Learning Context Cues for Synapse Segmentation

Carlos Becker, Karim Ali, Graham Knott, and Pascal Fua

Abstract—We present a new approach for the automated segmentation of synapses in image stacks acquired by electron microscopy. We rely on a large set of image features specifically designed to take spatial context into account and train a classifier that can effectively utilize cues such as the presence of a nearby post-synaptic region. As a result, our algorithm successfully distinguishes synapses from the numerous other organelles that appear within an EM volume, including those whose local textural properties are relatively similar. We evaluate our approach on three different datasets and compare it against the state-of-the-art in synapse segmentation.

Index Terms—Electron microscopy, segmentation, synapse, boosting.

I. INTRODUCTION

New imaging technologies have been a key driver of recent advances in neuroscience. In particular, Focus Ion Beam Scanning Electron Microscopy (FIBSEM) can now deliver 5nm nearly isotropic sampling and produce image stacks that reveal very fine neuronal structures. Stacks such as the one depicted by Fig. 1(a) can be used to analyse a variety of components, such as synapses, whose modeling is critical to understanding brain function. For example, evidence obtained from EM images suggests that the size, shape and distribution of synapses vary during the course of normal life but also under specific pathological conditions [25]. Similarly, EM imaging has provided new insights into synaptic signaling [19] and its relationship to mitochondrial activity [23] as well as to some neuro-degenerative diseases [17], [31].

Currently, analysis is carried out by manually segmenting the various structures of interest using tools such as Fiji [32]. This is not only tedious and time consuming but also error-prone. Thus, while the growing number of EM datasets offer a unique opportunity to unlock new concepts of neuronal function, the required amount of human effort remains a major bottleneck. There has therefore been great interest in automating the process.

EM data poses unique challenges for automatic segmentation algorithms in part because the volumes are heavily cluttered with structures that exhibit very similar textures and are difficult to distinguish solely on the basis of local image statistics. As shown in Fig. 1(b), synapse segmentation is very representative of this difficulty because a synapse can be reliably distinguished from other structures only by relying on contextual clues such as the presence of a nearby cluster of vesicles. This why human experts use well-established criteria such as densities on the pre-synaptic and post-synaptic membranes, presence of vesicles in the pre-synaptic axon terminal and of a synaptic cleft, as depicted in Fig. 1(c). It is therefore essential for an automatic segmentation method to proceed in a similar fashion.

Current methods for automated synapse detection either require first finding cell membranes [26] or operate on individual slices [12], thus failing to leverage the 3D structure of the data. By contrast, the recent method of [18] operates entirely in 3D. However, the latter does not exploit the contextual clues that allow human experts to distinguish synapses from other structures, such as endoplasmic reticula. More generally, though progress has been made towards the segmentation of various organelles from EM stacks, context has yet to be exploited in a meaningful way.

In this work, we propose an approach designed to take such contextual cues into account and emulate the human ability to distinguish synapses from regions that merely share a similar texture. Our method is fully automated, processes the data directly in 3D and is specifically designed to leverage context cues. We run various filters over the EM stack but compute our features over arbitrarily sized cubes placed at arbitrary locations inside an extended neighborhood of the voxel to be classified. As this generates a feature representation for each voxel in the order of tens of thousands, we rely on Boosting to select the relevant filter channels as well as the relevant cube locations and sizes. The resulting classifier is thus highly flexible, able to utilize context from a high variety of regions in the neighborhood of the voxel of interest.

We apply our classifier for the synapse segmentation task and compare our results with the state-of-the-art method of Kreshuk et al. [18] for synapse segmentation, a fully automated 3D approach which does not utilize context. By honing in on the presence of pre-synaptic vesicles and post-synaptic regions in addition to local texture, our method significantly outperforms the method of [18].

We validate our method on three datasets obtained from three different regions of the adult mammalian brain: the Somatosensory cortex, the Hippocampus, and the Cerebellum. We demonstrate our ability to automatically process large EM stacks, reliably collect density, shape and orientation statistics from thousands of synapses.

II. RELATED WORK

Prior work on segmenting neuronal structures from EM volumes has covered a range of approaches from early attempts at full manual tracing [24], [9], [8] to semi-automatic methods requiring user initialization [5], [20], and lately fully-automated methods [21], [14].
Manual segmentation has clear and well understood limitations for the analysis of EM stacks. One notable example can be found in [36], where the successful segmentation of the nervous system of a nematode worm, containing only 302 neurons, necessitated a sustained effort over a ten year period. The need for expert knowledge and the growing size of EM datasets render manual segmentation impractical and highlight the need for automation.

