMI challenge. To obtain the DTWS, one thresholds g, computes a distance transform of the background, i.e. the non-boundary pixels and runs the watershed algorithm on the inverted distances. According to [20], this is the best known heuristic to close boundary gaps in these data, but requires manual parameter tuning. We found the parameters of all baseline algorithms by grid search using the training dataset. To ensure fair comparison, we start region growing from ground truth seeds in all cases. Our algorithm takes the augmented image $\tilde{I}$ from eq. (2.24) as input and learns how to close boundary gaps.

Comparison to state-of-the-art: We show the 2D CREMI segmentation scores in Table 2.2. It is evident that the learned watershed transform significantly outperforms DTWS in both ARAND and VOI score. Quantitatively, we find that the flooding patterns and therefore the region shapes of the learned watershed prefer to adhere to biologically sensible structures. We illustrate this with our results on one CREMI test slice in Figure 2.6, as well as specific examples in Figure 2.5. We find throughout the dataset that especially *thin processes*, as depicted in the left panel of Figure 2.5, are a strength of our algorithm. Biologically sensible *shape completions* can also be found for roundish objects and is particularly noticeable when boundary evidence is weak, as shown in Figure 2.5 center. However, in rare cases, we find

	ARAND	VOI split	VOI merge
PowerWS	0.122 ± 0.003	0.340 ± 0.031	0.180 ± 0.019
ViscousWS	0.093 ± 0.003	0.328 ± 0.030	0.069 ± 0.003
RandomWalker	0.103 ± 0.004	0.355 ± 0.037	0.060 ± 0.004
Stochastic WS	0.193 ± 0.012	0.612 ± 0.080	0.077 ± 0.004
DTWS	0.085 ± 0.001	0.320 ± 0.029	0.070 ± 0.005
Learned WS	$\mathbf{0.082 \pm 0.001}$	$\mathbf{0.319 \pm 0.030}$	$\mathbf{0.057 \pm 0.004}$

Table 2.2: CREMI segmentation metrics evaluated on 2D slices: The Variation of Information (VOI) between a predicted segmentation and ground truth (lower is better) and the Adapted Rand Error (lower is better) [12].

incorrect shape completions (see right panel of Figure 2.5), mainly in areas of weak boundary evidence. It stands to reason that these errors could be fixed by providing more training data.

2.7 Conclusion

This paper proposes an end-to-end learnable seeded watershed algorithm that performs well an artificial data and neurosegmentation EM images. We found the following aspects to be critical success factors: First, we train a very powerful CNN to control region growing. Second, the CNN is trained in a structured fashion, allowing it to optimize segmentation performance directly, instead of treating pixels independently. Third, we improve modeling power by incorporating dynamic information about the current state of the segmentation. Specifically, feeding the current partial segmentation into the CNN provides shape clues for the next assignment decision, and maintaining a latent history along assignment paths allows to adjust growing parameters locally. We demonstrate experimentally that the resulting algorithm successfully solves difficult configurations like narrow region parts and low-contrast boundaries, where previous algorithms fail. In future work, we plan to include seed generation into the end-to-end learning scheme.

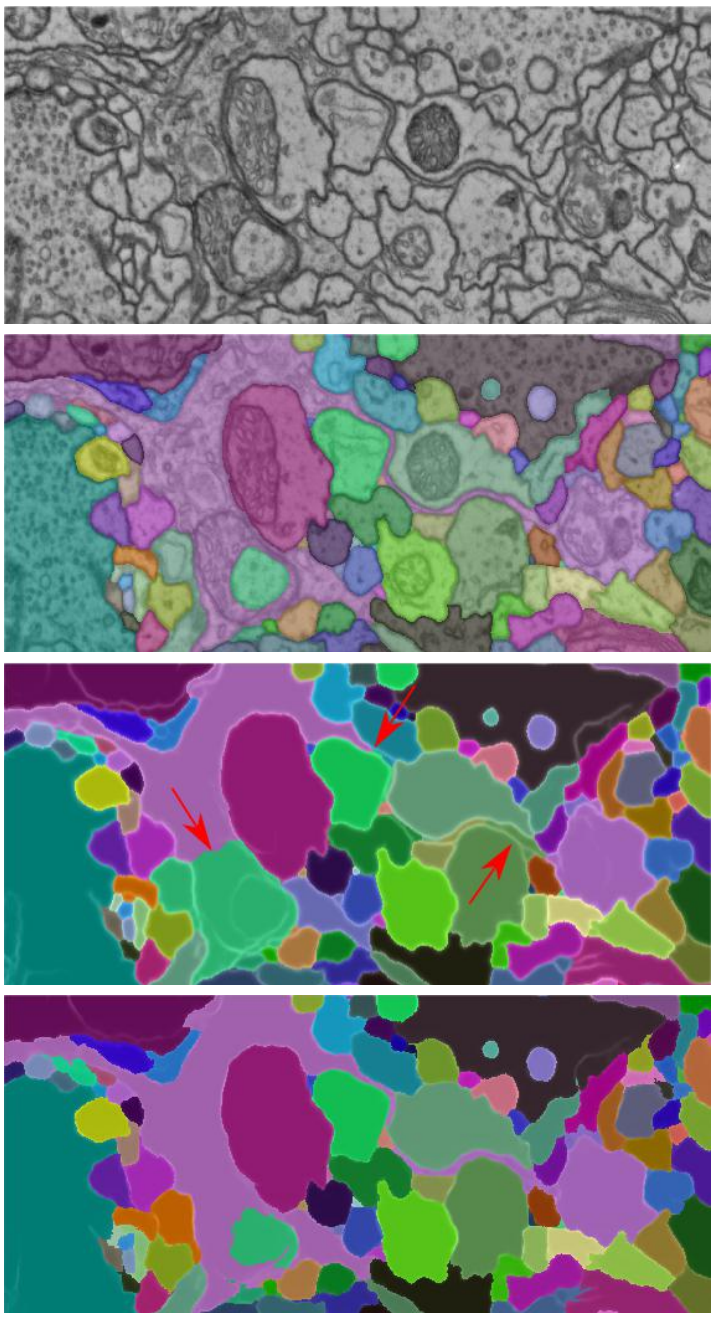


Figure 2.6: From top: Raw data. Ground truth. Result of distance transform WS (red arrows point out major errors). Result of our algorithm.

3 Mutex Watershed

In this chapter[1], we present our approach to incorporate repulsive interactions into the watershed algorithm. We show that this is beneficial in practice, obviating the need for seed prediction[2]. Furthermore, we prove that this simple algorithm finds a global optimum of an objective function that is intimately related to the multicut / correlation clustering integer linear programming formulation. When presented with short-range attractive and long-range repulsive cues from a deep neural network, the Mutex Watershed gives the best results currently known for the competitive ISBI 2012 EM segmentation benchmark.

3.1 Introduction

Most image partitioning algorithms are defined over a graph encoding purely attractive interactions. No matter whether a segmentation or clustering is then found agglomeratively (as in single linkage clustering / watershed) or divisively (as in spectral clustering or iterated normalized cuts), the user either needs to specify the desired number of segments or a termination criterion. An even stronger form of supervision is in terms of seeds, where one pixel of each segment needs to be designated either by a user or automatically. Unfortunately, clustering with automated seed selection remains a fragile and error-fraught process, because every missed or hallucinated seed causes an under- or oversegmentation error. Although the learning of good edge detectors boosts the quality of classical seed selection strategies (such as finding local minima of the boundary map, or thresholding boundary maps), non-local effects of seed placement along with strong variability in region sizes and shapes make it hard for any learned predictor to place *exactly one* seed in every true region.

In contrast to the above class of algorithms, multicut / correlation clustering partitions

[1] This chapter is based on our paper [163, 165], which was published in 2018/2019. The results in this chapter represent the current state at the time of publication. The algorithm was implemented and conceived by Constantin Pape and myself, whose neural network training pipeline for EM-images segmentation was indispensable for beating the state-of-the-art on the ISBI challenge. The theoretical characterization of the algorithm was done by Alberto Bailoni and me. Alberto especially developed the relation of the MWS to the Power watershed framework.

[2] Seed prediction for neuron segmentation is usually infeasible, due to the large, complex structures of the segments that also lack unique identification points.

vertices with both attractive and repulsive interactions encoded into the edges of a graph. Multicut has the great advantage that a "natural" partitioning of a graph can be found, without needing to specify a desired number of clusters, or a termination criterion, or one seed per region. Its great drawback is that its optimization is NP-hard.

The main insight of this paper is that when both attractive and repulsive interactions between pixels are available, then a generalization of the watershed algorithm can be devised that segments an image *without* the need for seeds or stopping criteria or thresholds. It examines all graph edges, attractive and repulsive, sorted by their weight and adds these to an active set iff they are not in conflict with previous, higher-priority, decisions. The attractive subset of the resulting active set is a forest, with one tree representing each segment. However, the active set can have loops involving more than one repulsive edge. See Fig. 3.1 for a visual abstract.

In summary, our principal contributions are, first, a fast deterministic algorithm for graph partitioning with both positive and negative edge weights that does not need prior specification of the number of clusters (section 3.4); and second, its theoretical characterization, including proof that it globally optimizes an objective related to the multicut correlation clustering objective (3.4).

Combined with a deep net, the algorithm also happens to define the state-of-the-art in a competitive neuron segmentation challenge (Section 3.5).

This is an extended version version of [163], with the second principal contribution (section 3.4) being new.

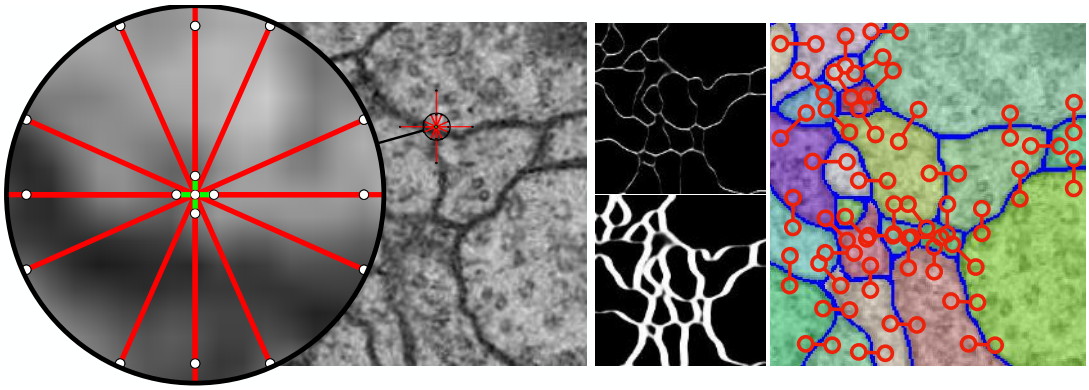


Figure 3.1: Left: Overlay of raw data from the ISBI 2012 EM segmentation challenge and the edges for which attractive (green) or repulsive (red) interactions are estimated for each pixel using a CNN. Middle: vertical / horizontal repulsive interactions at intermediate / long range are shown in the top / bottom half. Right: Active mutual exclusion (mutex) constraints that the proposed algorithm invokes during the segmentation process.

3.2 Related Work

In the original watershed algorithm [22, 157], seeds were automatically placed at all local minima of the boundary map. Unfortunately, this leads to severe over-segmentation. Defining better seeds has been a recurring theme of watershed research ever since. The simplest solution is offered by the seeded watershed algorithm [23]: It relies on an oracle (an external algorithm or a human) to provide seeds and assigns each pixel to its nearest seed in terms of minimax path distance.

In the absence of an oracle, many automatic methods for seed selection have been proposed in the last decades with applications in the fields of medicine and biology. Many of these approaches rely on edge feature extraction and edge detection like gradient calculation [4, 124]. Other types of methods generate seeds by first performing feature extraction [125, 166], whereas others first extract region of interests and then place seeds inside these regions by using thresholding [3], binarization [140], k-means [111] or other strategies [1, 2].

In applications where the number of regions is hard to estimate, simple automatic seed selection methods, e.g. defining seeds by connected regions of low boundary probability, do not work: The segmentation quality is usually insufficient because multiple seeds are in the same region and/or seeds leak through the boundary. Thus, in these cases seed selection may be biased towards over-segmentation (with seeding at all minima being the extreme case). The watershed algorithm then produces superpixels that are merged into final regions by more or less elaborate postprocessing. This works better than using watersheds alone because it exploits the larger context afforded by superpixel adjacency graphs. Many criteria have been proposed to identify the regions to be preserved during merging, e.g. region dynamics [51], the waterfall transform [21], extinction values [155], region saliency [115], and (α, ω)-connected components [144]. A merging process controlled by criteria like these can be iterated to produce a hierarchy of segmentations where important regions survive to the next level. Variants of such hierarchical watersheds are reviewed and evaluated in [123].

These results highlight the close connection of watersheds to hierarchical clustering and minimum spanning trees/forests [108, 113], which inspired novel merging strategies and termination criteria. For example, [137] simply terminated hierarchical merging by fixing the number of surviving regions beforehand. [101] incorporate predefined sets of generalized merge constraints into the clustering algorithm. Graph-based segmentation according to [47] defines a measure of quality for the current regions and stops when the merge costs would exceed this measure. Ultrametric contour maps [10] combine the gPb (global probability of boundary) edge detector with an oriented watershed transform. Superpixels are agglomerated until the ultrametric distance between the resulting regions exceeds a learned threshold. An optimization perspective is taken in [52, 69], which introduces h-increasing energy functions and builds the hierarchy incrementally such that merge decisions greedily minimize the energy.

The authors prove that the optimal cut corresponds to a different unique segmentation for every value of a free regularization parameter.

An important line of research is given by partitioning of graphs with both attractive and repulsive edges [66]. Solutions that optimally balance attraction and repulsion do not require external stopping criteria such as predefined number of regions or seeds. This generalization leads to the NP-hard problem of correlation clustering or (synonymous) multicut (MC) partitioning. Fortunately, modern integer linear programming solvers in combination with incremental constraint generation can solve problem instances of considerable size [8], and good approximations exist for even larger problems [119, 172] Reminiscent of strict minimizers [91] with minimal L_∞-norm solution, our work solves the multicut objective optimally when all graph weights are raised to a large power.

Related to the proposed method, the greedy additive edge contraction (GAEC) [65] heuristic for the multicut also sequentially merges regions, but we handle attractive and repulsive interactions separately and define edge strength between clusters by a maximum instead of an additive rule. The greedy fixation algorithm introduced in [92] is closely related to the proposed method; it sorts attractive and repulsive edges by their absolute weight, merges nodes connected by attractive edges and introduces no-merge constraints for repulsive edges. However, similar to GAEC, it defines edge strength by an additive rule, which increases the algorithm's runtime complexity compared to the presented Mutex Watershed. Also, it is not yet known what objective the algorithm optimizes globally, if any.

Another beneficial extension is the introduction of additional long-range edges. The strength of such edges can often be estimated with greater certainty than is achievable for the local edges used by watersheds on standard 4- or 8-connected pixel graphs. Such repulsive long-range edges have been used in [179] to represent object diameter constraints, which is still an MC-type problem. When long-range edges are also allowed to be attractive, the problem turns into the more complicated lifted multicut (LMC) [56]. Realistic problem sizes can only be solved approximately [19, 65], but watershed superpixels followed by LMC postprocessing achieve state-of-the-art results on important benchmarks [20]. Long-range edges are also used in [89], as side losses for the boundary detection convolutional neural network (CNN); but they are not used explicitly in any downstream inference.

In general, striking progress in watershed-based segmentation has been achieved by learning boundary maps with CNNs. This is nicely illustrated by the evolution of neurosegmentation for connectomics, an important field we also address in the experimental section. CNNs were introduced to this application in [59] and became, in much refined form [34], the winning entry of the ISBI 2012 Neuro-Segmentation Challenge [12]. Boundary maps and superpixels were further improved by progress in CNN architectures and data augmentation methods, using U-Nets [130], FusionNets [126] or inception modules [20]. Subsequent postprocessing with the GALA algorithm [73, 117], conditional random fields [154] or the lifted multicut [20]

pushed the envelope of final segmentation quality. MaskExtend [104] applied CNNs to both boundary map prediction and superpixel merging, while flood-filling networks [60] eliminated superpixels altogether by training a recurrent neural network to perform region growing one region at a time.

Most networks mentioned so far learn boundary maps on pixels, but learning works equally well for edge-based watersheds, as was demonstrated in [121, 185] using edge weights generated with a CNN [151, 152]. Tayloring the learning objective to the needs of the watershed algorithm by penalizing critical edges along minimax paths [151] or end-to-end training of edge weights and region growing [162] improved results yet again.

Outside of connectomics, [16] obtained superior boundary maps from CNNs by learning not just boundary strength, but also its gradient direction. Holistically-nested edge detection [76, 169] couples the CNN loss at multiple resolutions using deep supervision and is successfully used as a basis for watershed segmentation of medical images in [27].

We adopt important ideas from this prior work (hierarchical single-linkage clustering, attractive and repulsive interactions, long-range edges, and CNN-based learning). The proposed efficient segmentation framework can be interpreted as a generalization of [101], because we also allow for soft repulsive interactions (which can be overridden by strong attractive edges), and constraints are generated on-the-fly.

3.3 The Mutex Watershed Algorithm as an Extension of Seeded Watershed

In this section we introduce the Mutex Watershed Algorithm, an efficient graph clustering algorithm that can ingest both attractive and repulsive cues. We first reformulate seeded watershed as a graph partitioning with infinitely repulsive edges and then derive the generalized algorithm for finitely repulsive edges, which obviates the need for seeds.

3.3.1 Definitions and notation

Let $\mathcal{G} = (V, E, w)$ be a weighted graph. The scalar attribute $w : E \to \mathbb{R}$ associated with each edge is a merge affinity: the higher this number, the higher the inclination of the two incident vertices to be assigned to the same cluster. Conversely, large negative affinity indicates a greater desire of the incident vertices to be in different clusters. In our application, each vertex corresponds to one pixel in the image to be segmented. We call an edge $e \in E$ repulsive if $w_e < 0$ and we call it attractive if $w_e > 0$ and collect them in $E^- = \{e \in E \,|\, w_e < 0\}$ and $E^+ = \{e \in E \,|\, w_e > 0\}$ respectively.

In our application, each vertex corresponds to one pixel in the image to be segmented. The Mutex Watershed algorithm, defined in Section 3.3.3, maintains disjunct active sets $A^+ \subseteq E^+$, $A^- \subseteq E^-$, $A^+ \cap A^- = \emptyset$ that encode merges and mutual exclusion constraints, respectively. Clusters are defined via the "connected" predicate:

$$\begin{aligned} \forall i, j \in V: & \\ \Pi_{i \to j} &= \{\text{paths } \pi \text{ from } i \text{ to } j \text{ with } \pi \subseteq E^+\} \\ \text{connected}(i, j; A^+) &\Leftrightarrow \exists \text{ path } \pi \in \Pi_{i \to j} \text{ with } \pi \subseteq A^+ \\ \text{cluster}(i; A^+) &= \{i\} \cup \{j : \text{connected}(i, j; A^+)\} \end{aligned}$$

Conversely, the active subset $A^- \subseteq E^-$ of repulsive edges defines mutual exclusion relations by using the following predicate:

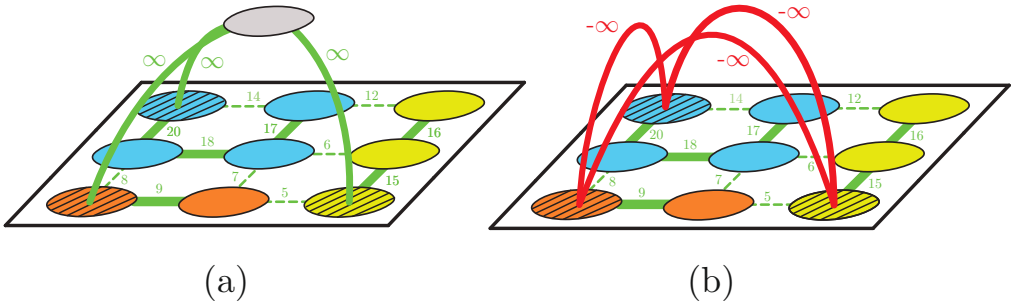
$$\begin{aligned} \text{mutex}(i, j; A^+, A^-) \Leftrightarrow\ & \exists\, e = (k, l) \in A^- \text{ with} \\ & k \in \text{cluster}(i; A^+) \text{ and} \\ & l \in \text{cluster}(j; A^+) \text{ and} \\ & \text{cluster}(i; A^+) \neq \text{cluster}(j; A^+) \end{aligned}$$

Admissible active edge sets A^+ and A^- must be chosen such that the resulting clustering is consistent, *i.e.* nodes engaged in a mutual exclusion constraint cannot be in the same cluster: $\text{mutex}(i, j; A^+, A^-) \Rightarrow \text{notconnected}(i, j; A^+)$. The "connected" and "mutex" predicates can be efficiently evaluated using a union find data structure.

3.3.2 Seeded watershed from a mutex perspective

One interpretation of the proposed method is in terms of a generalization of the edge-based watershed algorithm [106–108] or image foresting transform [45]. This algorithm can only ingest a graph with purely attractive interactions, $E^- = \emptyset$. Without further constraints, the algorithm would yield only the trivial result of a single cluster comprising all vertices. To obtain more interesting output, an oracle needs to provide seeds (e.g. one node per cluster). These seed vertices are all connected to an auxiliary node (see Fig. 3.2 (a)) by auxiliary edges with infinite merge affinity. A maximum spanning tree (MST) on this augmented graph can be found in linearithmic time; and the maximum spanning tree (or in the case of degeneracy: at least one of the maximum spanning trees) will include the auxiliary edges. When the auxiliary edges are deleted from the MST, a forest results, with each tree representing one cluster [45, 107, 108].

We now reformulate this well-known algorithm in a way that will later emerge as a special case of the proposed Mutex Watershed: we eliminate the auxiliary node and edges, and replace



(a) (b)

Figure 3.2: Two equivalent representations of the seeded watershed clustering obtained using (a) a maximum spanning tree computation or (b) Algorithm 2. Both graphs share the weighted attractive (green) edges and seeds (hatched nodes). The infinitely attractive connections to the auxiliary node (gray) in (a) are replaced by infinitely repulsive (red) edges between each pair of seeds in (b). The two final clusterings are defined by the active sets (bold edges) and are identical. Node colors indicate the clustering result, but are arbitrary.

Seeded Watershed:

WS$\big(\mathcal{G}(V,E)$, pos. weights $w : E \to \mathbb{R}^+$, seeds $S \subseteq V\big)$**:**

$A^+ \leftarrow \emptyset$

$A^- \leftarrow \{(s,t) \in S \times S \,|\, s \neq t\}$

▷ Equivalent to introducing infinitely repulsive edges between seeds

for $(i,j) = e \in E$ in descending order of w_e **do**

  if not connected$(i,j;A^+)$ **and not** mutex$(i,j;A^+,A^-)$ **then**

    $A^+ \leftarrow A^+ \cup e$

    ▷ merge i and j and inherit the mutex constraints of the parent clusters

return $A^+ \cup A^-$

Algorithm 2: Mutex version of seeded watershed algorithm. The output clustering is defined by the connected components of the final attractive active set A^+.

them by a set of infinitely repulsive edges, one for each pair of seeds (Fig. 3.2 (b)). Algorithm 2 is a variation of Kruskal's MST algorithm operating on the seed mutex graph just defined, and gives results identical to seeded watershed on the original graph.

This algorithm differs from Kruskal's only by the check for mutual exclusion in the if-statement. Obviously, the modified algorithm has the same effect as the original algorithm, because the final set A^+ is exactly the maximum spanning forest obtained after removing the auxiliary edges from the original solution.

In the sequel, we generalize this construction by admitting less-than-infinitely repulsive edges. Importantly, these can be dense and are hence much easier to estimate automatically than seeds with their strict requirement of only-one-per-cluster.

Mutex Watershed:

MWS$\big(\mathcal{G}(V, E)$**,** $w : E \to \mathbb{R}$**,** boolean connect_all$\big)$**:**

$A^+ \leftarrow \emptyset; \quad A^- \leftarrow \emptyset$

for $(i, j) = e \in E$ in descending order of $|w_e|$ **do**

  if $e \in E^+$ **then**

    if not mutex$(i, j; A^+, A^-)$ **then**

      if not connected$(i, j; A^+)$ **or** connect_all **then**

        merge(i, j): $A^+ \leftarrow A^+ \cup e$

          ▷ merge i and j and inherit the mutex constraints of the parent clusters

  else

    if not connected$(i, j; A^+)$ **then**

      addmutex(i, j): $A^- \leftarrow A^- \cup e$

        ▷ add mutex constraint between i and j

return $A^+ \cup A^-$

Algorithm 3: Mutex Watershed Algorithm. The output clustering is defined by the connected components of the final attractive active set A^+. The connect_all parameter changes the internal cluster connectedness from trees to fully connected, but does not change the output clustering. The connected predicate can be efficiently evaluated using union find data structures.

