Ruprecht-Karls-Universität Heidelberg
Faculty of Biosciences
Master Program Molecular Biotechnology

A Machine-Learning Pipeline for the Automated Segmentation of Neurons from Serial Block-Face Electron Microscopy Data

Master Thesis

Julia Milena Buhmann
born in Paris (France)

September 2014

The presented Master Thesis was performed at the Max Planck Institute for Medical Research in Heidelberg for completion of the Master's program Molecular Biotechnology at the Faculty of Bioscience at the University of Heidelberg in the time from 1st of April to the 31st of August 2014.

First Referee:
Dr. Rolf Sprengel
Department: Molecular Neurobiology
Max Planck Institute for Medical Research in Heidelberg

Second Referee:
Prof. Dr. Winfried Denk
Department: Biomolecular Optics
Max Planck Institute for Medical Research in Heidelberg

I herewith declare that

1. I wrote this Master Thesis independently under supervision and that I used no other sources and supporting materials than those indicated;
2. the adoption of quotation from the literature/internet as well as thoughts from other authors were indicated in the thesis;
3. my Master Thesis was not submitted to any other examination.

I am aware of the fact that a false declaration will have legal consequences.

Heidelberg, September 2014
Julia Buhmann

Abstract

High-resolution volume electron microscopy datasets of neural systems have the potential to change neuroscience by resolving entire neural circuits at synaptic level. The size of current datasets in the range of several terabytes however requires the automatization of image analysis and neural reconstruction.

We address the problem of neurite reconstruction by a machine-learning based workflow. The proposed pipeline predicts probabilities for merging adjacent supervoxels by incorporating voxel classifications generated by a Convolutional Neural Network. The resulting supervoxel merge-probability graph is further processed with a multicut graph-cut algorithm, resolving inconsistent merging decisions. We can show that the incorportation of various semantic biological classes such as mitchondrion, vesicle cloud and synaptic junction is beneficial for segmentation performance. Our best model scores an error free path length of about 10 µm on a difficult-to-analyze dataset. A preliminary comparison between the speed of manual skeletonization and proofreading of automated volume segmentation generated with our pipeline showed similar time requirements.

Zusammenfassung

Hochaufgelöste elektronenmikroskopische Bilddatensätze von neuronalen Systemen haben das Potential, die Neurowissenschaften zu verändern, indem komplette neuronale Schaltpläne aufgeklärt werden. Solche Datensätze sind gegenwärtig einige Terabytes groß. Diese Größe erfordert die Automatisierung von Bildanalyse und Nervenzellrekonstruktion.

In der vorliegenden Arbeit wird dem Problem der Nervenzellrekonstruktion mit einer Pipeline begegnet, die auf maschinellem Lernen basiert. Die Pipeline klassifiziert das Zusammenwachsen von benachbarten Supervoxeln. Die Entscheidung basiert hierbei auf Voxelklassifizierungen, die durch ein Convolutional Neural Network generiert werden. Der hierbei entstehende Graph, der die Wahrscheinlichkeiten des Zusammenwachsens beinhaltet, wird mit einem Multicut Graph-Cut Algorithmus weiter prozessiert. Der Multicut Algorithmus löst inkonsistente Fusionswahrscheinlichkeiten von Supervoxeln auf. Wir konnten zeigen, dass durch das Hinzufügen von verschiedenen biologischen Klassen wie Mitochondrion, Vesikelwolken und synaptischen Kontakten die Qualität der Segmentierung besser wird. Die beste Segmentierung erzielte einen Wert von 10 µm fehlerfreier Weglänge in einem schwer zu analysierendem Datensatz. Ein erster Geschwindigkeitsvergleich zeigt, dass das Korrekturlesen der automatisch erstellten Volumensegmentierung nicht länger dauert als die manuelle Skeletonisierung der Daten.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Structural neurobiology

The brain remains an unresolved enigma until to date, although multiple aspects at microscopic and macroscopic scale have been elucidated in the last 100 years. At first, the anatomic work of Golgi and Ramón y Cajal at the turn of the 19^{th} century has brought insight on the structural characteristics of neurons [1]. A detailed model of the electro-physiology of single neurons came up with the work of Hodgkin and Huxley explaining the ionic mechanism of an action potential [2] and with the insight on how individual neurons communicate via synapses [3–5]. Despite this progress, a cohesive model of the brain is still pending.
This discrepancy can be explained by the fact that one key aspect of the brain - neural circuit structure - remains elusive [6]. Indeed, neural networks might be essential in understanding how the algorithms of the brain are 'implemented' [7]. The knowledge on neural circuits will allow to test models of neural computation. Publications however on resolving neural networks at single-neurite resolution have been extremely rare not least because of the technical challenges associated with data acquisition [8].

1.1.1 Volume Electron Microscopy Techniques

Data acquisition for the purpose of neural reconstruction must fulfill two main requirements: Firstly, it must occur at very high resolution, e.g nerve cells can have diameters as low as 30 nm[1] [9] and synaptic connections must be detectable. Secondly, acquisition must be highly scalable, as neurons can branch out over long distances (e.g. pyramidal cell span over a cubic millimetre in the mouse cerebral cortex) [7]. Given

[1] measured in fly brain tissue

the needed high resolution and staining completeness, only electron microscopy (EM) is currently suitable. The challenge today lies in upscaling this technique for imaging very large data sets.

Different approaches have been developed to accomplish EM imaging in 3D. They are all based on section-wise imaging. The traditional strategy is to first cut-off a slice from the tissue block, that is then collected and recorded by means of transmission EM (TEM) (called serial-section TEM, ssTEM). In 1986, this technique was successfully applied for the reconstruction of the entire nervous system of *C. elegans* [10]. Sections can also be recorded by means of scanning EM (SEM). The automatization of the section-collection process (Automated Tape-collection Ultra Microtome, ATUM [11, 12][2]) allows recording of volume sizes of 400 µm or more at a resolution of 4x4x35 nm [13].

A conceptually different method is block-face imaging using SEM. The block face of a tissue sample is recorded first and the slice is taken off afterwards. The section is either removed by means of a diamond knife (Serial block face SEM - SBEM [14]) or by a focused ion beam (FIB-SEM[15]).

All presented techniques have their strengths and weaknesses. Block face imaging techniques generate thinner sections (FIB-SEM: 5 nm, SBEM: 20 nm) than traditional EM techniques (∼45 nm) [12], because the section does not have to remain intact. FIB-SEM is restricted to a total volume size of $100\,\mu m^3$, while SBEM and ATUM have the potential to record much larger volume blocks [13]. ATUM (and other collection techniques [16]) have the advantage that the recorded sections remain intact, enabling repeated recording, correlative fluorescence microscopy [17], potential parallelization, and random access. On the other side, those techniques are prone to section loss and physical distortion challenging subsequent registration post-processing [18].

1.1.2 Ultrastructural Features of Neurons

Neurons, together with glial cells make up the nervous tissue. A neuron is comprised of a cell body (called soma), dendrites and an axon (the term *neurite* denotes both dendrites and axons). Multiple dendritic branches project from the soma in order to receive and propagate information to the cell body. The axon transmits this information on to other neurons, muscles or glands. Some axons are wrapped by an insulating sheath, the myelin, in order to increase impulse speed [19]. Axons tend to be thin (0.1-0.5 µm, rat neocortex) but enlarge when they form synapses (0.7 µm in mean diameter, rat neocortex) [20]. Dendrites in contrast usually have a large dendritic shaft (e.g. dendrites in rat hippocampus have a diameter range of 0.3 to 1.0 µm [21, 22]), but can posses small

[2]First generation: Automated Lathe Tape-collection Ultra Microtome -ATLUM

protrusions, called *spines*, that receive input from axons. Spines are characterized by a tuberous head (the *spine head*, volume 0.001-1 µm) and a thin connection to the shaft (the *spine neck*, diameter < 0.1 µm) [23]. The predominant glial cell type in the nervous system are astrocytes. They sourround the neurons in an almost fluid-like manner, forming a supplying network for them. Astrocytes are potentially also involved in information processing [24].

In EM recordings, cells and their organelles are recognized by distinct features (see Figure 1.1). Cells are marked by an enclosing dark membrane [3]. The space within a cell (intracellular space, ICS) and the region in-between cells (extracellular space, ECS) are bright. Synapses are characterized by three main attributes: vesicles (*vesicle cloud*) at the presynaptic side, a 20-40 nm [25] wide synaptic cleft and a postsynaptic density. Intracellular organelles are also visible in EM data, first and foremost mitochondria. Those membrane-bound organelles appear dark in the data.

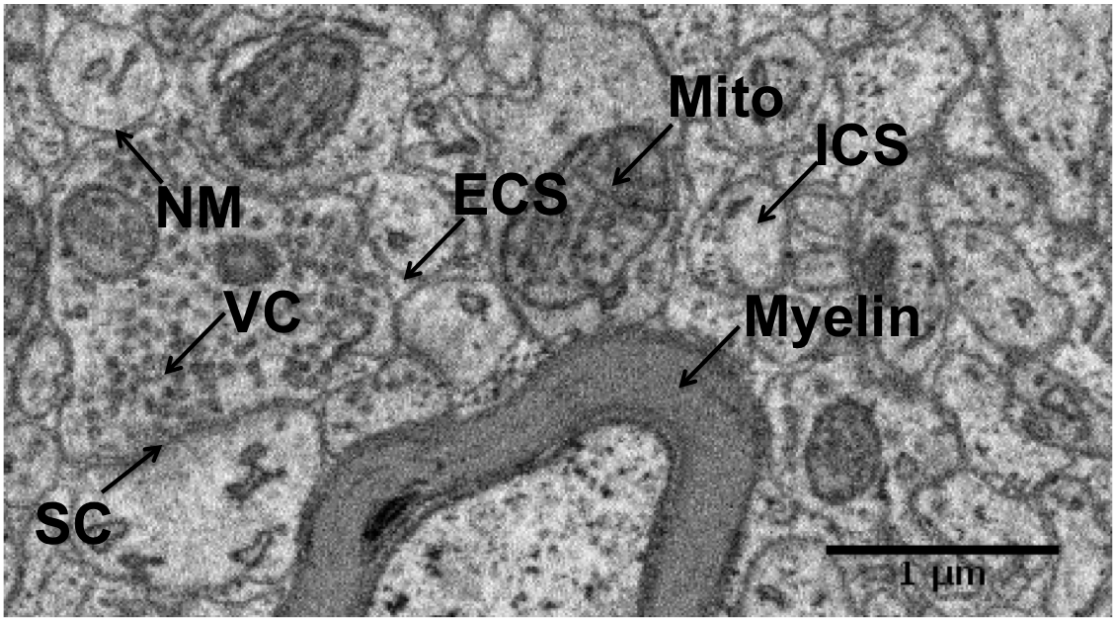


FIGURE 1.1: Appearance of neurite features in EM data of songbird brain tissue. NM: neurite membrane, SC: synaptic cleft, ECS: extracellular space, Mito: mitochondrion, ICS: intracellular space

[3]The exact appearance is staining specific. Here, we refer to EM recordings of osmium stained brain tissue

1.2 Computer vision for automated reconstruction of neural circuits

1.2.1 The reconstruction bottleneck

While data acquisition techniques (see 1.1.1) improve constantly in terms of quality and speed, image analysis is considered a true bottleneck of dense neural circuit reconstruction [9].
Firstly, neural reconstruction can be achieved manually by either tracing the center-line of the neuron (*skeletonization*) or drawing the contour of the neuron in each cross-section (*contouring*). The manual approach was successfully applied in recent local circuit reconstructions [26–28], but e.g. [28] required 30 000 hours of human annotation work. Considering the size of current datasets (10 TB, J. Kornfeld, unpublished data), this approach is certainly not an option for the longer term.
Enormous efforts have been undertaken to tackle the problem of neural reconstruction automatically [29–33, 46]. The complexity of the problem is related to the inherent nature of neurite wires: A few misclassified voxels potentially cause loosing track of a large branch of neuron. Or, in the opposite case, minor errors entail the false merging of two actually seperated neurons. Currently, the accuracy is far behind of what would be needed for the automated reconstruction of full-length neurons [28]. Machine learning algorithms so far had the most promising results and seem suitable for the sophisticated problem of neural reconstruction. Those algorithms will be subject of the following sections.