Semi-automated methods based on active contours and level sets [3], [20], [5], [30], [15], [22] as well as graphcuts [27] have achieved some measure of success on EM images. However, these methods require careful manual initialization of each object to be segmented, which is done supplying seed points and tuning of various parameters. Though active interactions and feedback may in the long term prove essential to the successful large-scale segmentation of EM stacks, the amount of user input required by these methods remains prohibitively high. Ultimately, when applied to large EM data sets containing millions or even billions of structures, these semi-automatic segmentation methods suffer from the same intractability issue as their manual counterparts.

Recent research has focused on methods relying on machine learning, requiring little to no user interaction. Among those, several follow the same methodology by performing a segmentation in individual 2D EM slices before linking the segmented regions across slices in 3D. For instance, in [26], a perceptron operating on Hessian ridge was shown to provide promising results in segmenting membranes. However, in addition to the post-processing required to link membranes across the various 2D slices, this method also suffers from the need to remove internal sub-cellular structures from the segmentation result. In [35] a Boosted classifier operating on Gabor filter based features is used to segment mitochondria in 2D slices while a connected component analysis generated the desired 3D segmentation. In [28], 2D mitochondria segmentation followed by simple 3D interpolation is obtained from a number of classifiers including Adaboost, Support Vector Machines and Nearest Neighbor trained on Texton features. Finally, in [16], a random forest classifier trained with Haar like features is used to detect membranes in individual EM slices, while a graph cut optimization is used to enforce perceptual grouping and 3D continuity constraints.

While slice-by-slice methods have been shown to provide both reasonable segmentation results and computational savings, they fail to leverage the consistency of the structures in all three dimensions. This situation arises in part from the fact these approaches were designed for anisotropic EM modalities, such as transmission electron microscopy (TEM). While the reduced resolution in the z-direction makes slice-by-slice approaches a reasonable choice, recent works [14], [29], [2], [33] have demonstrated the benefits of processing the data directly in 3D even in highly anisotropic image stacks. More generally, the appearance of objects in 2D slices can be significantly altered depending on the 3D orientation of the object with respect to the stack axes. Given that such variability is far less pronounced when observing the objects in 3D, processing EM stacks slice by slice significantly complicates the segmentation task and can prove exceedingly detrimental when compared to direct 3D processing. Such a strategy is clearly foolhardy in the case of 2D images where the analogue would consist of a column by column or a row by row processing.

For these reasons, a number of works have addressed the segmentation of various neuronal structures directly in 3D. For example, [14] uses a multilayer convolutional artificial neural network (ANN) to segment neuronal membranes. By employing a convolutional ANN, [14] removes the need to hand design features and learn instead the necessary filters.
directly from the data. Andres et al. [2] propose a bottom-up hierarchical segmentation framework that uses a random forest classifier and watersheds to segment neural tissue. Though both of these methods produce excellent membrane segmentation results, they are designed for datasets prepared with an extra-cellular stain which highlights cell membranes while suppressing the various intracellular structures. In [33], an affinity graph that can be paired with standard partitioning algorithms is generated using a convolutional ANN. Much as in [14], this method learns both the features as well as the decision function directly from the data.

Even though progress has been made towards the automatic segmentation of neural structures, none of the aforementioned methods, whether operating in 2D or in 3D, can reliably segment objects such as synapses, which are characterized by specific arrangements of structures in addition to local textural cues. Though current algorithms generally compute feature in a neighborhood around the voxel of interest, they do not exploit context in a meaningful way: features are either pooled into global histograms [21], [28], are computed in regions centered around the voxel of interest [35], [14], [2], [16], or operate on a limited neighborhood around the voxel of interest [14], [33]. The resulting classifiers are therefore unable to hone in on arbitrary localized context cues.

The importance of context for the purposes of segmentation has been highlighted by a few attempts to leveraging ad-hoc and heuristic contextual cues to improve segmentation. For instance, [35] uses vesicle detection cues to suppress false alarms on vesicle clusters that can interfere with mitochondria segmentation, while [15], [34] propose to sample features in a 2D stencil neighborhood around the pixel of interest. By allowing the classifier to measure features computed at various locations in addition to the pixel of interest, [15], [34] are better at identifying membranes at regions of minor discontinuities by leveraging this auxiliary information. However, by relying on a pre-determined set of locations from which features can be sampled, these approaches strongly restrict the use of context. By contrast, our approach learns the relevant context automatically, overcoming these limitations.

Closest to our work is the state-of-the-art method of Kreshuk et al. [18], specifically targeted to synapse segmentation in isotropic image stacks. This approach relies on voxel-wise classification, training a Random Forest classifier that employs a set of pre-defined features such as smoothed gradient magnitudes, Laplacian of Gaussians and Hessian and Structure Tensor eigenvalues, evaluated at the voxel of interest. Therefore, context can only be captured through the isotropic Gaussian filters applied to the image stack, ignoring the presence of the asymmetric context information generated by the pre- and post-synaptic regions.