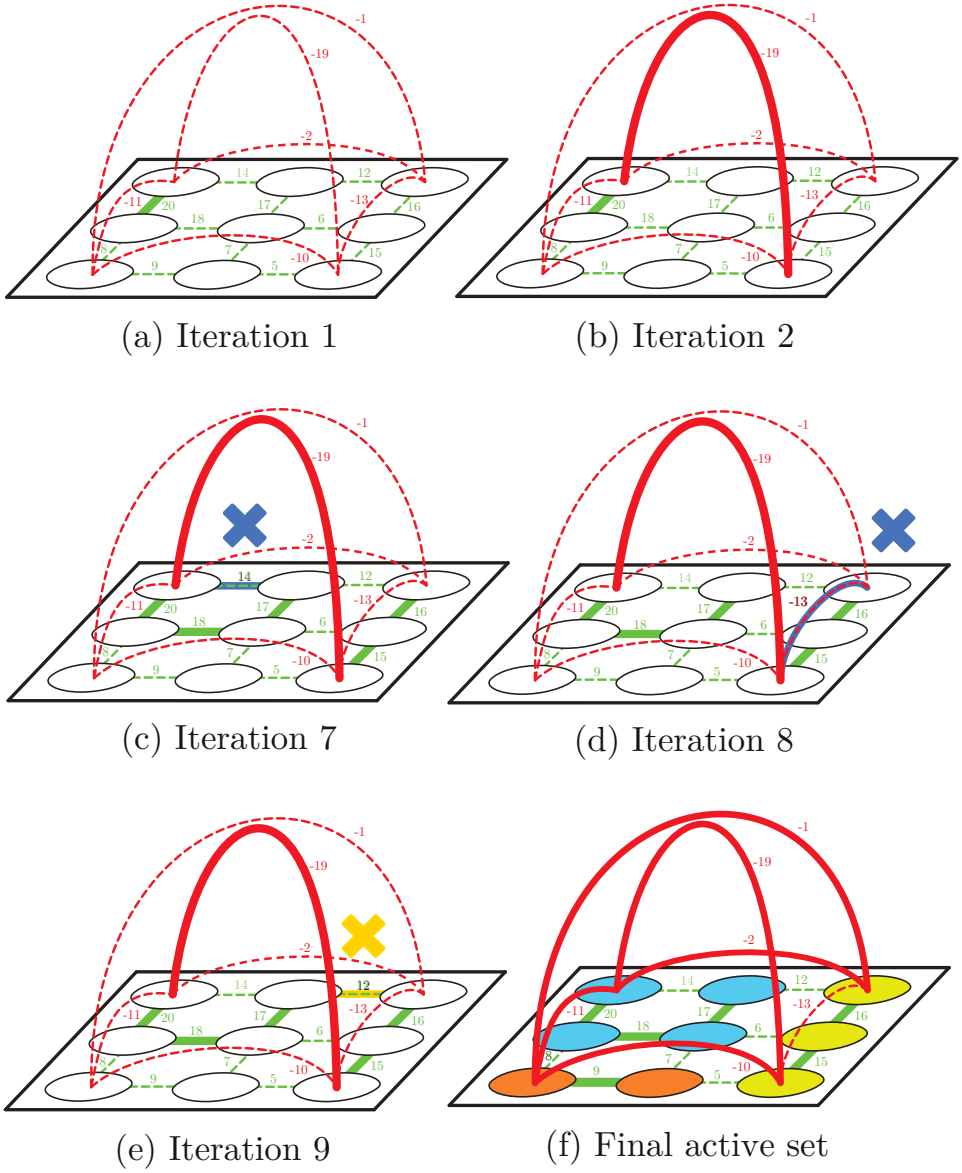


Figure 3.3: Some iterations of the Mutex Watershed Algorithm 3 applied to a graph with weighted attractive (green) and repulsive (red) edges. Edges accumulated in the active set A after a given number of iterations are shown in bold. The connect_all parameter of the algorithm is set to False, so that only the positive edges belonging to the maximum spanning tree of each cluster are added to the active set. Once the algorithm terminates, the final active set (f) defines the final clustering (indicated using arbitrary node colors). Some edges are not added to the active set because they are mutex constrained (yellow highlight) or because the associated nodes are already connected and in the same cluster (blue highlight).

3.3.3 Mutex Watersheds

We now introduce our core contribution: an algorithm that is empirically no more expensive than a MST computation; but that can ingest both attractive and repulsive cues and partition a graph into a number of clusters that does not need to be specified beforehand. Neither seeds nor hyperparameters that implicitly determine the number of resulting clusters are required.

The Mutex Watershed, Algorithm 3, proceeds as follows. Given a graph $\mathcal{G} = (V, E)$ with signed weights $w : E \to \mathbb{R}$, do the following: sort all edges E, attractive or repulsive, by their absolute weight in descending order into a priority queue. Iteratively pop all edges from the queue and add them to the active set one by one, provided that a set of conditions are satisfied. More specifically, assuming connect_all is False, if the next edge popped from the priority queue is attractive and its incident vertices are not yet in the same tree, then connect the respective trees provided this is not ruled out by a mutual exclusion constraint. If on the other hand the edge popped is repulsive, and if its incident vertices are not yet in the same tree, then add a mutual exclusion constraint between the two trees. The output clustering is defined by the connected components of the final attractive active set A^+.

The crucial difference to Algorithm 2 is that mutex constraints are no longer pre-defined, but created dynamically whenever a repulsive edge is found. However, new exclusion constraints can never override earlier, high-priority merge decisions. In this case, the repulsive edge in question is simply ignored. Similarly, an attractive edge must never override earlier and thus higher-priority must-not-link decisions.

The boolean value of the connect_all input parameter of the algorithm does not influence the final output clustering, but defines the internal cluster connectedness: when it is set to True, the algorithm adds all attractive intra-cluster edges to the active set A^+. When it is set to False, then a maximum spanning tree is built for each cluster similar to the seeded watershed. This variant of the algorithm will be helpful in the next section 3.4 to highlight the relation between the Mutex Watershed and the multicut problem.

Fig. 3.3 illustrates the proposed algorithm: Fig. 3.3a and Fig. 3.3b show examples of an unconstrained merge and an added mutex constraint, respectively; Fig. 3.3c and Fig. 3.3d show, respectively, an example of an attractive edge ($w_e = 14$) and repulsive edge ($w_e = -13$) that are not added to the active set because their incident vertices are already "connected" and belong to the same tree of the forest A^+; finally, Fig. 3.3e shows an attractive edge ($w_e = 12$) that is ruled out by a previously introduced mutual exclusion relation.

3.3.4 Time Complexity Analysis

Before analyzing the time complexity of algorithm 3 we first review the complexity of Kruskal's algorithm. Using a union-find data structure (with path compression and union by rank) the time

complexity of merge$(i,\ j)$ and connected$(i,\ j)$ is $\mathcal{O}(\alpha(V))$, where α is the slowly growing inverse Ackerman function, and the total runtime complexity is dominated by the initial sorting of the edges $\mathcal{O}(E \log E)$ [36].
To check for mutex constraints efficiently, we maintain a set of all active mutex edges

$$M[C_i] = \{(u,\ v) \in A^- | u \in C_i \vee v \in C_i\}$$

for every $C_i = \text{cluster}(i)$ using hash tables, where insertion of new mutex edges (i.e. addmutex) and search have an average complexity of $\mathcal{O}(1)$. Note that every cluster can be efficiently identified by its union-find root node. For mutex$(i,\ j)$ we check if $M[C_i] \cap M[C_j] = \emptyset$ by searching for all elements of the smaller hash table in the larger hash table. Therefore mutex$(i,\ j)$ has an average complexity of $\mathcal{O}(\min(|M[C_i]|, |M[C_j]|)$. Similarly, during merge$(i,\ j)$, mutex constraints are inherited by merging two hash tables, which also has an average complexity $\mathcal{O}(\min(|M[C_i]|, |M[C_j]|)$.
In conclusion, the average runtime contribution of attractive edges $\mathcal{O}(\max(|E^+| \cdot \alpha(V), |E^+| \cdot M))$ (checking mutex constraints and possibly merging) and repulsive edges $\mathcal{O}(\max(|E^-| \cdot \alpha(V), |E^-|))$ (insertion of one mutex edge) result in a total average runtime complexity of algorithm 3:

$$\mathcal{O}(\max(E \log E\,,\ EM)). \tag{3.1}$$

where M is the expected value of $\min(|M[C_i]|, |M[C_j]|)$ and therefore[3] $\alpha(V) \in \mathcal{O}(\log V) \in \mathcal{O}(\log E)$. In the worst case $\mathcal{O}(M) \in \mathcal{O}(E)$, the Mutex Watershed Algorithm has a runtime complexity of $\mathcal{O}(E^2)$. Empirically, we find that $\mathcal{O}(EM) \approx \mathcal{O}(E \log E)$ by measuring the runtime of Mutex Watershed for different sub-volumes of the ISBI challenge (see Figure 3.4), leading to a

$$\text{Empirical Mutex Watershed Complexity: } \mathcal{O}(E \log E) \tag{3.2}$$

3.4 Theoretical characterization

Towards the Multicut framework. In section 3.3.3, we have introduced the Mutex Watershed (MWS) algorithm as a generalization of seeded watersheds and the Kruskal algorithm in particular. However, since we are considering graphs with negative edge weights, the MWS is conceptually closer to the multicut problem and related heuristics such as GAEC and GF [92]. Fortunately, due to the structure of the MWS it can be analyzed using dynamic programming. This section summarizes our second contribution, i.e. the proof that the Mutex Watershed Algorithm globally optimizes a precise objective related to the multicut.

[3] In the worst case, G is a fully connected graph, with $|E| = |V|^2$, hence $\log |V| = \frac{1}{2} \log |E|$.

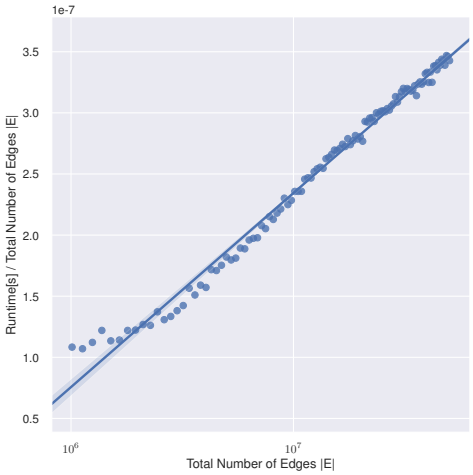
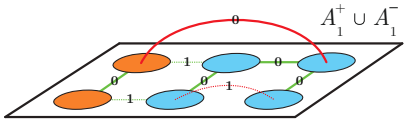


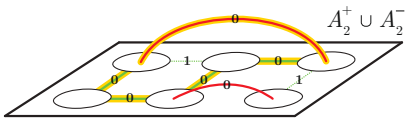
Figure 3.4: Runtime T of Mutex Watershed (without sorting of edges) measured on sub-volumes of the ISBI challenge of different sizes (thereby varying the total number of edges E). We plot $\frac{T}{|E|}$ over $|E|$ in a logarithmic plot, which makes $T \sim |E|log(|E|)$ appear as straight line. A logarithmic function (blue line) is fitted to the measured $\frac{T}{|E|}$ (blue circles) with ($R^2 = 0.9896$). The good fit suggests that empirically $T \approx \mathcal{O}(E \log E)$.



$\forall i, j \in V \;\; \texttt{connected}(i, j; A_1^+) \Rightarrow \textbf{not } \texttt{mutex}(i, j; A_1^+, A_1^-)$

$\mathcal{C}^-(A_1, \mathcal{G}, w) = \emptyset$

$x^{A_1} \in \mathsf{SC}(\mathcal{G}, w)$



$\exists\, i, j \in V \;\; \texttt{connected}(i, j; A_2^+) \;\; \text{and} \;\; \texttt{mutex}(i, j; A_2^+, A_2^-)$

$\mathcal{C}^-(A_2, \mathcal{G}, w) \neq \emptyset$

$x^{A_2} \notin \mathsf{SC}(\mathcal{G}, w)$

Figure 3.5: Consistent and inconsistent active sets – Two different active edge sets $A_1 \subseteq E$ (on the left) and $A_2 \subseteq E$ (on the right) on identical toy graphs with six nodes, attractive (green) and repulsive (red) edges. The value of the edge indicator $x^A \in \{0, 1\}^{|E|}$ defined in Eq. 3.4 is shown for every edge. Members of the active sets are shown as solid lines. **On the left**, the active set A_1 is *consistent*, i.e. does not include any conflicted cycle $\mathcal{C}^-(\mathcal{G}, w)$ (see Def. 3.4.1): Therefore, it is associated with a clustering (represented by arbitrary node colors). **On the right**, the active set A_2 is not consistent and includes at least one conflicted cycle (highlighted in yellow), thus it cannot be associated with a node clustering.

3.4.1 Review of the Multicut problem and its objective

In the following, we will review the multicut problem not in its standard formulation but in the *Cycle Covering Formulation* introduced in [85], which is similar to the MWS formulation as it also considers the set of *attractive* and *repulsive* edges separately. Previously, in Sec. 3.3.1, we defined a clustering by introducing the concept of an active set of edges $A = A^+ \cup A^- \subseteq E$ and the connected/mutex predicates. In particular, an active set describes a valid clustering if it does not include **both** a path of only attractive edges **and** a path with exactly one repulsive edge connecting any two nodes $i, j \in V$:

$$\text{connected}(i, j; A^+) \quad \Longrightarrow \quad \textbf{not } \text{mutex}(i, j; A^+, A^-). \tag{3.3}$$

In other words, an active set is *consistent* and describes a clustering if it does not contain any cycle with exactly one repulsive edge (known as conflicted cycles).

Definition 3.4.1. ***Conflicted cycles*** *– We call a cycle of $\mathcal{G}$ conflicted w.r.t. $(\mathcal{G}, w)$ if it contains precisely one repulsive edge $e \in E^-$, s.t. $w_e < 0$. We denote by $\mathcal{C}^-(\mathcal{G}, w) \subseteq \mathcal{C}(\mathcal{G}, w)$ the set of all conflicted cycles. Furthermore, given a set of edges $A \subseteq E$, we denote by $\mathcal{C}^-(A, \mathcal{G}, w) \subseteq \mathcal{C}^-(\mathcal{G}, w)$ the set of conflicted cycles involving only edges in A.*

From now on, in order to describe different clustering solutions in the framework of (integer) linear programs, we associate each active set A with the following edge indicator x^A

$$x^A := \mathbb{1}\{e \notin A)\} \in \{0, 1\}^{|E|}. \tag{3.4}$$

In this way, the cycle-free property $\mathcal{C}^-(A, \mathcal{G}, w) = \emptyset$ of an active set can be reformulated in terms of linear inequalities:

$$\forall C \in \mathcal{C}^-(\mathcal{G}, w) : \sum_{e \in E_C} x^A_e \geq 1 \quad \Longleftrightarrow \quad \mathcal{C}^-(A, \mathcal{G}, w) = \emptyset. \tag{3.5}$$

In words, the active set cannot contain conflicted cycles; or vice versa, every conflicted cycle must contain at least one edge that is not part of the active set. Following [85], via this property we describe the space of all possible clustering solutions by defining the convex hull $\mathsf{SC}(\mathcal{G}, w)$ of all edge indicators corresponding to valid clusterings of $(\mathcal{G}, w)$:

Definition 3.4.2. *Let $\mathsf{SC}(\mathcal{G}, w)$ denote the convex hull of all edge indicators $x \in \{0, 1\}^{|E|}$ satisfying the following system of inequalities:*

$$\forall C \in \mathcal{C}^-(\mathcal{G}, w) : \quad \sum_{e \in E_C} x_e \geq 1. \tag{3.6}$$

That is, $\mathsf{SC}(\mathcal{G}, w)$ contains all edge labelings for which every conflicted cycle is broken at least once. We call $\mathsf{SC}(\mathcal{G}, w)$ the set covering polyhedron with respect to conflicted cycles, similarly to [85].

Fig. 3.5 summarizes these definitions and provides an example of consistent and inconsistent active sets with their associated clusterings and edge indicators.

As shown in [85], the *multicut optimization problem* can be formulated with constraints over conflicted cycles in terms of the following integer linear program (ILP), which is NP-hard:

$$\min_{x \in \mathsf{SC}(\mathcal{G}, w)} \sum_{e \in E} |w_e| x_e. \tag{3.7}$$

The solution of the multicut problem is given by the clustering associated to the connected components of the active set $\hat{A}^+ = \{e \in E^+ | \hat{x}_e = 0\}$, where $\hat{x} \in \{0, 1\}^{|E|}$ is the solution of (3.7).

3.4.2 Mutex Watershed Objective

We now define the Mutex Watershed objective that is minimized by the Mutex Watershed Algorithm (proof in Section 3.4.3) and show how it is closely related to the multicut problem defined in Eq. (3.7). Lange et al. [85] introduce the concept of dominant edges in a graph. For example, an attractive edge $f \in E^+$ is called dominant if there exists a cut B with $f \in E_B$ such that $|w_f| \geq \sum_{e \in E_B \setminus \{f\}} |w_e|$. These highlight an aspect of the multicut problem that can be used to search for optimal solutions more efficiently. Not all weighted graphs contain dominant edges; but if, assuming no ties, we raise all graph weights to a large enough power a similar property emerges.

Definition 3.4.3. ***Dominant power:*** *Let $\mathcal{G} = (V, E, w)$ be an edge-weighted graph, with unique weights $w : E \to \mathbb{R}$. We call $p \in \mathbb{N}^+$ a dominant power if:*

$$|w_e|^p > \sum_{t \in E, w_t < w_e} |w_t|^p \qquad \forall e \in E, \tag{3.8}$$

In contrast to dominant edges [85], we do not consider edges on a cut but rather all edges with smaller absolute weight. Note that there exists a dominant power for any finite set of edges, since for any $e \in E$ we can divide (3.8) by $|w_e|^p$ and observe that the normalized weights $|w_t|^p / |w_e|^p$ (and any finite sum of these weights) converges to 0 when p tends to infinity.

By considering the multicut problem in Eq. (3.7) and raising the weights $|w_e|$ to a dominant power p, we fundamentally change the problem structure:

Definition 3.4.4. ***Mutex Watershed Objective:*** *Let $\mathcal{G} = (V, E, w)$ be an edge-weighted graph, with unique weights $w : E \to \mathbb{R}$ and $p \in \mathbb{N}^+$ a dominant power. Then the Mutex Watershed Objective is defined as the integer linear program*

$$\min_{x \in \mathsf{SC}(\mathcal{G}, w)} \quad \sum_{e \in E} |w_e|^p \, x_e \tag{3.9}$$

where $\mathsf{SC}(\mathcal{G}, w)$ is the convex hull defined in Def. 3.4.2.

In the following section, we will prove that this modified version of the multicut objective, which we call Mutex Watershed Objective, is indeed optimized by the Mutex Watershed Algorithm:

Theorem 3.4.1. *Let $\mathcal{G} = (V, E, w)$ be an edge-weighted graph, with unique weights $w : E \to \mathbb{R}$ and $p \in \mathbb{N}^+$ a dominant power. Then the edge indicator given by the Mutex Watershed Algorithm 3*

$$x^{\mathbf{MWS}} := \mathbb{1}\Big\{ e \notin \mathbf{MWS}\Big(\mathcal{G}, w, \textit{connect\_all=True}\Big) \Big\}$$

minimizes the Mutex Watershed Objective in Eq. (3.9).

3.4.3 Proof of optimality via dynamic programming

Conflicted-Cycles Mutex Watershed:

CCMWS$\big(\mathcal{G}(V, E), w : E \to \mathbb{R}\big)$**:**

```
A ← ∅
for (i, j) = e ∈ E in descending order of |w_e| do
    if C⁻(A ∪ {e}, G, w) = ∅ then
        A ← A ∪ e
return A
```

Algorithm 4: Equivalent formulation of the Mutex Watershed Algorithm 3, with input parameter connect_all=True. The set of conflicted cycles $\mathcal{C}^-(A, \mathcal{G}, w)$ is defined in Def. 3.4.1. The output clustering is defined by the connected components of the final attractive active set $A^+ = A \cap E^+$.

In this section we prove Theorem 3.4.1, i.e. that the Mutex Watershed Objective defined in 3.4.4 is solved to optimality by the Mutex Watershed Algorithm 4. Particularly, in the following Sec. 3.4.3 we show that the edge indicator associated to the solution of the MWS algorithm lies in $\mathsf{SC}(\mathcal{G}, w)$, whereas in Sec. 3.4.3 we prove that it solves Eq. 3.9 to optimality.

Cycle consistency

The Mutex Watershed algorithm introduced in Sec. 3.3 iteratively builds an active set $A = A^+ \cup A^-$ such that nodes engaged in a mutual exclusion constraint (encoded by edges in A^-) are never part of the same cluster. In other words, this means that the active set built by the Mutex Watershed at every iteration does never include a *conflicted cycle* and is always *consistent*. In particular, for any attractive edge $(i, j) = e^+ \in E^+$ and any consistent set A that fulfills $\mathcal{C}^-(A, \mathcal{G}, w) = \emptyset$:

$$\textbf{not } \text{mutex}(i, j, A^+, A^-) \quad \Leftrightarrow \quad \mathcal{C}^-(A \cup \{e^+\}, \mathcal{G}, w) = \emptyset$$

Similarly, for any repulsive edge $(s, t) = e^- \in E^-$:

$$\textbf{not } \text{connected}(s, t, A^+) \quad \Leftrightarrow \quad \mathcal{C}^-(A \cup \{e^-\}, \mathcal{G}, w) = \emptyset$$

Therefore, we can rewrite Algorithm 3 in the form of Algorithm 4. This new formulation makes it clear that

$$\mathcal{C}^-\Big(\mathbf{MWS}\big(\mathcal{G}, w, \text{connect\_all=True}\big)\Big) = \emptyset. \tag{3.10}$$

Thus, thanks to Eq. 3.5 and definition 3.4.2, it follows that the MWS edge indicator $x^{\mathbf{MWS}}$ defined in 3.4.1 lies in $\mathsf{SC}(\mathcal{G}, w)$:

$$x^{\mathbf{MWS}} \in \mathsf{SC}(\mathcal{G}, w). \tag{3.11}$$

Optimality

We first note that the Mutex Watershed Objective 3.4.4 and Theorem 3.4.1 can easily be reformulated in terms of active sets to minimize

$$\underset{A \subseteq E}{\arg\min} \quad -\sum_{e \in A} |w_e|^p \qquad \text{s.t.} \quad \mathcal{C}^-(A, \mathcal{G}, w) = \emptyset. \tag{3.12}$$

We now generalize the Mutex Watershed (see Algorithm 5) and the objective such that an initial consistent set of active edges $\tilde{A} \subseteq E$ is supplied:

Definition 3.4.5. ***Energy optimization subproblem.*** *Let $\mathcal{G} = (V, E, w)$ be an edge-weighted graph. Define the optimal solution of the subproblem as*

$$S(\mathcal{G}, \tilde{A}) := \underset{A \subseteq (E \setminus \tilde{A})}{\textit{argmin}}\, T(A) \qquad \textit{with} \quad T(A) := -\sum_{e \in A} |w_e|^p, \tag{3.13}$$

$$\textit{s.t.} \quad \mathcal{C}^-(A \cup \tilde{A}, \mathcal{G}, w) = \emptyset, \tag{3.14}$$

where $\tilde{A} \subseteq E$ is a set of initially activated edges such that $\mathcal{C}^-(\tilde{A}, \mathcal{G}, w) = \emptyset$.

Initialized Mutex Watershed:

IMWS$\big(\mathcal{G}(V, E), w : E \to \mathbb{R}$**,** initial active set $\tilde{A}\big)$**:**

```
A ← ∅
for e ∈ E \ Ã in descending order of weight do
    if C⁻(A ∪ Ã ∪ {e}, G, w) = ∅ then
        A ← A ∪ e
return A
```

Algorithm 5: Mutex Watershed algorithm starting from initial active set $\tilde{A}$. An initial set $\tilde{A}$ of active edges is given as additional input and the final active set is such that $A \subseteq E \setminus \tilde{A}$. Note that Algorithm 4 is a special case of this algorithm when $\tilde{A} = \emptyset$. Differences with Algorithm 4 are highlighted in blue.

We note that for $\tilde{A} = \emptyset$, the optimal solution $S(\mathcal{G}, \emptyset)$ is equivalent to the solution minimizing the Mutex Watershed Objective and Eq. (3.12).

Definition 3.4.6. ***Incomplete, consistent initial set:*** *For an edge-weighted graph $\mathcal{G} = (V, E, w)$ a set of edges $\tilde{A} \subseteq E$ is consistent if*

$$\mathcal{C}^-(\tilde{A}, \mathcal{G}, w) = \emptyset. \tag{3.15}$$

$\tilde{A}$ is incomplete if it is not the final solution and there exists a consistent edge $\tilde{e}$ that can be added to $\tilde{A}$ without violating the constraints.

$$\exists\, \tilde{e} \in E \setminus \tilde{A} \quad \textit{s.t.} \quad \mathcal{C}^-(\tilde{A} \cup \{\tilde{e}\}, \mathcal{G}, w) = \emptyset \tag{3.16}$$

Definition 3.4.7. ***First greedy step:*** *Let us consider an incomplete, consistent initial active set $\tilde{A} \subseteq E$ on $\mathcal{G} = (V, E, w)$. We define*

$$g := \underset{e \in (E \setminus \tilde{A})}{\textit{argmax}} |w(e)| \quad \textit{s.t.} \quad \mathcal{C}^-(\tilde{A} \cup \{e\}, \mathcal{G}, w) = \emptyset. \tag{3.17}$$

as the feasible edge with the highest weight, which is always the first greedy step of Algorithm 5.

In the following two lemmas, we prove that the Mutex Watershed problem has an *optimal substructure property* and a *greedy choice property* [36], which are sufficient to prove that the Mutex Watershed algorithm finds the optimum of the Mutex Watershed Objective.