1.2.2 Machine Learning

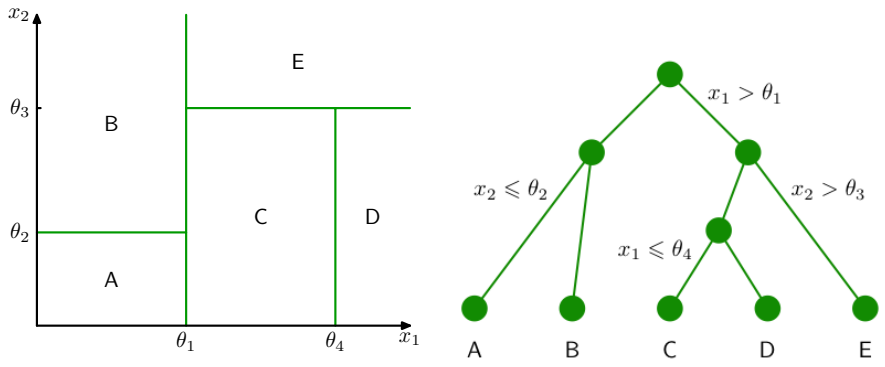
Supervised machine learning algorithms are defined by their capability to learn from data rather than following hand-crafted instructions. A training set $S = \{(x_1, y_1), ..., (x_N, y_N)\}$ consists of N training samples, where $x \in \mathbb{R}^m$ is the feature vector, $y \in Y$ a *target label* and m the number of different features. A training algorithm is applied to the training set in order to obtain a function $f(x)$, called a *classifier*. $f(x)$ maps a previously unknown sample x (that the classifier has never seen before) onto a prediction vector encoded in the same way as the target vector y. Solely based on the training sample, the classifier is hence able to categorize novel samples which is known as *generalization* [34]. In recent years, machine learning algorithms have received new impetus not least because the needed computational power has been reached to date.

1.2.2.1 Random Forest Classifier

One popular machine learning algorithm is the Random Forest Classifier (RFC) [35]. A RFC is an ensemble of decision trees. Given a test sample x, the output probability represents the fraction of all decision trees that voted for the specific class. Random Forests are hence an ensemble learning method, in which multiple learning algorithms are used to increase predictive performance [36].

A decision tree consists of nodes and edges. A node is called a *split node* when it has exactly two child nodes. A *leaf node* does not have any children, but a class label is assigned to it. A classification for a sample $x \in \mathbb{R}^m$ from the test set is made by going along the tree path. At each node, a test function is applied to x in order to decide which of the two child nodes to follow. This process is pursued until a leaf node is reached. The sample x is assigned the leaf node's class.

The test function for RFCs usually partitions the input feature space $\mathbb{R}^m$ with axis-aligned splits. This is illustrated in Figure 1.2. At the first node, the feature space is split into two regions. θ_1 represents the threshold for feature x_1 and its value is specific for the first node. The feature space is recursively split into a set of regions. Each region corresponds to a leaf node with an assigned class. Different regions can have same class labels. A RFC also quantifies the relative feature importance while constructing the tree, which can potentially be used for feature selection.



(A) Two-dimensional input space (B) Binary decision tree

FIGURE 1.2: Illustration of the functioning of a decision tree. (A) The input space has two features x_1 and x_2 and has been partitioned into five regions, which correspond to the leaf nodes of the decision tree in (B). Figures (A) and (B) were adapted from [34].

A decision tree is built up during training. A training sample is used to induce the tree and to set up test functions for each node. The tree is constructed in a greedy fashion. At each node, only that training set is considered, to which the test function of the previous node applied. At each node, only a subset of all features $F \subset \{1, ..., m\}$

is taken into account. In order to find the optimal test function $x_k < \theta$ for the given training subset, it is iterated over all splits θ and over all features $k \in F$ in order to find a split that best divides the set into separate classes. As a measure for a "good" split, the Gini index or entropy is used, quantifiying the impurity degree. Overfitting is prevented via two measures: Firstly, each single tree is only trained on a subset of the original training set (by means of *bagging*) [37]. Secondly, only a random subset of features is chosen to find the test function at each node.

1.2.2.2 Voxel Classification with Convolutional Neural Networks

Many models in computer vision operate on voxel level. Here, a voxel classification is performed based on local image features. In the field of neurite segmentation for instance, voxels are assigned a probability to belong to a neurite membrane or not. The outcome (called a *probability map*) can then be thresholded in order to obtain a segmentation. However, thresholding is only one example. Voxel classification is often only the first of multiple steps within a segmentation pipeline.
Voxel classification can be performed by any machine learning algorithm, although Convolutional Neural Networks (CNNs), a special type of Artificial Neural Networks (ANNs) have proven to be especially powerful.
ANNs are an example of *connectionism*, in which the interconnection of simple units can model any complex function [4]. A simple unit (also called *artificial neuron*) receives input, process the information through a weighted activation function and passes the result on to other units. The parameters of each unit can be learned by means of back-propagation.
In priniciple, an ANN could yield good results for the task of voxel classification given a large training sample and sufficient computational power (a high amount of free parameters would be needed) [34]. This would however ignore a decisive feature of an image: nearby voxels are more correlated than distant voxels. A CNN, introduced in 1980 by a paper of Kunihiko Fukushima [38], explores this property. A CNN consists of layers of feature maps, which can be thought of as 2D images. Feature maps in a given layer are generated by convolutions of feature maps in the previous layer, followed by the application of a nonlinear activation function. The convolution kernels are much smaller than the feature maps. Convolutional layers usually alternate with max-pooling (downsampling) layers. The feature maps in the last layer correspond to desired pixel classification. The kernels for the convolutions are learned by back-propagation [39]. The number of convolutional layers, max-pooling layers and convolutions per convolutional layer are usually optimized empirically.

[4] given a non-linear activation function and a network with at least two-hidden layers [**?**]

1.2.3 Algorithms operating on supervoxel-level

A supervoxel is defined as set of connected voxels that ideally represent a meaningful unit within the image. Operating on supervoxel level reduces redundancy and allows to calculate region based features [40, 41]. One popular method to generate supervoxels is the watershed algorithm, introduced by Beucher and Lantuéj in 1979 [42]. Using the same analogy as Beucher and Lantuéj in their original paper, the grey scale image is pictured as a topographical surface, whereby intensity values are interpreted as heights. In the segmentation process, each valley is a water source from which the relief is flooded. Between water from different sources, a watershed is built up. Those watersheds form the contours in the segmented image. As each local minimum is treated as a separated water source, the algorithm has overall the tendency to produce oversegmented images. Different approaches have been applied of how to segment an image based on supervoxels. One approach is to find for each pair of adjacent supervoxels a label whether the pair should be merged or not. The label can either be calculated hand-designed or predicted based on a machine learning algorithm. Here, local merge decisions generate the final image segmentation (single linkage clustering). [30]. A more global solution is approached by agglomerative clustering of supervoxels, as proposed in [43]. Pairs of clusters are iteratively merged by maximizing a learned similarity function (agglomerative clustering also used in [44]). A third approach reformulates the image segmentation as a graph partitioning problem in order to find a globally optimal solution. The latter will be discussed in more detail in the next section.

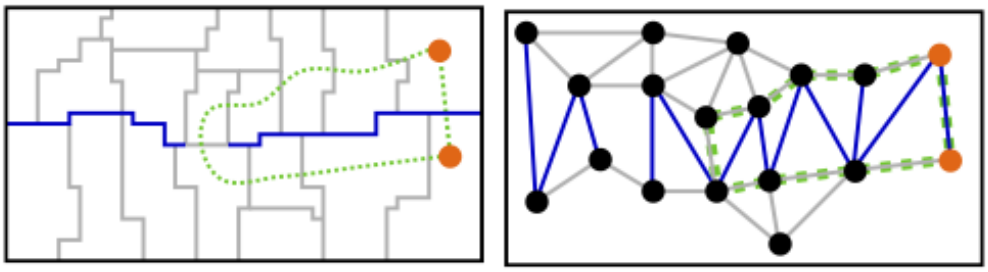
1.2.3.1 A globally optimal solution for image segmentation

Let a labeled image be defined as a cell complex $\zeta = (C, \prec, dim)$, where C constitutes a set of elements $C = \bigcup C_i$ related to each other via $\prec$. dim is a function, that maps each cell $c \in C$ to its dimension: $dim : C \to \mathbb{N}$. $c \prec c'$ indicates that c bounds c' implying that $dim(c) < dim(c')$. For instance, an image with labelled supervoxels can be represented as a 1-dimensional cell complex

$$C = C_0 \cup C_1 \tag{1.1}$$

where $s \in C_1$ are the *segments* and $e \in C_0$ are the *faces* between segments. In this representation, a segmentation can be represented by a joint face labelling $y \in \{0, 1\}^{|C_2|}$. The labeling indicates, whether the two adjacent supervoxels s_1 and s_2 belong to the same object (0) or not (1). Learned likelihood for merging most often produces inconsistent labelings. This means that a face between two adjacent supervoxels might indicate a split though there exists a path connecting the two supervoxels marking them

as belonging to the same segment (all faces along this path are labelled as 0). Figure 1.3 illustrates this duality. The resulting segmentation has thus open surfaces.



(A) (B)

FIGURE 1.3: Duality of edge labeling for image segmentation. Superpixel arrangement (A) and the corresponding region adjacency graph (B) is displayed. Gray edges indicate that the adjacent segments belong to same classes, while blue edges indicate the contrary. Two orange nodes are claimed to belong to different classes via their direct connecting edge, while there exist a path (green dots) declaring them as belonging to the same class. Figures (A) and (B) adapted from [45].

In order to resolve those inconsistencies and potentially close surfaces, a graphical probabilistic approach has been proposed by [45], which models posterior probability for a joint face labeling of all faces given

- features $f_c \in \mathbb{R}^m$ of each face ($\boldsymbol{F}$)
- the connectivity of faces and segments ($\boldsymbol{T}$) encoded in the bounding relation $\prec$.

The posterior probability is given through [5]

$$p(y|\boldsymbol{F},\boldsymbol{T}) \propto \prod_{c\in C_0} \hat{p}(y_c|\boldsymbol{f})p(y_c) \tag{1.2}$$

In order to find the likelihood that best describes the model, 1.2 must be maximized. This is known as the maximal aposteriori probability (MAP) estimate. Instead of maximizing 1.2, we minimize the negative log likelihood by solving the *Integer Linear Program* (ILP). If we consider the prior $p(y_c)$ being the same for all faces, specified through the parameter $\beta \in \{1, 0\}$

$$p(y_c) = \begin{cases} \beta & \text{if } y = 1 \\ 1-\beta & \text{if } y = 0 \end{cases} \tag{1.3}$$

[5] please refer to [46] and [45] for a detailed derivation. Equations mostly adapted from [47]

we derive from (1.2) following ILP

$$\min_{\substack{y\in\{0,1\}^{|C_0|} \\ \text{subject to } y\in MC}} \sum_{c\in C_0} \omega_c y_c \tag{1.4}$$

with

$$\omega_c = \log \frac{p(y_c = 0|f_c)}{p(y_c = 1|f_c)} + \log \frac{1-\beta}{\beta}. \tag{1.5}$$

MC denotes the multicut polytope [48]. Faces within the multicut polytope are consistent. This imposes hence the constraint on the minimization term that all faces must be consistent. Solving this ILP is a NP-hard problem. A global optimal solution can however still be found in reasonable time by using the branch and cut approach [49]: While the number of possible inconsistent labelings is exponential, the number of violated constraints can be found in polynomial time. The workflow is as follows

1. Solve (trivial) ILP by thresholding.
2. Find inconsistent faces [6] and add them to the constraint pool
3. Solve ILP with additional constraints
4. Iterate over 2) and 3) until no inconsistent faces are found

One can picture the algorithm as follows. The original face predictions are transformed into face weights through the term ω. The minimization term of the ILP is a formulation of how inconsistent faces should best be resolved. For each iteration step, all inconsistent configurations are resolved by solving the ILP. The resulting configuration produces new inconsistencies that are resolved in the following iteration. If there are no inconsistent faces anymore, the original problem has been solved. A coherent (or almost coherent) original face probability needs thus less iteration steps than randomly generated face weighting. Refer to [45] for a detailed description of the algorithm.