### III. Proposed Approach

We describe here our approach, which was first introduced in [4]. Let \( x \in \mathcal{X} = [0, 1]^{W \times H \times D} \) be an EM volume of width \( W \), height \( H \) and depth \( D \). Voxels are indexed by \( i \in \{1, \ldots, W \times H \times D\} \), and the location of each voxel is designated \( \ell_i \in \mathbb{N}^3 \).

Our goal is to find a function \( \varphi(x, \ell_i) \in \mathbb{R} \) that yields high scores at locations \( \ell_i \) in the volume that are part of synaptic tissue, and lower score values at those that are not.

As shown in Fig. 1(b), it can be difficult to distinguish synapses from other structures based solely on local texture. Human experts confirm their presence by looking nearby for post-synaptic densities and vesicles. This protocol cannot be emulated simply by measuring filter responses at the target voxel [18], pooling features into a global histogram [21], [28] or relying on hand-determined locations for feature extraction [15], [34].

To emulate the human ability to identify synapses, we design features, termed context cues, that can be extracted in any cube contained within a large volume centered on the voxel to be classified at \( \ell_i \), as depicted by Fig. 2(b). They are computed in several image channels using a number of Gaussian kernels, as shown in Fig. 3. As will be discussed in Sec. IV-B this yields more than 50,000 potential features. We therefore rely on AdaBoost [11] to select the most discriminative ones.

Given that synapses have arbitrary 3D orientations, we ensure that our context cues are computed at consistent locations across differently oriented synapses. We rely on the pose-indexing framework of [10] to enforce this consistency.

In the remainder of this section, we describe briefly the main structure of our context features. Their implementation is discussed in more detail later in § IV.

#### A. Context Cue Location

Let us consider a voxel located at \( \ell_i \) and an associated unit vector \( \mathbf{n}_i \in \mathbb{R}^3 \), as in Fig. 2(a). This unit vector is computed so that it is normal to the synaptic cleft. Let

\[
\mathbf{c}_p \in \mathbb{R}^3, \quad p = 1, \ldots, P
\]
denote a set of \( P \) locations expressed in the common \( x_0, y_0, z_0 \) reference frame shown at the center of Fig. 2(a). These locations are translated and rotated to occur at consistent locations relative to a target voxel by defining,

\[
e^{l_i} = l_i + R(l_i)c_p
\]

where \( R(l_i) \) is a 3x3 rotation matrix such that \( R(l_i)(0,0,1)^T = n_i \).

**B. Context Cue Features**

Given the \( e^{l_i}_p \) locations of Eq. 2, our goal now is to compute image statistics inside cubic neighborhoods \( N_r(c^{l_i}_p) \) of edge length \( 2r \) centered around these locations, such as those depicted in Fig. 2(b).

To this end, we precompute gradient magnitudes, Laplacians of Gaussian and eigenvalues of structure tensors and Hessians everywhere in the EM volume, as depicted by Fig. 3. Each of the resulting data cubes, in addition to the original one, is treated as a data channel \( m, 1 \leq m \leq 5 \), and is smoothed using isotropic Gaussian kernels of increasing variance \( \sigma_n \), as in [18]. We denote the gray levels in the resulting data volumes as

\[
H_{m,\sigma_n}(x,z) \in \mathcal{X},
\]

where \( x \) is the original EM volume and \( z \) represents the 3D location. We take context cue features to be

\[
f_{c_p,m,\sigma_n}(x,l_i) = \sum_{z \in N_r(c^{l_i}_p)} H_{m,\sigma_n}(x,z).
\]

In other words, we sum the smoothed channel output over the cubic boxes centered at all \( c_p \) for all possible values of \( m, \sigma_n \), and \( r \). This yields a set of \( K \) features, which we will denote for simplicity

\[
f_k(x,l_i), \; k = 1, \ldots, K,
\]

and which we use for classification purposes as explained next.

**C. Contextual Classifier**

Given the context features \( f_k \), we create decision stumps by simple thresholding and combine these stumps via a standard AdaBoost procedure [11] into a strong learner of the form

\[
\varphi(x,l_i) = \sum_{t=1}^{T} \alpha_t 1\{f_t(x,l_i) > \rho_t\}.
\]

This classifier is pose-indexed as its constituent features translate and rotate according to \( l_i \) and \( R(l_i) \) respectively.

**IV. IMPLEMENTATION DETAILS**

In what follows, the specifics of our implementation are provided. We follow the same notation as in §III.