Lemma 3.4.2. ***Greedy-choice property.*** *For an incomplete, consistent initial active set $\tilde{A}$ of the Mutex Watershed, the first greedy step g is always part of the optimal solution*

$$g \in S(\mathcal{G}, \tilde{A}).$$

Proof. We will prove the theorem by contradiction by assuming that the first greedy choice is not part of the optimal solution, i.e. $g \notin S(\mathcal{G}, \tilde{A})$. Since g is by definition the feasible edge with highest weight, it follows that:

$$|w(e)| < |w(g)| \quad \forall e \in S(\mathcal{G}, \tilde{A}). \tag{3.18}$$

We now consider the alternative active set $A' = \{g\}$, that is a consistent solution, with

$$T(A') = -|w_g|^p \overset{(3.8)}{<} - \sum_{t \in S(\mathcal{G}, \tilde{A})} |w_t|^p = T\Big(S(\mathcal{G}, \tilde{A})\Big) \tag{3.19}$$

which contradicts the optimality of $S(\mathcal{G}, \tilde{A})$. □

Lemma 3.4.3. ***Optimal substructure property.*** *Let us consider an initial active set $\tilde{A}$, the optimization problem defined in Equation 3.13, and assume to have an incomplete, consistent problem (see Def. 3.4.6). Then it follows that:*

1. *After making the first greedy choice g, we are left with a subproblem that can be seen as a new optimization problem of the same structure;*
2. *The optimal solution $S(\mathcal{G}, \tilde{A})$ is always given by the combination of the first greedy choice and the optimal solution of the remaining subproblem.*

Proof. After making the first greedy choice and selecting the first feasible edge g defined in Equation 3.17, we are clearly left with a new optimization problem of the same structure that has the following optimal solution: $S(\mathcal{G}, \tilde{A} \cup \{g\})$.
In order to prove the second point of the theorem, we now show that:

$$S(\mathcal{G}, \tilde{A}) = \{g\} \cup S(\mathcal{G}, \tilde{A} \cup \{g\}). \tag{3.20}$$

Since algorithm 5 fulfills the greedy-choice property, $g \in S(\mathcal{G}, \tilde{A})$ and we can add the edge g as an additional constraint to the optimal solution:

$$\begin{aligned} S(\mathcal{G}, \tilde{A}) = \underset{A \subseteq (E \setminus \tilde{A})}{\text{argmin}} \quad & T(A) \\ \text{s. t.} \quad & \mathcal{C}^-(A \cup \tilde{A}, \mathcal{G}, w) = \emptyset; \quad g \in A \end{aligned} \tag{3.21}$$

Then it follows that:

$$\begin{aligned} S(\mathcal{G}, \tilde{A}) = &\{g\} \cup \underset{A \subseteq E \setminus (\tilde{A} \cup \{g\})}{\operatorname{argmin}} \quad T(A) \\ &\text{s. t.} \quad \mathcal{C}^-\Big(A \cup \{g\} \cup \tilde{A}, \mathcal{G}, w\Big) = \emptyset \end{aligned} \tag{3.22}$$

which is equivalent to Equation 3.20. □

Proof of Theorems 3.4.1. In Lemmas 3.4.2 and 3.4.3 we have proven that the optimization problem defined in 3.12 has the optimal substructure and a greedy choice property. It follows through induction that the final active set $\mathbf{MWS}\big(\mathcal{G}, w, \text{connect\_all=True}\big)$ found by the Mutex Watershed Algorithm 4 is the optimal solution for the Mutex Watershed objective (3.12) [36]. □

3.4.4 Relation to the extended Power Watershed framework

The Power Watershed [37] is an important framework for graph-based image segmentation that includes several algorithms like seeded watershed, random walker and graph cuts. Recently, [114] extended the framework to even more general types of hierarchical optimization algorithms thanks to the use of Γ-theory and Γ-convergence [24, 41]. In this section, we show how the Mutex Watershed algorithm can also be included in this extended framework[4] and how the framework suggests an optimization problem that is solved by the Mutex Watershed.

Mutex Watershed as hierarchical optimization algorithm

We first start by introducing the extended Power Watershed framework and restating the main theorem from [114]:

Theorem 3.4.4. ***[114] Extended Power Watershed Framework.*** *Consider three strictly positive integers $p, m, t \in \mathbb{N}^+$ and t real numbers*

$$1 \geq \lambda_0 > \lambda_1 > \ldots \lambda_{t-1} > 0 \tag{3.23}$$

Given t continuous functions $Q_k : \mathbb{R}^m \to \mathbb{R}$ with $0 \leq k < t$, define the function

$$Q^p(x) := \sum_{0 \leq k < t} \lambda_k^p Q_k(x). \tag{3.24}$$

[4] The connection between the Mutex Watershed and the extended Power Watershed framework was kindly pointed out by an anonymous reviewer.

Generic hierarchical optimization:

GHO$(Q_0, \dots, Q_{t-1})$**:**

$M_0 = \arg\min_{x \in \mathbb{R}^m} Q_0(x)$

for $k \in 1, \dots, t-1$ **do**

$M_k = \arg\min_{x \in M_{k-1}} Q_k(x)$

return some $x^* \in M_{t-1}$

Algorithm 6: Generic hierarchical optimization algorithm introduced in [114]. The sequence of continuous functions $Q_k : \mathbb{R}^m \to \mathbb{R}$ is sorted according to the associated scales λ_k (Eq. 3.23).

Then, if any sequence $(x_p)_{p>0}$ of minimizers x_p of $Q^p(x)$ is bounded (i.e. there exists $C > 0$ such that for all $p > 0$, $||x_p||_\infty \leq C$), the sequence is convergent, up to taking a subsequence, toward a point of M_{t-1}, which is the set of minimizers recursively defined in Algorithm 6.

Proof. See [114] (Theorem 3.3). □

We now show that the Mutex Watershed algorithm can be seen as a special case of the generic hierarchical Algorithm 6, for a specific choice of scales λ_k and functions $Q_k(x) : \mathbb{R}^m \to \mathbb{R}$ (see definitions (3.25, 3.26) below) .

Scales λ_k: Let $\tilde{w}_k$ be the signed edge weights $w : E \to \mathbb{R}$ ordered by decreasing absolute value $|\tilde{w}_1| > |\tilde{w}_2| > \dots > |\tilde{w}_{t-1}|$. If two edges share the same weight, then the weight is called $\tilde{w}_k$ for both and $E_k \subseteq E$ denotes the set of all edges with weight $\tilde{w}_k$. We then define *the scales* λ_k as

$$\lambda_k := \begin{cases} 1 & \text{if } k = 0 \\ \left| \frac{\tilde{w}_k}{2\tilde{w}_1} \right| & \text{otherwise.} \end{cases} \tag{3.25}$$

The continuous functions $Q_k(x) : \mathbb{R}^{|E|} \to \mathbb{R}$ are defined as follows

$$Q_k(x) := \begin{cases} |E| \cdot \min_{x' \in \mathsf{ISC}(\mathcal{G}, w)} ||x' - x|| & \text{if } k = 0 \\ \sum_{e \in E_k} x_e & \text{otherwise,} \end{cases} \tag{3.26}$$

where $\mathsf{ISC}(\mathcal{G}, w)$ is defined as:

$$\mathsf{ISC}(\mathcal{G}, w) := \mathsf{SC}(\mathcal{G}, w) \cap \{0, 1\}^{|E|}. \tag{3.27}$$

In words, $Q_0(x)$ is proportional to the distance between x and the closest point on the set $\mathsf{ISC}(\mathcal{G}, w)$, whereas $Q_k(x)$ depends only on the indicators x_e of edges in E_k, for $k > 0$.

Algorithm 7 is obtained by substituting the scales λ_k and functions $Q_k(x)$ (respectively defined in Eq. (3.25) and (3.26)) into Algorithm 6 . The algorithm starts by setting M_0 to $\mathsf{ISC}(\mathcal{G}, w)$, i.e. by restricting the space of the solutions only to integer edge labelings x that do not include any conflicted cycles. Then, in the following iterations $k \in 1, \dots, t-1$, the algorithm solves a series of minimization sub-problems that in the most general case are NP-hard, even though they involve a smaller set of edges $E_k \subseteq E$. Nevertheless, if we assume that all weights are distinct, then $|E_k| = 1$ for all k and the solution to the sub-problems amounts to checking if the new edge can be labeled with $x_e = 0$ without introducing any conflicted cycles. This procedure is identical to Algorithm 3: at every iteration, the Mutex Watershed tries to add an edge to the active set A, provided that no mutual exclusion constraints are violated.

In summary, using the framework in [114] allows generalizing the Mutex Watershed Algorithm to graphs with tied edge weights. In practice, when edge weights are estimated by a CNN, we do not expect tied edge weights.

PWSMWS$(Q_0, \dots, Q_{t-1})$**:**
$M_0 = \arg\min_{x \in \mathbb{R}^{|E|}} Q_0(x) = \mathsf{ISC}(\mathcal{G}, w)$
for $k \in 1, \dots, t-1$ **do**
  $M_k = \arg\min_{x \in M_{k-1}} \sum_{e \in E_k} x_e$
end
return some $x^* \in M_{t-1}$

Algorithm 7: Special case of the general hierarchical Algorithm 6 obtained by substituting Def. (3.25) and (3.26). With the additional assumption of unique signed weights $w : E \to \mathbb{R}$, this algorithm is equivalent to the Mutex Watershed Algorithm 4. The sequence of functions $Q_k : \mathbb{R}^m \to \mathbb{R}$ defined in Eq. 3.26 is sorted according to the associated scales λ_k in Eq. 3.25. $\mathsf{ISC}(\mathcal{G}, w)$ is defined in Eq. 3.27

Convergence of the sequence of minimizers

In this section, we see how Theorem 3.4.4 also suggests a minimization problem that is solved by the Mutex Watershed algorithm. A short summary is given in the final paragraph of the section.
First, we make sure that the conditions of Theorem 3.4.4 are satisfied when we apply it to Algorithm 7:

Lemma 3.4.5. *Let us consider the scales λ_k and continuous functions $Q_k(x) : \mathbb{R}^{|E|} \to \mathbb{R}$ respectively defined in Eq. (3.25) and (3.26). For any value of $p \in \mathbb{N}^+$, let $x_p \in \mathbb{R}^{|E|}$ be a minimizer of the function $Q^p(x)$ defined in Eq. (3.24). Then, the minimizer x_p lies in the set $\mathsf{ISC}(\mathcal{G}, w)$. From this, it follows that any sequence of minimizers $(x_p)_{p>0}$ is bounded and the conditions of Theorem 3.4.4 are satisfied.*

Proof. See Appendix A.1. □

Then, given any $p \in \mathbb{N}^+$ and the Def. (3.25, 3.26), we have that the minimization of the function $Q^p(x)$ defined in Eq. (3.24) is given by the following problem:

$$\begin{aligned}
\operatorname*{arg\,min}_{x \in \mathbb{R}^m} Q^p(x) &= \operatorname*{arg\,min}_{x \in \mathbb{R}^m} \sum_{0 \leq k < t} \lambda_k^p Q_k(x) && (3.28)\\
&= \operatorname*{arg\,min}_{x \in \mathsf{ISC}(\mathcal{G}, w)} \sum_{1 \leq k < t} \left| \frac{\tilde{w}_k}{2\tilde{w}_1} \right|^p \sum_{e \in E_k} x_e && (3.29)\\
&= \operatorname*{arg\,min}_{x \in \mathsf{ISC}(\mathcal{G}, w)} \frac{1}{|2\tilde{w}_1|^p} \sum_{e \in E} |w_e|^p \; x_e && (3.30)
\end{aligned}$$

where we used Lemma 3.4.5 and restricted the domain of the $\arg\min$ operation to $\mathsf{ISC}(\mathcal{G}, w)$, so that $Q_0(x) = 0$ for all $x \in \mathsf{ISC}(\mathcal{G}, w)$.

It follows from Lemma 3.4.5 and Theorem 3.4.4 that a sequence of minimizers $(x_p)_{p>0}$ of the problem (3.30) converge, up to taking a subsequence, to the solution x^* returned by Algorithm 7. More specifically, we know that any minimizer x_p of (3.30) is in the discrete set $\mathsf{ISC}(\mathcal{G}, w)$. Hence, the convergent sequence of minimizers $(x_p)_{p>0}$ eventually becomes constant and there exists a $p' \in \mathbb{N}^+$ large enough such that $x_p = x^*$ for all $p \geq p'$. In other words, in the case of unique weights and $p \geq p'$ large enough, the solution x^* of the Mutex Watershed Algorithm 7 solves the problem (3.30), which is just a rescaled version of the Mutex Watershed Objective we introduced in Sec. 3.4.2.

To summarize, we used the extended Power Watershed framework to show that the Mutex Watershed provides a solution to the minimization problem in Eq. (3.30) for p large enough. In particular, this problem suggested by the Power Watershed framework is the same one previously derived in Sec. 3.4.2 by linking the Mutex Watershed Algorithm to the multicut optimization problem.

3.5 Experiments

We evaluate the Mutex Watershed on the challenging task of neuron segmentation in electron microscopy (EM) image volumes. This application is of key interest in connectomics, a field of neuro-science that strives to reconstruct neural wiring digrams spanning complete central nervous systems. The task requires segmentation of neurons from electron microscopy images of neural tissue – a challenging endeavor, since segmentation has to be based only on boundary information (cell membranes) and some of the boundaries are not very pronounced. Besides, cells contain membrane-bound organelles, which have to be suppressed in the segmentation. Some of the neuron protrusions are very thin, but all of those need to be preserved in the segmentation to arrive at the correct connectivity graph. While a lot of progress is being made, currently only manual tracing or proof-reading yields sufficient accuracy for correct circuit reconstruction [139].

We validate the Mutex Watershed algorithm on the most popular neural segmentation challenge: ISBI2012 [12]. We estimate the edge weights using a CNN as described in Section 3.5.1 and compare with other entries in the leaderboard as well as with other popular post-processing methods for the same network predictions in Section 3.5.2.

3.5.1 Estimating edge weights with a CNN

The common first step to EM segmentation is to predict which pixels belong to a cell membrane using a CNN. Different post-processing methods are then used to obtain a segmentation, see Section 3.2 for an overview of such methods. The CNN can either be trained to predict boundary pixels [20, 34] or undirected affinities [49, 89] which express how likely it is for a pixel to belong to a different cell than its neighbors in the 6-neighborhood. In this case, the output of the network contains three channels, corresponding to left, down and next imaging plane neighbors in 3D. The affinities do not have to be limited to immediate neighbors – in fact, [89] have shown that introduction of long-range affinities is beneficial for the final segmentation even if they are only used to train the network. Building on the work of [89], we train a CNN to predict short- and long-range affinities and then use those directly as weights for the Mutex Watershed algorithm.

We estimate the affinities / edge weights for the neighborhood structure shown in Figure 3.6. To that end, we define local attractive and long-range repulsive edges. When attractive edges are only short-range, the solution will consist of spatially connected segments that cannot comprise “air bridges”. This holds true for both (lifted) multicut and for Mutex Watershed. We use a different pattern for in-plane and between-plane edges due to the great anisotropy of the data set. In more detail, we pick a sparse ring of in-plane repulsive edges and additional longer-range in-plane edges which are necessary to split regions reliably (see Figure 3.6). We

also added connections to the indirect neighbors in the lower adjacent slice to ensure correct 3D connectivity (see Figure 3.6). In our experiments, we pick a subset of repulsive edges, by using strides of 2 in the XY-plane in order to avoid artifacts caused by occasional very thick membranes. Note that the stride is not applied to local (attractive) edges, but only to long-range (repulsive) edges. The particular pattern used was selected after inspecting the size of typical regions. The specific pattern is the only one we have tried and was *not* optimized over.

In total, C^+ attractive and C^- repulsive edges are defined for each pixel, resulting in $C^+ + C^-$ output channels in the network. We partition the set of attractive / repulsive edges into subsets H^+ and H^- that contain all edges at a specific offset: $E^+ = \bigcup_{c=1}^{C^+} H_c^+$ for attractive edges, with H^- defined analogously. Each element of the subsets H_c^+ and H_c^- corresponds to a specific channel predicted by the network. We further assume that weights take values in $[0, 1]$.

Network architecture and training

We use the 3D U-Net [33, 130] architecture, as proposed in [49].

Our training targets for attractive / repulsive edges $\overset{*}{w}{}^{\pm}$ can be derived from a groundtruth label image $\overset{*}{L}$ according to

$$\overset{*}{w}{}^{+}_{e=(i,j)} = \begin{cases} 1, & \text{if } \overset{*}{L}_i = \overset{*}{L}_j \\ 0, & \text{otherwise} \end{cases} \tag{3.31}$$

$$\overset{*}{w}{}^{-}_{e=(i,j)} = \begin{cases} 0, & \text{if } \overset{*}{L}_i = \overset{*}{L}_j \\ 1, & \text{otherwise} \end{cases} \tag{3.32}$$

Here, i and j are the indices of vertices / image pixels. Next, we define the loss terms

$$\mathcal{J}_c^+ = -\frac{\sum_{e \in H_c^+} (1 - w_e^+)(1 - \overset{*}{w}{}_e^+)}{\sum_{e \in H_c^+} ((1 - w_e^+)^2 + (1 - \overset{*}{w}{}_e^+)^2)} \tag{3.33}$$

$$\mathcal{J}_c^- = -\frac{\sum_{e \in H_c^-} w_e^- \overset{*}{w}{}_e^-}{\sum_{e \in H_c^-} ((w_e^-)^2 + (\overset{*}{w}{}_e^-)^2)} \tag{3.34}$$

for attractive edges (i.e. channels) and repulsive edges (i.e. channels).

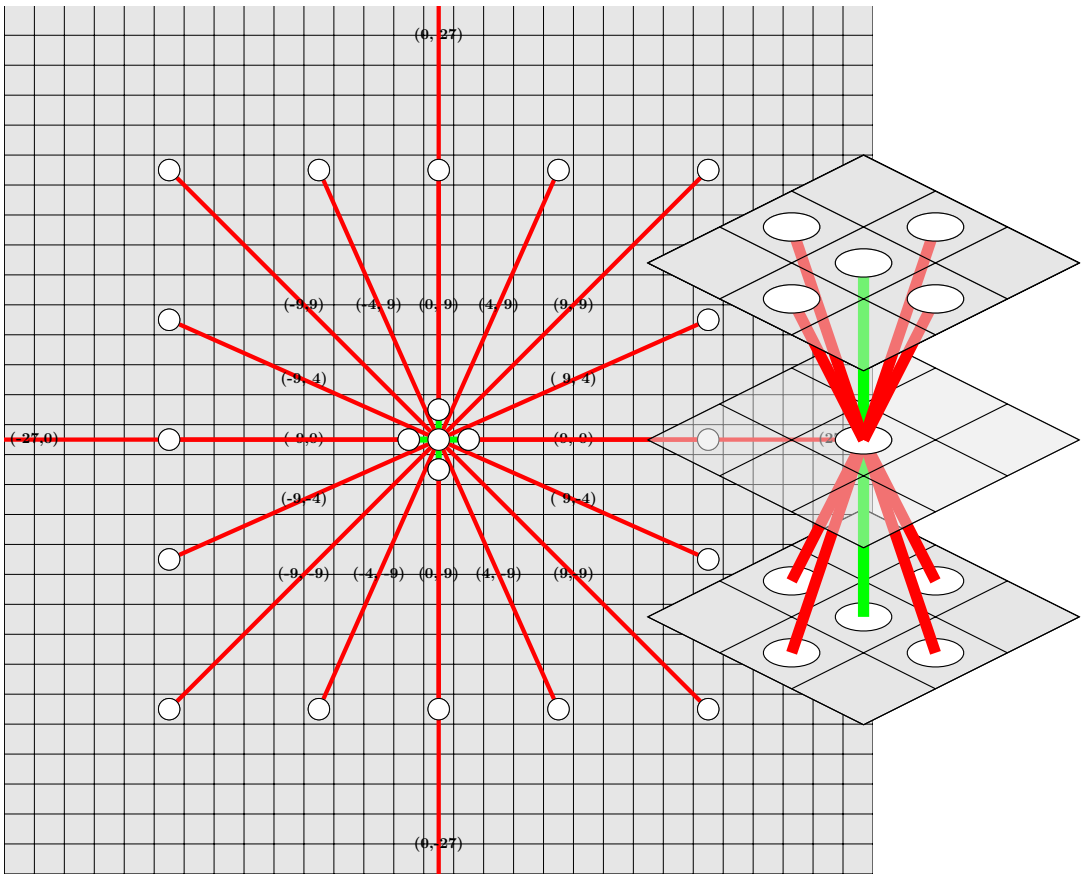


Figure 3.6: Local neighborhood structure of attractive (green) and repulsive (red) edges in the Mutex Watershed graph.

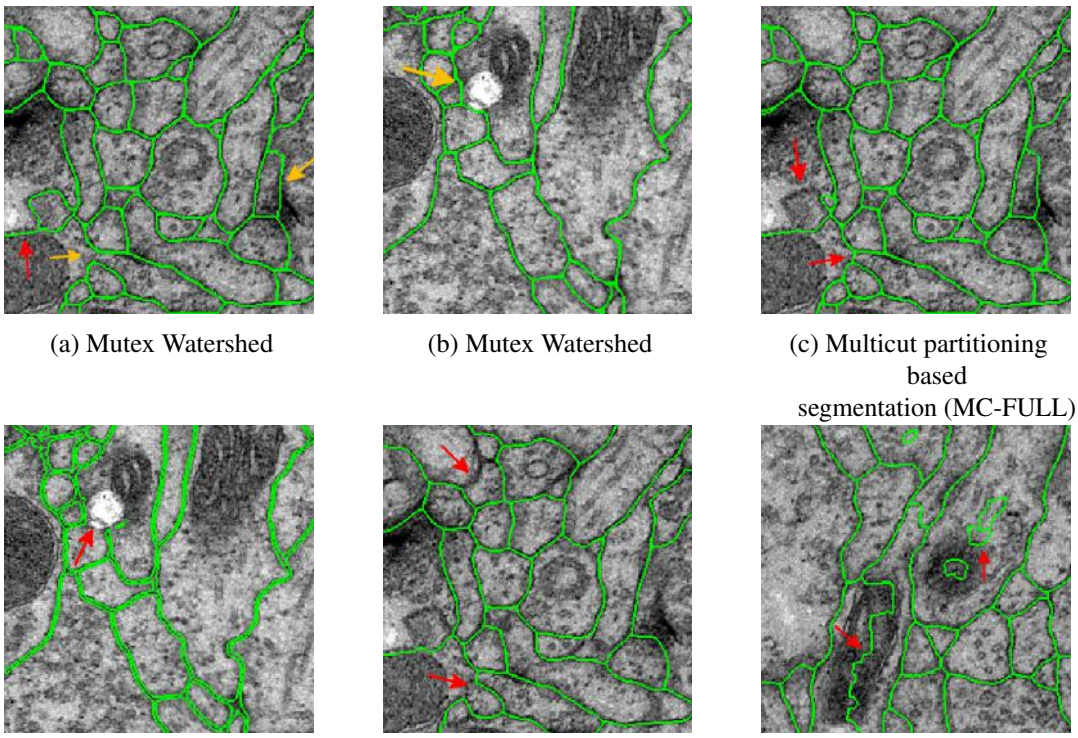
Equation 3.33 is the Sørensen-Dice coefficient [42, 145] formulated for fuzzy set membership values. During training we minimize the sum of attractive and repulsive loss terms $\mathcal{J} = \sum_c^{C^+} \mathcal{J}_c^+ + \sum_c^{C^-} \mathcal{J}_c^-$. This corresponds to summing up the channel-wise Sørensen-Dice loss. The terms of this loss are robust against prediction and / or target sparsity, a desirable quality for neuron segmentation: since membranes are locally two-dimensional and thin, they occupy very few pixels in three-dimensional the volume. More precisely, if w_e^+ or $\overset{*}{w}{}_e^+$ (or both) are sparse, we can expect the denominator $\sum_e((w_e^+)^2 + (\overset{*}{w}{}_e^+)^2)$ to be small, which has the effect that the numerator is adaptively weighted higher. In this sense, the Sørensen-Dice loss at every pixel i is conditioned on the global image statistics, which is not the case for a Hamming-distance

based loss like Binary Cross-Entropy or Mean Squared Error.

We optimize this loss using the Adam optimizer [68] and additionally condition learning rate decay on the Adapted Rand Score [12] computed on the training set every 100 iterations. During training, we augment the data set by performing in-plane rotations by multiples of 90 degrees, flips along the X- and Y-axis as well as elastic deformations. At prediction time, we use test time data augmentation, presenting the network with seven different versions of the input obtained by a combination of rotations by a multiple of 90 degrees, axis-aligned flips and transpositions. The network predictions are then inverse-transformed to correspond to the original image, and the results averaged.

3.5.2 ISBI Challenge

The ISBI 2012 EM Segmentation Challenge [12] is the neuron segmentation challenge with the largest number of competing entries. The challenge data contains two volumes of dimensions $1.5 \times 2 \times 2$ microns and has a resolution of $50 \times 4 \times 4$ nm per pixel. The groundtruth is provided as binary membrane labels, which can easily be converted to a 2D, but not 3D segmentation. To train a 3D model, we follow the procedure described in [20].



(a) Mutex Watershed

(b) Mutex Watershed

(c) Multicut partitioning based segmentation (MC-FULL)

(d) Thresholding of local boundary maps (THRESH)

(e) Watershed, seeded at local minima of the smoothed input map (WS)

(f) Distance Transform Watershed (WSDT)

Figure 3.7: Mutex Watershed and baseline segmentation algorithms applied on the ISBI Challenge test data. Red arrows point out major errors. Orange arrows point to difficult, but correctly segmented regions. All methods share the same input maps.