1.3 Objective

The aim of this work was to build up an image segmentation pipeline for neurite reconstruction. A multi-class approach which combines semantic biological classes, such as mitochondrion and vesicle cloud is used to deal with this sophisticated problem. Furthermore, a metric that evaluates segmentation performance with regards

[6] found by double path search

to application of neurite reconstruction is established.
The pipeline should allow to directly assess segmentation performance. Additionally, flexibility of the workflow is required. Flexibility allows the easy replacement of individual classifiers or algorithms within the workflow. It thus permits the comparison of different models to choose the best performing model.

Chapter 2

Methods

2.1 Description of the dataset

We subjected the proposed segmentation pipeline to a volume image of songbird (area X) acquired with SBEM (see Section 1.1.1) at $10 \times 10 \times 20$ nm^3 resolution. The data set had a size of $10 \times 10 \times 5$ kilovoxels, which corresponds to a physical size of 100 µm^3. With a 8 bit gray resolution, this leads to a total size of 1 TB.

2.2 Ground Truth generation with KNOSSOS

KNOSSOS is a software package which was explicitly developed for the purpose of visualization and skeletonization (see 1.2.1) of volume EM data at the Max Planck Institute for Medical Research in Heidelberg. The data is displayed in three orthogonal slice viewports: (x,y), (x,z), (y,z). KNOSSOS allows the user to navigate through the data set in either x, y, or z direction. It is to note, that KNOSSOS handles arbitrarily large datasets. This is achieved by not displaying the entire dataset at once. The dataset instead is stored in form of *mipmaps* at multiple zoom levels that are loaded on demand into memory. A neuron is skeletonized by setting a node at a given position. Several nodes placed in sequence are linked by edges. A forth viewport provides the user with a 3D view of already traced skeletons. In recent years, KNOSSOS was already successfully used to reconstruct neural circuits [28, 50].

2.2.1 Proof reading functionality in KNOSSOS

In the course of my project, the program was extended by several features allowing the visualization and modification of segmentation data. Those features are grouped under

the term knossos-segmentation [1]. The implementation occurred in an interactive way, incorporating the needs of the automated segmentation project. Further on, alpha versions were constantly tested by human annotators, and feedback was taken into account. The overall goal was to build up a proofreading tool, that meets the need of usability, performance and accuracy. Great importance was also placed on scalability. First of all, knossos-segmentation enables the visualization of 64 bit-per-voxel segmentation data, where every 64-bit ID corresponds to a supervoxel ID (called subobject). The data is displayed in form of overlays over the raw gray data.
Secondly, knossos-segmentation allows the user to interact with segmentation data in order to perform volume annotation. Subobject IDs can be grouped using a so-called *mergelist*. The mergelist can be auto-generated by a segmentation algorithm. Here, all subobjects overlapping with one segmented object are grouped together. During proofreading, the mergelist is modified accordingly, so that all subobjects representing e.g. a neuron are mapped together. It is to note, that manual proofreading operates on mergelist level, i.e. finer-grained editing of the raw subobject overlay is currently unsupported, but in development.

2.2.2 Pipeline of Ground Truth generation with human annotators

Knossos-segmentation does not allow for corrections on voxel-level. This implies that the original shape of one subobject can not be modified. For that reason, the size of subobjects was chosen to be very small (~ 1000 voxels), so that the right contour of the biological objects could still be approached with sufficiently high accuracy. Ground truth was generated in an iterative approach: Initially, automated segmentation was based on a very small training set corresponding to 5 neurons in a cube of size $(2.56\,\mu m)^3$. This initial training set was built up from scratch. The training set was used to generate a first segmentation, which was proofread by human annotators. The so derived ground truth could again be used to refine automated segmentation for further cubes. In total the automated segmentation of 6 cubes was proofread, located at different positions of the dataset.

2.3 Image Segmentation Pipeline

The aim of this project was to automatically segment neurons in SBEM data. Segmentation was achieved in several steps, explained herewith. Firstly, images were interpolated in z-direction in order to obtain isotropic scaling. Secondly, voxel

[1] Features were mainly implemented by My-Tien Nguyen, Norbert Pfeiler, and Michael Pronski

probability maps of different classes (neurite membrane, vesicle cloud, etc.) were generated by means of a RFC (Ilastik) or a CNN (see Section 2.3.1.2). A watershed algorithm generated small supervoxels, the basic unit of the pipeline (2.3.2). The likelihood for merging adjacent supervoxels were then calculated using a RFC (see Section2.3.3). RFC prediction was based on features calculated on the original image and the CNN output. In order to guarantee closed surfaces, a multicut was applied to the predicted faces (see Section2.3.4). The results of the multicut (binary faces) were used to obtain the final segmentation.

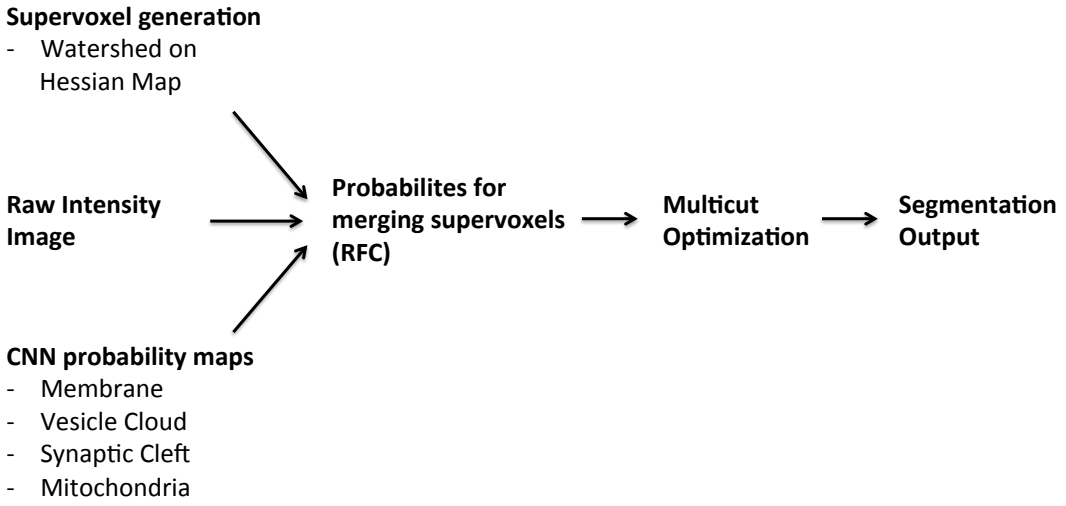


FIGURE 2.1: Image segmentation pipeline scheme. Hessian Map denotes the largest Eigenvalue of the 3D Hessian Matrix.

2.3.1 Classification of multiple classes on voxel-level

2.3.1.1 Classification with a RFC (ilastik)

Ilastik is an interactive learning and segmentation toolkit which is based on a RFC [51]. The RFC is trained on various image features, assigning a voxel-wise prediction for each class. The toolkit was used to train a classifier for membrane, intracellular space, and mitochondria. The training occurs interactively with real-time prediction feedback. This allows voxel-classification without any previous training set. Interactive training was performed on a $256x256x1283$ voxel cube for two hours. In the course of the training, voxels were subsequently labeled and the classifier retrained. The so generated classifier was embedded in the image segmentation pipeline and used for performing membrane prediction.

2.3.1.2 Classification with CNNs

Voxel probability maps were also generated on the basis of a CNN. Gregor Urban[2] mainly implemented the CNN using the python package Theano [52]. The CNN was adapted and extended by Sven Dorkenwald, who also carried out all training. A 2D CNN was trained on a set consisting of three classes: mitochondrion, vesicle cloud, and synaptic junction (2D CNN architecture: 5 hidden layers, 62196 free parameters, receptive field 50 x 50 pixels). The CNN was then used to predict probability maps for those classes. A 3D CNN was further trained on a membrane training set (3D CNN architecture: 4 hidden layers, 231780 free parameters, 38 x 38 x 38 pixels receptive field). The training set for membrane was inferred from the labeled ground truth cubes (see Section 2.2.2) by applying an edge detector filter (gradient magnitude). The membrane thickness was further varied by applying a dilatation filter (structuring element size=1). The membrane was enlarged in an iterative way, applying the dilatation filter on the outcome of the previous dilatation step. In total, iteration was carried out 3 times, using all iteration steps as a separate membrane training set (dil0, dil1, dil2, dil3). dil0 denotes the original membrane ground truth, and dil1 the outcome of the first dilatation operation. The CNN was additionally trained recursively. Here, the 4 CNN output probability maps (dil0-dil3) were used as additional feature maps together with the raw image. 4 different CNNs were trained recursively on the same training set (dil0-dil3).

2.3.2 Generation of Supervoxels

The original gray level is highly oversegmented by means of a watershed algorithm. The oversegmentation is wanted since it nearly guarantees complete boundary evidence. The largest Eigenvalue of the 3D Hessian matrix (called Hessian map) is computed which is used for boundary detection. A seeded watershed algorithm is applied on the computed Hessian map, where the seeds are local minima of the Gaussian smoothed ($sigma = 0.8$) Hessian map. Generated supervoxels serve two purposes: Firstly, they are used for GT generation with knossos-segmentation (see Section 2.2.1). Secondly, they form the basic unit for my image segmentation pipeline.

2.3.3 Classification of faces by means of a RFC

The classification problem of merging adjacent segments is addressed by supervised learning by means of a RFC (see Section 1.2.2.1). The contact areas between two adjacent supervoxels are named *faces* in the following. Features were calculated for

[2] from the HCI working group of Prof. Fred Hamprecht

voxels located along each face f (face-feature) and for the two adjacent segments s_1 and s_2 (region features), respectively. Features included statistics of distributions inferred from different intensity maps. The intensity maps comprised the original raw image itself along with voxel probability maps for membrane, mitochondrion and others. Region features of two adjacent segments were added to the face feature vector in three different ways. The sum, the minimal, and the maximal value of the two feature values were appended to the respective face feature vector. All used features are listed in Table 2.1.

face feature
mean, variance, 0.25 quantile, median, 0.75-quantile, kurtosis, skewness
region feature
mean, variance, (0.00, 0.10, 0.25, 0.50, 0.75, 0.90, 1.00)-quantile, skewness, sum, power sum N=2*, power sum N=3*

*$\sum_N x_N$

TABLE 2.1: Statistical distribution of intensity is used for face and region features

The ground truth (labeled image cubes, see Section 2.2.2) was converted into a training set of labeled faces. A face f was labeled as inactive (0) when s_1 and s_2 had the largest overlap with the same connected component, otherwise the face was labeled as active (1). Note, that this allows to use sparse ground truth [3]. Faces within the objects are labeled as inactive, faces at the border as active and faces being part of the background/mask are not included in the training set at all.

In total, 4 RFCs (255 trees) were trained each using a different combination of features (listed in Table 2.2). All RFCs were trained on the same balanced training set consisting of 919 162 faces (derived from 4 dense labeled ground truth cubes). For each face, we obtain a probability between 0 (s_1 and s_2 should be merged) and 1 (s_1 and s_2 should remain separated). The probability reflects the fraction of the trees that classified the face as active (1).