**A. Image Channels**

Following the methodology employed by [18], we process each EM volume with several different filters, resulting in different data channels such as those of Fig. 3. Each channel is subsequently smoothed using isotropic Gaussian kernels of increasing variance \( \sigma_n \). The image filters used are as follows:

- Identity
- Gradient magnitude
- Laplacian Of Gaussian
- Hessian eigenvalues
- Structure Tensor eigenvalues

For the subsequent Gaussian smoothing step, isotropic filters with \( \sigma = \{1.0,1.6,3.5,5.0\} \) were used with the exception of the structure tensor for which a \( \rho = \{1.0,1.6,3.5,5.0\} \) with \( \sigma = \frac{\rho}{2} \) was used. This results in 20 different preprocessings of the original EM volume, each of which is available to construct boosting weak learners.
TABLE I
ALGORITHM PARAMETERS AND DEFAULT VALUES.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaBoost iterations</td>
<td>( T )</td>
<td>2000</td>
</tr>
<tr>
<td>Weighting-by-resampling ratio</td>
<td>( M = \frac{N_{\text{ad}}}{N_{\text{ps}}} )</td>
<td>2</td>
</tr>
<tr>
<td>Number of weak learners explored per iteration</td>
<td>( \Omega )</td>
<td>4000</td>
</tr>
<tr>
<td>Context cue maximum distance</td>
<td>( |c_p|_{\text{max}} )</td>
<td>40 voxels</td>
</tr>
<tr>
<td>Context cue distance quantization steps</td>
<td>( Q_{</td>
<td>\varepsilon</td>
</tr>
<tr>
<td>Context cue maximum box size</td>
<td>( r_{\text{max}} )</td>
<td>20 voxels</td>
</tr>
<tr>
<td>Context cue box size quantization steps</td>
<td>( Q_{\sigma} )</td>
<td>11</td>
</tr>
<tr>
<td>Context cue ( \varphi ) quantization steps</td>
<td>( Q_{\varphi} )</td>
<td>9</td>
</tr>
<tr>
<td>Context cue ( \theta ) quantization steps</td>
<td>( Q_{\theta} )</td>
<td>9</td>
</tr>
<tr>
<td>Supervoxel seed size</td>
<td>( SV_{n} )</td>
<td>2 voxels</td>
</tr>
<tr>
<td>Supervoxel cubeness</td>
<td>( SV_{m} )</td>
<td>16</td>
</tr>
<tr>
<td>Hessian scale for orientation estimation</td>
<td>( \sigma_{\text{Ho}} = \frac{w_a}{2\sqrt{2}} )</td>
<td>18 nm</td>
</tr>
</tbody>
</table>

B. Context Cue Parametrization

Context cue locations \( c_p, p = 1, \ldots, P \) in the common reference frame are parametrized in spherical coordinates as

\[
c_p = \left( \|c_p\| \cos \varphi_p \sin \theta_p, \|c_p\| \sin \varphi_p \sin \theta_p, \|c_p\| \cos \theta_p \right)
\]

with \( 0 \leq \|c_p\| \leq \|c_p\|_{\text{max}}, 0 \leq \theta_p \leq \pi \) and \( 0 \leq \varphi_p \leq 2\pi \). The parameter space is quantized uniformly in \( Q_{|\varepsilon|}, Q_{\sigma} \) and \( Q_{\varphi} \) bins respectively. This is also applied to the cube edge length \( 2r \), which is quantized in \( Q_r \) steps with \( \frac{\xi}{2} \leq r \leq r_{\text{max}} \).

To efficiently compute our context cue features of Eq. 4 efficient, we employ 3D integral images for each channel \( H_{m,\sigma_n} \). This allows us to compute the sum of any channel inside an arbitrary cube in constant time. Note that to allow for maximum consistency across the differently oriented synapses, the cubes over which sums of image channel values are computed should also be pose-indexed and hence rotate according to \( R(\ell_i) \). However, this would either impose a heavy memory burden if rotated integral volumes were used or a heavy computational burden otherwise. For this reason, we do not pose-index the cubes and restrict the boxes over which channel voxels are summed to be axis-aligned as shown in Fig. 2(b).

C. Estimating Synaptic Cleft Orientation

The context cues defined above are located relative to a normal estimate \( n_i \), which induces the rotation matrix \( R(\ell_i) \) of Eq. 2, such that

\[
R(\ell_i)(0, 0, 1)^T = n_i
\]  

(7)

Let \( \{\xi_1(\ell_i), \xi_2(\ell_i), \xi_3(\ell_i)\} \) be the Eigenvectors of the Hessian matrix at the voxel of interest, ordered by decreasing magnitude of their respective eigenvalues. Hence, \( \xi_3(\ell_i) = n_i \) corresponds to the eigenvector with the highest-magnitude eigenvalue, and is perpendicular to the synaptic cleft, assuming there is one at \( \ell_i \). We write

\[
R(\ell_i) = \begin{pmatrix} \omega_1(\ell_i) & \omega_2(\ell_i) & \omega_3(\ell_i) \end{pmatrix}
\]