The test volume has private groundtruth; results can be submitted to the leaderboard. They are evaluated based on the Adapted Rand Score (Rand-Score) and the Variation of Information Score (VI-Score) [12].

Our method holds the top entry in the challenge's leader board[5] at the time of submission, see Table 3.1a. This is especially remarkable insofar as it is simpler than the methods holding the other top entries. Three out of four rely on a CNN to predict boundary locations and postprocess its output with the complex pipeline described in [20]. This post-processing first generates superpixels via distance transform watersheds. Then it computes a merge cost for local and long-range connections between superpixels. Based on this, it defines a lifted multicut

[5] http://brainiac2.mit.edu/isbi_challenge/leaders-board-new, the leader-board of the ISBI segmentation challenge

partioning problem that is solved approximately. In contrast, our method operates purely on the pixel level and does not involve a NP-hard partioning step.

Comparison with other segmentation methods

The weights predicted by the CNN described above can be post-processed directly by the Mutex Watershed algorithm. To ensure a fair comparison, we transform the same CNN predictions into a segmentation using basic and state-of-the-art post-processing methods. We start from simple thresholding (THRESH) and seeded watershed. Since these cannot take long-range repulsions into account, we generate a boundary map by taking the maximum[6] values over the attractive edge channels. Based on this boundary map, we introduce seeds at the local minima (WS) and at the maxima of the smoothed distance transform (WSDT). For both variants, the degree of smoothing was optimized such that each region receives as few seeds as possible, without however causing severe under-segmentation. The performance of these three baseline methods in comparison to Mutex Watershed is summarized in Table 3.1b. The methods were applied only in 2D, because the high degree of anisotropy leads to inferior results when applied in 3D. In contrast, the Mutex Watershed can be applied in 3D out of the box and yields significantly better 2D segmentation scores.

Qualitatively, we show patches of results in Figure 3.7. The major failure case for WS (Figure 3.7e) and WSDT (Figure 3.7f) is over-segmentation caused by over-seeding a region. The major failure case for THRESH is under-segmentation due to week boundary evidence (see Figure 3.7d). In contrast, the Mutex Watershed produces a better segmentation, only causing minor over-segmentation (see Figure 3.7a, Figure 3.7b).

Note that, in contrast to most pixel-based postprocessing methods, our algorithm can take long range predictions into account. To compare with methods which share this property, we turn to the multicut and lifted multicut-based partitioning for neuron segmentations as introduced in [8] and [56]. As proposed in [7], we compute costs corresponding to edge cuts from the affinities estimated by the CNN via:

$$s_e = \begin{cases} \log \frac{w_e^+}{1-w_e^+}, & \text{if } e \in E^+ \\ \log \frac{1-w_e^-}{w_e^-}, & \text{otherwise,} \end{cases} \tag{3.35}$$

We set up two multicut problems: the first is induced only by the short-range edges (MC-LOCAL), the other by short- and long-range edges together (MC-FULL). Note that the solution to the full connectivity problem can contain “air bridges”, i.e. pixels that are connected only by long-range edges, without a path along the local edges connecting them. However, we found

[6] The maximum is chosen to preserve boundaries.

this not to be a problem in practice. In addition, we set up a lifted multicut (LMC) problem from the same edge costs.

Both problems are NP-hard, hence it is not feasible to solve them exactly on large grid graphs. For our experiments, we use the approximate Kernighan Lin [64, 65] solver. Even this allows us to only solve individual 2D problems at a time. The results for MC-LOCAL and MC-FULL can be found in Table 3.1b. The MC-LOCAL approach scores poorly because it under-segments heavily. This observation emphasizes the importance of incorporating the longer-range edges. The MC-FULL and LMC approaches perform well. Somewhat surprisingly, the Mutex Watershed yields a better segmentation still, despite being much cheaper in inference. We note that both MC-FULL, LMC and the Mutex Watershed are evaluated on the same long-range affinity maps (i.e. generated by the same CNN with the same set of weights).

3.6 Conclusion

We have presented a fast algorithm for the clustering of graphs with both attractive and repulsive edges. The ability to consider both gives a valid alternative to other popular graph partitioning algorithms that rely on a stopping criterion or seeds. The proposed method has low computational complexity in imitation of its close relative, Kruskal's algorithm. We have shown which objective this algorithm optimizes exactly, and that this objective emerges as a specific case of the multicut objective. It is possible that recent interesting work [85] on partial optimal solutions may open an avenue for an alternative proof.

Finally, we have found that the proposed algorithm, when presented with informative edge costs from a good neural network, outperforms all known methods on a competitive bioimage partitioning benchmark, including methods that operate on the very same network predictions.

Method	Rand-Score	VI-Score
UNet + MWS	**0.98792**	**0.99183**
ResNet + LMC [167]	0.98788	0.99072
SCN + LMC [160]	0.98680	0.99144
M2FCN-MFA [141]	0.98383	0.98981
FusionNet + LMC [126]	0.98365	0.99130

(a) Top five entries at time of submission. Our Mutex Watershed (MWS) is state-of-the-art without relying on the complex lifted multicut postprocessing used by most other top entries.

Method	Rand-Score	VI-Score	Time [s]
MWS	**0.98792**	**0.99183**	43.3
MC-FULL	0.98029	0.99044	9415.8
LMC	0.97990	0.99007	966.0
THRESH	0.91435	0.96961	0.2
WSDT	0.88336	0.96312	4.4
MC-LOCAL	0.70990	0.86874	1410.7
WS	0.63958	0.89237	4.9

(b) Comparison to other segmentation strategies, all of which are based on our CNN.

Table 3.1: Results on the ISBI 2012 EM Segmentation Challenge.

4 Semantic Mutex Watershed

In this chapter[1], we present our approach for joint graph partitioning and labeling to adress the problem of semantic instance segmentation.

The development of our algorithm was motivated by the connection of Mutex Watershed and multicut (see Chapter 3) that suggested the existence of a similar, efficient algorithm connected to the Asymmetric Multiway Cut. We propose an extension to the Mutex Watershed, that we call Semantic Mutex Watershed (SMWS) and show that it optimizes an objective, which is connected to the Asymmetric Multiway Cut. Furthermore, we apply the SWMS on the Cityscapes dataset (2D urban scenes) and on 3D microscopy volumes. We show that our technique, combined with the use of current deep neural networks, outperforms the strong baseline of 'Panoptic Feature Pyramid Networks' by Kirillov et al. [71]. In the special case of 3D electron microscopy images, we show explicitly that our joint formulation outperforms a separate optimization of the partition problem and labeling problem.

4.1 Introduction

Image segmentation literature distinguishes *semantic segmentation* - associating each pixel with a class label - and *instance segmentation*, i.e. detecting and segmenting individual objects while ignoring the background. The joint task of simultaneously assigning a class label to each pixel and grouping pixels to instances has been addressed under different names, including semantic instance segmentation, scene parsing [148], image parsing [150], holistic scene understanding [171] or instance-separating semantic segmentation [93]. Recently, a new metric and evaluation approach to such problems has been introduced under the name of *panoptic segmentation* [72].

From a graph theory perspective, semantic instance segmentation corresponds to the simultaneous partitioning and labeling of a graph. Most greedy graph partitioning algorithms are defined on graphs encoding attractive interactions only. Clusters are then formed through

[1] This chapter is based on our paper [164] which was published in 2019. The results in this chapter represent the current state at the time of publication. The training of the deep neural network used for the experiments of this chapter has been carried out by Yuyan Li (2D urban scenes) and Constantin Pape (3D electron microscopy images). Yuyan additionally aided in the implementation and characterization of the SMWS.

agglomeration or division until a user-defined termination criterion is met (often a threshold or a desired number of clusters). These algorithms perform pure instance segmentation. The semantic labels for the segmented instances need to be generated independently.

If repulsive - as well as attractive - forces are defined between the nodes of the graph, partitioning can be formulated as a Multicut problem [5]. In this formulation clusters emerge naturally without the need for a termination criterion. Furthermore, the Multicut problem can be extended to include the labeling of the graph, delivering a semantic instance segmentation from a joint optimization of partitioning and labeling [81].

We propose to solve the joint partitioning and labeling problem by an efficient algorithm which we term Semantic Mutex Watershed (SMWS), inspired by the Mutex Watershed [165]. In more detail, in this contribution we:

- propose a fast algorithm for joint graph partitioning and labeling
- prove that the algorithm minimizes (exactly) an objective function closely related to the Asymmetric Multiway Cut objective
- demonstrate competitive performance on natural and biological images.

4.2 Related Work

Semantic segmentation. State-of-the-art semantic segmentation algorithms are based on convolutional neural networks (CNNs) which are trained end-to-end. The networks commonly follow the design principles of image classification networks (*e.g.* [53, 80, 143]), replacing the fully connected layers at the end with convolutional layers to form a fully convolutional network [99]. This architecture can be further extended to include encoder-decoder paths [131], dilated or atrous convolutions [28, 175] and pyramid pooling modules [29, 182].

Instance segmentation. Many instance segmentation methods use a detection or a region proposal framework as their basis; object segmentation masks are then predicted inside region proposals. A cascade of multiple networks is employed by [40], each solving a specific subtask to find the instance labeling. Mask-RCNN [54] builds on the bounding box prediction capabilities of Faster-RCNN [127] to simultaneously produce masks and class predictions. An extension of this method with an additional semantic segmentation branch has been proposed in [71] as a single network for semantic instance segmentation.

In contrast to the region-based methods, proposal-free algorithms often start with a pixel-wise representation which is then clustered into instances [46, 78, 174]. Alternatively, the distance transform of instance masks can be predicted and clustered by thresholding [15].

Graph-based segmentation. Graph-based methods, used independently or in combination with machine learning on pixels, form another popular basis for image segmentation algorithms [47]. In this case, the graph is built from pixels or superpixels of the image and the instance segmentation problem is formulated as graph partitioning. When the number of instances is not known in advance and repulsive interactions are present between the graph nodes, graph partitioning can in turn be formulated as a Multicut or correlation clustering problem [5]. This NP-hard problem can be solved reasonably fast for small problem sizes with integer linear programming solvers [7] or approximate algorithms [20, 119]. A modified Multicut objective is introduced by [165] together with the Mutex Watershed - an efficient clustering algorithm for its optimization.

The Multicut objective can be extended to solve a joint graph partitioning and labeling problem [62, 81] for simultaneous instance and semantic segmentation. In practice, the computational complexity of the joint problem only allows for approximate solutions [93], possibly combined with reducing the problem size by over-segmentation into superpixels. This formulation has been applied to natural images by [70] and to biological images by [79].

Similar to the semantic segmentation use case, CNNs can be used to predict pixel and superpixel affinities which serve as edge weights in the graph partitioning problem [89, 98, 100].

4.3 The Semantic Mutex Watershed

In this section, we introduce an extension to the Mutex Watershed algorithm for semantic instance segmentation. Similar to instance segmentation algorithms discussed in Chapter 3, we build a graph of image pixels (voxels) or superpixels and formulate the semantic instance segmentation problem as the joint partitioning and labeling of a graph.

Weighted graph with terminal nodes. To partition an undirected weighted graph $G = G(V, E, w)$ graph into instances Wolf et al. [165] differentiate between *attractive edges* $E^+ = \{e \in E \,|\, w_e >= 0\}$ and *repulsive edges* $E^- = \{e \in E \,|\, w_e < 0\}$. In other words, the weights encode the attraction and repulsion between the incident nodes of each edge. Since we will augment this graph with additional nodes, we will refer to V as *internal nodes* and edges $E = E^+ \cup E^-$ as *internal edges*.

Semantic instance segmentation can be achieved by clustering the internal nodes and assigning a semantic label $l \in \{l_0, ..., l_k\}$ to each cluster. We extend G by k *terminal* nodes $\{t_0, ..., t_k\} \in T$ where each t_i is associated with a label l_i. Every internal node $v \in V$ is connected to every t by a weighted *semantic edge* $e \in E^S$. Here, a large *semantic weight* $w_{ut} \subseteq \mathbb{R}^+$ implies a strong association of internal node u with the label of the terminal node

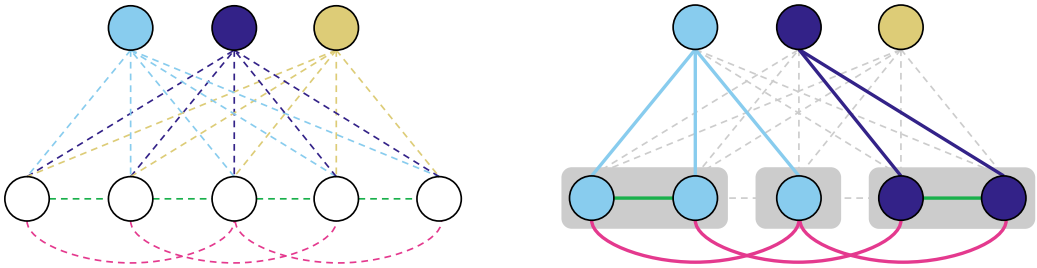


Figure 4.1: *Left:* An example of an extended graph. Nodes on the top are terminal nodes whereby each color represents a label class. The associated semantic edges are colored correspondingly. The internal nodes are on the bottom with attractive (green) and repulsive (red) edges between them. *Right:* Semantic instance segmentation. Edges that are part of the active set are shown in bold. Clusters are depicted in grey. Note that two adjacent nodes with the same label are not necessarily clustered together.

t. The extended graph thus becomes $G'(V', E', W')$ with $V' = V \cup T,\ E' = E \cup E^S$ and $W' = W \cup W^S$. Figure 4.1 shows an example of such an extended graph.

4.3.1 The Semantic Mutex Watershed Algorithm.

We will now extend the Mutex Watershed Algorithm to the extended graph G' for joint graph partitioning and labeling. The algorithm finds a clustering and label assignment described by a set of *active* edges: $A \subseteq E'$ where $A^+ := A \cap E^+$, $A^- := A \cap E^-$ and $A^S := A \cap E^S$ encode clusters, mutual exclusions and label assignments, respectively. In order to restrict A to a consistent partitioning and labeling we will make the following definitions:

We define two internal nodes $i, j \in V$ as connected if they are connected by active attractive edges, i.e.

$$\forall i, j \in V : \tag{4.1}$$

$$\Pi_{i \to j} = \{\text{paths } \pi \text{ from } i \text{ to } j \text{ with } \pi \subseteq E'\} \tag{4.2}$$

$$\text{connected}(i, j; A^+) \Leftrightarrow \exists \text{ path } \pi \in \Pi_{i \to j} \text{ with } \pi \subseteq A^+ \tag{4.3}$$

$$\text{cluster}(i; A^+) = \{i\} \cup \{j : \text{connected}(i, j; A^+)\} \tag{4.4}$$

Semantic Mutex Watershed:

SMWS$\big(\mathcal{G}(V, E')$**,** $w : E' \to \mathbb{R}$**,** boolean connect_all$\big)$**:**

$A^+ \leftarrow \emptyset$; $A^- \leftarrow \emptyset$

for $(i, j) = e \in E'$ *in descending order of* $|w_e|$ **do**

  if $e \in E^+$ **then**

    if *not* *mutex*$(i, j; A^+, A^-)$

    and not *differentclass*(i, j, A^+, A^S) **then**

      if not connected$(i, j; A^+)$ **or** connect_all **then**

        merge(i, j): $A^+ \leftarrow A^+ \cup e$

        ▷ merge i and j and inherit the mutex constraints of the parent clusters

  else if $e \in E^-$ **then**

    if not connected$(i, j; A^+)$ **then**

      addmutex(i, j): $A^- \leftarrow A^- \cup e$

      ▷ add mutex constraint between i and j

  else if $e \in E^S$ **then**

    if *class*$(i, A^+, A^S) = \emptyset$ **or** *class*$(i, A^+, A^S) = l_j$ **then**

      assignLabel(i, j): $A \leftarrow A \cup e$

return A

Algorithm 8: The Semantic Mutex Watershed algorithm. The differences to the Mutex Watershed are marked in blue.

and the mutual exclusion between two nodes as

$$\begin{aligned} \text{mutex}(i, j; A^+, A^-) \quad \Leftrightarrow \quad & \exists\, e = (k, l) \in A^- \text{ with} && (4.5)\\ & k \in \text{cluster}(i; A^+) \text{ and} && (4.6)\\ & l \in \text{cluster}(j; A^+) \text{ and} && (4.7)\\ & \text{cluster}(i; A^+) \neq \text{cluster}(j; A^+) && (4.8) \end{aligned}$$

Two nodes are thus mutual exclusive if they are connected by a path from i to j with exactly one repulsive edge. Furthermore, a label l_j is assigned to a node i if this node is connected to the corresponding terminal node t_j by attractive and semantic edges:

$$\text{class}(i, A^+, A^S) = l_j \Leftrightarrow \exists\, \pi \in \Pi_{i \to j} \text{ with } \pi \subseteq A^+ \cup A^S. \tag{4.9}$$

For unlabeled nodes i, where $\text{class}(i, A^+, A^S) \neq c \quad \forall c \in \{l_0, ..., l_k\}$, we use the notation $\text{class}(i, A^+, A^S) = \emptyset$ and use it to define the following predicate

$$\begin{aligned} \text{differentclass}(i, j, A^+, A^S) \quad \Leftrightarrow \quad & \text{class}(i, A^+, A^S) \neq \text{class}(j, A^+, A^S) \text{ and} && (4.10) \\ & \text{class}(i, A^+, A^S) \neq \emptyset \text{ and} && (4.11) \\ & \text{class}(j, A^+, A^S) \neq \emptyset && (4.12) \end{aligned}$$

Algorithm. The Semantic Mutex Watershed algorithm is an extension of the Mutex Watershed algorithm introduced by Wolf et al. [165]. It augments the partitioning of the latter with a consistent labeling. The algorithm is shown in algorithm 8 with the additions to [165] highlighted. In the following we explain the syntax and procedure of the shown pseudocode.

All edges E' are considered to be added to the active set A. The decisions are made in descending order of their absolute edge-weights and depend on the type of each edge:
Attractive edges: The edge is added if the incident nodes are not mutual exclusive and not labeled differently.
Repulsive edges: The edge is added if the incident nodes are not connected.
Semantic edges: The edge is added if the node is either unlabeled or already has the same label as the edge's terminal node.

After following these rules, the set of attractive edges in the final set $A \cap E^+$ form clusters in the graph G, which are each connected to a single terminal node indicating the labeling. Figure 4.1(b) shows a simple example of such an active set. Note, that the Mutex Watershed algorithm is embedded in the Semantic Mutex Watershed for the special case when there are zero or one label ($|T| \in \{0, 1\}$).

Efficient Implementation with Maximum-Spanning-Trees. The SMWS is similar to the efficient Kruskal's maximum spanning tree algorithm [84] and can feasibly be applied to pixel-graphs of large images and even image volumes. Our implementation utilizes an efficient union-find data structure; mutex relations are realized through a hash table.

4.3.2 The Semantic Mutex Watershed Objective

The Semantic Mutex Watershed, introduced in the previous section, operates on a graph with semantic nodes identical to the Asymmetric Multiway Cut. In this section we prove that the Semantic Mutex Watershed optimizes a precise objective and show how it relates to the Asymmetric Multiway Cut objective. To this end, we will extend the proof of [165] to the Semantic Mutex Watershed. Let us first recall the definitions of *dominant powers* and *mutex constraints* in [165].

Dominant power. Let $\mathcal{G} = (V', E', w)$ be an edge-weighted graph, with unique weights $w : E' \to \mathbb{R}$. We call $p \in \mathbb{N}^+$ a dominant power if:

$$|w_e|^p > \sum_{t \in E',\, w_t < w_e} |w_t|^p \qquad \forall e \in E', \tag{4.13}$$

Note that there exists a dominant power for any finite set of edges, since for any $e \in E$ we can divide (4.13) by w_e^p and observe that the normalized weights w_s^p / w_e^p (and any finite sum of these weights) converges to 0 when p tends to infinity.

Conflicted cycles. (see Definition 3.4.1) We call a cycle of $\mathcal{G}$ conflicted w.r.t. $(\mathcal{G}, w)$ if it contains precisely one repulsive edge $e \in E^-$, s.t. $w_e < 0$. We denote by $\mathcal{C}^-(\mathcal{G}, w) \subseteq \mathcal{C}(\mathcal{G}, w)$ the set of all conflicted cycles. Furthermore, given a set of edges $A \subseteq E$, we denote by $\mathcal{C}^-(A, \mathcal{G}, w) \subseteq \mathcal{C}^-(\mathcal{G}, w)$ the set of conflicted cycles involving only edges in A. If there are no conflicted cycles $\mathcal{C}^-(G, A, w) = \emptyset$ then A implies a consistent graph partitioning[85]. In other words, it ensures that two nodes that are mutual exclusive can not be connected.

Furthermore, we define the set $\mathcal{P}(A)$ of all paths π that connect two distinct terminal nodes through attractive and semantic edges:

$$\mathcal{P}(A) := \{\, \pi \,|\, \pi \in \Pi_{t \to t'},\, \pi \in A \cap (E^+ \cup E^S),\, t, t' \in T\,, t \neq t' \,\} \tag{4.14}$$

The algorithm must never connect two terminal nodes through such a path, thus we define the **label constraint** $\mathcal{P}(A) = \emptyset$. This ensures the consistency between the partitioning and labeling.

Lemma 4.3.1 (Optimality of the Semantic Mutex Watershed).
Let $G' = (V', E', w) = (V \cup T, E \cup E^S, w)$ be an edge-weighted graph extended by terminal nodes T, with unique weights $w' : E' \to \mathbb{R}$, $w_t > 0 \,\forall\, t \in T$ and $p \in \mathbb{R}^+$ a dominant power. The edge indicator given by the Semantic Mutex Watershed algorithm 3

$$x^{\mathbf{SMWS}} := \mathbb{1}$$

is the optimal solution to the integer linear program

$$\begin{aligned}
&\underset{x \in \{0,1\}^{|E'|}}{\arg\min} \sum_{e \in E'} |w_e|^p x_e && (4.15)\\
\text{s.t.}\quad & \mathcal{C}^-(G, A, w) = \emptyset, && (4.16)\\
& \mathcal{P}(A) = \emptyset, && (4.17)\\
\text{with}\quad & A := \{\, e \in E \mid x_e = 0 \,\}. && (4.18)
\end{aligned}$$

Proof. This proof is completely analogous to Theorem 3.4.1 and even identical for $T = \emptyset$. The SMWS finds the optimal solution because it enjoys the properties *optimal substructure* and *greedy choice*. The proof of Theorem 3.4.1 showing optimal substructure does not rely on the specific constraints in the ILP. Thus it can also be applied with the additional constraint in eq. (4.17), giving the ILP eqs. (4.15) to (4.18) optimal substructure.

In every iteration the SMWS adds the feasible edge e with the largest weight to the active set. Due to the dominant power, its energy contribution is larger than for any combination of edges e' with $w'_e < w_e$. Thus, SMWS has the greedy choice property [36]. It follows by induction that the SMWS algorithm finds the globally optimal solution to the SMWS objective.

□

Relation to the Asymmetric Multiway Cut. To understand the relation of the Semantic Mutex Watershed to the Asymmetric Multiway Cut we will transform the SMWS problem (eqs. (4.15) to (4.18)) into an ILP with the same minimal energy solution as the Asymmetric Multiway Cut.

First, we identify the indicator variables x in eq. (4.15) with the AMWC indicators y in eq. (1.3). For attractive and semantic edges both indicators represent the same graph partitions and class assignments. In particular, given the associated indicators x and y of any graph partitioning and labeling, $x_e = y_e \quad \forall\, e \in E^+ \cup E^S$ holds. For repulsive edges $e^- \in E^-$ however, x_{e^-} indicates a mutex edge and therefore a necessary cut, hence $y_{e^-} = 1 - x_{e^-}$. Additionally, the Asymmetric Multiway Cut introduces repulsive edges between terminal nodes and constrains them to be always cut. In conclusion we can translate between both indicators with

$$y_e(x, e) = \begin{cases} x_e & \text{if } e \in E^+ \cup E^S \\ 1 - x_e & \text{if } e \in E^- \\ 1 & \text{if } e \in (T \times T) \end{cases} \tag{4.19}$$

Using eq. (4.19) we translate the SWMS objective eq. (4.15)

$$\sum_{e \in E'} |w_e|^p x_e = \sum_{e \in E^+} |w_e|^p y_e + \underbrace{\left(\sum_{e \in E^-} 1 \right)}_{\mathcal{L}_{\text{triv}}} - \sum_{e \in E^-} |w_e|^p y_e + \sum_{e \in E^S} |w_e|^p y_e \tag{4.20}$$

$$= \sum_{e \in E'} \text{sign}(w_e) |w_e|^p y_e + \mathcal{L}_{\text{triv}} \tag{4.21}$$

Note that the constant $\mathcal{L}_{\text{triv}}$ does not affect the minimum energy solution.