2.3.4 Multicut algorithm

The multicut algorithm [4] (see Section2.3.4) was applied on the faces incorporating the RFC outcome (see Section 2.3.3). The final multicut output consists of a binary labeling

[3]The term sparse is here used in the sense that not all neurons in a given cube have to be annotated

[4]Software and support for the multicut was provided by Thorben Kröger, HCI working group of Prof. Fred Hamprecht

Classifier name	Input feature map (face feature)	Input feature map (region feature)
model 1	raw image, RFC membrane, synaptic junction	raw image, vesicle cloud, mitochondrion
model 2	raw image, synaptic junction, CNN membrane (dil0)	raw image, vesicle cloud, mitochondrion
model 3	raw image, synaptic junction, CNN membrane (dil0, dil1, dil2, dil3)	raw image, vesicle cloud, mitochondrion
model 4	raw image, synaptic junction, CNN membrane (rec_dil0, rec_dil1, rec_dil2, rec_dil3)	raw image, vesicle cloud, mitochondrion
model 5	raw image, CNN membrane (dil0, dil1, dil2, dil3)	raw image

TABLE 2.2: Overview on classifier models, that vary with regards to input features.

of faces, which can be used to relabel the given segmentation. The free parameter β (called prior) can be considered as a tuner for the greediness of the multicut. A low value results in a segmentation with large segments, while a high value is more conservative by generating more but smaller segments. β was tuned on a validation set in order to obtain a valid prior for processing the test cube.

2.3.5 Chunkwise Array Processing

The overall goal - neural reconstruction of large EM datasets - demands the processing of the dataset in blocks for several reasons. Firstly, the processing is RAM limited. Secondly, blockwise processing potentially entails speed up by parallelization, when appropriate computational capacity is given. For the purpose of chunkwise processing, we developed a python class called *Chunky*. A wrapper function allows the chunkwise application of any arbitrary function. Intermediate results can either be stored on disk or kept in memory.

Once each chunk is segmented, corresponding regions need to be stitched together. For each two neighbouring chunks, the IDs of overlapping objects are saved as disjunct subsets within a list. In the final global list, each object is represented as a set of elements partitioned in a number of disjoints subsets. The process of finding all connected components (one connected component corresponding to one object) is achieved by using the network package NetworkX [5]. The list of disjoint subsets (each subset represents one pair of IDs) is converted into a graph object and connected components extracted

[5] http://networkx.github.io

(see Figure 2.2). Based on the connected components, the local segments are relabeled accordingly.

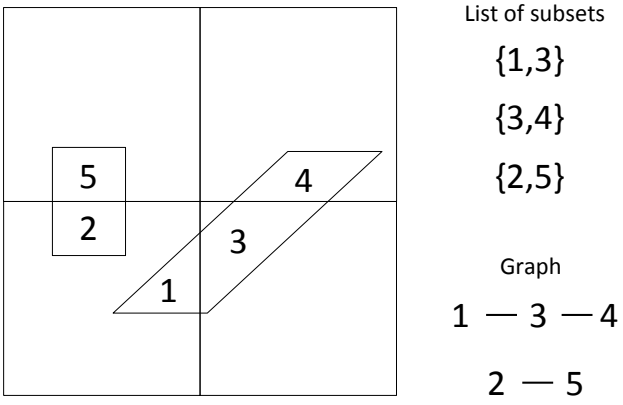


FIGURE 2.2: Schematic representation of how chunks are stitched together.4 adjacent chunks with two objects are shown (overlap is not included). Each subset within the list of subsets represents a pair of two overlapping segments in different chunks. A graph is constructed from the list of subsets which allows to extract connected component in order to identify all segment IDs that belong to the same object.

2.4 Validation of segmentation

2.4.1 Splits and mergers

Segmentation validation is performed measuring the number of splits and mergers per true segment, as it has been done before [31, 53, 54]. While in [31], the mergers and splits are counted per object, we propose to measure mergers and splits as a function of neuron length.

A bipartite graph is constructed in order to count false mergers and false splits. One set of nodes represents the ground truth in the graph (one node for each object), the second set of nodes corresponds to all supervoxels in the test set. An edge is drawn between ground truth node and test node, if the corresponding object and supervoxel overlap. In order to account for minor errors that do not affect the overall segmentation, an overlap size threshold of 7000 voxels is used for those supervoxels that overlap with more than one object. In an ideal segmentation, each node of the ground truth would map to exactly one supervoxel in the test set. The number of false mergers is derived from the number of neighbours per test set node. More than one neighbour implies that the underlying supervoxel is located within more than one ground truth object. On the other hand, the number of false splits can be read off from the number of neighbours per ground truth node. This reflects the number of supervoxels that the object is split

into. One neighbour is equal to no split, two neighbours to one split and so forth.

2.4.1.1 Reconstruction metric

We found that this metric does not reflect the perceived number of false splits in the context of neural reconstruction. For instance, a segment that accurately indicates start and end point of an object might still be quantified with a high number of false splits because of minor errors at the border of the object. Those minor errors do not have any impact on the reconstruction of this object. We therefore propose to compute additionally a reconstruction split count, that takes this issue into account.
The reconstruction split count is calculated with the help of both skeletons and volume annotated ground truth data. A ground truth object defines the set of all supervoxels in the test data belonging to it. With the corresponding skeleton, supervoxels are identified that represent endpoints of the object. Based on the cell complex representation (see Section 2.3.4), the shortest path that connects two endpoint supervoxels is computed. The shortest path reflects the minimal number of supervoxels that needs to be passed in order to get from one endpoint to the other (see Figure 2.3). For instance, a number of two would correspond to one false split, because two supervoxels are needed to reconstruct the ground truth object.

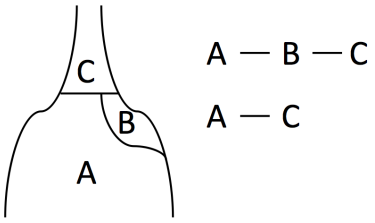


FIGURE 2.3: Schematic representation of reconstruction split count. Two endpoint supervoxels A and C are connected via two different paths A-B-C and A-C. The full split count would indicate two splits (because the object consists of three different supervoxels), while the reconstruction split count indicates only one split (because the shortest path A-C consists of two supervoxels).

2.4.1.2 Distinction between object-ECS-merger and object-object merger

An object is either merged with another object or with the ECS. If an object is merged with both ECS and another object we consider it to be an object-object merger. Merges that occur between the same objects but at different locations are counted once since the same objects are affected.

2.4.2 The Jaccard Index quantifies similarity between two clusters

The Jaccard Index is used in this work for assessing how well the automated generated object volumes represent the volume of the ground truth objects. The Jaccard Index, proposed in [55], quantifies similarity between two clusters A and B ans is defined as

$$J(A, B) = \frac{A \cap B}{A \cup B} \tag{2.1}$$

Chapter 3

Results

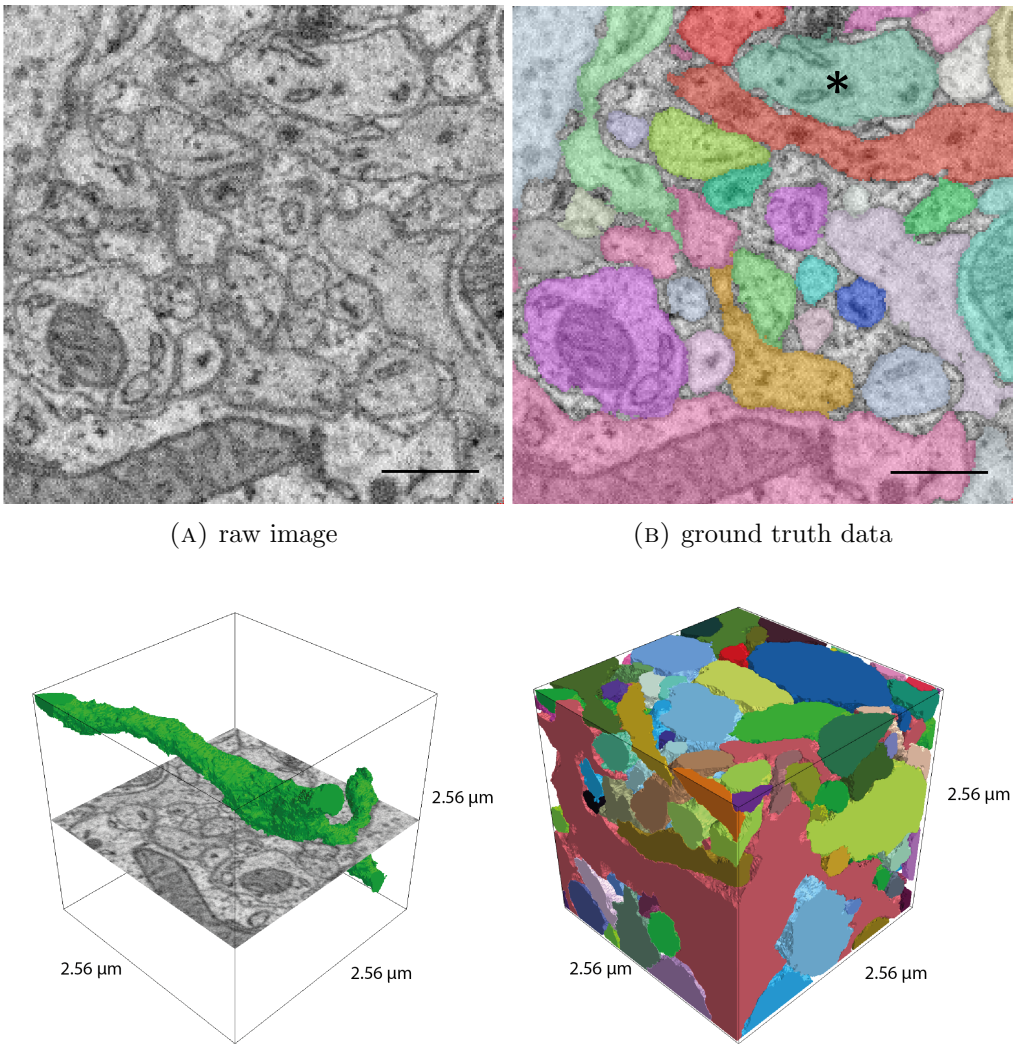
3.1 Characteristics of ground truth data

We densely annotated 6 ground truth cubes (size: $256x256x128$ voxels, $(2.56\,\mu m)^3$) [1]. Each object (neuron, glial cell or myelin) in these data was labelled with a unique ID. One such cube is shown in Figure 3.1.
Each cube contained on average ($72\pm10 s.d.$) objects, that shared ($1656\pm280 s.d.$) contact surfaces. $(18.8 \pm 3 s.d.)\%$ of the total volume (measured in voxels) consists of ECS. On average, one object had $8.9 \pm 5.5 s.d.$ faces with other objects. We further computed the areas for object-object and object-ECS contacts. Object-object faces had a total area percentage of $(22.2 \pm 0.5 s.d.)\%$, while the rest represented object-ECS connections.
Two cubes were additionally densely skeletonized. The total neuron path length in one cube was 114 µm and for the other 87 µm.

3.2 Oversegmentation by means of a watershed algorithm

Small supervoxels were generated with the proposed watershed based pipeline (Figure 3.2). On a $(2.56\,\mu m)^3$ cube, the oversegmentation resulted on average in ($40500\pm348 s.d.$) supervoxels, that had a size of $(416 \pm 238 s.d.)$ voxels. The supervoxels were connected via $(376 \pm 3 s.d.) \times 10^3$ faces.

[1] The chosen cubes did not contain somata or blood vessels, only neuropil; The data was interpolation in z-direction.