(8)

with

\[
\omega_1(\ell_i) = \xi_3(\ell_i)
\]

(9)

\[
\omega_2(\ell_i) = \text{sign} \left[ D_u(\ell_i)(\ell_i) \right] \xi_2(\ell_i)
\]

(10)

\[
\omega_3(\ell_i) = \frac{\omega_1(\ell_i) \times \omega_2(\ell_i)}{\omega_2(\ell_i)}
\]

(11)

where \( D_u(x) \) is the directional derivative of the image along \( u \) at location \( x \). The motivation behind the latter is that \( R(\ell_i) \) in Eq. 7 is only defined up to a rotation in the \( \omega_1(\ell_i)/\omega_2(\ell_i) \) plane. However, it is preferable to define a rotation matrix that is consistent, particularly at \( \ell_i \) close to the edge of the synapses. As shown in Fig. 4, introducing Eq. 10 makes \( \omega_2(\ell_i) \) point towards the outside of the synaptic cleft when \( \ell_i \) is close to the border of the synapse. Towards the center of the synaptic cleft, there exists a rotational ambiguity in the \( \omega_1(\ell_i)/\omega_2(\ell_i) \) plane which can in fact be ignored due to the symmetry of synaptic structures there. Both the Hessian and the derivatives \( D_u(x) \) are computed at scale \( \sigma_{\text{Ho}} = \frac{w_a}{2\sqrt{2}} \) where \( w_a \) is the average synaptic cleft width, estimated once per EM volume from a single slice.

D. Learning Method

To make training computationally tractable, we specialize the AdaBoost learning procedure [11] as follows:
1) Weighting-by-sampling: Denote \( W = \{ w_1, ..., w_N \} \) and \( Y = y_1, ..., y_N \) the weights and labels for each training sample, with \( 1 \leq 1 \leq N \). Assume that the weights have been normalized such that \( \sum_{i=1}^{N} w_i = 1 \). At each AdaBoost iteration, instead of searching for the weak learner that minimizes the weighted error over all \( N \) training samples, we employ the weighting-by-sampling scheme [10] to approximate the distribution \( W \). This is done by finding the weak learner that minimizes a weighted error computed on a subset \( S \) of the training data, formally

\[
S = \{ S_P, S_N \} \tag{12}
\]

where \( S_P \) comprises all the positive samples and \( S_N \) is a subset of the negative samples, obtained by weighted sampling with replacement according to the weights \( W \). The weights of each sample in \( S_P \) is its respective weight in \( W \), while the samples in \( S_N \) are assigned a constant weight \( \frac{\sum_{i=1}^{N} w_i}{|| S_N ||} \).

In situations where the amount of negative samples outnumbers the number of positives, approaches such as weighting-by-sampling can reduce training time significantly. We call \( M = \frac{|| S_N ||}{|| S_P ||} \) the ratio between the number of negative and positive samples selected by weighting-by-sampling, which is a parameter for our algorithm. In our experiments we have observed that segmentation performance is robust against the value of \( M \). We set \( M = 2 \), which yields faster training without decreasing the performance of the final classifier.

2) Random Weak Learner Search: Due to the large number of possible weak learners \( f_k(x, \ell_i), k = 1, ..., K \), it is impractical to explore them all at each AdaBoost iteration. Instead, we only explore a subset of size \( \Omega \), obtained by randomly sampling from the pool of \( K \) possible weak learners. This also speeds up training significantly. We have experimented with different values of \( \Omega \) and observed that segmentation performance is fairly independent of its value. For all the results presented here we used \( \Omega = 4000 \).

E. Pose Indexing

1) Pose Annotations: Learning this classifier requires annotated training data. Since our contextual features are computed both for a given location and orientation, our training data must include both. While the location of synaptic voxels is manually specified by user annotation, synaptic orientation \( \omega_3(\ell_i) \) is automatically extracted as explained in § IV-C, using eigen analysis on the Hessian matrix computed at \( \ell_i \). Unfortunately, the obtained \( \omega_3(\ell_i) \) vector is only defined up to a polarity, which is problematic since pre-synaptic and post-synaptic regions are starkly different in appearance. We therefore follow the standard pose-indexing methodology [10] by labeling the polarity of the \( \omega_3(\ell_i) \) vector during training. During testing, our learned contextual classifier \( \varphi(\cdot) \) is evaluated for both polarities of the extracted \( \omega_3(\ell_i) \) vector and the maximum response retained. Note that the orientation labeling procedure in training is extremely efficient requiring a single user input per synapse to consistently direct all \( \omega_3(\ell_i) \) vectors to the pre-synaptic region.