Second, we will add the constraints

$$\sum_{t \in T} y_{tv} = |T| - 1 \qquad \forall v \in V \tag{4.22}$$

$$y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'} \qquad \forall C \in \text{ cycles } (G) \forall e \in C \tag{4.23}$$

to the Semantic Mutex Watershed ILP eqs. (4.15) to (4.17) and observe, since $y(x^{\mathbf{SMWS}})$ always fulfills eqs. (4.22) and (4.23). Therefore, $y(x^{\mathbf{SMWS}})$ also minimizes eq. (4.15) subject to the tighter constraints eqs. (4.17), (4.18), (4.22) and (4.23). Using Equation (4.22) and Lemma B.1.1 (see Appendix section B.1) we can replace the path constraints eq. (4.17) by

$$\begin{aligned}
\mathcal{P}(A) = \emptyset \quad &\Leftrightarrow \quad \sum_{e \in P} y_e \geq 1 && \forall P \in \pi_{t \rightsquigarrow t'} \; \forall t, t' \in T, \, t \neq t' && (4.24)\\
&\Leftrightarrow \quad y_{ut} + y_{uv} + y_{vt'} \geq 1 && \forall (u, v) \in E \; \forall t, t' \in T, \, t \neq t' && (4.25)\\
&\Leftrightarrow \quad y_{tu} + y_{uv} \geq y_{tv}, && \forall uv \in E, t \in T && (4.26)\\
&\qquad\; y_{tv} + y_{uv} \geq y_{tu}, && \forall uv \in E, t \in T. && (4.27)
\end{aligned}$$

We conclude that $y(x^{\mathbf{SMWS}})$ minimizes the objective:

$$\begin{aligned}
&\arg\min_{y \in \{0,1\}^{|E'|}} \sum_{e \in E'} \text{sign}(w_e) |w_e|^p y_e && && (4.28)\\
\text{subject to} \quad &y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'} && \forall C \in \text{ cycles } (G) \forall e \in C && (4.29)\\
&\sum_{t \in T} y_{tv} = |T| - 1, && \text{if } T \neq \emptyset, \forall v \in V \setminus T && (4.30)\\
&y_{tt'} = 1, && \forall t, t' \in T, t \neq t' c, f && (4.31)\\
&y_{tu} + y_{uv} \geq y_{tv}, && \forall (u, v) \in E, t \in T && (4.32)\\
&y_{tv} + y_{uv} \geq y_{tu}, && \forall (u, v) \in E, t \in T && (4.33)
\end{aligned}$$

highlighting the close connection to the Asymmetric Mutiway Cut objective. In fact, although unlikely in practical applications, for graphs $\mathcal{G}'$ where $d = 1$ is a dominant power, the Semantic Mutex Watershed solves the Asymmetric Mutiway Cut to optimality.

4.4 Experiments

We will now demonstrate how to apply the SMWS algorithm to semantic instance segmentation of 2D and 3D images. We start from showing how existing CNNs can be used as graph weight

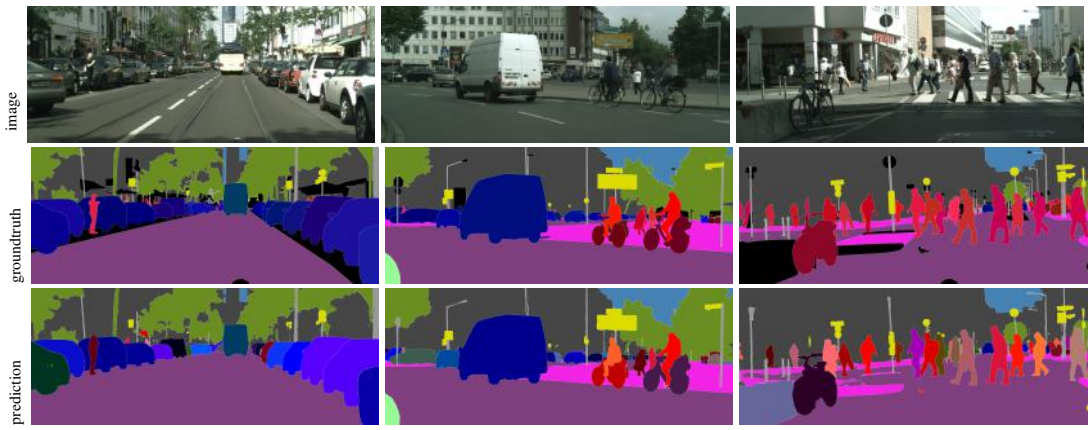


Figure 4.2: Semantic instance segmentation. Results on Cityscapes using semantic unaries (Deeplab 3+ network) and affinities derived from Mask-RCNN foreground probability. Colors indicate predicted semantic classes with color tone variations for separate instances.

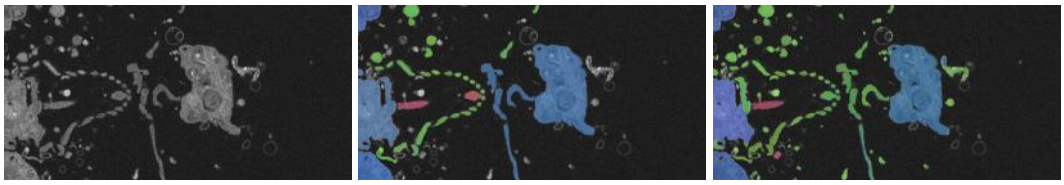


Figure 4.3: Results for the sponge dataset. From left to right: Raw data. Ground truth. Result of the Semantic Mutex Watershed. Cell-bodies are colored in blue, microvilli in green and flagella in red.

estimators and compare different sources of edge weights on the Cityscapes dataset. Additionally, we apply the SMWS algorithm to a 3D electron microscopy volume and demonstrate its efficiency and scalability.

4.4.1 Affinity Generation with Neural Networks

The only input to the SMWS are the graph weights; it does not require any hyperparamters such as thresholds. Consequently, its segmentation quality relies on good estimates of the graph weights $W' = W \cup W^S$. In this section we present how state-of-the-art CNNs can be used as sources for these weights.

Affinity Learning. Affinities are commonly used in instance segmentation; many modern algorithms train CNNs to directly predict pixel affinities. A universal approach is to employ a stencil pattern that describes for each pixel which neighbours to consider for the affinity computation. Regularly spaced, multi-scale stencil patterns are widely used for natural images [98, 100] and bio-medical data [89, 163].

The predicted affinities are usually in the interval $[0, 1]$ and can be interpreted as pseudo-probabilities. We use these affinities directly as weights for the attractive edges and invert them to get the repulsive edge weights.

Mask-RCNN produces overlapping masks that have to be resolved for a consistent panoptic segmentation. We achieve this with the SMWS by deriving affinities from the foreground probabilities of each mask. A straightforward approach is to compute the (attractive) affinity $a(i, j)$ of two pixels as their joint foreground probability, weighted by the classification score s: $a(i, j) = s\, p(i)\, p(j)$.

We find that sparse repulsive edges work well in practice, as they lead to faster inference and reduced over-segmentation on the instance boundaries. +For this reason, we sample random points from all pairs of masks and add (repulsive) edges with weight proportional to a soft intersection over union of two masks m and n:

$$w_{nm} = 1 - \frac{\sum_{q \in V} p_m(q) p_n(q)}{\sum_{q \in V} \max\left(p_m(q),\, p_n(q)\right)}. \tag{4.34}$$

Semantic Segmentation CNNs. State of the art CNNs [30, 182] achieve high quality results on semantic segmentation tasks. The output of the last softmax layer usually used in these networks can be interpreted as the normalized probability of each pixel belonging to each class. Thus, we can use these predictions directly as semantic weights W^S.

Additionally, we derive affinities from the stuff class probabilities; we treat each stuff class separately and again compute the affinity of two pixels as their joint probability of being in each stuff class c, i.e.: $a_c(i, j) = p_{\mathrm{c}}(i)\, p_{\mathrm{c}}(j)$. This cannot be done for thing classes since they can have multiple instances.

4.4.2 Panoptic Segmentation on Cityscapes

We apply the SMWS on the challenging task of panoptic segmentation on the Cityscapes dataset [35]. We illustrate how the different sources of affinities can be used and combined and show their different strengths and weaknesses.

MRCNN[54]		GMIS[98]		Deeplab[30]			Cityscapes		
att	rep	att	rep	att	rep	sem	PQ	PQ^{Th}	PQ^{St}
✓	✓					✓	59.3	50.6	65.7
	✓	✓				✓	58.6	48.8	65.7
		✓	✓			✓	56.1	42.8	65.7
✓	✓	✓	✓	✓	✓	✓	48.7	38.7	55.9
		✓	✓	✓	✓	✓	47.3	35.5	55.9
				✓	✓	✓	46.3	33.1	56.0

Table 4.1: Panoptic segmentation quality PQ of the SMWS on top of diverse sources of graph weights. We distinguish between attractive (att), repulsive (rep) and semantic (sem) graph weights extracted from the respective methods.

Dataset. The Cityscapes dataset consists of urban street scene images taken from a driver's perspective. It has 5k densely annotated images separated into train (2975), val (500) and test (1525) set. Since there is no public evaluation server for panoptic segmentation on the test set, we report all results on the validation set. There are 19 classes with 11 stuff classes and 8 thing classes.

Implementation Details. We employ and combine multiple sources of graph weights to build the SMWS graph. We train a Deeplab 3+ [30] network for semantic edge weight and affinities prediction following [98]. We employ the Mask-RCNN [54] implementation provided by [102] and train a model on Cityscapes following [54]'s training configuration. Further implementation details can be found in section B.2.1.

Study of Affinity Sources. We evaluate the semantic instance segmentation performance of the SMWS in terms of the "panoptic" metric using different combinations of the graph weight sources discussed above. In table 4.1 we compare the PQ metric on the Cityscapes dataset.

The best performance can be achieved with a combination of Mask-RCNN affinities and Deeplab 3+ for semantic predictions outperforming the strong baseline of [71] listed in table 4.2 and shown in fig. 4.2 and the supplementary fig. B.1. Through observations on the images, we find that Mask-RCNN affinities are more reliable in detecting small objects as well as in connecting fragmented instances. Note that PQ mostly measures detection quality which is

Cityscapes	PQ	PQ^{Th}	PQ^{St}
SMWS	**59.3**	50.6	**65.7**
PFPN[71]	58.1	**52.0**	62.5
DIN[13]	53.8	42.5	62.1
Sponge			
SMWS	**51.6**	**62.1**	20.0
MWS-MAX	48.1	56.2	**23.8**
CC_{sem}	43.4	55.6	06.7
CC_{aff}	24.3	27.7	13.9

Table 4.2: Comparison to other segmentation strategies in measuring the panoptic segmentation quality PQ.

then weighted by the segmentation quality of the found instances, hence the detection strength of the Mask-RCNN shines through.

We observe that using all sources together leads to a performance drop of 10 percentage points below the best result. We believe this is due to the greedy nature of the SMWS which selects the strongest of all provided edges. This example demonstrates how important it is to carefully select/train the algorithm input.

4.4.3 Semantic Instance Segmentation of 3D EM Volumes

Semantic instance segmentation is an important task in bio-medical image analysis where classes naturally arise through cellular structure. We use a 3D EM image dataset to compare the SMWS to algorithms that separately optimize instance segmentation and semantic class assignment.

Dataset. The data-set consists of two FIBSEM volumes of a sponge choanocye chamber. The data was acquired in [112] to investigate proto-neural cells in sponges using the segmentation approach introduced in [120]. These cells filter nutrients from water by creating a flow with the beating of a flagellum and absorbing the nutrients through microvilli that surround the flagellum in a collar [86] (see fig. 4.2). In order to investigate this process in detail, a precise semantic instance segmentation of the cell-bodies, flagella and microvilli is needed. The dataset consists of three EM image volumes of size $96 \times 896 \times 896$ pixel ($2 \times 18 \times 18$ μm).

Implementation Details. We predict affinities with two separate 3D U-Nets [33] to derive graph edge weights and semantic class probabilities respectively. We adopt the training procedure of [163] which uses the Dice Coefficient as the loss function. We use two volumes for training and one for testing.

We also implement baseline approaches which start from the same network predictions, but do not perform joint labeling and partitioning. First, we compare to instance segmentation with the Mutex Watershed, followed by assigning instances the semantic label of the strongest semantic edge (MWS-MAX). In addition, we compute connected components of the semantic predictions (CC_{sem}) and short-range affinities (CC_{aff}).

Results. The PQ values in table 4.2 show that the SMWS outperforms the baselines approaches that separately optimize instance segmentation and semantic class assignment. An additional analysis can be found in the appendix fig. B.2, where we measure the runtimes for different volume sizes and observe almost linear scaling behavior.

4.5 Conclusion

We have introduced a new method for joint partitioning and labeling of weighted graphs as a generalization of the Mutex Watershed algorithm. We have shown that it optimally solves an objective function closely related to the objective of the Asymmetric Multiway Cut problem. Our experiments demonstrate that SMWS with graph edge weights predicted by convolutional neural networks outperform strong baselines on natural and biological images. Any improvement in the CNN performance will translate directly to an improvement of the SMWS results. However, we also observe that the extreme value selection used by the SMWS to assign edges to the active set can lead to sub-optimal performance when diverse edge weights sources are combined. Empirically, the algorithm scales almost linearly with the number of graph edges N making it applicable to large images and volumes without prior over-segmentation into superpixels. The source code will be made available upon publication.

5 Self-Supervised Affinities

Labeling "ground truth" for biological medical image segmentation, especially cell segmentation and neuron reconstruction, is a challenging and labor-intensive task [152]. It is therefore of great value in practice if algorithms can provide meaningful segmentation results fully unsupervised, without the need correctly labeled training data. In Chapter 3 we have evaluated the segmentation quality of the Mutex Watershed algorithm in conjunction fully-supervised affinity networks. In this chapter[1], we present our approach to derive affinities in an unsupervised way. We observe that deep neural networks trained to inpaint partially occluded images show a deep understanding of image composition and have even been shown to remove objects from images convincingly. We investigate how this implicit knowledge of image composition can be leveraged for fully self-supervised instance separation. We propose a measure for the independence of two image regions given a fully self-supervised inpainting network and separate objects by maximizing this independence. We evaluate our method on two microscopy image datasets and show that it reaches comparable segmentation performance to fully supervised methods.

5.1 Motivation

Recent inpainting neural networks demonstrate a remarkable ability to remove distortions in natural images (*e.g.*, text overlays, watermarks, or pixel-wise independent noise) and are even able to entirely remove foreground objects (*e.g.*, a flagpole as demonstrated here). Trained on large datsets, these networks learn the statistics that underlie images in a way that goes well beyond low level features. In this work, we aim to leverage those learnt statistics to distinguish individual objects in images from each other, without any form of supervision.

In order to intuitively understand how these statistics can be used, let us consider a high-capacity inpainting network trained on a very large corpus of natural images and imagine the following scenario: Given the image of a busy street with a region in the center masked out to inpaint, such a network will be able to continue inpainting cars that are partially visible. If, however, the masked-out region is large enough to contain entire objects, the provided context will be uninformative about their location, shape, and texture and will therefore not be able to

[1] This chapter is based on our paper [161] which was published in 2020. The results in this chapter represent the current state at the time of publication.

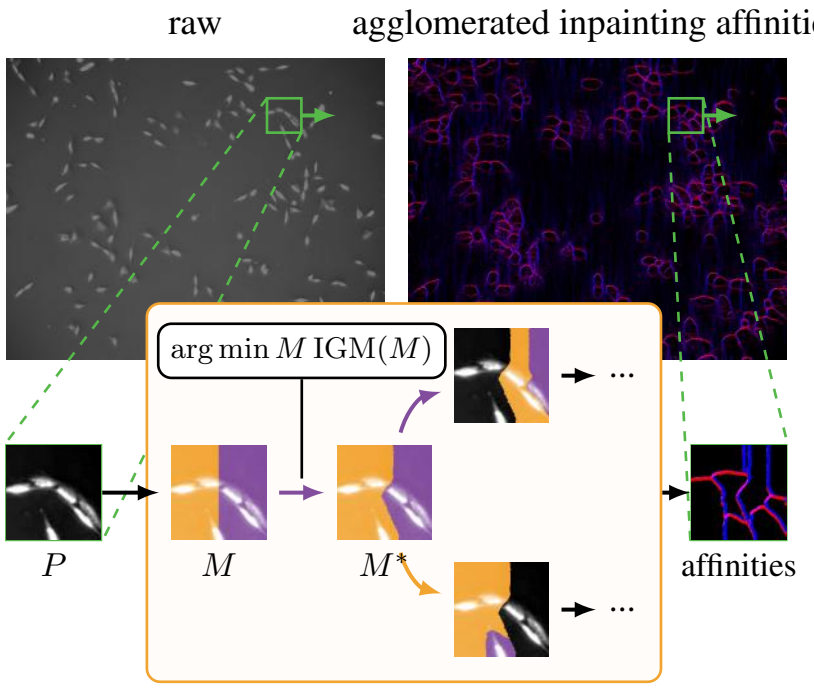


Figure 5.1: Extraction of instance separating affinities from an inpainting network. Given an image patch P, we optimize a set of pixels M (shown in purple) to minimize the *information gain measure* IGM, which is based on the predictions of a probabilistic inpainting network (see Section 5.2.2 and Fig. 5.2 for details). This optimization ensures that pixels in M^* provide minimal information about the intensity values of pixels in the complement $\overline{M}^*$ (shown in orange). We apply this procedure recursively to M^* and $\overline{M}^*$ to obtain a hierarchical segmentation of the image patch from which we extract affinities (shown in blue/red for x-/y-direction, respectively). These affinities are computed and averaged over a set of sliding image patches (green box) to obtain the final affinity estimates.

recover those objects. In other words, the success of predicting masked out objects depends on the information about those object contained in the surrounding context.

Here, we propose to exploit the predictability of image regions given partial information to separate instances. We do so by maximizing the surprise of the inpainting network when trying to predict image content from one segment to another, or, equivalently, by minimizing the information gain between segments. This optimization can be carried out using only the implicit knowledge of inpainting networks about instances and thus gives rise to a self-supervised instance separation.

In particular, we define an information gain measure between image segments that can be approximated efficiently given an inpainting network. We show that minimizing this measure, through a hierarchical optimization algorithm yields useful image decompositions. We represent those decompositions by *affinities*, *i.e.*, attractive or repulsive edges between pairs of pixels, which we average over a set of image patches in a sliding window fashion to obtain affinities for arbitrarily large images. An overview of this method is shown in Fig. 5.1. The resulting affinities require only minimal post-processing to obtain a segmentation. We apply our method to the challenging problem of cell segmentation in microscopy images, where we show that the unsupervised instance separation finds non-trivial splits and is competitive with supervised methods.

5.2 Self-Supervised Segmentation

In general, self-supervised segmentation is an under-constrained problem. What exactly constitutes a correct segmentation of an image depends not only on the application context (*e.g.*, segment all cells in a microscopy image), but also on a subjective level of detail (*e.g.*, segment nuclei and cell membrane individually). Without constraining assumptions or instructions, several different segmentations of the same image are plausible, leading to an intrinsic ambiguity. This ambiguity can be prominently observed as the inter-human variance for segmentation tasks where the concept of a segment is not precisely defined (see, *e.g.*, human generated segmentations of the BSD500 dataset for fruits, fences, or flowers)[11].

In the case of supervised image segmentation, this ambiguity is resolved by a set of training object instances in the form of, *e.g.*, affinities, labeled images, bounding boxes, or polygons. For self-supervised segmentation, on the other hand, assumptions about what constitutes a segmentation have to fill in for the lack of training data.

Here, we propose to resolve this ambiguity by assuming that pixels of the same instance are more predictable from each other than across instances. We define the similarity between two pixels (and therefore the likelihood to be part of the same instance) as the information gained about the value of one pixel by observing the value of the other one. In the following we will derive this similarity from a measure of inpainting accuracy.

5.2.1 Self-supervised Inpainting

Let x_i be a random variable representing the intensity of pixel $i \in \Omega$, and x_M with $M \subseteq \Omega$ a set of random variables $\{x_i | i \in M\}$. Probabilistic inpainting is equivalent to learning a parameterized function $p_\theta(x_i | x_M)$, *i.e.*, the conditional distribution over intensities of pixel i, given known intensities of a partial observation M. The parameters θ of the distribution p_θ

can be learned by maximizing the likelihood of a measurement $x = x^*$, or equivalently by minimizing the following negative log-likelihood:

$$\mathcal{L}_{\text{inpaint}}(\theta; M) = \sum_{i \notin M} -\log p_\theta \left(x_i = x_i^* | x_M = x_M^*\right) \tag{5.1}$$

It is worth noting that this loss formulation resembles the objective of probabilistic NOISE2VOID [83], highlighting the close connection between inpainting and denoising. In the next subsection, we will derive a similar connection between inpainting ("predictability") and instance separation ("affinity").

5.2.2 Predictability is Affinity

Our central assumption is that the intensity value of a pixel in an instance is conditionally independent of all pixels outside the instance.

In other words, pixel values should be well predictable given the values of other pixels in the same instance (high affinity). Conversely, pixel values from other instances should provide *no additional* information (low affinity). More formally, let $S = \{S_u \subseteq \Omega\}$ be a segmentation of Ω (*i.e.*, $\bigcup_u S_u = \Omega$ and $\forall u \neq v : S_u \cap S_v = \varnothing$), and let $S(i) \subseteq \Omega$ denote the segment containing pixel i. We assume that for the true instance segmentation S^*

$$p(x_i | x_{\Omega \setminus \{i\}}) = p(x_i | x_{S^*(i) \setminus \{i\}}), \tag{5.2}$$

or, equivalently, that there is *no further* information gain provided by Ω compared to $S^*(i)$ for estimating the value of x_i. For general subsets $M \subseteq \Omega$, let $\mathrm{IG}(i|M)$ denote the additional information gained for estimating the value of x_i when observing Ω compared to M alone, *i.e.*,

$$\mathrm{IG}(i|M) = D_{\mathrm{KL}}\Big(p(x_i | x_{\Omega \setminus \{i\}}) \Big|\Big| p(x_i | x_{M \setminus \{i\}})\Big), \tag{5.3}$$

where D_{KL} denotes the Kullback-Leibler divergence. In the following, we will use $\mathrm{IG}(i|M)$ as a measure of how much x_i depends on values *not* contained in M.

Considering our assumption stated in (5.2), a sensible objective to recover a single segment of the true segmentation S^* would be to minimize (5.3) with respect to M. In practice, however, it would be unreasonable to assume that even for a correct segment M the information gain for pixels in this set from pixels outside this set is exactly zero. In other words, dilating M would trivially decrease $\mathrm{IG}(i|M)$ until $M = \Omega$. Therefore, instead of minimizing (5.3) directly, we propose to minimize a symmetric information gain measure. Let $\overline{M} = \Omega \setminus M$ be the complement of M. Recall that $\mathrm{IG}(i|M)$ measures the dependency of x_i on values in $\overline{M}$. We

introduce a *relative* information gain that indicates whether M or $\overline{M}$ provide more information about the value of x_i:

$$\mathrm{RIG}(i|M) = \mathrm{IG}(i|M) - \mathrm{IG}(i|\overline{M}). \tag{5.4}$$

The quality of a single segment M can now be assessed by the following symmetric information gain measure over all pixels i:

$$\begin{aligned}\mathrm{IGM}(M) &= \sum_{i \in M} \mathrm{RIG}(i|M) + \sum_{i \in \overline{M}} \mathrm{RIG}(i|\overline{M}) && (5.5)\\ &= \sum_{i \in M} \mathrm{RIG}(i|M) - \sum_{i \in \overline{M}} \mathrm{RIG}(i|M). && (5.6)\end{aligned}$$

5.2.3 Efficient Implementation

In its current form, $\mathrm{IGM}(M)$ requires evaluation of $\mathrm{IG}(i|M)$ for every pixel $i \in \Omega$. For each of these evaluations, $p_\theta(x_i|\cdot)$ has to be computed two times (conditioned on M and $\overline{M}$), which is too inefficient for a practical implementation.

To remedy this inefficiency, we make two approximations: First, we take advantage of convolutional neural network architectures that can inpaint an arbitrary set of pixels N for the same conditional [97]:

$$\prod_{i \in N} p_\theta(x_i|M \setminus \{i\}) \approx \prod\nolimits_{i \in N} p_\theta(x_i|M \setminus N) \tag{5.7}$$

A similar approximation technique was first proposed by Krull, Buchholz, and Jug [82] who argue that this approximation is error-free for convolutional neuronal networks, if all pixels in N are spaced further apart than the field of view of the network. In our experiments, we find that even much denser subsets can be chosen without significant impact. We will refer to $\mathrm{RIG}(i|M)$ using this approximation as $\mathrm{RIG}_N(i|M)$ in the following.

Second, due to the limited field of view of the inpainting network, pixels far away from the conditional set have to be estimated via a constant prior and the relative information gain can therefore be computed without evaluating the neural network. Similarly, the complement conditional contains all pixels in the field of view. This is exactly the denoising setup of NOISE2VOID [82]. Therefore, for low-noise-images one can directly approximate $\mathrm{IG}(i|\Omega) \approx 0$ and otherwise apply the NOISE2VOID as a preprocessing step to our method. Thus, $\mathrm{RIG}(i|M) \approx$ const for pixels far away from the boundary between M and $\overline{M}$.

In conclusion, limiting the computation of IGM to a specified region N close to the boundary combined with the approximate RIG_N leads to the following approximation:

$$\mathrm{IGM}_N(M) = \sum_{i \in M \cap N} \mathrm{RIG}_N(i|M) - \sum_{i \in \overline{M} \cap N} \mathrm{RIG}_N(i|M) \tag{5.8}$$

$$\approx \mathrm{IGM}(M) + \text{const} \tag{5.9}$$

5.2.4 Segmentation from Maximal Independent Regions

Although the approximation IGM_N introduced above reduces the computational burden of evaluating IGM, finding an optimal mask

$$M^* = \arg\min M \, \mathrm{IGM}_N(M) \tag{5.10}$$

still remains intractable in general due to the combinatorial number of possible masks. We propose to solve this optimization problem by following a greedy optimization strategy that generates a sequence of masks M^t for $t \in \{0, \ldots, T\}$ such that $\mathrm{IGM}_N(M^{t+1}) \leq \mathrm{IGM}_N(M^t)$, illustrated in Fig. 5.2.