(A) raw image

(B) ground truth data

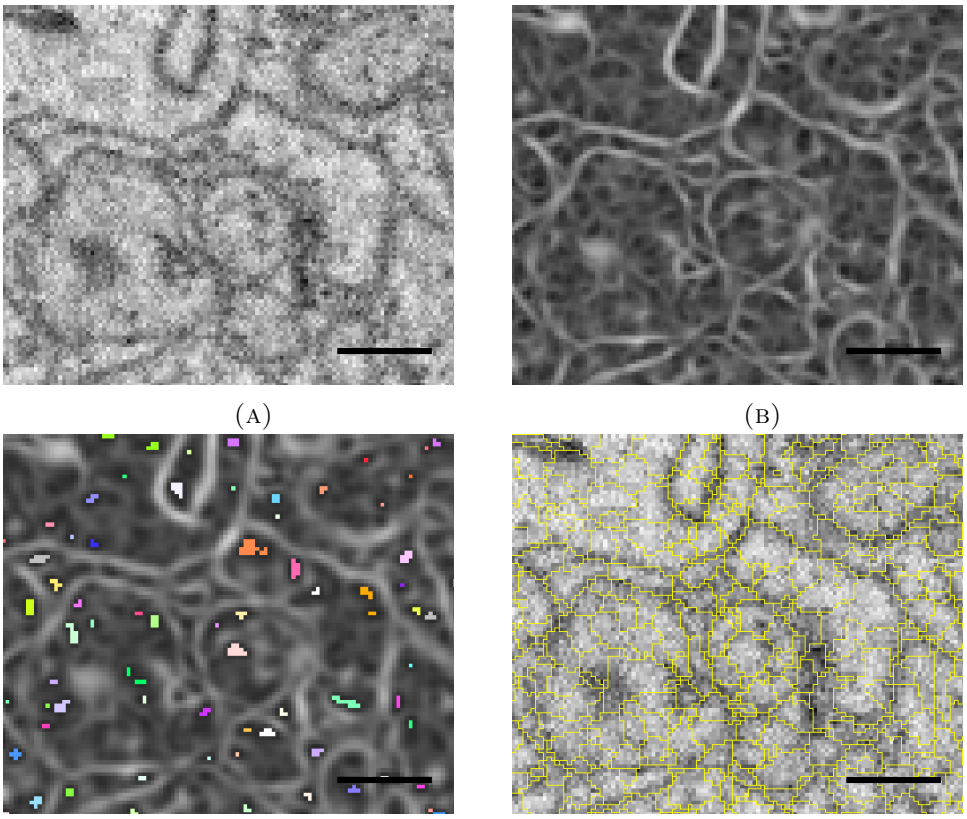
(C) 3D rendering of one segment

(D) 3D rendering of all objects

FIGURE 3.1: Ground truth labels generated with knossos-segmentation. The whole slice of a $(2.56\,\mu m)^3$ ground truth cube is shown. (B) Each id in the ground truth data is displayed with a different color and laid over on the raw image. The object rendered in (C) is marked with an asterisk. ECS remains fully visible. scale bar: 0.5 µm (D) All ground truth objects displayed as 3D renderings.

3.3 Voxel probability maps for various classes

A 2D CNN (see Section 2.3.1.2) was used for the prediction of mitochondrion, vesicle cloud and synaptic junction. The results of such a prediction and the corresponding ground truth data are displayed in Figure 3.3. Although vesicle cloud and mitochondrion predictions were not perfect, the generated probability maps rarely crossed cell membrane boundaries.



(A) (B)

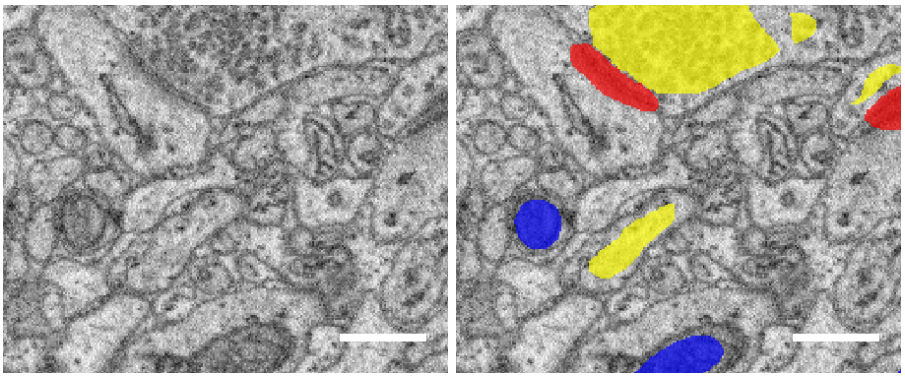
(C) (D)

FIGURE 3.2: Generation of small supervoxels. (A) raw image; (B) Largest Eigenvalue of 3D Hessian matrix (C) seeds displayed on the Gaussian smoothed Hessian map. Results from the seeded watershed algorithm are displayed in (D). All figures are 2d slices from 3D computation; scale bar: 0.25 µm

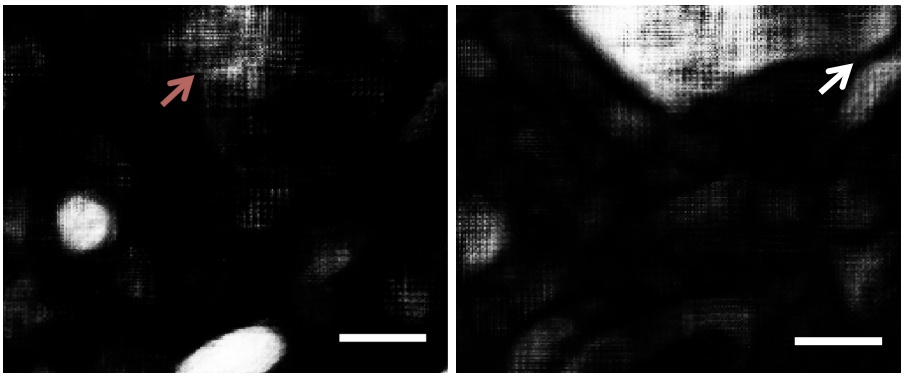
Membrane probability maps were generated with a 3D CNN. Ground truth data of membrane were modified with regards to the thickness of the membrane and resulted in 4 different training sets (dil0-dil4). Two of them (dil0 and dil3) are displayed together with the 3D CNN output for a simple and for a recursive approach in Figure 3.4. The different outputs (simple vs recursive; dil0 vs. dil3) display variable performance with regards to hole closing, removing of intracellular organelles and the intactness of intracelluar space of thin neurites.

3.4 Validation of reconstruction metric

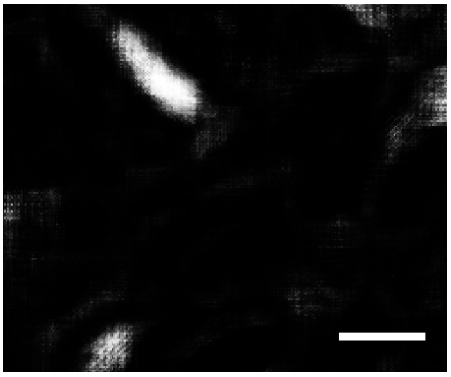
We evaluated two ways to calculate the number of splits. The first is to take all supervoxels into account that are covered by the ground truth object. The second is to ignore those supervoxels that are not essential for the path reconstruction of the object (reconstruction split count, see Section 2.4.1.1). The two approaches yield very different results (Figure 3.5a). The full split metric has a much higher split



(A) raw image (B) GT



(C) prob map mitochondrion (D) prob map vesicle cloud

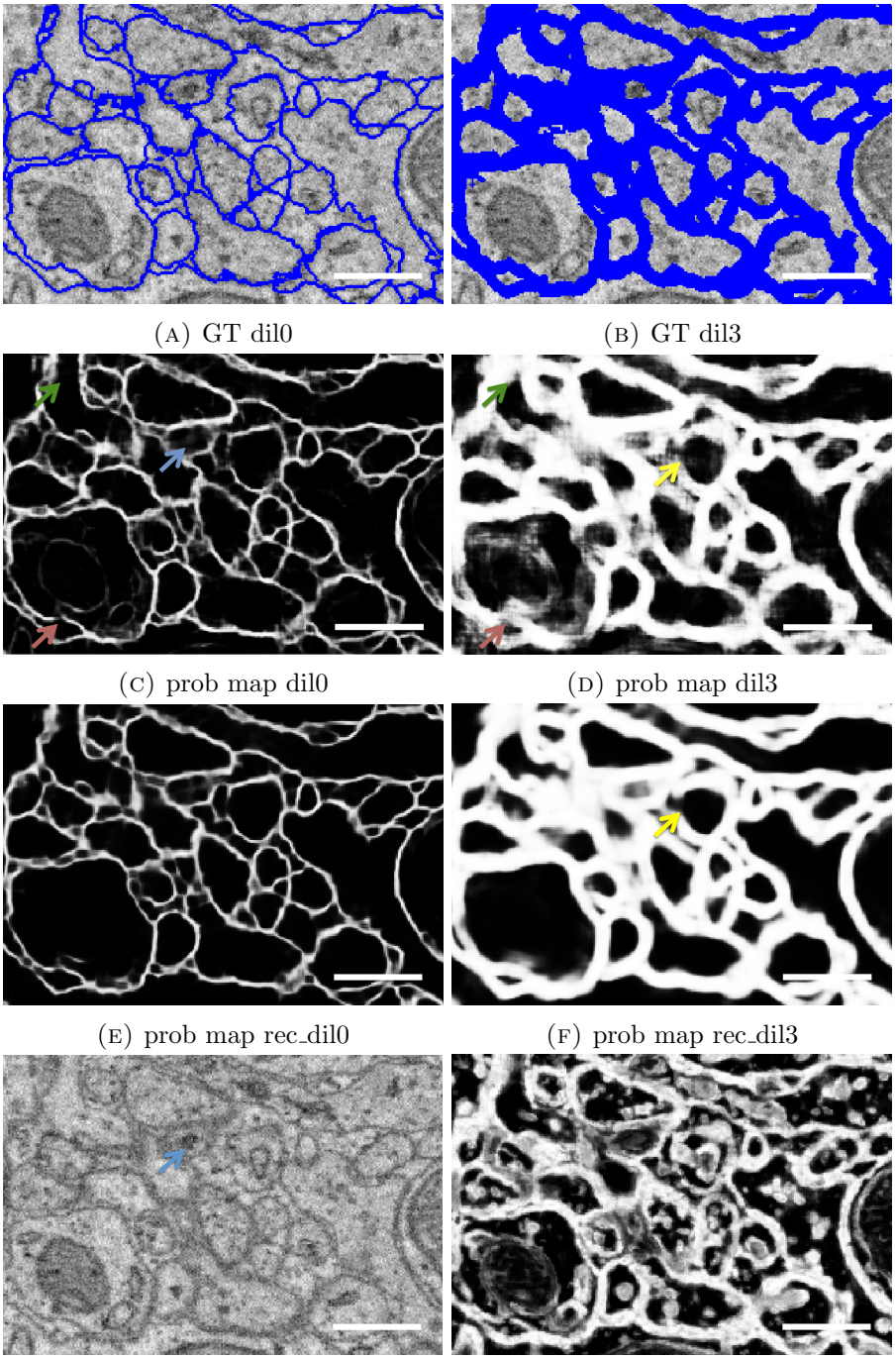


(E) prob map synaptic junction

FIGURE 3.3: Ground truth data and probability maps for mitochondrion, vesicle cloud, and synaptic junction;(B) Ground truth for mitochondrion (blue), for vesicle cloud (yellow), and for synaptic junction (red) is overlaid on the raw image. (C) Error with regards to the distinction between veiscle cloud and mitochondrion is marked with a red arrow. (D) The prediction 'avoids' membrane locations (marked with a white arrow); GT: Ground Truth; prob map: probability map; scale bar: 0.5 µm

count than the reconstruction split count. As argued in Section 2.4.1.1, the full split count might not reflect the actual error split rate in the context of neurite reconstruction.

In order to evaluate if the reconstruction split count can be used instead of the full split count, the quality of the reconstructed object is quantified with the Jaccard Index (see Section 2.4.2). A high Jaccard Index of two objects means that the similarity of those



(A) GT dil0 (B) GT dil3

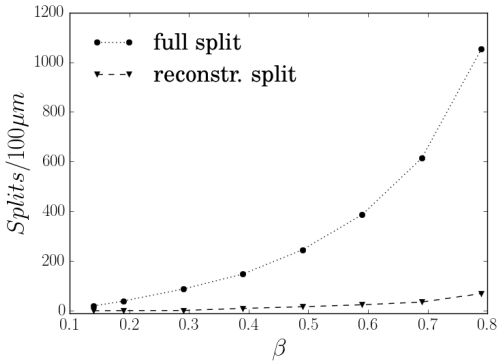
(C) prob map dil0 (D) prob map dil3

(E) prob map rec_dil0 (F) prob map rec_dil3

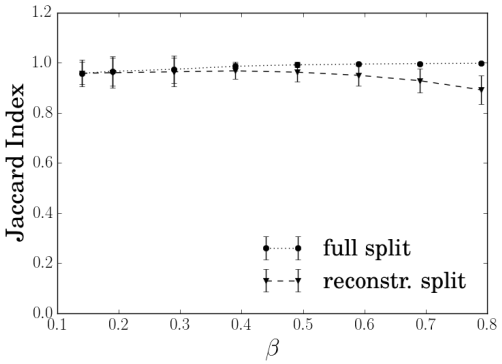
(G) raw image (H) prob map RFC

FIGURE 3.4: Ground truth data and probability maps for membrane. (A) & (B) show ground truth data for dil0 and dil3 (different thickness of membrane) and (C) and (D) the corresponding CNN outputs. Intracellular organelles are largely removed (blue arrow). Note the membrane hole that is present for dil0 but not for dil3 (red arrow). Intracellular space is intact for dil0 but not for dil3 (green arrow).(E) and (F) show CNN output from recursive training. The hole in the membrane not resolved in (D) is still present and more pronounced in (F) (yellow arrow). The overall presence of intracellular objects could be reduced through the recursive step. (F) Prediction obtained with RFC in ilastik, but on a smaller and different training set; scale bar: 0.5 µm

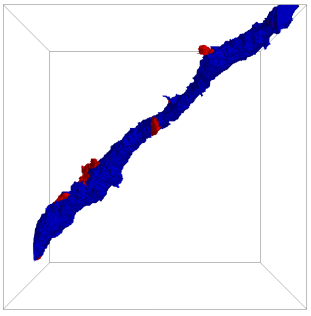
objects is high. 1 indicates full congruence while 0 denotes complete disagreement. The Jaccard Index for the reconstructed objects is high suggesting that the overall volume shape of the object is not strongly altered (Figure 3.5b and 3.5c). This is a prerequisite for using the reconstruction metric. The reconstruction split count will be used in the following for measuring segmentation performance.



(A) Full split vs reconstr. split



(B) Jaccard Index



(C) reconstructed object (blue) with missing parts (red); Jaccard Index = 0.98

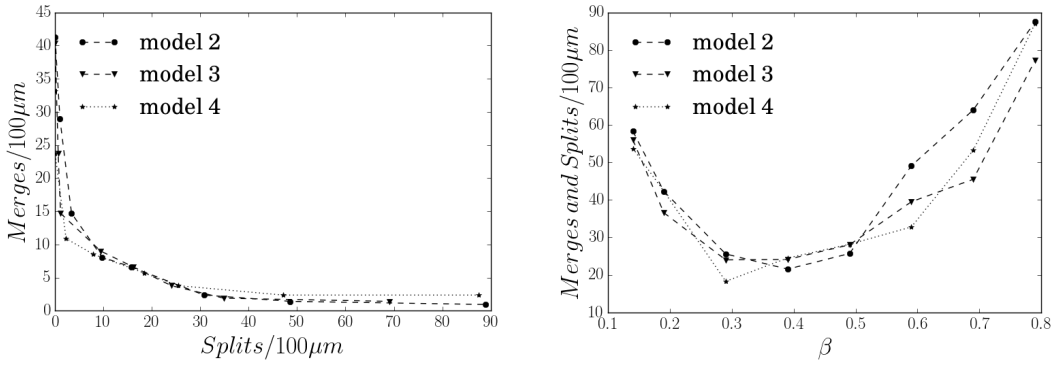
FIGURE 3.5: Reconstruction split and full split yield different results, but reconstructed objects have still high similarity to ground truth objects. Results are shown for model 3. (C) One reconstructed object (Jaccard Index = 0.98) is shown with small volume errors with regards to the ground truth object indicated in red (model 3, $\beta = 0.39$, test cube)

3.5 Segmentation performance

3.5.1 Error free path length of best performing model

The segmentation outcome depends on the prior β which is considered to be a tuner for the degree of over- or under-segmentation. The smaller the prior β, the more the

original supervoxels are merged together. Figure 3.6a shows the number of mergers as a function of reconstruction split count for various prior β. In order to get a final estimate for the segmentation performance, the optimal prior β was determined on a validation cube. The number of mergers together with the reconstruction split count is added up in order to obtain the minimal error count (for an example, total error count for three models are plotted in Figure 3.6). Segmentation performance is then computed with the obtained prior β on the test cube.
Segmentation performance was evaluated for 5 different classifiers (model 1 - model 5) which varied with regards to the input features that they were trained on (see Section 2.3.3). Results are displayed in Table 3.1. Model 4 (based on the membrane probability map of the recursive CNN, rec_dil0-rec_dil3) scores the best performance with an error free path length of 10.4 µm. No error on the results can be computed for lack of additional test cubes. In the following several aspects of the pipeline are analysed in more detail.



(A) All mergers vs. reconstr. split count

(B) Tweaking of β

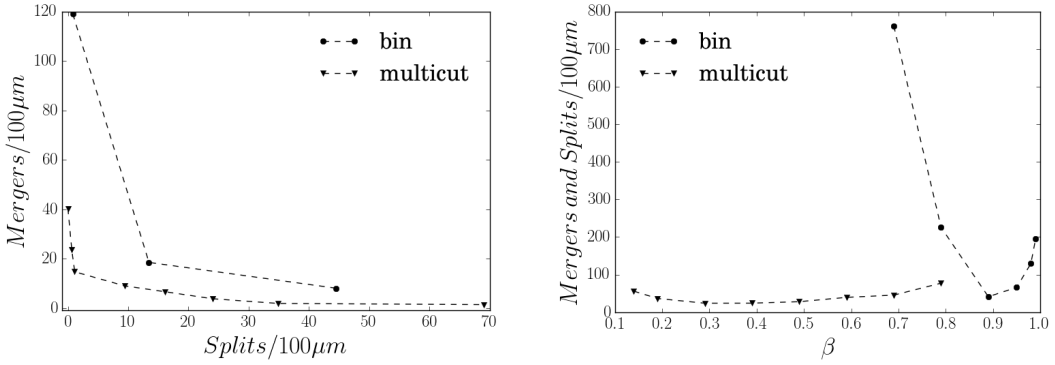
FIGURE 3.6: Tweaking the prior β minimizing the sum of all mergers and reconstruction splits. (A) Number of mergers and splits for model 2-4 are plotted for various β. One data point corresponds to one β. The lower β the more false mergers, but the less false splits exist. (B) The parameter β is optimized by minimizing the total error count (mergers and reconstruction splits).

3.5.2 Multicut algorithm increases segmentation performance

We compared segmentation obtained by simply thresholding face predictions (binary case) to segmentation obtained by resolving inconsistent faces by applying the multicut (see Section 2.3.4). The segmentation performance for the multicut is 10 µm error free path length compared to the binary case where the error free path length is 3.8 µm. We found that the multicut optimization is more robust with regards to β. Binary thresholding has only a small range in which the threshold produces meaningful results (Figure 3.7b).

Classifier name	optimal bias β	mergers per 100 µm	reconstr. splits per 100 µm	total error count	EFP [µm]
model 1	0.39	14.2	9.7	23.9	4.2
model 2	0.39	3.9	8.9	12.8	7.8
model 3	0.29	1.9	8.1	10	10
model 4	0.29	3.9	5.7	9.6	**10.4**
model 5	0.39	4.9	8.1	13	7.7

TABLE 3.1: Neurite reconstruction performance for various models. The models differ with regards to their input probability maps: model 1 is based on RFC membrane, model 2 on single CNN membrane (dil0), model 3 on dil0, dil1, dil2, and dil3, model 4 on the recursice CNN output (rec_dil0, rec_dil1, rec_dil2, rec_dil3), model 5 on dil0, but in contrast to model 1-4, all other classes (mitochondrion, synapse) were omitted (see Section 2.3.3 for exact model overview). oob: out-of-bag error; reconstr. splits: reconstruction splits; EFP: error free path length



(A) Reconstruction metric (B) Parameter tuning

FIGURE 3.7: Performance of multicut segmentation versus segmentation obtained by binary thesholding (bin). (A) Metrics were calculated wrt. human ground truth (3.1) on segmentations generated at various priors (multicut) or thresholds (bin) for model 3. For the binary case, only thresholds in the range of 0.79 and 0.95 are displayed. (B) The range for a valid β is much larger for the multicut than for the binary case. β refers either to the threshold (bin) or to the prior (multicut). bin: binary thresholding

3.5.3 Combination of different membrane probability maps produces better results than single-membrane model

The best two models (model 3, model 4; error free path length = 10 µm, 10.4 µm) combine both membrane probability maps from all 4 CNNs trained with various dilated membranes (dil0-dil4). Model 2 (error free path length= 7.8 µm) had the same input as model 4 except for membrane probability maps (dil1-dil3). The combination of the four different probability maps seems to outperform the model using only one single membrane probability map. This notion is also supported by the feature importance of the full model (model 3). Here, the feature importance of all 4 CNN membrane probability maps are rated equally high. In order to test this further, additional

comparisons are required in which also the other CNN outputs (dil1-dil3) are compared in form of single-models against the full model (using all 4 CNN membrane probability maps as inputs).

3.5.4 CNN membrane probability map versus RFC probability map

Model 1 and model 2 had the same features as input except for the membrane probability map. Model 1 was based on a probability map obtained with a RFC while model 2 was based on a CNN probability map. Model 1 scores an error free path length of 4.1 µm while model 2 has a performance of 8 µm error free path length. CNNs in this context seem to produce better results. However, this statement is only preliminary because the RFC for membrane prediction was trained on a different and smaller subset than the CNN.

3.5.5 Combination of multiple classes outperforms single-class (membrane) model

Model 3 and model 5 had the same four membrane predictions as input (dil0-dil4), but model 5 did not get any other class probability maps as input (mitochondrion, vesicle cloud, synaptic junction). Model 5, without the additional classes, scored an error free path length of 7.7 µm, while model 3, with the additional classes, had a performance of 10 µm error free path length. The incorporation of biological semantic information, such as mitochondrion and synapse labels, therefore appears to increase the overall neurite reconstruction performance.

3.5.6 Many false mergers are with ECS

We found that most mergers were object-ECS mergers ($\sim 85\%$)(see Figure 3.8). Figure 3.9 shows mergers that extent exclusively into ECS. Most of those mergers are locally limited, while a few disperse more globally. However, (74 ± 3) % (for model 3) of all ECS-mergers were touching the edge of the cube. we can therefore only confidently classify 25 % of ECS mergers as such, the remaining 75 % may be indirect object mergers not fully contained in the test cube. Our current metric is conservative because it includes all mergers regardless of them being an object-ECS merger or an object-object merger. If we exclude the object-ECS mergers, the best model (model 3) scores an error free path length of 15.4 µm comprising 6.5 reconstruction splits per 100 µm and no object-object merger at all.

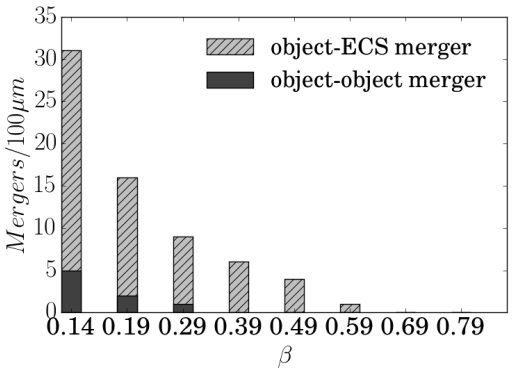


FIGURE 3.8: Object-ECS merger have a higher proportion of the total merger count than object-object merger. The percentage is shown for model 2 on the test cube at various priors β.