2) Sampling Negatives: Under the pose-indexing framework [10], samples that do not exhibit the same pose (location and orientation) as positive samples should be considered as negative during training. However, samples that are too close in pose-space to the positives should be excluded from training, as their appearance can be similar to that of true positives, which may decrease performance considerably if they were included as negatives.

For example, following [10], any voxel lying on the synaptic cleft with an incorrect orientation polarity should be treated as a negative sample. However, doing this would force AdaBoost to learn a model solely based on context information, discarding local appearance. Given that experts find synapses through a mix of local appearance as well as pre- and post-synaptic regions, this would be counter-productive.

Thus, we setup conservative training exclusion zones in pose-space around our positive examples and sample negative examples outside these exclusion zones. In practice, we do not use positive voxels with the wrong orientation polarity, as discussed above. Moreover, we also exclude voxels that are outside the synaptic cleft and less than 10 voxels away from a positive-labeled voxel.

F. Supervoxels

Our entire algorithm including feature extraction, training and testing is designed and implemented to operate on individual voxels of the EM volumes. However, significant computational savings can be achieved by grouping voxels into supervoxels [1] for specific operations for training and testing. Thus, during training, instead of using every voxel as a positive or negative data sample, we restrict our method to training only on voxels corresponding to centers of super-voxels. In effect, this amounts to a spatially-driven sampling of the training data which significantly speeds up training while maintaining performance. Likewise, during testing, instead of evaluating our learned contextual classifier \( \varphi(\cdot) \) on every voxel in the EM test volume, we only evaluate \( \varphi(\cdot) \) on voxels corresponding to super-voxel centers while off-center voxels are assigned a response equal to that of the center.
Fig. 6. Slice cut of a region from the hippocampus dataset. Synapses indicated with circles. Labeling voxels close to the synaptic boundary is an ill-posed problem.

V. EXPERIMENTS

We evaluated our method the three different EM stacks acquired from different regions of the rat brain depicted by Fig. 5. We assessed performance both in terms of voxel-wise segmentation and synapse detection.

In this section, we first describe these datasets and our training and evaluation methodology. We then use them to evaluate both the voxel-wise precision of our method and its accuracy in terms of how many entire synapses are correctly detected. We use the method of [18] as a baseline against which we compare our results. Finally, we show that they can be used to compute biologically relevant statistics and discuss computational complexity issues.

A. Datasets and Evaluation Methodology

We used three different datasets from (A) the Somatosensory Cortex, (B) Hippocampus, and (C) Cerebellum of an adult rat. Example slice cuts of each dataset are shown in Fig. 5.

The amounts of training and test data for each dataset are summarized in Table II. The volumes were annotated in a voxel-wise fashion using Fiji [32]. Note that testing volumes were fully annotated, each voxel being assigned a synapse or a non-synapse label, in order to generate as large as possible a test set and report meaningful results. Training volumes on the other hand, in particular the large datasets A and C, were only partially annotated in order to reduce labeling cost. In those cases, an approach similar to what was used in [18] was followed, labeling a fraction of the voxels inside the volume as positive or negative, leaving most of the voxels un-annotated and therefore not using them for training. In the case of Dataset B, two test subvolumes were extracted from different regions of the Hippocampus.

1) Data Annotation: Synapse labeling close to the synaptic boundary is an ill-posed problem because this boundary is generally blurry, as shown in Fig. 6.

For training purposes, we adopted a conservative labeling policy whereby voxels are labeled positive only if experts are highly confident. This yields a ground truth volume whose positive samples are mainly located at the center of synapses.

For testing purposes, the behavior of a method on ambiguous voxels may be of particular interest to the practitioner. Thus, we adopted a different procedure for the annotation of test volumes. Experts were given more freedom in deciding whether or not a specific voxel lies on a synaptic cleft while our evaluation procedure, explained next, was designed to study the behavior of our detector at synaptic boundaries.

2) Evaluation Methodology: Voxel-wise evaluation is essential to assess and compare the performance of different segmentation methods. However, such an evaluation must take into account the aforementioned boundary issue to provide meaningful performance measures.

Much as we defined a training exclusion zone, we define a testing exclusion zone about the labeled border of the synapse with an exterior radius of $d^+$ and an interior radius of $d^-$, as depicted by Fig. 7. Voxel overlap between ground truth and exclusion zone shown in black in (d). Note the labeling ambiguity in (b) for the voxels close to the vesicles.

When the exclusion zone is very small, the performance measure may degrade mainly because of misclassification of boundary voxels; in contrast, as the exclusion zone size...
increases, the measure will experience stronger influence from voxels far away from the synapses.