To this end, we first separate Ω into two equally sized components M^0 and $\overline{M^0}$ by randomly splitting them horizontally or vertically. We then evolve the boundary of the split by evaluating $\mathrm{RIG}_N(i|M^t)$ for all pixels $i \in N$ in close proximity to the current boundary between M^t and $\overline{M^t}$. The sign of $\mathrm{RIG}_N(i|M^t)$ indicates whether M^t or $\overline{M^t}$ provide more information about the pixel i. We update M accordingly, *i.e.*,

$$M^{t+1} = (M^t \setminus N) \cup \{i \in N | \mathrm{RIG}_N(i|M^t) > 0\}, \tag{5.11}$$

which, by definition of (5.8), monotonically decreases IGM_N.

Finally, in order to obtain a decomposition of an image into arbitrarily many maximally independent regions, we apply the minimization recursively to already identified regions, *i.e.*, we repeat the optimization procedure described above on regions M^* and $\overline{M^*}$, until either M^* or $\overline{M^*}$ are empty. Further implementation details on our neighborhood selection can be found in the appendix.

In order to extract affinities for a full image we compute maximally independent regions on a set of overlapping, sliding image patches and average their affinities. This procedure is illustrated in Fig. 5.1.

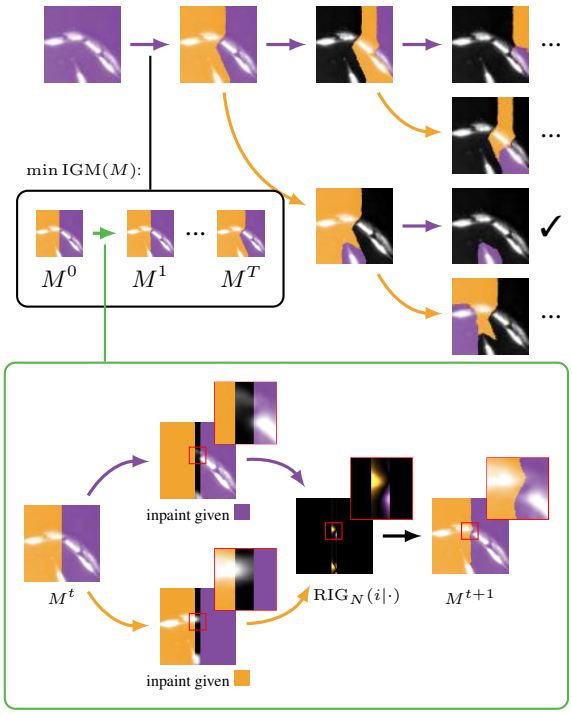
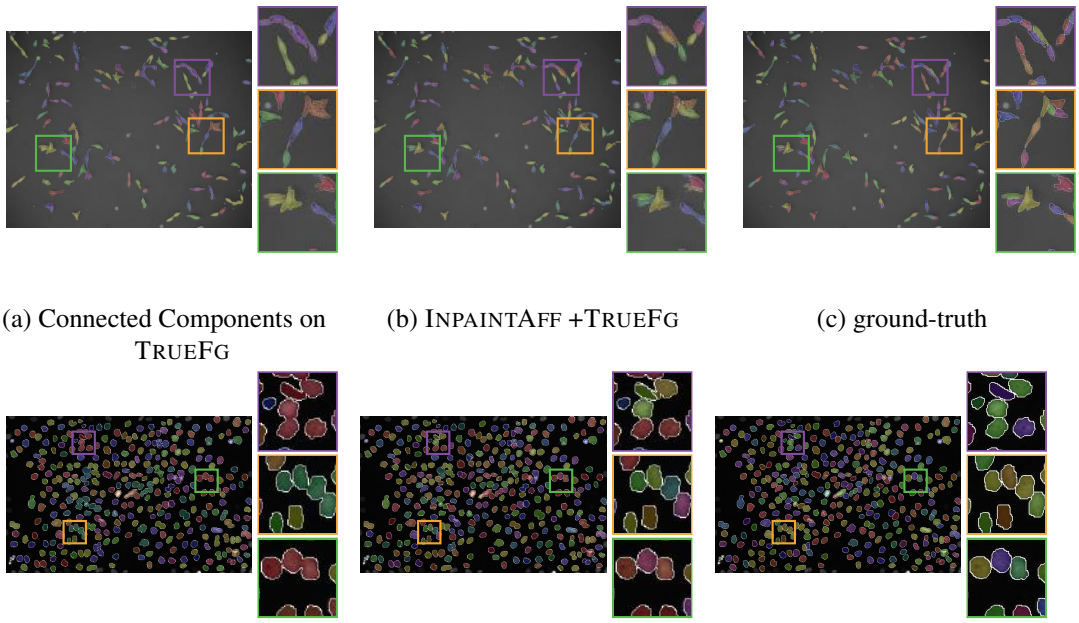


Figure 5.2: Details of the hierarchical segmentation of an image patch from an inpainting network. Given an image patch (top left), we recursively find optimal splits (shown in orange and purple) by evolving a randomly chosen horizontal or vertical split over T iterations (black box). For each step (illustrated in the green box), we evolve the boundary of the split by consulting a probabilistic inpainting network to predict the intensity of pixels in a region N around the boundary, once given only the information contained in M and once in its complement $\overline{M}$. We then measure the relative information gain RIG_N in the inpainting region to determine which component (orange or purple) provided more information about the pixels in N and reassign M accordingly.

5.3 Experiments on Microscopy Image Instance Segmentation

Instance separation is of particular importance for the identification and tracking of individual cells in microscopy images, where cells frequently form densely packed clusters and thus pose a challenging segmentation problem [153]. In many cases, those cells are freely moving in a substrate and can thus be considered as many independent instances of the same kind, which



(a) Connected Components on TRUEFG (b) INPAINTAFF +TRUEFG (c) ground-truth

(d) Connected Components on TRUEFG (e) INPAINTAFF +TRUEFG (f) ground-truth

Figure 5.3: Instance separation results assuming an accurate foreground detection TRUEFG on the PANC dataset (top row) and the HELA dataset (bottom row). A foreground detection alone is not sufficient to segment touching cells (a, d). INPAINTAFF extracted from an inpainting network find non-trivial splits between instances (b, e).

makes them suitable for an inpainting based approach like the one we propose here and in particular for the independence assumption we made in (5.2). In the following, we will refer to the affinities extracted using the proposed method as INPAINTAFF.

5.3.1 Cell Segmentation Benchmark Dataset

We evaluate INPAINTAFF on a subset of the ISBI Cell Segmentation Benchmark, which includes a diverse set of 2D microscopy videos covering a wide range of cell types and imaging quality.

In particular, we selected two datasets that contain cells of irregular shape in close proximity for which instance separation is needed to obtain a correct segmentation: (1) HELA contains cervical cancer cells expressing H2b-GFP and (2) PANC contains pancreatic stem cells on a polystyrene substrate (see Fig. 5.7 for samples and the CTC website for further information

about the datasets).

The PANC dataset arguably belongs to the more difficult datasets of the ISBI Cell Segmentation Benchmark (reflected in the comparatively low test scores on the challenge), which we attribute to two factors that are found in both HELA and PANC: First, they contain a large amount of touching cells with little boundary evidence, which renders a mere foreground segmentation ineffective for the detection of individual cells. Second, both datasets contain only little labeled training data (815 instances[2] for HELA and 514 for PANC in fully labeled frames), which challenges fully supervised segmentation approaches.

5.3.2 Results

As argued earlier, completely unsupervised segmentation is an under-constrained problem. As such, INPAINTAFF alone is unlikely to give rise to a segmentation capturing the intuition of a human annotator. We recall that the main guiding principle for INPAINTAFF is predictability of pixel intensities. Depending on the distribution of cells in images used to train the inpainting network, this predictability might equally well apply to a background region around each cell. This effect is visible in both datasets (compare Fig. 5.7) and demonstrates that the method is agnostic about the intensity of pixels and merely clusters pixels that are mutually predictable.

Therefore, we investigate first how well INPAINTAFF *separates* instances. We then turn to the problem of instance segmentation, where we assume that at least a small amount of ground-truth labels is available to capture the notion of objects of interest—an assumption that arguably holds for any realistic application in practice, where an accurate segmentation is required.

We report results using the ISBI Cell Segmentation Benchmark segmentation accuracy (SEG score), a metric that is based on the Jaccard similarity index and measures average IoU of all segments that overlap at least 50% with the ground truth (further details are given on the challenge website). The detection score is the percentage of matches that surpass a set IoU threshold.

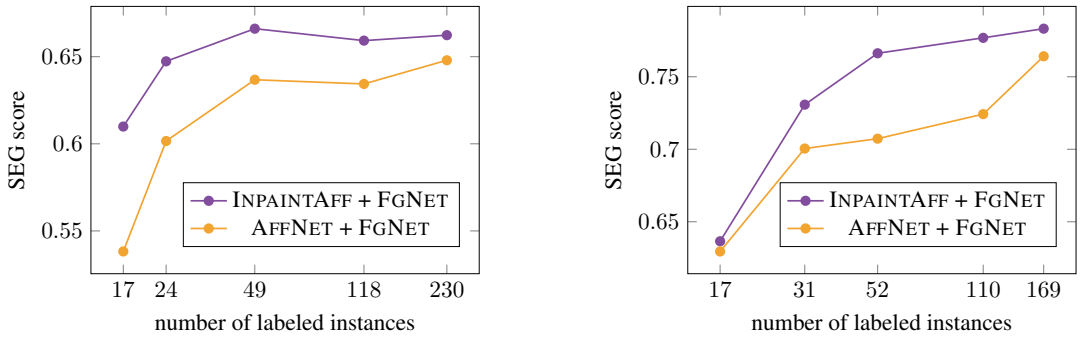
Instance Separation We investigate how well INPAINTAFF separates instances, assuming that an accurate foreground segmentation is already available. For that, we use the ground-truth segmentation provided in the datasets and convert it into a binary foreground segmentation TRUEFG, while connecting all segments separated by a one pixel wide gap.

As we show in Table 5.1 (and qualitatively in Fig. 5.3), TRUEFG alone is not sufficient to achieve an accurate instance segmentation, due to merges of cells in close proximity. Separating

[2]The HELA dataset has 571 additional instances, in partially labeled frames which can not trivially be used to train neural networks.

Method		HeLa	Panc
Connected Components on TrueFg		0.785	0.748
InpaintAff + TrueFg		0.858	0.914
InpaintAff + FgNet50		0.766	0.666
HIT-CN*	MU-Lux-CZ*	0.919	0.715
FR-Ro-GE*	CVUT-CZ*	0.903	0.682
PURD-US*	HD-Hau-GE*	0.902	0.665

Table 5.1: Segmentation scores assuming an accurate foreground detection TrueFg and FgNet50 (trained with 52/49 labeled instances for HeLa/Panc). For reference, we include the official challenge scores [153] of supervised methods on the same datasets (marked with a star), which have been trained on more labeled instances and evaluated on a different testing dataset than our method.



(a) Panc (b) HeLa

Figure 5.4: Segmentation score on the test data of Panc and HeLa datasets, for varying amounts of labeled instances used to train FgNet and AffNet.

those cells using InpaintAff, however, results in an almost perfect instance segmentation, in the case of Panc even significantly exceeding the scores of the best performing methods (albeit on different testing data and constrained to the ground-truth foreground). Those results suggest that (1) InpaintAff is accurately separating instances, and (2) a foreground segmentation is necessary and sufficient to constrain the boundaries of found objects to obtain a competitive segmentation.

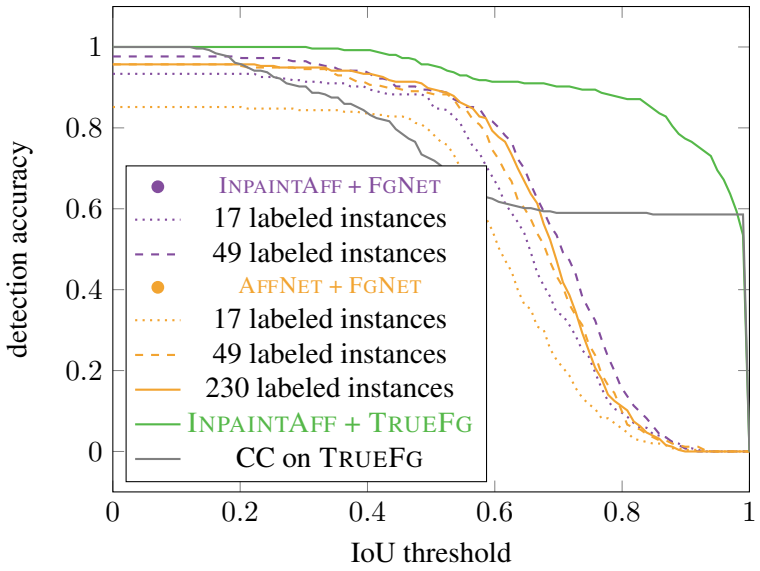


Figure 5.5: Detection accuracy over different IoU thresholds on PANC. Over a large range of IoU thresholds, INPAINTAFF in combination with a foreground network FGNET trained on 49 labeled instances has a higher detection accuracy than the fully supervised method AFFNET trained on 230 labeled instances.

Instance Segmentation from Foreground Prediction Since a foreground segmentation is crucial to capture the application specific notion of what constitutes an object, we next investigate the segmentation accuracy of our method when combined with a foreground prediction network trained on few instances only, which we will refer to as FGNET (details in Section 5.3.3). We train FGNET on varying amounts of labeled instances to predict a binary foreground mask and use this prediction in combination with our INPAINTAFF to obtain an instance segmentation. As a baseline, we also train a second network AFFNET to predict affinities directly from the same labeled instances used to train the foreground network.

The segmentation scores for either approach on the test dataset are shown in Fig. 5.4, for varying amounts of labeled instances used for training. Remarkably, INPAINTAFF consistently outperform trained affinities in terms of the SEG score . This effect is most visible in dataset PANC, where cells tend to cluster more compactly and the separation of individual cells is therefore more challenging. In particular, INPAINTAFF on this dataset in combination with FGNET trained on as few as 24 labeled cells produce a segmentation that outperforms the fully supervised AFFNET using one order of magnitude more training data. As shown in Fig. 5.5, this observation also holds in terms of the detection score over varying IoU thresholds.

Furthermore, on the PANC datasets the obtained segmentation score using only around 50 labeled instances for the foreground prediction together with unsupervised affinities is on par with the third leading submissions to the ISBI Cell Segmentation Benchmark, which have been trained on 514 instances (albeit evaluated on a different testing dataset then used here).

5.3.3 Experiment Details

Training and Testing Split Since INPAINTAFF requires a considerable amount of computational resources (see discussion in Section 5.5) a direct evaluation on the CTC servers on the official testing data is not possible. Therefore, we split the publicly available data for each dataset into a train and testing dataset, each containing one video of sparsely labeled cells.

Model Architectures For the inpainting network underlying INPAINTAFF, we use a downscaled version of the architecture proposed by Liu et al. [97], *i.e.*, a U-NET architecture with a depth of four resulting in five levels with 64, 128, 256, 512, and 512 feature maps, each. We train the network for 1M iterations using the ADAM optimizer and the loss proposed by Liu et al. [97] that is comprised of a perceptual, style, total variation and reconstruction loss.

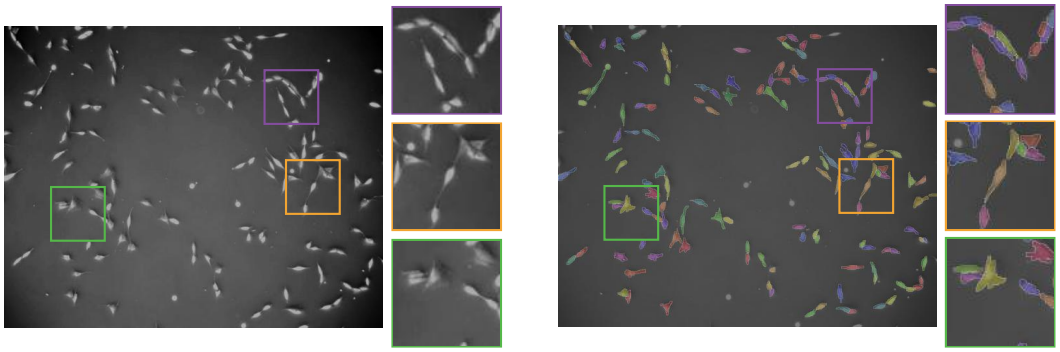
FGNET is a PIX2PIX network [58, 184] with a depth of six layers, containing 64 initial features maps, trained using ADAM to minimize a binary cross-entropy loss [68].

Since we use the MUTEXWATERSHED to post-process affinity predictions, we use the same training procedure proposed by Wolf et al. [163] for AFFNET (PIX2PIX architecture). In particular, we also use the Sørensen-Dice coefficient [42, 145] loss and the same affinity neighborhood (12 distances, up to 27 pixels).

Affinity-Based Segmentation We use the MUTEXWATERSHED to derive a segmentation from affinities [163], where we introduce a single parameter α to control for over- and undersegmentation by multiplying all long range affinities (that are used to split) with α. The optimal α for each evaluated method was determined on the validation dataset.

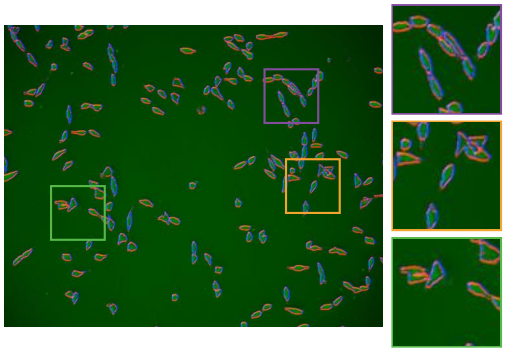
5.4 Related Work

While classical patch-based inpainting methods such as [17, 43, 146] synthesize high quality images, they fundamentally cannot make semantically aware decisions for intensity predictions. Deep inpainting networks, on the other hand, trained on large corpuses of data are known to develop an intrinsic understanding of images [88], which raises the question what aspects are captured by these networks. The usefulness of these inpainting models for image segmentation was shown by Pathak et al. [122], who demonstrate that features extracted from a trained

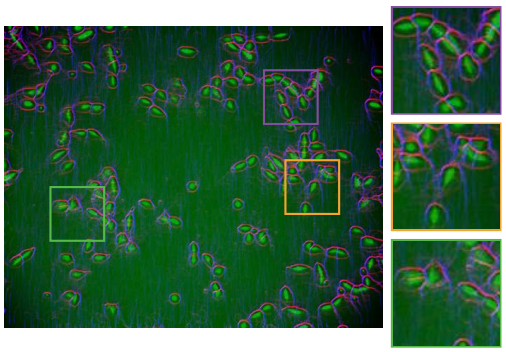


(a) PANC raw

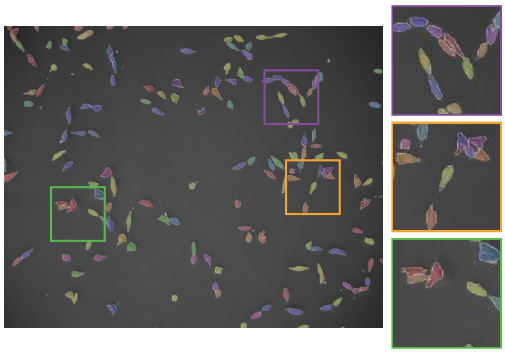
(b) ground-truth



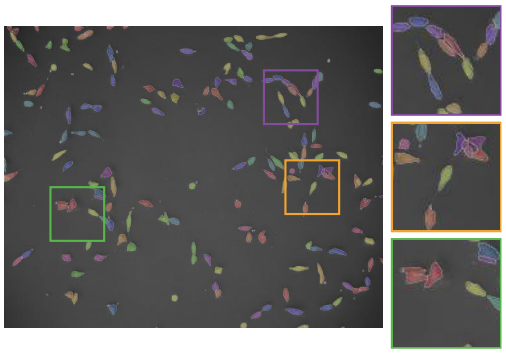
(c) AFFNET



(d) INPAINTAFF



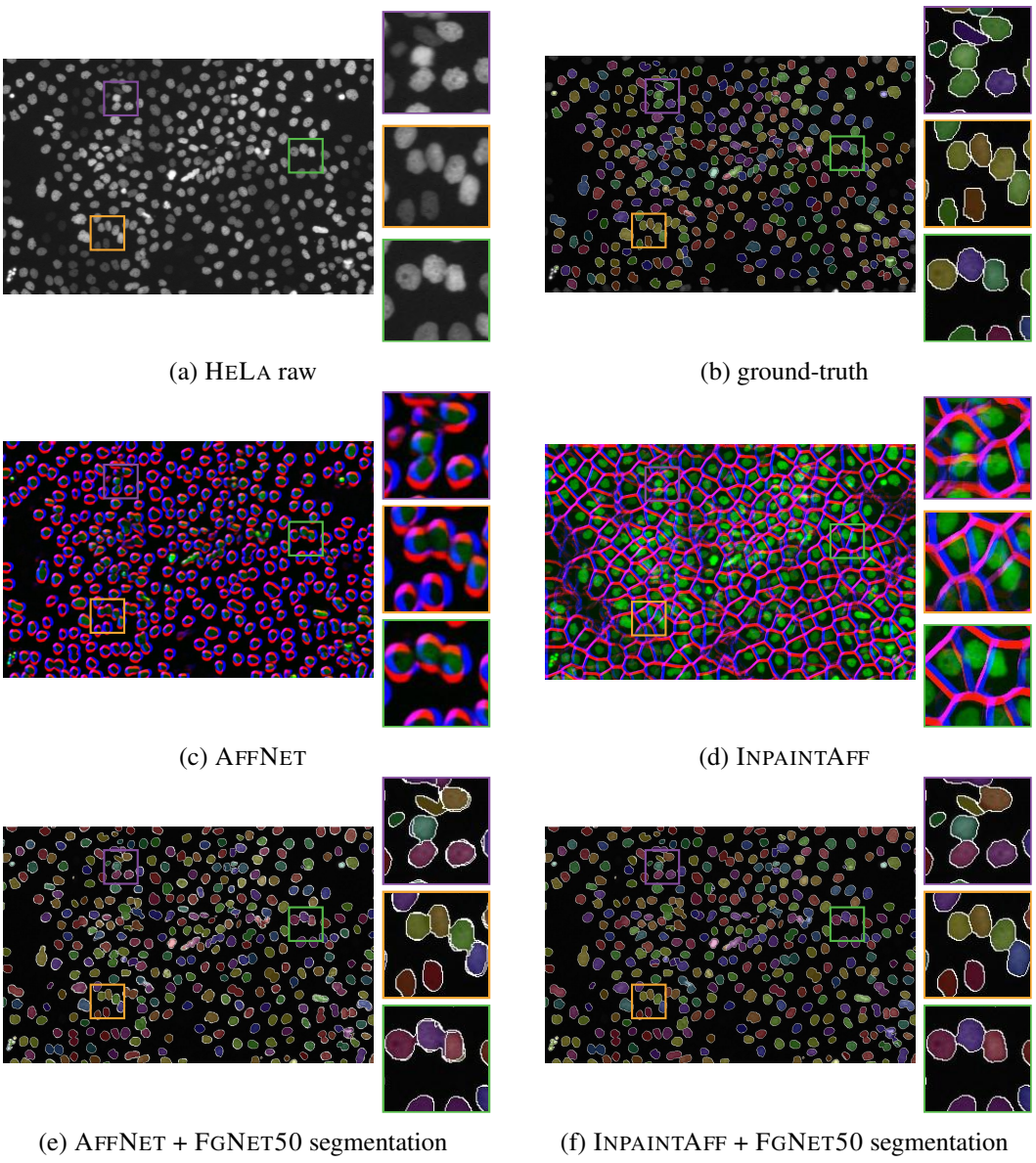
(e) AFFNET + FGNET50 segmentation



(f) INPAINTAFF + FGNET50 segmentation

Figure 5.6: Sample test images of PANC. Affinities are shown as blue/red for x-/y-direction, respectively.

inpainting network capture appearance and semantics of visual structures aiding in the pre-training of classification, detection, and segmentation tasks. Extending inpainting networks



(a) HeLa raw (b) ground-truth

(c) AffNet (d) InpaintAff

(e) AffNet + FgNet50 segmentation (f) InpaintAff + FgNet50 segmentation

Figure 5.7: Sample test images of HeLa. Affinities are shown as blue/red for x-/y-direction, respectively.

that directly minimize the reconstruction error [74, 168] with texture and structure aware loss, such as multi-scale neural patch synthesis [170] or Structure-aware Appearance Flow [128]

leads to high-fidelity images and prediction and modeling of higher order relations

In parallel, specialized architectures and convolutions have been developed that make it possible to realistically inpaint arbitrary masks [97, 176].

In this work, we use the network architecture and loss proposed by Liu et al. [97] which is designed to inpaint arbitrary masks and is trained with an additional style component loss. Since we leverage the network's learned distribution by measuring information gain between image patches, we intentionally avoid networks trained with an additional GAN loss [32, 116, 178]. Although GANs produce extremely realistic looking images, they are prone to mode collapse that affects our estimate of information gain.

More generally, inpainting falls under the broader category of unsupervised prediction of left-out data, also known as *self-supervised* learning [136]. This includes tasks such as image colorization [87, 180], co-occurrence [57], predicting permutations [138], and denoising [82]. These methods are highly effective at extracting robust features for further transfer learning [181] and image embeddings [149] and can be considered a proxy task for developing a semantic understanding [88].