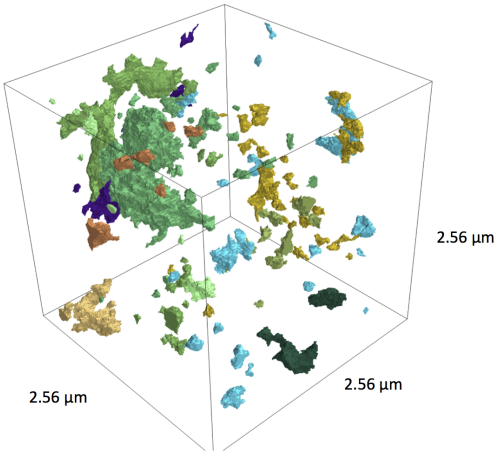
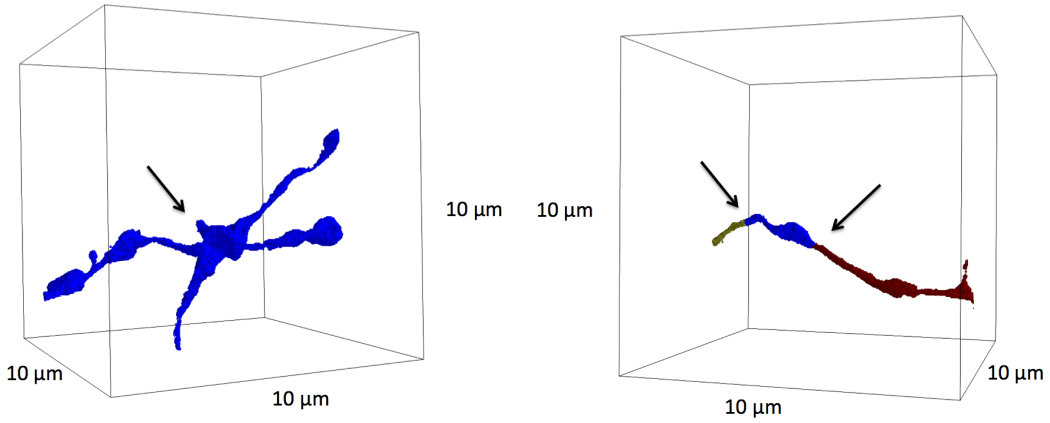


FIGURE 3.9: 3D rendering of object-ECS mergers. Most object-ECS mergers are locally limited. All ECS mergers of the same object are displayed with the same color.(segmentation of test cube: model 3, $\beta = 0.19$)

3.6 Segmentation of a $(10\,\mu m)^3$ cube

So far, we only considered a test cube of size $(2.56\,\mu m)^3$. A cube sized $(10\,\mu m)^3$ was additionally segmented. The large cube was partitioned into 260 chunks, processed and stitched together, demonstrating that the pipeline is suitable for larger data. However, due to a lack of ground truth data, this cube could not be validated yet. In Figure 3.10, an example of two false splits and one false merger is shown. The size of the supervoxels did now allow the computation of high-level features for a potential iterative merging approach.



(A) false merger (B) false split

FIGURE 3.10: 3D rendering of false splits and false mergers. Errors are marked with an arrow. (Segmentation with model 1, $\beta = 0.59$)

3.7 Speed of skeletonization versus speed of proofreading

The generation of ground truth was performed with two different methods: skeletonization, and the proofreading of automated segmentation with knossos-segmentation. The processing of a $(2.56\,\mu m)^3$ cube yielded $4.9 \pm 4.5 s.d.$(n=3) hours for skeletonization and $5.5 \pm 0.75 s.d$ (n=3) for proofreading with knossos-segmentation. It is to note that the times for skeletonization and proofreading refer to dense annotations, in which the user also had to guarantee the completeness of annotation. The time is likely to decrease when tasks are distributed object-wise. These results are preliminary because of the small N. However, if the overall trend can be confirmed, proofreading and skeletonization would fall into the same time range. With this method, it is hence potentially possible to dramatically decrease the time for volume labelling, which was until now estimated to be 10-50 times slower than skeletonization [50].

3.8 Run time considerations

The presented image segmentation pipeline must handle terabytes of data to process. Therefore, runtime is a decisive factor for the feasibility of a neural reconstruction pipeline. Runtimes for all steps of the pipeline are listed individually (Table 3.2). The total time for the current set up for a $(2.56\,\mu m)^3$ cube would amount to 30 min for model 1 (without CNN probability maps included), and 1 hour for model 2 (with 3D CNN probability maps). Including probability maps from the recursive network (model 4) would result in 4 hours runtime. Times are also listed for the processing of a cube

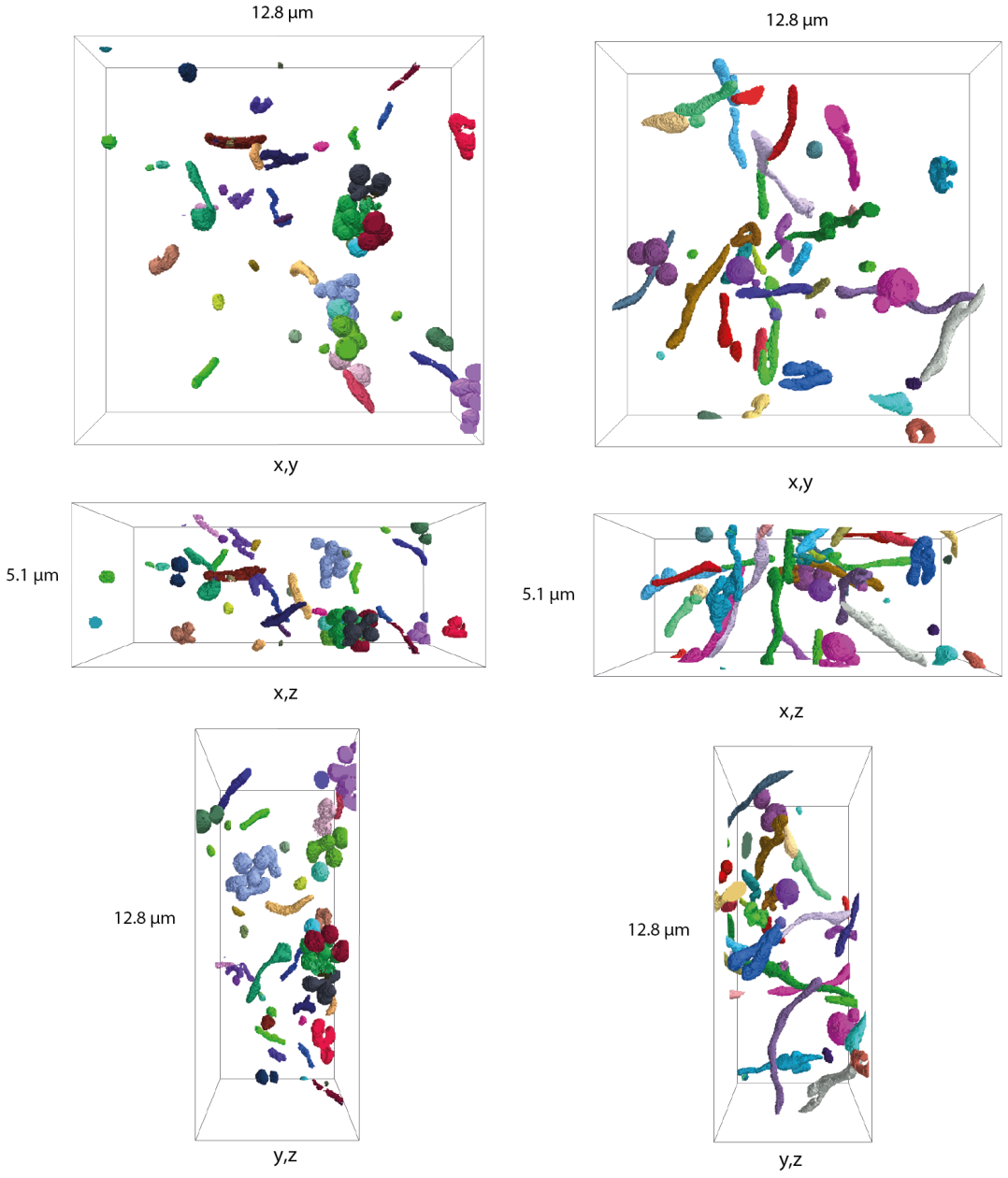
recursively. In this case, the original supervoxels are much larger, which decreases the time for the computation of face features by a factor of three.

Computation	CPU(s)	GPU	Runtime A [sec]	Runtime B [sec]
Watershed	1	-	182	
CellComplex generation	1	-	54	34
CNN prediction 2D	1	1	62	
CNN prediction 3D	1	1	1630	
CNN prediction 3D rec	1	1	13 440	
ilastik prediction	1	-	121	
Face Feature	1	-	187	60
Region Feature	1	-	9	12
Face Classification	1	-	273	
Multicut	8	-	292 ± 15	

TABLE 3.2: Absolute runtime of segmentation pipeline. Times are given for a 2 x Intel Quad Xeon (E5-2609) running at 2.5 GHz. CNN prediction was carried out on one GPU (GeForce GTX Titan). Runtime A refers to a 256^3 cube with 41 002 segments and 379 557 faces. Runtime (B) refers to the automated segmentation result from (A) with 7529 segments and 61 748 faces. Times for face and region feature are given for one input probability map. If several input maps (mitochondrion, membrane etc.) are used, this time is multiplied accordingly.

3.9 Axonal and dendritic mitochondria have different appearance

Mitochondria segmentation was subjected to a qualitative analysis. Figure 3.11 shows dendritic mitochondria next to axonal mitochondria. Dendritic mitochondria appear elongated while axonal mitochondria tend to be smaller and globular shaped.



(A) Axonal mitochondria (B) Dendritic mitochondria

FIGURE 3.11: Axonal and dendritic mitochondria have different appearance. Displayed are 42 dendritic and 49 axonal mitochondria.

Chapter 4

Discussion

4.1 Incorporation of semantic biological classes improves the automated neurite reconstruction

The incorporation of CNN generated probability maps for mitochondrion, vesicle cloud and synaptic junction resulted in a gain of 2.7 µm error free path length. It has been already reported that other classes (for instance the presence of vesicles) disturb automated neurite reconstruction [53]. If voxel classification improves further and other disturbing classes are already removed at voxel level, there might be no need to incorporate additional classes at a higher level. One could on the other hand argue, that the incorporation of biological priors can still help automated segmentation, discussed in the following.

We found for instance that axonal mitochondria differ in appearance from dendritic mitochondria. There are potentially more features in EM data that provide information on biological function. This would allow to not only differentiate between dendrites and axons but also to distinguish between different cell types.

If those features can be quantified (such as the shape properties of a mitochondrion), they have the potential to be incorporated into the segmentation pipeline. An obvious dendritic segment for instance should never be merged with an obvious axonal segment.

The incorporation of biological priors has yet to be treated with caution. It is dangerous in the sense that new unknown facts might be missed. This phenomenon is already a problem for scientists themselves. They are biased by their expectations and prone to overlook new facts that contradict the 'scientific' dogma. It is therefore even more dangerous to incorporate such priors in an automatization, where humans do not have any chance to discover unknown deviations.

4.2 Comparability of segmentation pipelines

In our opinion, neurite reconstruction pipelines are currently difficult to compare for two main reasons.
First, pipelines are developed for specific types of datasets. Datasets differ with regards to several aspects such as imaging technique (resolution), staining (e.g. ECS preservation), and specimen (species, region). There is therefore an obvious need for publicly available datasets that would allow an objective comparison.
The second problem is the lack of a gold standard metric. Variation of Information is more suitable than the Rand Index [56] but to our opinion only of limited significance for neurite reconstruction. The count of false mergers and false splits is intuitively convenient and has already been used for segmentation evaluation [31]. However, the error count should occur as a function of path length, and not as a function of cube size or object count. The application in mind, one could even try using human interaction time as a metric, ie the time required by an annotator to correct the mistakes of an automated pipeline to achieve a (perceived) error-free segmentation of a certain path length of a neurite. This would be convenient and summarize all pipeline aspects as long as human interaction time remains the limiting factor for neurite reconstruction.
A thourough comparison of the proposed pipeline with other reconstruction pipelines is still pending. Only such a comparison would allow to decide whether algorithms and proposed approaches are promising and should be pursued further.

4.3 The combination of two powerful approaches: CNN and multicut

To our opinion, our pipeline currently produces promising results because it combines two powerful approaches: low-level CNN voxel classifiers and a high level supervoxel-linking multicut algorithm. The pipeline allows to easily replace the currently used classifiers and algorithms on the voxel and supervoxel level, thus offering a large amount of flexibility. This flexibility could be used to incorporate upcoming better-performing algorithms to stay state-of-the art.
This implies that our pipeline can also be used to estimate and compare the performance of various low- and high-level algorithms. We used this to compare different membrane voxel probability maps, an important tool with regards to the development of new voxel-level classification algorithms. Furthermore, when using the same voxel probability maps, we could compare different strategies for combining supervoxels. So far, only the multicut algorithm was used, but it would be interesting to compare it against GALA

(graph-based active learning of agglomeration) [44], LASH (Learning Agglomeration of Superpixel Hierarchies) [43], "Cut, Glue and Cut" [57] and others.

4.4 The role of ECS for neurite reconstruction

We found that only 20 % of the total contact area represents touches. This suits the result that only ~ 15 % of all mergers were object-object mergers. If we assume the error probability to be the same for object-object contact area and object-ECS contact area and that ECS mergers are often locally contained, ECS preservation during staining could inherently reduce the number of false object mergers in neurite reconstruction. For the segmentation workflow, one option would be to treat ECS as an explicit class and create a constraint that those areas should not be merged. In such a case, ECS-object mergers would be favoured to remain locally limited, essentially generating a wall of fragmented ECS supervoxels preventing object-object mergers. In staining without ECS preservation, every false merger that occurs is with another object. ECS preservation has thus the potential to improve neurite reconstruction. On the other hand, it is possible that ECS preservation favours the formation of membrane holes and/or that the presence of ECS is harmful for neurite reconstruction by leading to thinner neurites. In order to test this, the next step would be to evaluate how many object-object mergers are actually caused indirectly by object-ECS mergers.

4.5 Segmentation performance in the context of dataset size

In the presented dataset, a $(10\,\mu m)^3$ cube has a total path length of ~ 6 mm [1]. Our current error free path length of 10 µm is thus far away from allowing the full automated segmentation of even such a small cube and is impossible for current dataset sizes ($(300\,\mu m)^3$). To tackle this problem, a combination of human interaction and automated segmentation has been proposed previously [9]. For instance, a current strategy is to avoid any false mergers to happen by merging very conservatively. The location for potential errors is derived directly from the segmentation pipeline (Manuel Berning, personal communication).

The nature of current errors additionally favours the idea of a top down approach. For instance, false mergers can easily be identified by humans by simply looking at a 3D rendering of an object (see Figure 3.10). In a top down approach, those errors could

[1] projected from a $(2.56\,\mu m)^3$ cube

be detected in an automated fashion (and potentially guide human proofreading). It could be convenient to have such an automated error detector independent of underlying segmentation pipelines.

Bibliography

[1] Y Ramon. Cajal (1888) estructura de los centros nerviosos de las aves. *Rev. Trim. Histol. Norm. Pat*, 1:1–10.

[2] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

[3] P Fatt and B Katz. Some observations on biological noise. *Nature*, 166(4223):597, 1950.

[4] SL Palay. Electron microscope study of the cytoplasm of neurons. In *Anatomical Record*, volume 118, pages 336–336. WILEY-LISS DIV JOHN WILEY & SONS INC, 605 THIRD AVE, NEW YORK, NY 10158-0012, 1954.

[5] Eduardo DP De Robertis and H Stanley Bennett. Some features of the submicroscopic morphology of synapses in frog and earthworm. *The Journal of biophysical and biochemical cytology*, 1(1):47–58, 1955.

[6] Stephen J Smith. Circuit reconstruction tools today. *Current opinion in neurobiology*, 17(5):601–608, 2007.

[7] Winfried Denk, Kevin L Briggman, and Moritz Helmstaedter. Structural neurobiology: missing link to a mechanistic understanding of neural computation. *Nature Reviews Neuroscience*, 13(5):351–358, 2012.

[8] Jeff W Lichtman and Winfried Denk. The big and the small: challenges of imaging the brain's circuits. *Science*, 334(6056):618–623, 2011.

[9] Moritz Helmstaedter. Cellular-resolution connectomics: challenges of dense neural circuit reconstruction. *Nature methods*, 10(6):501–507, 2013.

[10] John G White, Eileen Southgate, J Nichol Thomson, and Sydney Brenner. The structure of the nervous system of the nematode caenorhabditis elegans. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 314(1165):1–340, 1986.

[11] KJ Hayworth, N Kasthuri, R Schalek, and JW Lichtman. Automating the collection of ultrathin serial sections for large volume tem reconstructions. *Microscopy and Microanalysis*, 12(S02):86–87, 2006.

[12] Masaaki Kuwajima, John M Mendenhall, and Kristen M Harris. Large-volume reconstruction of brain tissue from high-resolution serial section images acquired by sem-based scanning transmission electron microscopy. In *Nanoimaging*, pages 253–273. Springer, 2013.

[13] Moritz Helmstaedter, Kevin L Briggman, and Winfried Denk. 3d structural imaging of the brain with photons and electrons. *Current opinion in neurobiology*, 18(6): 633–641, 2008.

[14] Winfried Denk and Heinz Horstmann. Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. *PLoS biology*, 2(11):e329, 2004.

[15] Graham Knott, Herschel Marchman, David Wall, and Ben Lich. Serial section scanning electron microscopy of adult brain tissue using focused ion beam milling. *The Journal of Neuroscience*, 28(12):2959–2964, 2008.

[16] Kristina D Micheva and Stephen J Smith. Array tomography: a new tool for imaging the molecular architecture and ultrastructure of neural circuits. *Neuron*, 55(1):25–36, 2007.

[17] Siddharth Nanguneri, Benjamin Flottmann, Heinz Horstmann, Mike Heilemann, and Thomas Kuner. Three-dimensional, tomographic super-resolution fluorescence imaging of serially sectioned thick samples. *PloS one*, 7(5):e38098, 2012.

[18] Kevin L Briggman and Davi D Bock. Volume electron microscopy for neuronal circuit reconstruction. *Current opinion in neurobiology*, 22(1):154–161, 2012.

[19] Daniel K Hartline. What is myelin? *Neuron glia biology*, 4(02):153–163, 2008.

[20] Laurent Descarries, Beaudet Alain, and Kenneth C Watkins. Serotonin nerve terminals in adult rat neocortex. *Brain research*, 100(3):563–588, 1975.

[21] Kristen M Harris and John K Stevens. Dendritic spines of ca 1 pyramidal cells in the rat hippocampus: serial electron microscopy with reference to their biophysical characteristics. *The Journal of neuroscience*, 9(8):2982–2997, 1989.

[22] Jun Noguchi, Masanori Matsuzaki, Graham CR Ellis-Davies, and Haruo Kasai. Spine-neck geometry determines nmda receptor-dependent ca 2+ signaling in dendrites. *Neuron*, 46(4):609–622, 2005.

[23] Esther A Nimchinsky, Bernardo L Sabatini, and Karel Svoboda. Structure and function of dendritic spines. *Annual review of physiology*, 64(1):313–353, 2002.

[24] Alfonso Araque, Vladimir Parpura, Rita P Sanzgiri, and Philip G Haydon. Tripartite synapses: glia, the unacknowledged partner. *Trends in neurosciences*, 22(5):208–215, 1999.

[25] Eric R Kandel, James H Schwartz, Thomas M Jessell, et al. *Principles of neural science*, volume 4. McGraw-Hill New York, 2000.

[26] Davi D Bock, Wei-Chung Allen Lee, Aaron M Kerlin, Mark L Andermann, Greg Hood, Arthur W Wetzel, Sergey Yurgenson, Edward R Soucy, Hyon Suk Kim, and R Clay Reid. Network anatomy and in vivo physiology of visual cortical neurons. *Nature*, 471(7337):177–182, 2011.

[27] Kevin L Briggman, Moritz Helmstaedter, and Winfried Denk. Wiring specificity in the direction-selectivity circuit of the retina. *Nature*, 471(7337):183–188, 2011.

[28] Moritz Helmstaedter, Kevin L Briggman, Srinivas C Turaga, Viren Jain, H Sebastian Seung, and Winfried Denk. Connectomic reconstruction of the inner plexiform layer in the mouse retina. *Nature*, 500(7461):168–174, 2013.

[29] 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 *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[30] Björn Andres, Ullrich Köthe, Moritz Helmstaedter, Winfried Denk, and Fred A Hamprecht. Segmentation of sbfsem volume data of neural tissue by hierarchical classification. In *Pattern recognition*, pages 142–152. Springer, 2008.

[31] 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. *Neural Computation*, 22(2):511–538, 2010.

[32] John A Bogovic, Gary B Huang, and Viren Jain. Learned versus hand-designed feature representations for 3d agglomeration. *arXiv preprint arXiv:1312.6159*, 2013.

[33] Gary B Huang and Viren Jain. Deep and wide multiscale recursive networks for robust image labeling. *arXiv preprint arXiv:1310.0354*, 2013.

[34] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[35] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[36] Lior Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2): 1–39, 2010.

[37] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[38] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[39] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[40] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34 (11):2274–2282, 2012.

[41] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 670–677. IEEE, 2009.

[42] Serge Beucher and Christian Lantuéjoul. Use of watersheds in contour detection. 1979.

[43] Viren Jain, Srinivas C Turaga, K Briggman, Moritz N Helmstaedter, Winfried Denk, and H Sebastian Seung. Learning to agglomerate superpixel hierarchies. In *Advances in Neural Information Processing Systems*, pages 648–656, 2011.

[44] Juan Nunez-Iglesias, Ryan Kennedy, Toufiq Parag, Jianbo Shi, and Dmitri B Chklovskii. Machine learning of hierarchical clustering to segment 2d and 3d images. *PloS one*, 8(8):e71715, 2013.

[45] Bjoern Andres, Joerg H Kappes, Thorsten Beier, Ullrich Kothe, and Fred A Hamprecht. Probabilistic image segmentation with closedness constraints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2611–2618. IEEE, 2011.

[46] 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 *Computer Vision–ECCV 2012*, pages 778–791. Springer, 2012.

[47] Thorben Kroeger. *Learning-based Segmentation for Connectomics*. PhD thesis, 2014.

[48] Sunil Chopra and MR Rao. The partition problem. *Mathematical Programming*, 59(1-3):87–115, 1993.

[49] Abilio Lucena and John E Beasley. Branch and cut algorithms. *Advances in linear and integer programming*, 4:187–221, 1996.

[50] Moritz Helmstaedter, Kevin L Briggman, and Winfried Denk. High-accuracy neurite reconstruction for high-throughput neuroanatomy. *Nature neuroscience*, 14(8):1081–1088, 2011.

[51] Christoph Sommer, Christoph Straehle, Ullrich Kothe, and Fred A Hamprecht. Ilastik: Interactive learning and segmentation toolkit. In *Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on*, pages 230–233. IEEE, 2011.

[52] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4, page 3, 2010.

[53] Yuriy Mishchenko. Automation of 3d reconstruction of neural tissue from large volume of conventional serial section transmission electron micrographs. *Journal of neuroscience methods*, 176(2):276–289, 2009.

[54] Verena Kaynig, Amelio Vazquez-Reina, Seymour Knowles-Barley, Mike Roberts, Thouis R Jones, Narayanan Kasthuri, Eric Miller, Jeff Lichtman, and Hanspeter Pfister. Large-scale automatic reconstruction of neuronal processes from electron microscopy images. *arXiv preprint arXiv:1303.7186*, 2013.

[55] Paul Jaccard. The distribution of the flora in the alpine zone. 1. *New phytologist*, 11(2):37–50, 1912.

[56] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[57] Thorsten Beier, Thorben Kroeger, Jorg Kappes, Ullrich Kothe, and Fred Hamprecht. Cut, glue & cut: A fast, approximate solver for multicut partitioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 73–80, 2013.