To facilitate interpretation, we fix the ratio $\eta = \frac{d^+}{d^-}$ and plot the performance measure as a function of $d = d^+$. In our experiments we considered $0 \leq d \leq 5$ voxels and fixed $\eta = 2.5$ so that the maximum $d^-$ is 2 voxels. Limiting $d^-$ is essential to confine the exclusion zone to boundary voxels only, thus preserving most of the labeled synaptic voxels such that the evaluation remains meaningful. In practice, the exclusion zone is found by pre-computing a chamfer distance volume w.r.t. the synaptic boundaries in the ground truth.

B. Voxel-wise Accuracy

We employ precision-recall (PR) curves and the Jaccard index, also known as the VOC score [21], to evaluate the voxel-wise segmentation performance of our approach and compare it to [18].

Precision-recall curves at different exclusion zone sizes $d$ are shown in Fig. 9. Our approach clearly outperforms the baseline for all recall values as well as exclusion zone sizes $d$. We also show the value of the highest Jaccard index obtained at different values of $d$ in Fig. 10, which manifests the same behavior.

For comparison, we also evaluate the performance of our approach when only using local information, that is with $\|c_p\|_{\text{max}} = 0$. This means that post- and pre-synaptic context is not exploited, yielding a degenerate version of our method. Its performance is labeled Our approach: local only in the plots, and it shows that ignoring context results in a significant performance drop with respect to our full approach, evidencing
Fig. 9. Precision-recall curves for each dataset for different values of $d$. Our approach always yields better performance than the baseline of [18].

Fig. 10. Highest Jaccard index (VOC score) as a function of exclusion zone size $d$ for the different datasets. Our approach outperforms [18] for all values of $d$. 
the importance of contextual information for synapse segmentation.

1) Consistency of Contextual Cues: Context cue locations are rotated according to the orientation estimate at each voxel of interest $\ell_i$, which allows us to exploit synaptic context in a meaningful way. Therefore, it is interesting to observe the effects of setting the orientation estimate of all voxels to a fixed value. To do so, we have fixed $R(\ell_i)$ to the identity matrix for all locations $\ell_i$, which makes $\omega_3(\ell_i)$ point in the $z$ direction. This is equivalent to our approach without pose-indexing, similar to the method proposed in [14], [33]. We call this particular implementation Our approach: fixed orientation estimate, and a comparison with our full method is presented in Fig. 11.

As expected, performance drops significantly since context is relative to the orientation of the synaptic cleft, and fixing $R(\ell_i)$ to an identity transformation yields inconsistent contextual cues.

C. Detection Accuracy

So far, we have evaluated the different approaches for voxel-wise segmentation. However, it is also interesting to evaluate synapse detection performance, that is if a synapse as a whole is detected by the algorithm or not. We measure detection performance by clustering thresholded score volumes. This can be summarized as:

1) Threshold the score volume at the value that yields the best VOC score for all $d$.
2) Run connected component analysis on the resulting binary volume.
3) Remove detected clusters with less than 1000 voxels, as in [18].
4) Count the number of missed and false-positive clusters/detections.

False-positive detections are clusters of positive-predicted voxels who do not intersect with the ground truth, while missed (false-negative) detections are ground truth clusters that do not intersect with the predicted score volume after step 3 above.

As shown in Table III both the baseline and our technique detect all synapses. The difference lies in a significant reduction in the number of false positives when using our approach. Furthermore, because our method yields much better voxel-wise accuracy the shape of the recovered synapses is much closer to the ground-truth, as depicted by Fig. 12.

Table III shows that both approaches detect all synapses, but our method reduces the false positive count significantly. See text for detailed description.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Our approach</th>
<th>Kreshuk et al. [18]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Somatosensory</td>
<td>Missed</td>
<td>0</td>
</tr>
<tr>
<td>Cortex</td>
<td>False Positives</td>
<td>0</td>
</tr>
<tr>
<td>Hippocampus</td>
<td>Missed</td>
<td>0</td>
</tr>
<tr>
<td>False Positives</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>Cerebellum</td>
<td>Missed</td>
<td>0</td>
</tr>
<tr>
<td>False Positives</td>
<td>9</td>
<td>35</td>
</tr>
</tbody>
</table>

Fig. 12. Examples of detected synaptic voxels thresholding for the best Jaccard index. Synapses are detected by both methods, but the baseline method yields poor results from a segmentation perspective.

D. Biological Statistics

We applied the classifier trained on the Somatosensory Cortex data to a large volume consisting of 1500x1125x750 voxels or 9.8 x 7.4 x 4.9 $\mu$m. This is the original volume from which the train and test sub-volumes were extracted.

The resulting score volume was smoothed with a Gaussian filter with unit variance and thresholded at the value that
corresponds to the maximum VOC score in the test volume. Afterwards, clusters of positive detections of less than 1000 voxels were discarded, as in [18]. This resulted in a total of 405 clusters of voxels that our approach labeled as synapses.

Finally, an expert went through the resulting segmentation volume, discarding 31 false positive synapses, obtaining a total of 374 verified synapses. This number is in agreement with the expected synapse density in the Somatosensory Cortex region (layer II) [6].

A 3D visualization of the detected synapses is shown in Fig. 14. It is interesting to observe the large variation in synapse shape and size, which is evidenced in the histograms of synapse size and synapse flatness1 Fig. 14(a) and Fig. 14(b).

Another interesting observation comes from Fig. 14(c), which is a scatter plot of synapse volume and flatness. There is a strong correlation between synapse volume and flatness. This occurs because synapses are membrane-like structures and the synaptic cleft width is constant across different synapses, independently of their size. Therefore, larger synapses are flatter than smaller ones, which is evidenced in Fig. 14(c).

In addition to generating segmentation results, our approach can also be used to determine the location of the post and pre-synaptic regions. This can be highly relevant in practice to determine the location of the axons and dendrites relative to a given synapse, helping reveal neural circuit connectivity.

As discussed Sec. IV-E, our technique evaluates two different scores for location $\ell_i$ at test time, one for each possible orientation polarity. The polarity with the highest score can be employed as a polarity estimate. Once the score image is thresholded, majority voting can be used to determine the most likely polarity.

We applied this technique to the predicted scores for the three datasets and observed 100% accuracy at predicting synapse polarity. Examples are shown in Fig. 13, where the synapse normal has been aligned to the horizontal axis. This is also an indication that our approach is exploiting context information.

E. Computational Complexity

In terms of computational cost, our approach is faster in both training and testing than [18]. Training time is reduced with the techniques mentioned in Section IV-D. Using stumps as weak learners is also an important advantage at test time, in contrast to Random Forest’s trees which are built with as many splits as necessary to separate training data perfectly.

Another speed up is obtained by using supervoxels to oversegment the EM volumes. The chosen seed size of 2 voxels translates in an average supervoxel size of $2 \times 2 \times 2 = 2^3$ voxels [1], which yields a 8x speed factor since only supervoxel centers are classified.

Table IV summarizes timings obtained for both methods. The number of trees in the Random Forest classifier was set to 500. To make comparison fair, we have modified the Random Forest implementation used by Kreshuk et al. [18] to use multiple threads, speeding up training and testing substantially. Note that, in the case of using the default Vigra implementation, the timings for the baseline method in Table IV would be an order of magnitude higher.

The parameters used for our approach are the default ones described in Table I. Note that the test timings for our approach already consider evaluating the two possible polarities for each location $\ell_i$.

VI. CONCLUSION

We presented a novel approach to synapse segmentation. It relies on a large set of image features, specifically designed to take spatial context into account, which are selected, weighted and combined using AdaBoost. We used three different EM datasets to demonstrate that our algorithm effectively distinguishes true synapses from other organelles that exhibit the same local texture.

Even though our approach was designed for synapse segmentation, we believe that it can be applied to segment other types of tissue where context plays an important role, such as in mitochondria membranes. Another potential extension is to handle anisotropic volumes by scaling our features differently in the isotropic and anisotropic axes.

REFERENCES


1 Flatness is computed as the ratio of the second-largest principal moment and the largest principal moment of the voxel positions in each voxel cluster, and was obtained with the Insight Toolkit [13].

2 The original version is found in the Vigra library (http://hci.iwr.uni-heidelberg.de/vigra/) and our version can be found online at http://cvlab.epfl.ch/%7ECjbecker
TABLE IV

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Our approach</th>
<th>Kreshuk et al. [18]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Single stump</td>
</tr>
<tr>
<td></td>
<td>($T = 2000$)</td>
<td>(average)</td>
</tr>
<tr>
<td>Somatosensory</td>
<td>Train</td>
<td>0:57 hs</td>
</tr>
<tr>
<td>Cortex</td>
<td>Test</td>
<td>0:13 hs</td>
</tr>
<tr>
<td>Hippocampus</td>
<td>Train</td>
<td>2:40 hs</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>1:20 hs</td>
</tr>
<tr>
<td>Cerebellum</td>
<td>Train</td>
<td>1:41 hs</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>1:35 hs</td>
</tr>
</tbody>
</table>

Fig. 14. Results of our approach on a large Somatosensory Cortex volume of 1500x1125x750 voxels: (top) 3D Visualization of the detected synapses and (bottom) some of the statistics than can be extracted from voxel-wise segmentation. The correlation between synapse volume and flatness in (c) evidences the fact that synapses are membrane-like objects.