In some cases, the self-supervised task can be used as a free supervisory signal that directly translates to classically supervised tasks. For example, object tracking emerges from video colorization [158] (which inspired our title) or through obeying cycle-consistency in time [159]. When provided with background images and images with objects, Ostyakov et al. [118] learn to segment by predicting masks and paste patches from the object domain onto the background domain constrained by an adversarial and a cycle consistency loss.

Our work uses the statistical properties of instances to derive a method for separating instances, which closely relates to other self-supervised segmentation approaches that utilize different properties to identify objects. Burgess et al. [25] utilize compressibility, in a compositional generative model, where image regions are reconstructed through a low dimensional bottleneck. They show that their model is capable of discovering useful decompositions of scenes by identifying segments that can be represented in a common format. Another approach by Chen, Artières, and Denoyer [31] learns to find masks of objects by learning to replace the masked content content that corresponds with altering the masked objects properties (e.g. altering the color of flowers).

5.5 Conclusion

It remains an open question as to how far completely unsupervised segmentation based on image statistics alone will find real world applications. As we already observed on the segmentation of cells in microscopy images studied here, an experimentalist's intention of what constitutes a good cell segmentation does not necessarily match the clustering of pixels based on information content. Only at least partially supervised methods with application specific losses can ultimately produce predictions tailored to a specific application, provided enough labeled training data is available. We see the contribution of this work therefore primarily as an aid to supervised methods, especially in scenarios in which labeled training data is scarce. As our experiments demonstrate, INPAINTAFF allow practitioners to obtain competitive segmentations from very few labeled instances. Given the high rate and diversity of microscopy images acquired in the life sciences, self-supervised segmentation has the potential to significantly reduce the amount of human interaction needed. Our work shows that in this domain the inherent knowledge captured by inpainting networks provides competitive performance with very few labeled instances.

A limitation of the method proposed here is the runtime: INPAINTAFF requires around 48h to process a 700x1100 image on a single GPU. Although inference can be trivially parallelized, the current implementation might be prohibitively slow for many applications. Increasing the efficiency of the inference by, *e.g.*, training networks directly on IGM, will be subject of future work.

6 Conclusion

In this work, we have tackled the problem of instance segmentation in large 3D volumes, focusing on the application of neuron reconstruction in particular. Before this work, watershed algorithms were not used to segment large volumes despite their theoretical efficiency. This is because they rely on seed points, which are very difficult to generate automatically for neurons. Instead, graph partitioning methods, like the multicut, that use repulsions instead of seeds were preferred, but could only be applied on reduced problem sizes, where pixels had been preliminarily clustered. In Chapter 3, we invent a new watershed algorithm that, for the first time, uses repulsions instead of seeds. We characterize this algorithm by proving that it finds the global optimum of an objective function. This objective function shows the close connection to the multicut objective as well as its relationship to the power watershed framework. Further extensions of this algorithm, to the problem of joint semantic instance segmentation, are explored in Chapter 4.

We demonstrate that this algorithm, in combination with deep neural network, outperforms multicut approaches on the ISBI neuron segmentation challenge. Since these results were published, our method has become a part of two state-of-the-art segmentation pipelines by Hirsch, Mais, and Kainmueller [55] and Lee et al. [90].

Our results offer several opportunities for further research. The strong ties to multicut and Multi-Way Cut suggest that other optimization problems with similar structure might yield further watershed algorithms. A notable example is moral lineage tracing [61], for joint segmentation and tracking in video time series. Another research direction is the supervised learning of graph weights, that we explore in Chapter 2. Since the objective function of the MWS is known, techniques such as structured learning could be applied to learn weights that optimize the segmentation performance directly. The fact that the MWS objective can be optimized so efficiently suggests that there is a similarly efficient structured learning method. Finally, our approach for unsupervised learning, that we propose in chapter Chapter 5 should be explored further. We have proposed to separate instances by directly maximizing an independence measure. This process is very resource-intensive, requiring many hours to process a full image. Therefore, one should investigate methods that learn to predict these independent regions directly, circumventing the costly optimization step.

Appendices

A Mutex Watershed

A.1 Property of the minimizers of $Q^p(x)$

Recall Lemma 3.4.5:

Lemma A.1.1. *Let us consider the scales λ_k and continuous functions $Q_k(x) : \mathbb{R}^{|E|} \to \mathbb{R}$ respectively defined in Eq. (3.25) and (3.26). For any value of $p \in \mathbb{N}^+$, let $x_p \in \mathbb{R}^{|E|}$ be a minimizer of the function $Q^p(x)$ defined in Eq. (3.24). Then, the minimizer x_p lies in the set $\mathsf{ISC}(\mathcal{G}, w)$. From this, it follows that any sequence of minimizers $(x_p)_{p>0}$ is bounded and the conditions of Theorem 3.4.4 are satisfied.*

Proof. The function $Q^p(x)$ can be explicitly written as (see Eq. 3.24, 3.25 and 3.26):

$$Q^p(x) = \sum_{0 \leq k < t} \lambda_k^p Q_k(x) \tag{A.1}$$

$$= |E| \min_{x' \in \mathsf{ISC}(\mathcal{G}, w)} ||x - x'|| + \sum_{1 \leq k < t} \left| \frac{\tilde{w}_k}{2\tilde{w}_1} \right|^p \sum_{e \in E_k} x_e \tag{A.2}$$

$$= |E| \min_{x' \in \mathsf{ISC}(\mathcal{G}, w)} ||x - x'|| + \sum_{e \in E} \left| \frac{w_e}{2\tilde{w}_1} \right|^p x_e. \tag{A.3}$$

We then denote these two terms by:

$$Q^p_{\mathrm{A}}(x) := |E| \min_{x' \in \mathsf{ISC}(\mathcal{G}, w)} ||x - x'||, \tag{A.4}$$

$$Q^p_{\mathrm{B}}(x) := \sum_{e \in E} \left| \frac{w_e}{2\tilde{w}_1} \right|^p x_e. \tag{A.5}$$

Intuitively, we now prove that the minimizer x_p of $Q^p(x)$ lies in $\mathsf{ISC}(\mathcal{G}, w)$ by showing that the first term $Q^p_{\mathrm{A}}(x)$ is always "dominant" as compared to $Q^p_{\mathrm{B}}(x)$.
First, we note that the gradient of the first term $Q^p_{\mathrm{A}}(x)$ has always norm equal to $|E|$ and points in the direction of the closest point $x' \in \mathsf{ISC}(\mathcal{G}, w)$. Given a generic point $y \in \mathbb{R}^{|E|}$, the only two cases when the gradient $\nabla_x Q^p_{\mathrm{A}}(x)$ does not exists are: i) if $y \in \mathsf{ISC}(\mathcal{G}, w)$; ii) if there

are at least two points $x'', x''' \in \mathsf{ISC}(\mathcal{G}, w)$ such that $||y - x''|| = ||y - x'''||$. Clearly, $Q^p_{\mathrm{A}}(x)$ presents minima only in the first case, when $y \in \mathsf{ISC}(\mathcal{G}, w)$.
On the other hand, the second term $Q^p_{\mathrm{B}}(x)$ is always differentiable and the norm of its gradient is never greater than $\sqrt{|E|}$:

$$||\nabla_x \, Q^p_{\mathrm{B}}(x)|| < \left|\left| \nabla_x \left(\sum_{e \in E} x_e \right) \right|\right| = \sqrt{|E|} \tag{A.6}$$

where we used the fact that $\tilde{w}_k / 2\tilde{w}_1 < 1$ for every $1 \leq k < t$. Thus, the magnitude of the gradient given by the first term is always larger compared to the one given by the second term. We then conclude that the objective can always be reduced unless x_p is a point of $\mathsf{ISC}(\mathcal{G}, w)$. □

B Semantic Mutex Watershed

B.1 Redundant Path Constraints

Lemma B.1.1 (Redundant Paths)**.** *Let $G' = (V', E', w) = (V \cup T, E \cup E^S, w)$ be an edge-weighted graph extended by terminal nodes T. For any edge indicator $y \in 0, 1^{|E \cup S|}$ that satisfies*

$$\sum_{t \in T} y_{tv} = |T| - 1 \, \forall v \in V \tag{B.1}$$

the following set of constraints are equivalent:

$$y_{ut} + y_{uv} + y_{vt'} \geq 1 \qquad \forall (u, v) \in E \; \forall t, t' \in T, \; t \neq t' \tag{B.2}$$

$$\Leftrightarrow y_{tu} + y_{uv} \geq y_{tv}, \qquad \forall uv \in E, t \in T \tag{B.3}$$

$$y_{tv} + y_{uv} \geq y_{tu}, \qquad \forall uv \in E, t \in T. \tag{B.4}$$

Proof. This lemma is trivially fulfilled for $|T| \leq 1$. We will prove the lemma for $|T| > 1$ by contradiction in each direction.

"⇒" Assume eq. (B.2) holds and $\exists y_{tv} > y_{tu} + y_{uv}$. In case $y_{tv} = 1$, eq. (B.1) implies that $\exists t' \neq t : y_{tv} = 0$ which leads to the contradiction

$$y_{tv} > y_{tu} + y_{uv} \geq 1 - y_{t'u} = 1 \tag{B.5}$$

In case $y_{tv} = 0$, eq. (B.1) implies that $\forall t' \neq t : y_{t'v} = 0$ leading to the contradiction

$$y_{tv} > y_{tu} + y_{uv} \geq 1 - y_{t'u} = 0. \tag{B.6}$$

The proof for eq. (B.4) is analogous.

"⇐" Assume eqs. (B.3) and (B.4) hold and $\exists (u, v) \in E \, t, t' \in T, \; t \neq t' : y_{ut} + y_{uv} + y_{vt'} < 1$. This leads to the contradiction

$$y_{ut} + y_{uv} + y_{vt'} < 1 \tag{B.7}$$

$$\Rightarrow y_{ut} = 0, y_{uv} = 0 \text{ and } \underbrace{y_{vt'} = 0}_{\Rightarrow y_{vt} = 1} \tag{B.8}$$

$$\Rightarrow 1 = y_{vt} \leq y_{ut} + y_{uv} = 0. \tag{B.9}$$

□

B.2 Additional Details of the Cityscapes Experiments

B.2.1 Implementation Details

We use the class probabilities from a Deeplab 3+ [30] as semantic edge weights. We use a trained model provided by Tensorflow. employ the Mask-RCNN [54] implementation provided by [102] and trained a model on Cityscapes following [54]'s training configuration. The graph weights are derived as explained above. We derive graph weights for different offsets: for attractive edges we use (1) 8-neighbourhood with distances of {1, 2, 4} pixels, (2) random pairs inside each bounding box. For repulsive edges we sample 5 random pixel pairs for each mask and compute the soft IOU (eq. (4.34)). [98] trained a Deeplab 3+ to predict affinities for their graph-clustering algorithm. They kindly provided their trained models allowing us to use the same affinities. Since their clustering utilizes a threshold, we treat the threshold as the splitting point between attractive and repulsive edge weights; affinities below the threshold are inverted and scaled to [0, 1]. In addition to the model by GMIS that is trained on scaled bounding boxes, we train a Deeplab3+ for affinity predictions on the full images. Because [98] only tackle instance segmentation, their model does not predict affinities for stuff classes. We train the network with Sorensen Dice Loss and the same stencil pattern as [98]. The training protocol follows the settings in [30], using a batch size of 12 and 70k training iterations. We do not employ any test time augmentations.

B.2.2 Additional images

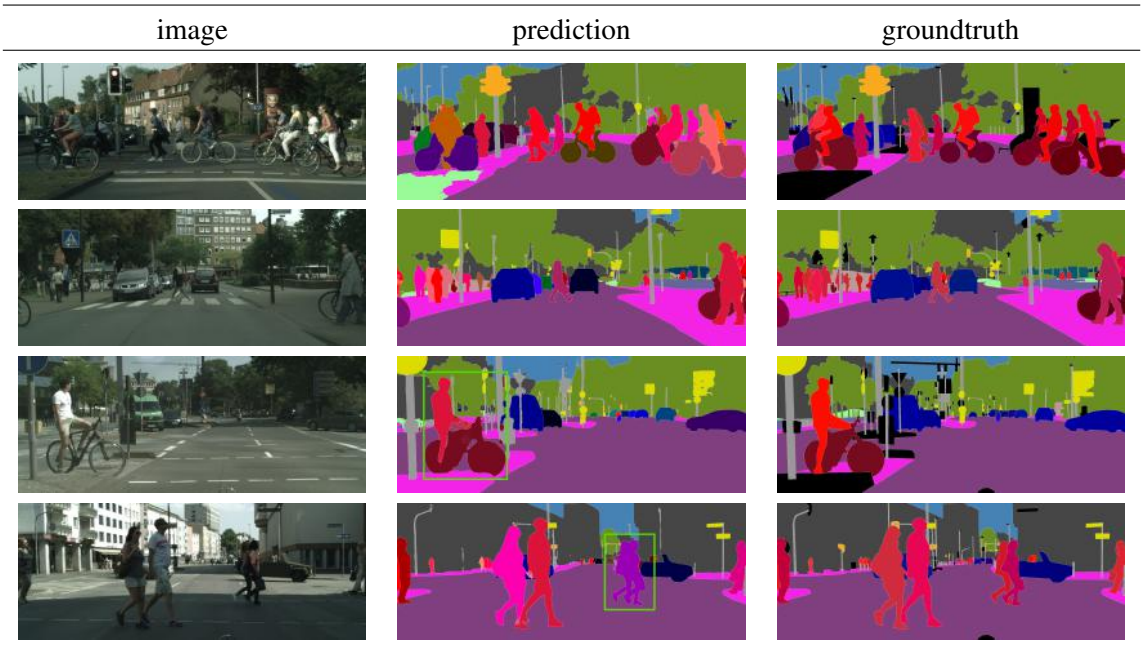


Figure B.1: Further examples panoptic results on Cityscapes using using semantic unaries (Deeplab 3+ network [96]) and affinities derived from Mask-RCNN [54] foreground probability. Prediction errors are highlighted in green.

B.3 Scaling Behavior

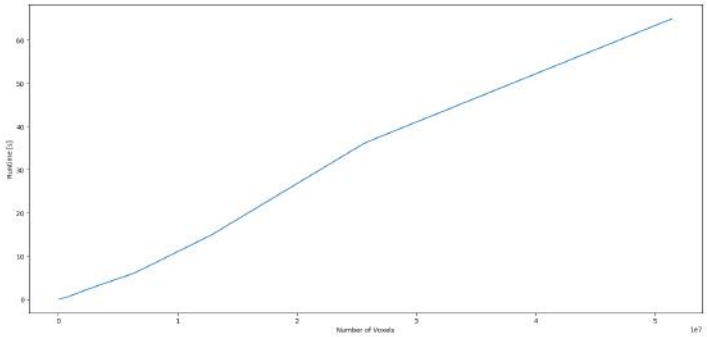


Figure B.2: Runtime scaling of the SMWS. We evaluate the runtime of the SMWS for different volume sizes of the 3D Sponge dataset. We find an almost linear relation between runtime and number of voxels.

Publications

I contributed to the following peer reviewed publications:

- **Steffen Wolf**, Lukas Schott, Ullrich Köthe, and Fred A. Hamprecht. „Learned Watershed: End-to-End Learning of Seeded Segmentation.“ In: *Proceedings of the International Conference on Computer Vision (ICCV)* (2017).
- **Steffen Wolf**, Constantin Pape, Alberto Bailoni, Nasim Rahaman, Anna Kreshuk, Ullrich Kothe, and Fred A. Hamprecht. „The mutex watershed: efficient, parameter-free image partitioning.“ In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- **Steffen Wolf**, Alberto Bailoni, Constantin Pape, Nasim Rahaman, Anna Kreshuk, Ullrich Köthe, and Fred A. Hamprecht. „The Mutex Watershed and Its Objective: Efficient, Parameter-Free Image Partitioning.“ In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).
- Carsten Haubold, Janez Aleš, **Steffen Wolf**, and Fred A Hamprecht. „A generalized successive shortest paths solver for tracking dividing targets.“ In: *European Conference on Computer Vision*. Springer. 2016.
- Elke Kirschbaum, Manuel Haußmann, **Steffen Wolf**, Hannah Sonntag, Justus Schneider, Shehabeldin Elzoheiry, Oliver Kann, Daniel Durstewitz, and Fred A Hamprecht. „LeMoNADe: Learned Motif and Neuronal Assembly Detection in calcium imaging videos.“ In: *International Conference on Learning Representations*. 2019.
- Nasim Rahaman, **Steffen Wolf**, Anirudh Goyal, Roman Remme, and Yoshua Bengio. „Learning the Arrow of Time for Problems in Reinforcement Learning.“ In: *International Conference on Learning Representations*. 2020.
- Martin Schiegg, Ben Heuer, Carsten Haubold, **Steffen Wolf**, Ullrich Koethe, and Fred A Hamprecht. „Proof-reading guidance in cell tracking by sampling from tracking-by-assignment models.“ In: *International Symposium on Biomedical Imaging (ISBI)*. 2015.

- Vladimír Ulman et al. „An objective comparison of cell-tracking algorithms.“ In: *Nature methods* 14.12 (2017), p. 1141.
- Shaofei Wang, **Steffen Wolf**, Charless Fowlkes, and Julian Yarkony. „Tracking objects with higher order interactions via delayed column generation.“ In: *Artificial Intelligence and Statistics*. 2017.

The following publications are currently under peer review:

- **Steffen Wolf**, Fred A. Hamprecht, and Jan Funke. „Instance Separation Emerges from Inpainting.“ In: *arXiv preprint arXiv:2003.00891* (2020).
- **Steffen Wolf**, Yuyan Li, Constantin Pape, Alberto Bailoni, Anna Kreshuk, and Fred A. Hamprecht. „The Semantic Mutex Watershed for Efficient Bottom-Up Semantic Instance Segmentation.“ In: *arXiv preprint arXiv:1912.12717* (2019).

Bibliography

[1] Mohammed Abdelsamea. „An Enhancement Neighborhood Connected Segmentation for 2D-Cellular Image.“ In: *International Journal of Bioscience, Biochemistry and Bioinformatics* 1.4 (2011).

[2] Ali Qusay Al-Faris, Umi Kalthum Ngah, Nor Ashidi Mat Isa, and Ibrahim Lutfi Shuaib. „Breast MRI tumour segmentation using modified automatic seeded region growing based on particle swarm optimization image clustering.“ In: *Soft Computing in Industrial Applications*. Springer, 2014, pp. 49–60.

[3] Ali Qusay Al-Faris, Umi Kalthum Ngah, Nor Ashidi Mat Isa, and Ibrahim Lutfi Shuaib. „Computer-aided segmentation system for breast MRI tumour using modified automatic seeded region growing (BMRI-MASRG).“ In: *Journal of digital imaging* 27.1 (2014), pp. 133–144.

[4] Mustafa A Alattar, Nael F Osman, and Ahmed S Fahmy. „Myocardial segmentation using constrained multi-seeded region growing.“ In: *International Conference Image Analysis and Recognition*. Springer. 2010, pp. 89–98.

[5] Bjoern Andres, Jörg H. Kappes, Thorsten Beier, Ullrich Köthe, and Fred A. Hamprecht. „Probabilistic Image Segmentation with Closedness Constraints.“ In: *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 2611–2618.

[6] Bjoern Andres, Ullrich Koethe, Thorben Kroeger, Moritz Helmstaedter, Kevin L Briggman, Winfried Denk, and Fred A. Hamprecht. „3D segmentation of SBFSEM images of neuropil by a graphical model over supervoxel boundaries.“ In: *Medical image analysis* 16.4 (2012), pp. 796–805.

[7] Bjoern Andres, Thorben Kroeger, Kevin L Briggman, Winfried Denk, Natalya Korogod, Graham Knott, Ullrich Koethe, and Fred A. Hamprecht. „Globally optimal closed-surface segmentation for connectomics.“ In: *European Conference on Computer Vision*. Springer. 2012, pp. 778–791.

[8] Björn Andres, Thorben Kröger, K. L. Briggmann, W. Denk, N. Norogod, G. Knott, Ullrich Köthe, and Fred A. Hamprecht. „Globally Optimal Closed-Surface Segmentation for Connectomics.“ In: *Proc. ECCV'12, part 2*. 7574. 2012, pp. 778–791.

[9] Jesús Angulo and Dominique Jeulin. „Stochastic watershed segmentation.“ In: *PROC. of the 8th International Symposium on Mathematical Morphology*. 2007, pp. 265–276.

[10] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. „Contour Detection and Hierarchical Image Segmentation.“ In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.5 (2011), pp. 898–916.

[11] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. „Contour Detection and Hierarchical Image Segmentation.“ In: *IEEE Transactions Pattern Analysis Machine Intelligence* 33.5 (2011), pp. 898–916.

[12] Ignacio Arganda-Carreras, Srinivas Turaga, Daniel Berger, et al. „Crowdsourcing the creation of image segmentation algorithms for connectomics.“ In: *Front. Neuroanatomy* 9 (2015), p. 142.

[13] Anurag Arnab and Philip HS Torr. „Pixelwise Instance Segmentation with a Dynamically Instantiated Network.“ In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 441–450.

[14] Shai Bagon and Meirav Galun. „Large Scale Correlation Clustering Optimization.“ In: *CoRR* abs/1112.2903 (2011).

[15] Min Bai and Raquel Urtasun. „Deep Watershed Transform for Instance Segmentation.“ In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 2858–2866.

[16] Min Bai and Raquel Urtasun. „Deep watershed transform for instance segmentation.“ In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017, pp. 2858–2866.

[17] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. „PatchMatch: A randomized correspondence algorithm for structural image editing.“ In: *ACM Transactions on Graphics (ToG)*. Vol. 28. 3. ACM. 2009, p. 24.

[18] Thorsten Beier. „Multicut Algorithms for Neurite Segmentation.“ PhD thesis. 2018.

[19] Thorsten Beier, Björn Andres, Ullrich Köthe, and Fred A. Hamprecht. „An efficient fusion move algorithm for the minimum cost lifted multicut problem.“ In: *European Conference on Computer Vision*. Springer. 2016, pp. 715–730.

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

[21] Serge Beucher. „Watershed, Hierarchical Segmentation and Waterfall Algorithm.“ In: *Proc. ISMM'94*. Vol. 94. 1994, pp. 69–76.

[22] Serge Beucher and Christian Lantuéjoul. „Use of Watersheds in Contour Detection.“ In: *Int. Workshop on Image Processing*. CCETT/IRISA. 1979.

[23] Serge Beucher and Fernand Meyer. „The morphological approach to segmentation: the watershed transformation.“ In: *Optical Engineering* 34 (1992), pp. 433–433.

[24] Andrea Braides. „A handbook of Γ-convergence.“ In: *Handbook of Differential Equations: stationary partial differential equations*. Vol. 3. Elsevier, 2006, pp. 101–213.

[25] Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. „Monet: Unsupervised scene decomposition and representation.“ In: *arXiv preprint arXiv:1901.11390* (2019).

[26] Challenge CREMI. *MICCAI Challenge on Circuit Reconstruction from Electron Microscopy Images*. 2017. URL: http://cremi.org/ (visited on 02/23/2017).

[27] Jinzheng Cai, Le Lu, Zizhao Zhang, Fuyong Xing, Lin Yang, and Qian Yin. „Pancreas Segmentation in MRI Using Graph-Based Decision Fusion on Convolutional Neural Networks.“ In: *Proc. MICCAI*. 2016.

[28] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. „Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs.“ In: *ICLR*. 2016.

[29] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. „Rethinking Atrous Convolution for Semantic Image Segmentation.“ In: *CoRR* abs/1706.05587 (2017).

[30] 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)*. 2018, pp. 801–818.

[31] Mickaël Chen, Thierry Artières, and Ludovic Denoyer. „Unsupervised object segmentation by redrawing.“ In: *Advances in Neural Information Processing Systems*. 2019, pp. 12705–12716.

[32] Zeyuan Chen, Shaoliang Nie, Tianfu Wu, and Christopher G. Healey. „High Resolution Face Completion with Multiple Controllable Attributes via Fully End-to-End Progressive Generative Adversarial Networks.“ In: *CoRR* abs/1801.07632 (2018).

[33] Özgün Çiçek, 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.

[34] Dan C Ciresan, Alessandro Giusti, Luca M Gambardella, and Jurgen Schmidhuber. „Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images.“ In: *Proc. NIPS'12* (2012).

[35] 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: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[36] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009.

[37] Camille Couprie, Leo Grady, Laurent Najman, and Hugues Talbot. „Power watershed: A unifying graph-based optimization framework.“ In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.7 (2011).

[38] Michel Couprie and Gilles Bertrand. „Topological gray-scale watershed transformation.“ In: *Vision Geometry VI*. Vol. 3168. International Society for Optics and Photonics. 1997, pp. 136–146.

[39] 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* (2009).

[40] Jifeng Dai, Kaiming He, and Jian Sun. „Instance-Aware Semantic Segmentation via Multi-Task Network Cascades.“ en. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 3150–3158.

[41] Gianni Dal Maso. *An introduction to Γ-convergence*. Vol. 8. Springer Science & Business Media, 2012.

[42] Lee R Dice. „Measures of the amount of ecologic association between species.“ In: *Ecology* 26.3 (1945), pp. 297–302.

[43] Iddo Drori, Daniel Cohen-Or, and Hezy Yeshurun. „Fragment-based image completion.“ In: *ACM SIGGRAPH*. 2003, pp. 303–312.

[44] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. „Dermatologist-level classification of skin cancer with deep neural networks.“ In: *Nature* 542.7639 (2017), pp. 115–118.

[45] Alexandre X Falcão, Jorge Stolfi, and Roberto de Alencar Lotufo. „The image foresting transform: Theory, algorithms, and applications.“ In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.1 (2004), pp. 19–29.

[46] Alireza Fathi, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama, and Kevin P. Murphy. „Semantic Instance Segmentation via Deep Metric Learning.“ en. In: *arXiv:1703.10277 [cs]* (2017).

[47] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. „Efficient Graph-Based Image Segmentation.“ en. In: *International Journal of Computer Vision* 59.2 (2004), pp. 167–181.

[48] Charless Fowlkes, David R. Martin, and Jitendra Malik. „Learning affinity functions for image segmentation: combining patch-based and gradient-based approaches.“ In: *Proc. CVPR*. 2003.

[49] Jan Funke, Fabian David 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* (2018).

[50] Leo Grady. „Random walks for image segmentation.“ In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.11 (2006), pp. 1768–1783.

[51] Michel Grimaud. „New measure of contrast: the dynamics.“ In: *Proc. Image Algebra and Morphological Processing*. Ed. by P. D. Gader, E. R. Dougherty, & J. C. Serra. Vol. 1769. SPIE Conf. Series. 1992, pp. 292–305.

[52] Laurent Guigues, Jean Pierre Cocquerez, and Hervé Le Men. „Scale-sets image analysis.“ In: *International Journal of Computer Vision* 68.3 (2006), pp. 289–317.

[53] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. „Deep Residual Learning for Image Recognition.“ In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[54] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. „Mask R-CNN.“ In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988.

[55] Peter Hirsch, Lisa Mais, and Dagmar Kainmueller. „PatchPerPix for Instance Segmentation.“ In: *arXiv preprint arXiv:2001.07626* (2020).

[56] 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.

[57] Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H Adelson. „Learning visual groups from co-occurrences in space and time.“ In: *arXiv preprint arXiv:1511.06811* (2015).

[58] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. „Image-to-image translation with conditional adversarial networks.“ In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134.

[59] Viren Jain, Joseph F Murray, Fabian Roth, Srinivas Turaga, Valentin Zhigulin, Kevin L Briggman, Moritz N Helmstaedter, Winfried Denk, and H Sebastian Seung. „Supervised learning of image restoration with convolutional networks.“ In: *Proc. ICCV'07* (2007), pp. 1–8.

[60] Michał Januszewski, Jörgen Kornfeld, Peter H Li, Art Pope, Tim Blakely, Larry Lindsey, Jeremy Maitin-Shepard, Mike Tyka, Winfried Denk, and Viren Jain. „High-precision automated reconstruction of neurons with flood-filling networks.“ In: *Nature methods* (2018), p. 1.

[61] Florian Jug, Evgeny Levinkov, Corinna Blasse, Eugene W Myers, and Bjoern Andres. „Moral lineage tracing.“ In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 5926–5935.

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

[63] Jörg Hendrik Kappes, Markus Speth, Gerhard Reinelt, and Christoph Schnörr. „Higher-order segmentation via multicuts.“ In: *Computer Vision and Image Understanding* 143 (2016), pp. 104–119.

[64] 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.

[65] Margret Keuper, Evgeny Levinkov, Nicolas Bonneel, Guillaume Lavoué, Thomas Brox, and Bjorn Andres. „Efficient decomposition of image and mesh graphs by lifted multicuts.“ In: *Proc. ICCV'15*. 2015, pp. 1751–1759.

[66] Margret Keuper, Siyu Tang, Yu Zhongjie, Bjoern Andres, Thomas Brox, and Bernt Schiele. „A multi-cut formulation for joint segmentation and tracking of multiple objects.“ In: *arXiv preprint arXiv:1607.06317* (2016).

[67] Sungwoong Kim, Chang D Yoo, Sebastian Nowozin, and Pushmeet Kohli. „Image segmentation using higher-order correlation clustering.“ In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.9 (2014), pp. 1761–1774.

[68] D Kinga and J Ba Adam. „A method for stochastic optimization.“ In: *International Conference on Learning Representations (ICLR)*. Vol. 5. 2015.

[69] B. Ravi Kiran and Jean Serra. „Global–local optimizations by hierarchical cuts and climbing energies.“ In: *Pattern Recognition* 47.1 (2014), pp. 12–24.

[70] 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*. 2017, pp. 5008–5017.

[71] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. „Panoptic feature pyramid networks.“ In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6399–6408.

[72] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. „Panoptic segmentation.“ In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019, pp. 9404–9413.

[73] Seymour Knowles-Barley, Verena Kaynig, Thouis Ray Jones, Alyssa Wilson, Joshua Morgan, Dongil Lee, Daniel Berger, Narayanan Kasthuri, Jeff W Lichtman, and Hanspeter Pfister. „RhoanaNet Pipeline: Dense Automatic Neural Annotation.“ In: *arXiv:1611.06973* (2016).

[74] Rolf Köhler, Christian Schuler, Bernhard Schölkopf, and Stefan Harmeling. „Mask-specific inpainting with deep neural networks.“ In: *German Conference on Pattern Recognition*. Springer. 2014, pp. 523–534.

[75] Pushmeet Kohli, Alexander Shekhovtsov, Carsten Rother, Vladimir Kolmogorov, and Philip Torr. „On partial optimality in multi-label MRFs.“ In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 480–487.

[76] Iasonas Kokkinos. „Surpassing Humans in Boundary Detection using Deep Learning.“ In: *4th International Conference on Learning Representations, ICLR 2016, Conference Track Proceedings*. 2016.

[77] Vladimir Kolmogorov and Ramin Zabin. „What energy functions can be minimized via graph cuts?“ In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.2 (2004), pp. 147–159.

[78] Shu Kong and Charless C. Fowlkes. „Recurrent Pixel Embedding for Instance Grouping.“ In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 2018, pp. 9018–9028.

[79] Niko Krasowski, Thorsten Beier, Graham Knott, Ulrich Kothe, Fred A. Hamprecht, and Anna Kreshuk. „Neuron Segmentation With High-Level Biological Priors.“ en. In: *IEEE Transactions on Medical Imaging* 37.4 (2018), pp. 829–839.

[80] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. „ImageNet Classification with Deep Convolutional Neural Networks.“ en. In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[81] Thorben Kroeger, Jörg H. Kappes, Thorsten Beier, Ullrich Koethe, and Fred A. Hamprecht. „Asymmetric Cuts: Joint Image Labeling and Partitioning.“ en. In: *Pattern Recognition*. Vol. 8753. Springer International Publishing, 2014, pp. 199–211.

[82] Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. „Noise2void-learning denoising from single noisy images.“ In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2129–2137.

[83] Alexander Krull, Tomas Vicar, and Florian Jug. „Probabilistic Noise2Void: Unsupervised content-aware denoising.“ In: *arXiv preprint arXiv:1906.00651* (2019).

[84] Joseph B Kruskal. „On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem.“ en. In: *Proceedings of the American Mathematical Society* (1956), p. 3.

[85] Jan-Hendrik Lange, Andreas Karrenbauer, and Bjoern Andres. „Partial Optimality and Fast Lower Bounds for Weighted Correlation Clustering.“ In: *International Conference on Machine Learning*. 2018, pp. 2898–2907.

[86] Paul-Friedrich Langenbruch and Norbert Weissenfels. „Canal systems and choanocyte chambers in freshwater sponges (Porifera, Spongillidae).“ In: *Zoomorphology* 107.1 (1987), pp. 11–16.

[87] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. „Learning representations for automatic colorization.“ In: *European Conference on Computer Vision*. Springer. 2016, pp. 577–593.

[88] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. „Colorization as a proxy task for visual understanding.“ In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6874–6883.

[89] 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).

[90] Kisuk Lee, Ran Lu, Kyle Luther, and H Sebastian Seung. „Learning Dense Voxel Embeddings for 3D Neuron Reconstruction.“ In: *arXiv preprint arXiv:1909.09872* (2019).

[91] Zohar Levi and Denis Zorin. „Strict Minimizers for Geometric Optimization.“ In: *ACM Transactions on Graphics* 33.6 (2014), 185:1–185:14.

[92] Evgeny Levinkov, Alexander Kirillov, and Bjoern Andres. „A Comparative Study of Local Search Algorithms for Correlation Clustering.“ In: *German Conference on Pattern Recognition*. Springer. 2017, pp. 103–114.

[93] Evgeny Levinkov, Jonas Uhrig, Siyu Tang, Mohamed Omran, Eldar Insafutdinov, Alexander Kirillov, Carsten Rother, Thomas Brox, Bernt Schiele, and Bjoern Andres. „Joint Graph Decomposition & Node Labeling: Problem, Algorithms, Applications.“ en. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 1904–1912.

[94] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. „Continuous control with deep reinforcement learning.“ In: *4th International Conference on Learning Representations, ICLR 2016, Conference Track Proceedings*. 2016.

[95] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. „Feature Pyramid Networks for Object Detection.“ In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[96] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L. Yuille, and Li Fei-Fei. „Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation.“ In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[97] Guilin Liu, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. „Image Inpainting for Irregular Holes Using Partial Convolutions.“ In: *The European Conference on Computer Vision (ECCV)*. 2018.

[98] Yiding Liu, Siyu Yang, Bin Li, Wengang Zhou, Ji-Zeng Xu, Houqiang Li, and Yan Lu. „Affinity Derivation and Graph Merge for Instance Segmentation.“ en. In: *The European Conference on Computer Vision (ECCV)*. 2018, p. 18.

[99] 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.

[100] Michael Maire, Takuya Narihira, and Stella X. Yu. „Affinity CNN: Learning Pixel-Centric Pairwise Relations for Figure/Ground Embedding.“ In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 174–182.

[101] Filip Malmberg, Robin Strand, and Ingela Nyström. „Generalized hard constraints for graph segmentation.“ In: *Scandinavian Conference on Image Analysis*. Springer. 2011, pp. 36–47.

[102] Francisco Massa and Ross Girshick. „Maskrcnn-Benchmark: Fast, Modular Reference Implementation of Instance Segmentation and Object Detection Algorithms in PyTorch.“ In: (2018).

[103] Marina Meila. „Comparing clusterings: an axiomatic view.“ In: *Proc. ICML'05*. 2005, pp. 577–584.

[104] Yaron Meirovitch, Alexander Matveev, Hayk Saribekyan, David Budden, David Rolnick, Gergely Odor, Seymour Knowles-Barley Thouis Raymond Jones, Hanspeter Pfister, Jeff William Lichtman, and Nir Shavit. „A Multi-Pass Approach to Large-Scale Connectomics.“ In: *arXiv preprint:1612.02120* (2016).

[105] Yaron Meirovitch, Lu Mi, Hayk Saribekyan, Alexander Matveev, David Rolnick, and Nir Shavit. „Cross-classification clustering: An efficient multi-object tracking technique for 3-d instance segmentation in connectomics.“ In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8425–8435.

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

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

[108] Fernand Meyer. „Morphological multiscale and interactive segmentation.“ In: *NSIP*. 1999, pp. 369–377.

[109] Fernand Meyer. „Watersheds on weighted graphs.“ In: *Pattern Recognition Letters* 47 (2014), pp. 72 –79.

[110] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, et al. „Asynchronous methods for deep reinforcement learning.“ In: *Proc. ICML'16*. 2016.

[111] D Muhammad Noorul Mubarak, M Mohamed Sathik, S Zulaikha Beevi, and K Revathy. „A hybrid region growing algorithm for medical image segmentation.“ In: *International Journal of Computer Science & Information Technology* 4.3 (2012), p. 61.

[112] Jacob M Musser, Klaske J Schippers, Michael Nickel, Giulia Mizzon, Andrea B Kohn, Constantin Pape, Jörg U Hammel, Florian Wolf, Cong Liang, Ana Hernández-Plaza, et al. „Profiling cellular diversity in sponges informs animal cell type and nervous system evolution.“ In: *BioRxiv* (2019), p. 758276.

[113] Laurent Najman. „On the equivalence between hierarchical segmentations and ultrametric watersheds.“ In: *Journal of Mathematical Imaging and Vision* 40.3 (2011), pp. 231–247.

[114] Laurent Najman. „Extending the power watershed framework thanks to Γ-convergence.“ In: *SIAM Journal on Imaging Sciences* 10.4 (2017), pp. 2275–2292.

[115] Laurent Najman and Michel Schmitt. „Geodesic saliency of watershed contours and hierarchical segmentation.“ In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18.12 (1996), pp. 1163–1173.

[116] Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Qureshi, and Mehran Ebrahimi. „EdgeConnect: Structure Guided Image Inpainting using Edge Prediction.“ In: *The IEEE International Conference on Computer Vision (ICCV) Workshops*. 2019.

[117] Juan Nunez-Iglesias, Ryan Kennedy, Toufiq Parag, Jianbo Shi, and Dmitri Chklovskii. „Machine learning of hierarchical clustering to segment 2D and 3D images.“ In: *PLoS one* 8 (2013), e71715.

[118] Pavel Ostyakov, Roman Suvorov, Elizaveta Logacheva, Oleg Khomenko, and Sergey I Nikolenko. „Seigan: Towards compositional image generation by simultaneously learning to segment, enhance, and inpaint.“ In: *arXiv preprint arXiv:1811.07630* (2018).

[119] Constantin Pape, Thorsten Beier, Peter Li, Viren Jain, Davi D Bock, and Anna Kreshuk. „Solving large multicut problems for connectomics via domain decomposition.“ In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2017, pp. 1–10.

[120] 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.

[121] Toufiq Parag, Fabian Tschopp, William Grisaitis, Srinivas C Turaga, Xuewen Zhang, Brian Matejek, Lee Kamentsky, Jeff W Lichtman, and Hanspeter Pfister. „Anisotropic EM Segmentation by 3D Affinity Learning and Agglomeration.“ In: *arXiv preprint 1707.08935* (2017).

[122] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. „Context encoders: Feature learning by inpainting.“ In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2536–2544.

[123] Benjamin Perret, Jean Cousty, Silvio Jamil F Guimaraes, and Deise S Maia. „Evaluation of hierarchical watersheds.“ In: *IEEE Transactions on Image Processing* 27.4 (2018), pp. 1676–1688.

[124] Regina Pohle and Klaus D Toennies. „Segmentation of medical images using adaptive region growing.“ In: *Medical Imaging 2001: Image Processing*. Vol. 4322. International Society for Optics and Photonics. 2001, pp. 1337–1346.

[125] S Poonguzhali and G Ravindran. „A complete automatic region growing method for segmentation of masses on ultrasound images.“ In: *2006 International Conference on Biomedical and Pharmaceutical Engineering*. IEEE. 2006, pp. 88–92.

[126] Tran Minh Quan, David GC Hilderbrand, and Won-Ki Jeong. „FusionNet: A deep fully residual convolutional neural network for image segmentation in connectomics.“ In: *arXiv:1612.05360* (2016).

[127] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.“ In: *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015, pp. 91–99.

[128] Yurui Ren, Xiaoming Yu, Ruonan Zhang, Thomas H Li, Shan Liu, and Ge Li. „StructureFlow: Image Inpainting via Structure-aware Appearance Flow.“ In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 181–190.

[129] Jos BTM Roerdink and Arnold Meijster. „The watershed transform: Definitions, algorithms and parallelization strategies.“ In: *Fundamenta informaticae* 41.1, 2 (2000), pp. 187–228.

[130] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. „U-Net: Convolutional Networks for Biomedical Image Segmentation.“ In: *Proc. MICCAI'15* (2015), pp. 234–241.

[131] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. „U-Net: Convolutional Networks for Biomedical Image Segmentation.“ In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[132] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. „U-net: Convolutional networks for biomedical image segmentation.“ In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[133] Carsten Rother, Vladimir Kolmogorov, Victor Lempitsky, and Martin Szummer. „Optimizing binary MRFs via extended roof duality.“ In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2007, pp. 1–8.

[134] Olga Russakovsky et al. „ImageNet Large Scale Visual Recognition Challenge.“ In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252.

[135] Challenge SNEMI3D. *ISBI 2013 challenge: 3D segmentation of neurites in EM images*. 2017. URL: http://brainiac2.mit.edu/SNEMI3D/ (visited on 01/01/2020).

[136] Virginia R de Sa. „Learning classification with unlabeled data.“ In: *Advances in neural information processing systems*. 1994, pp. 112–119.

[137] Philippe Salembier and Luis Garrido. „Binary Partition Tree as an Efficient Representation for Image Processing, Segmentation, and Information Retrieval.“ In: *IEEE Transactions on Image Processing* 9 (2000), pp. 561–576.

[138] Rodrigo Santa Cruz, Basura Fernando, Anoop Cherian, and Stephen Gould. „DeepPermNet: Visual Permutation Learning.“ In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 3949–3957.

[139] Philipp Schlegel, Marta Costa, and Gregory S X E Jefferis. „Learning from connectomics on the fly.“ In: *Current opinion in insect science* 24 (2017), pp. 96–105.

[140] Juan Shan, Heng-Da Cheng, and Yuxuan Wang. „A novel automatic seed point selection algorithm for breast ultrasound images.“ In: *2008 19th International Conference on Pattern Recognition*. IEEE. 2008, pp. 1–4.

[141] Wei Shen, Bin Wang, Yuan Jiang, Yan Wang, and Alan L. Yuille. „Multi-stage Multi-recursive-input Fully Convolutional Networks for Neuronal Boundary Detection.“ In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2410–2419.

[142] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, et al. „Deterministic Policy Gradient Algorithms.“ In: *Proc. ICML'14*. 2014.

[143] Karen Simonyan and Andrew Zisserman. „Very Deep Convolutional Networks for Large-Scale Image Recognition.“ In: *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*. 2015.

[144] Pierre Soille. „Constrained Connectivity for Hierarchical Image Decomposition and Simplification.“ In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.7 (2008), pp. 1132–1145.

[145] Thorvald Sørensen. „A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons.“ In: *Biol. Skr.* 5 (1948), pp. 1–34.

[146] Jian Sun, Lu Yuan, Jiaya Jia, and Heung-Yeung Shum. „Image completion with structure propagation.“ In: *ACM SIGGRAPH*. 2005, pp. 861–868.

[147] Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. „Policy gradient methods for reinforcement learning with function approximation.“ In: *Proc. NIPS'99*. 1999.

[148] Joseph Tighe, Marc Niethammer, and Svetlana Lazebnik. „Scene Parsing with Object Instance Inference Using Regions and Per-Exemplar Detectors.“ en. In: *International Journal of Computer Vision* 112.2 (2015), pp. 150–171.

[149] Trieu H Trinh, Minh-Thang Luong, and Quoc V Le. „Selfie: Self-supervised pretraining for image embedding.“ In: *arXiv preprint arXiv:1906.02940* (2019).

[150] Zhuowen Tu, Xiangrong Chen, Alan L. Yuille, and Song-Chun Zhu. „Image Parsing: Unifying Segmentation, Detection, and Recognition.“ en. In: *International Journal of Computer Vision* 63.2 (2005), pp. 113–140.

[151] Srinivas C. Turaga, Kevin L. Briggman, Moritz Helmstaedter, Winfried Denk, and H. Sebastian Seung. „Maximin Affinity Learning of Image Segmentation.“ In: *Proceedings of the 22nd International Conference on Neural Information Processing Systems*. NIPS'09. Curran Associates Inc., 2009, 1865–1873.

[152] 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.

[153] Vladimír Ulman, Martin Maška, Klas EG Magnusson, Olaf Ronneberger, Carsten Haubold, Nathalie Harder, Pavel Matula, Petr Matula, David Svoboda, Miroslav Radojevic, et al. „An objective comparison of cell-tracking algorithms.“ In: *Nature methods* 14.12 (2017), p. 1141.

[154] Mustafa Gökhan Uzunbaş, Chao Chen, and Dimitris Metaxas. „Optree: a learning-based adaptive watershed algorithm for neuron segmentation.“ In: *Int. Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI'14)*. 2014, pp. 97–105.

[155] Corinne Vachier and Fernand Meyer. „Extinction value: a new measurement of persistence.“ In: *Worksh. Nonlinear Signal and Image Processing*. Vol. 1. 1995, pp. 254–257.

[156] Corinne Vachier and Fernand Meyer. „The viscous watershed transform.“ In: *J. Math. Imaging and Vision* 22.2 (2005), pp. 251–267.

[157] Luc Vincent and Pierre Soille. „Watersheds in digital spaces: an efficient algorithm based on immersion simulations.“ In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6 (1991), pp. 583–598.

[158] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. „Tracking emerges by colorizing videos.“ In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 391–408.

[159] Xiaolong Wang, Allan Jabri, and Alexei A Efros. „Learning correspondence from the cycle-consistency of time.“ In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2566–2576.

[160] Maurice Weiler, Fred A. Hamprecht, and Martin Storath. „Learning Steerable Filters for Rotation Equivariant CNNs.“ In: 2018, pp. 849–858.

[161] Steffen Wolf, Fred A. Hamprecht, and Jan Funke. „Instance Separation Emerges from Inpainting.“ In: *arXiv preprint arXiv:2003.00891* (2020).

[162] Steffen Wolf, Lukas Schott, Ullrich Köthe, and Fred A. Hamprecht. „Learned Watershed: End-to-End Learning of Seeded Segmentation.“ In: *Proc. ICCV'17* (2017).

[163] 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).

[164] Steffen Wolf, Yuyan Li, Constantin Pape, Alberto Bailoni, Anna Kreshuk, and Fred A. Hamprecht. „The Semantic Mutex Watershed for Efficient Bottom-Up Semantic Instance Segmentation.“ In: *arXiv preprint arXiv:1912.12717* (2019).

[165] Steffen Wolf, Alberto Bailoni, Constantin Pape, Nasim Rahaman, Anna Kreshuk, Ullrich Köthe, and Fred A. Hamprecht. „The Mutex Watershed and Its Objective: Efficient, Parameter-Free Image Partitioning.“ In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[166] Jie Wu, Skip Poehlman, Michael D Noseworthy, and Markad V Kamath. „Texture feature based automated seeded region growing in abdominal MRI segmentation.“ In: *2008 International Conference on BioMedical Engineering and Informatics*. Vol. 2. IEEE. 2008, pp. 263–267.

[167] Chi Xiao, Jing Liu, Xi Chen, Hua Han, Chang Shu, and Qiwei Xie. „Deep contextual residual network for electron microscopy image segmentation in connectomics.“ In: *Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on*. IEEE. 2018, pp. 378–381.

[168] Junyuan Xie, Linli Xu, and Enhong Chen. „Image denoising and inpainting with deep neural networks.“ In: *Advances in neural information processing systems*. 2012, pp. 341–349.

[169] Saining Xie and Zhuowen Tu. „Holistically-nested edge detection.“ In: *Proc. ICCV'15*. 2015, pp. 1395–1403.

[170] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. „High-resolution image inpainting using multi-scale neural patch synthesis.“ In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6721–6729.

[171] Jian Yao, S. Fidler, and R. Urtasun. „Describing the Scene as a Whole: Joint Object Detection, Scene Classification and Semantic Segmentation.“ en. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 702–709.

[172] Julian Yarkony, Alexander Ihler, and Charless C Fowlkes. „Fast planar correlation clustering for image segmentation.“ In: *European Conference on Computer Vision*. Springer. 2012, pp. 568–581.

[173] Julian Yarkony, Thorsten Beier, Pierre Baldi, and Fred A. Hamprecht. „Parallel multicut segmentation via dual decomposition.“ In: *International Workshop on New Frontiers in Mining Complex Patterns*. Springer. 2014, pp. 56–68.

[174] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Sang Nong. „Learning a Discriminative Feature Network for Semantic Segmentation.“ In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1857–1866.

[175] Fisher Yu and Vladlen Koltun. „Multi-Scale Context Aggregation by Dilated Convolutions.“ In: *4th International Conference on Learning Representations, ICLR 2016, Conference Track Proceedings*. 2016.

[176] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. „Free-form image inpainting with gated convolution.“ In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 4471–4480.

[177] Charles T Zahn. „Graph-theoretical methods for detecting and describing gestalt clusters.“ In: *IEEE Transactions on computers* 100.1 (1971), pp. 68–86.

[178] Yanhong Zeng, Jianlong Fu, Hongyang Chao, and Baining Guo. „Learning pyramid-context encoder network for high-quality image inpainting.“ In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019, pp. 1486–1494.

[179] C. Zhang, Julian Yarkony, and Fred A. Hamprecht. „Cell detection and segmentation using correlation clustering.“ In: *Proc. MICCAI'14*. 2014, pp. 9–16.

[180] Richard Zhang, Phillip Isola, and Alexei A Efros. „Colorful image colorization.“ In: *European conference on computer vision*. Springer. 2016, pp. 649–666.

[181] Richard Zhang, Phillip Isola, and Alexei A Efros. „Split-brain autoencoders: Unsupervised learning by cross-channel prediction.“ In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1058–1067.

[182] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. „Pyramid scene parsing network.“ In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2881–2890.

[183] Zhihao Zheng, J Scott Lauritzen, Eric Perlman, Camenzind G Robinson, Matthew Nichols, Daniel Milkie, Omar Torrens, John Price, Corey B Fisher, Nadiya Sharifi, et al. „A complete electron microscopy volume of the brain of adult Drosophila melanogaster.“ In: *Cell* 174.3 (2018), pp. 730–743.

[184] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. „Unpaired image-to-image translation using cycle-consistent adversarial networks.“ In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232.

[185] Aleksandar Zlateski and H. Sebastian Seung. „Image Segmentation by Size-Dependent Single Linkage Clustering of a Watershed Basin Graph.“ In: *CoRR* abs/1505.00249 (2015).

List of Figures

List of Tables

List of Algorithms

Colophon

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX: