Supervoxel-Based Segmentation of EM Image Stacks with Learned Shape Features

Aurélien Lucchi, Kevin Smith, Radhakrishna Achanta, Graham Knott, and Pascal Fua

***Abstract*—Immense amounts of high resolution data are now routinely produced thanks to recent advances in EM imaging. While a strong demand for automated analysis now exists, it is stifled by the lack of robust automatic 3D segmentation techniques. State-of-the-art Computer Vision algorithms designed to operate on natural 2D images tend to perform poorly when applied to EM image stacks for a number of reasons. The sheer size of a typical EM image stack renders many segmentation schemes intractable. Most approaches rely on local statistics that easily become confused when confronted with the noise and textures found within EM image stacks. The assumption that strong image gradients always correspond to object boundaries is violated by cluttered membranes belonging to numerous objects.**

In this work, we propose an automated graph partitioning scheme that addresses these issues. It reduces the computational complexity by operating on supervoxels instead of voxels, incorporates global shape features capable of describing the 3D shape of the target objects, and learns to recognize the distinctive appearance of true boundaries. Our experiments demonstrate that, when applied to segment mitochondria from neural tissue, our approach closely matches the performance of human annotators and outperforms a state-of-the-art 3D segmentation technique.

***Index Terms*—Electron microscopy, segmentation, supervoxels, mitochondria, shape features.**

I. Introduction

Electron microscopy (EM) is key to mapping the morphology of neural structures. Recent techniques, such as *Focused Ion Beam Scanning Electron Microscopy* (FIB-SEM), can now deliver image stacks at the nanometer resolution in all three dimensions, such as those depicted by Fig. 1. Such stacks show very fine structures that are critical to unlocking new insights into brain function, but are still mostly analyzed by hand, which can require months of tedious labor [36]. As a result, the vast majority of this very high quality data goes unused. Furthermore, although they contain tens of millions of voxels, these stacks span volumes smaller than $10 \times 10 \times 10$ μm, which presents less than a billionth of the volume of the entire mouse brain. If it is ever to be mapped in its entirety, automation will be required.

Manual segmentation remains dominant in part because most state-of-the-art automated algorithms that are reported in the Computer Vision literature perform well on standard natural image benchmarking data sets such as the Pascal VOC data set [11], but much less well when applied to EM imagery. Furthermore, many automated segmentation algorithms specifically designed to handle EM images tend to work on individual image slices [24], [38] because other EM modalities, such as *Transmission Electron Microscopy* (TEM), deliver image stacks with much lower resolution across slices than within them. As a consequence, they rarely take full advantage of the consistency in all three dimensions. Neither do they usually take into account global 3D geometric constraints.

To overcome these limitations we advocate a graph partitioning approach that combines the following components.

- **Operating on supervoxels instead of voxels.** We start by clustering groups of similar voxels into regularly spaced *supervoxels* of nearly uniform size, which can then be used to compute robust local statistics. This reduces the computational cost by several orders of magnitude without sacrificing accuracy because supervoxels naturally respect image boundaries.
- **Including global shape cues.** The supervoxels are connected to their neighbors by edges and form a graph that we want to partition. Most graph current segmentation approaches techniques rely on very local statistics. By contrast, we introduce features that capture non-local geometric properties and use them to evaluate how likely supervoxels are to be part of the target structure.
- **Learning boundary appearance.** EM data is notoriously noisy and complex. The standard assumption that strong image gradients always correspond to significant boundaries does not hold, as illustrated in Fig 1. Spatial cues and textural cues must be considered when determining where true object boundaries lay. We therefore train a classifier to recognize which pairs of supervoxels are most likely to straddle a relevant boundary. This prediction determines which edges of the supervoxel graph should most likely be cut during segmentation.

We demonstrate our approach for the purpose of segmenting mitochondria in FIB-SEM images and show it to be much more accurate than a state-of-the-art 3D segmentation approach [47]. Furthermore, we believe the concepts exposed here to be generic and applicable to many other target structures and image modalities.

II. Related Work

In this section, we first examine earlier works that specifically address the problem of EM image segmentation. We then focus on graph-partitioning methods and discuss why their extensions to EM imagery is far from straightforward, thus motivating our design choices.

A. Lucchi and K. Smith contributed equally to this work.

A. Lucchi, R. Achanta, K. Smith, and P. Fua are in the Computer, Communication, and Information Sciences Department; G. Knott is with the Interdisciplinary Center for Electron Microscopy, EPFL, Lausanne CH-1015 Switzerland. E-mail: firstname.lastname@epfl.ch.

Manuscript received December 23, 2010.

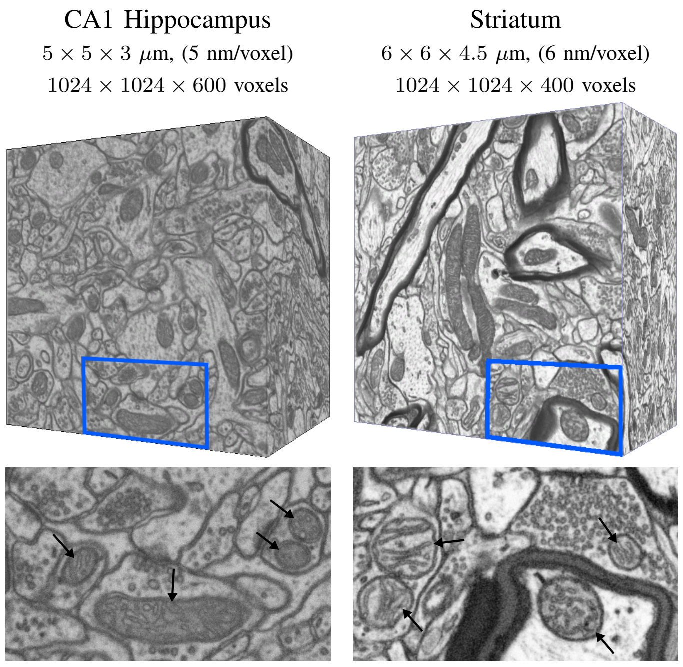
CA1 Hippocampus
$5 \times 5 \times 3$ μm, (5 nm/voxel)
$1024 \times 1024 \times 600$ voxels
Striatum
$6 \times 6 \times 4.5$ μm, (6 nm/voxel)
$1024 \times 1024 \times 400$ voxels

Fig. 1. *FIB-SEM data sets.* The top row contains 3D image stacks acquired using FIB-SEM microscopy. The bottom row depicts details contained in the blue boxes overlaid on the stacks. Mitochondria, which we wish to segment, are indicated by black arrows. The very high resolution allows neuroscientists to see key details but poses unique challenges. As shown in the top row, the image stack dimensions are very large, making inference intractable for most segmentation algorithms. Objects with distracting textures such as vesicles are abundant and mitochondrial membranes are easily confused with various others. The conventional assumption that strong contrasts denote object boundaries is unhelpful due the large number of high-contrast ones belonging to other structures.

A. Segmentation in neural EM imagery

Segmentation is a crucial step in automatic analysis of EM stacks, but such large data sets pose unique challenges. In addition to the very large number of voxels involved, algorithms have to contend with a variety of different structures including mitochondria, vesicles, synapses, and axonal or dendritic membranes, as can be seen in Fig. 1. They are difficult to distinguish from each other solely on the basis of local image statistics, especially given the often low signal-to-noise ratios of the data. This is one of the reasons why Computer Vision algorithms that perform well on natural images are far less successful on microscopy images and that special-purpose ones have been developed. Algorithms designed for EM imagery typically either detect membranes, reconstruct dendrites and axons, or segment structures such as mitochondria. We briefly review the three corresponding technique classes below and refer the interested reader to an excellent recent survey [20] for further details.

EM segmentation techniques are often tailored to the type of data they work on and can be further categorized according to it. Most EM data is acquired either via transmission electron microscopy (TEM) or scanning electron microscopy (SEM). TEM image stacks are produced by transmitting electrons through thinly sliced specimens, which must be aligned. The slicing process usually introduces a significant gap between slices, which results in anisotropic image stacks. By contrast, SEM techniques achieve much smaller gaps and higher isotropy by repeatedly milling ultra-thin layers from the specimen's surface and collecting backscattered electrons from it.

1) Finding membranes: Membrane detection is often a necessary step in other tasks such as stack registration or axon and dendrite reconstruction. The work reported in [51] is an early example of interactive membrane delineation in 2D TEM images. After image regularization by anisotropic diffusion, the user clicks points on the membrane and the algorithm traces a geodesic path joining them. In [48], an Hessian-based partial differential equation (PDE) is proposed to enhance boundaries in TEM image stacks. More recent work [22], [30], [55] relies on Machine Learning techniques for automated membrane detection. For example, in [22], Neural Networks relying on feature vectors composed of intensities sampled over stencil neighborhoods are trained to recognize membranes in TEM image stacks. Similarly, in [55], an Adaboost classifier is trained to detect cell membranes based on eigenvalues and orientation of Hessian features.

2) Segmenting axons and dendrites: The largest body of EM segmentation work focuses on axons and dendrites, a task also known as process reconstruction. The differences between TEM and SEM acquisition, discussed above, have given rise to a split between classes of algorithms that must explicitly handle the large gaps between image slices in the z direction and those that are freed from this burden. Ones from the first group operate on aligned TEM slices. They typically adopt either level-set or tracking-based approaches that enforce smoothness constraints to account for gaps in the data. Methods from the second group can treat the data in each direction similarly, allowing for natural extensions of state-of-the-art 2D segmentation approaches such as graph partitioning.

Because of the large gaps between slices, most TEM approaches outline the processes within slices and then follow a tracking-inspired approach to connect them across slices. The Kalman filter and the Dijkstra's algorithm have both been used to find an optimal path linking such outlines by applying a cost function that relates the description in one section to that in the next [21], [23]. In [37], neuron profiles are detected using a boundary detector and sequentially linked from adjacent sections by accounting for shape similarity and image texture on a weighted graph. In [52], geometrical arrangements known *a priori* are represented by active ribbons [13] within a level set framework and allowed to evolve from slice to slice. In [56], neighboring sections are jointly clustered to form segmentations in a process called co-clustering. Graph-cut segmentation is used in [38] in conjunction with a gradient flux field to find membranes, but it is restricted to individual slices due to the data anisotropy.

Since modern microscopy techniques can now deliver nearly isotropic high-resolution volumes, many of the difficulties that arise from having to link features extracted independently from different slices, can now be avoided by directly working with volumetric features. For example, in [19], a Convolutional Network (CN) performs neuronal segmentation by binary image restoration. This work is extended in [18] by incorporating topological constraints. In [50], CNs are used to predict an affinity graph that expresses which pixels should be grouped together using the Rand index [44], a quantitative measure of segmentation performance. In [58], the same combination of graph-cuts and gradient flux as in [38] is used but directly on volumetric data. Another recent graph-partitioning approach [24] invokes a random forest classifier to predict whether or not any given pixel belongs to a membrane on a volume of ultra-thin TEM image slices. None of these techniques, however, take advantage of global shape cues to learn the shape of the objects they segment.

3) Segmenting mitochondria: While mitochondria segmentation has received comparatively little attention, it is nevertheless an important problem because many neurodegenerative diseases are thought to be tied to mitochondrial abnormalities. In [39], shape-driven watersnakes that exploit prior knowledge about the shape of membranes are used to segment liver mitochondria in TEM stacks. In [57], a Gentle-Boost classifier is trained to detect mitochondria based on textural features. In [42], texton-based mitochondria classification of melanoma cells from Ion-Absorption SEM image stacks is performed using a variety of classifiers including k-NN, SVM, and Adaboost. While this technique achieves reasonable results by considering textural cues while ignoring shape information, it takes a similar approach to the work [47] we compare ourselves against with in Section IV.

B. Segmentation by graph-partitioning

While active contours and level sets have been successfully applied to many medical imaging problems [10], they suffer from two important limitations: first, each target object requires individual initialization and, second, each contour requires a shape prior that may not generalize well to variations in the target objects. EM image stacks contain hundreds of mitochondria, which vary greatly in size and shape, making the use these approaches problematic for our purposes.

In recent years, graph partitioning approaches to segmentation have become very popular. They have seen much success in 2D natural image segmentation [45], [40]. They currently dominate competitions such as the VOC segmentation challenge [11] and, in 2010, the top two competitors [9], [15] were relying on them. While these methods naturally extend from 2D images to 3D image stacks, computational complexity and memory requirements become limiting factors. Even for moderately-sized stacks, standard minimization techniques [27], [59], [29] become intractable. This is, in part, because most current methods operate on graphs whose vertices represent individual pixels or voxels. A straightforward solution is to crop, segment, and stitch sub-volumes. Since this introduces a number of undesirable effects [34], a more effective fix, albeit one that has only rarely been used, is to define the graph over groups of pixels or voxels, called superpixels [14] or supervoxels [54], respectively. This drastically reduces the size of the graph and thus the complexity. We adopt this approach in our work and introduce an algorithm that produces high-quality supervoxels without being computationally demanding in Section III-A.

Having addressed the complexity issue, we may formalize the graph partitioning segmentation approach. Graph partitioning approaches minimize a global objective function defined on an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ whose nodes i correspond to pixels, voxels, superpixels, or supervoxels, and whose edges connect to neighboring pixels, voxels, superpixels or supervoxels [5], [7], [2]. This energy function usually takes the standard form,

$$E(y|x,\lambda) = \sum_i \underbrace{\psi(y_i|x_i)}_{\text{unary term}} + \lambda \sum_{(i,j)\in\mathcal{E}} \underbrace{\phi(y_i, y_j|x_i, x_j)}_{\text{pairwise term}} \ , \quad (1)$$

which we adopt in our work where $\mathcal{E}$ is the set of edges and $y_i \in \{0, 1\}$ is a class label assigned to i corresponding to the foreground and the background. The so-called *unary* term ψ encourages agreement between a node's label c_i and the local image evidence x_i. ϕ is known as the *pairwise* term, which promotes consistency between labels of neighboring nodes i and j. The weight λ controls the relative importance of the two terms. Supplementary terms can be added into the energy function of Eq. 1. For example, a term favoring cuts that maximize the object's surface gradient flux was introduced in [26]. It alleviates the tendency of graph-cut algorithms to pinch off long or convoluted shapes, which is important when tracking elongated processes [38]. However, as noted in [24], it cannot entirely compensate for weakly detected membranes and further terms may have to be added.

When the pairwise term is submodular, which is often the case in practice, global minima of the energy function can be found. However, this does not mean that resulting segmentations are necessarily perfect, or even good. This is often because the criterion being minimized often fails to take global shape information into account, even though it is crucial to effective recognition and segmentation. Another contributing factor can be that the pairwise term does not necessarily properly encode the likelihood that edges of the graph should be cut and others retained. We briefly discuss current approaches to overcoming these shortcomings below.

1) Accounting for global shape information: The comparatively few graph-partitioning approaches addressing this issue can be categorized as either template or fragment-based.

Template-based approaches, such as [1], [12], [38], incorporate a shape template fitted to the image in an alignment or detection step. Templates can be either contours [12] or silhouettes [1], [38] representing target objects, which is learned or painstakingly constructed beforehand. It is used to modulate either the unary [1], [38] or pairwise [12] energy terms, usually through a distance transform. The complexity of this approach and the difficulty of simultaneously aligning multiple templates tends to confine it to detections of single well-centered objects.

Fragment-based approaches match image patches extracted around a node to a predefined fragment code book to incorporate shape information [3], [32]. However, for highly deformable objects, a prohibitively large code book is necessary, making this approach computationally expensive.

By contrast, in this paper, we will show that we can introduce global shape information directly into the unary term of Eq. 1 at a very modest computational cost.

2) Cutting the right edges: Most graph-partitioning approaches define the pairwise term as a simple function such as the one proposed in [5]

$$\phi(y_i, y_j | x_i, x_j) = \begin{cases} \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right) & \text{, if } y_i \neq y_j \\ 0 & \text{, otherwise,} \end{cases} \quad (2)$$

where the observation x_i is simply I_i, the color or intensity taken from node i, and σ is a constant. It favors cutting edges at locations of abrupt color or intensity changes. And, while expressions based on Laplacian zero-crossings, gradient orientations, local histograms exist [5], very few works go much further in the graph-partitioning context.

For images such as the ones we are dealing with which include many distracting contours, this is not optimal and may lead to semantically meaningless cuts. A recent exception can be found in [2] where the authors define an interaction term that encodes geometric relations between multi-region objects. In [41], the pixel intensities of patches situated around edges are used train a class-specific edge classifier whose result is used as pairwise term in a Markov Random Field. In our work, we follow a similar path by training a classifier to recognize which pairs of supervoxels are most likely to straddle a relevant boundary.

III. Our Approach

Our first step is to over-segment the image stack into *supervoxels*, that is, small clusters of similar intensity voxels. All subsequent steps operate on supervoxels instead of individual voxels, speeding up the algorithm by at least two orders of magnitude. For each supervoxel, we extract a feature vector that includes shape information and is fed to a classifier that predicts how likely it is to be within a mitochondrion. Feature vectors corresponding to pairs of neighboring supervoxels are fed to a second classifier that returns the probability that a mitochondrial boundary passes between them. These classifier outputs provide the unary and pairwise potential terms in Eq. 1. Finally, a segmentation is produced by minimizing this criterion using the max-flow min-cut approach of [6]. These steps are detailed below and we list parameters we used in Section IV-B.

A. Supervoxel Segmentation

Using supervoxels decreases image redundancy and greatly reduces the cost of subsequent image processing steps. Assuming that they naturally respect image boundaries, little is sacrificed in terms of segmentation accuracy. However, to be useful in our context, the algorithm used to generate them must meet several important requirements. It must be efficient in terms of computation and memory consumption while providing a way to control the size and number of the resulting supervoxels.

To meet these requirements, we extended our earlier *simple linear iterative clustering* (SLIC) superpixel algorithm [43] to produce 3D supervoxels such as those depicted in Fig. 2. The corresponding procedure is summarized in the Algorithm 1 Figure. At its heart is a k-means clustering step modified to limit distance calculations to a modest local volume. Initial cluster centers are chosen by sampling the image stack at regular intervals of length S in all three dimensions. Next, the centers are moved to the nearest gradient local minimum within their immediate neighborhood. The algorithm then assigns voxels to the nearest cluster center, recomputes the centers, and iterates. After n iterations, the final cluster members define the supervoxels. The key to fast clustering is a distance function measuring the spatial and intensity similarities of voxels within a $2S \times 2S \times 2S$ region

$$\delta = \sqrt{\frac{(I_k - I_i)^2}{m^2} + \frac{(u_k - u_i)^2 + (v_k - v_i)^2 + (z_k - z_i)^2}{S^2}}, \quad (3)$$

where I is image intensity; u_i, v_i, and z_i are the spatial coordinates of voxel i; u_k, v_k, and z_k are those of cluster center k. The spatial proximity and intensity similarity terms are normalized by S and m, which are the average expected spatial and intensity distances within a supervoxel, respectively. The process is repeated until it stabilizes. Finally, because this computation does not guarantee the spatial connectivity of the resulting supervoxels, a post-processing step assigns orphan voxels to the most similar nearby supervoxels using a flood-fill algorithm to check for size and connectivity. We refer the interested reader to [4] for further details.

S controls how many supervoxels are generated while m regulates compactness. Higher m values produce more compact supervoxels. Lower m values produce less compact ones that more tightly fit the image boundaries. Limiting the distance calculations to a $2S \times 2S \times 2S$ volume around the seed points ensures that the total number of distance calculations remains constant, irrespective of the number of supervoxels. This makes the complexity of the clustering step $O(Nn)$, whereas a conventional k-means implementation would be of complexity of $O(kNn)$ where n is the number of iterations required for convergence, N is the number of voxels or pixels in the image, and k is the number of cluster centers. The speed of the post-processing required to enforce connectivity depends on how many unconnected cluster components are generated. In practice, it also appears to be roughly $O(Nn)$ and takes less than 10% of the total computation time.

We could have extended other popular superpixel algorithms [31], [40], [53] for supervoxel generation. However, we felt SLIC [43] to be particularly well adapted because if delivers memory efficiency and speed in addition to size and compactness control. By contrast, the 2D version of [53] has a complexity of $O(N^2)$ and does not offer explicit control on the number or size of superpixels. The $O(NlogN)$ computational complexity of [40] is closer to that of SLIC but the algorithm requires seven times more memory in 3D in order to store the graph edges and thresholds assuming 6-connectivity, and even more assuming 26-connectivity. Furthermore, the algorithm cannot control the number or size of superpixels. While [31] also has an $O(N)$ complexity, we found it to be ten times slower than SLIC in the 2D case and noticed that it has an undesirable tendency to merge small superpixels that should not be. These comparisons are documented in [4].

B. Feature Vectors

To describe the image properties of individual supervoxels, we construct a feature vector $\mathbf{f}$ that combines two types of features: Ray descriptors and intensity histograms. For a given supervoxel i, the feature vector is written as

$$\mathbf{f}_i = [\mathbf{f}_i^{\text{Ray}\top}, \ \mathbf{f}_i^{\text{Hist}\top}]^\top , \quad (4)$$

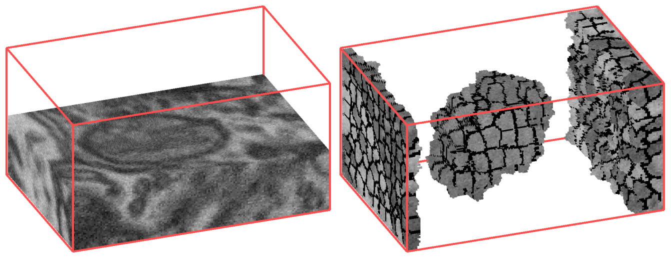

Fig. 2. *Segmenting an image stack into supervoxels.* On the left, a cropped FIB-SEM image stack containing a mitochondrion is shown. On the right, the cropped stack has been segmented using the SLIC algorithm to form small groups of voxels known as *supervoxels*. For ease of visualization, we have removed most of the supervoxels in the center of the image stack, leaving supervoxels belonging to the mitochondrion interior and supervoxels on the caps of the image stack. Boundaries between supervoxels are marked in black. Notice that SLIC groups voxels with similar intensities while respecting natural boundaries within the image stack.

where $\mathbf{f}_i^{\text{Ray}}$ represents a Ray descriptor and $\mathbf{f}_i^{\text{Hist}}$ represents an intensity histogram. For simplicity, we will omit the i subscript for $\mathbf{f}_i$ in the remainder of the section.

1) Ray Descriptors: Rays are a class of image features designed to capture non-local shape information around a supervoxel i. Because they capture shape relative to a given point, they fit naturally into our graph partitioning framework.

We introduced 2D Ray features in [46] and extend them here to 3D. As shown in Fig. 3, computing them involves repeatedly casting an imaginary ray in an arbitrary direction (θ_l, γ_l) from the center c_i of supervoxel i and measuring an image property at a distant point $\mathbf{r} = r(\mathbf{I}, c_i, \theta_l, \gamma_l)$ where the ray encounters an edge. In our implementation, these edges are found by a 3D extension of the Canny algorithm [17]. Given L orientations uniformly spaced over a geodesic sphere and defined by polar angles $\mathbf{\Theta_l} = \{\theta_1, \dots, \theta_L\}$ and $\mathbf{\Gamma_l} = \{\gamma_1, \dots, \gamma_L\}$, as depicted by Fig. 4, we take our *Ray descriptor* $\mathbf{f}^{\text{Ray}}$ to be a set of $3 \times L$ Ray features emanating from the center c_i of supervoxel i.

Algorithm 1 SLIC Supervoxels

```
1: Initialize cluster centers $C_k = [I_k, u_k, v_k, z_k]^T$ by sampling voxels at regular grid steps $S$.
2: Move each cluster center to the nearest gradient local minimum within its local neighborhood.
3: repeat
4:   for each cluster center $C_k$ do
5:     Assign the best matching voxels from a $2S \times 2S \times 2S$ cubic neighborhood around the cluster center according to the distance measure $\delta$ of Eq. 3.
6:   end for
7:   Compute new cluster centers and residual error $\epsilon$, the $L_1$ distance between previous centers and recomputed centers.
8: until $\epsilon \leq \text{threshold}, T$ or $n = n_{\text{max}}$
9: Enforce connectivity.
```

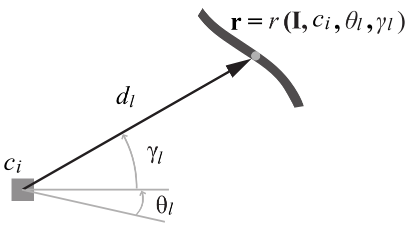

Fig. 3. *Ray feature function* $r(\mathbf{I}, c_i, \theta_l, \gamma_l)$. All components of the Ray descriptor depend on this basic function. For a given location c_i, it returns the location of the closest boundary point $\mathbf{r}$ in direction l defined by angles (θ_l, γ_l). d_l is the corresponding distance from c_i to the boundary.

This ensures that the Rays are uniformly spaced. We write

$$\mathbf{f}^{\text{Ray}}(\mathbf{I}, c_i, \theta_l, \gamma_l) = \begin{array}{l} [f_{\text{ndist}}(\mathbf{I}, c_i, \theta_l, \gamma_l), \\ f_{\text{norm}}(\mathbf{I}, c_i, \theta_l, \gamma_l), \\ f_{\text{ori}}(\mathbf{I}, c_i, \theta_l, \gamma_l)]^\top , \end{array} \tag{5}$$

where

$$\begin{aligned} \mathbf{f}_{\text{ndist}}(\mathbf{I}, c_i, \theta_l, \gamma_l) &= \frac{\|r(\mathbf{I}, c_i, \theta_l, \gamma_l) - c_i\|}{D} , \\ \mathbf{f}_{\text{norm}}(\mathbf{I}, c_i, \theta_l, \gamma_l) &= \|\nabla \mathbf{I}(r(\mathbf{I}, c_i, \theta_l, \gamma_l))\| , \\ \mathbf{f}_{\text{ori}}(\mathbf{I}, c_i, \theta_l, \gamma_l) &= \frac{\nabla \mathbf{I}(r(\mathbf{I}, c_i, \theta_l, \gamma_l))}{\|\nabla \mathbf{I}(r(\mathbf{I}, c_i, \theta_l))\|} \cdot \frac{\mathbf{r} - c_i}{\|\mathbf{r} - c_i\|} , \end{aligned} \tag{6}$$

and $\mathbf{I}$ and $\nabla \mathbf{I}$ are the image stack and its gradient.

In other words, the Ray descriptor $\mathbf{f}^{\text{Ray}}$ is made of three types of features that measure different image characteristics at the edge point $\mathbf{r}$:

- The most basic one, f_{ndist}, simply encodes the distance $d_l = \|r(\mathbf{I}, c_i, \theta_l, \gamma_l) - c_i\|$ to the closest edge. To make it scale-invariant, this distance is normalized by dividing it by D, the average of all such distances over the L directions being used.
- The other two, f_{norm} and f_{ori}, encode the gradient orientation and strength of the closest edge.

As shown in Fig. 4(a), two perpendicular axes $\mathbf{n_1}$ and $\mathbf{n_2}$ define a canonical frame of reference for the descriptor. The axes $\mathbf{n_1}$ and $\mathbf{n_2}$ are assigned specific locations in the feature vector shown in Fig. 4(b), all other elements being ordered according to their angular offsets from $\mathbf{n_1}$ and $\mathbf{n_2}$. To achieve rotational invariance, we re-order the descriptor such that $\mathbf{n_1}$ and $\mathbf{n_2}$ align with a local orientation estimate. Because the terminal points of the Ray descriptor capture the local shape, they can be used to estimate the local orientation. Applying Principle Component Analysis (PCA) to the set of terminal points yields two orthogonal vectors e_1 and e_2 in the directions of maximal variance of the local shape. Because e_1 and e_2 do not necessarily correspond to any of the Ray vectors, we pick the two closest Ray vectors $\mathbf{e_1}$ and $\mathbf{e_2}$ to be the principle axes, as shown in Fig. 4(d). Finally, the extracted feature vector shown in Fig. 4(e) is re-ordered such that the Rays corresponding to $\mathbf{e_1}$ and $\mathbf{e_2}$ fall in the bins assigned to $\mathbf{n_1}$ and $\mathbf{n_2}$ in the canonical feature vector shown in Fig. 4(b). Note that the accuracy of the pose estimation depends on the number of Rays in the descriptor, which therefore has to be taken to be sufficiently large.

2) Histogram Features: Recall from Eq. 4 that the feature vector $\mathbf{f}$ contains intensity histograms $\mathbf{f}^{\text{Hist}}$ extracted for a given supervoxel i and its neighborhood. It complements the

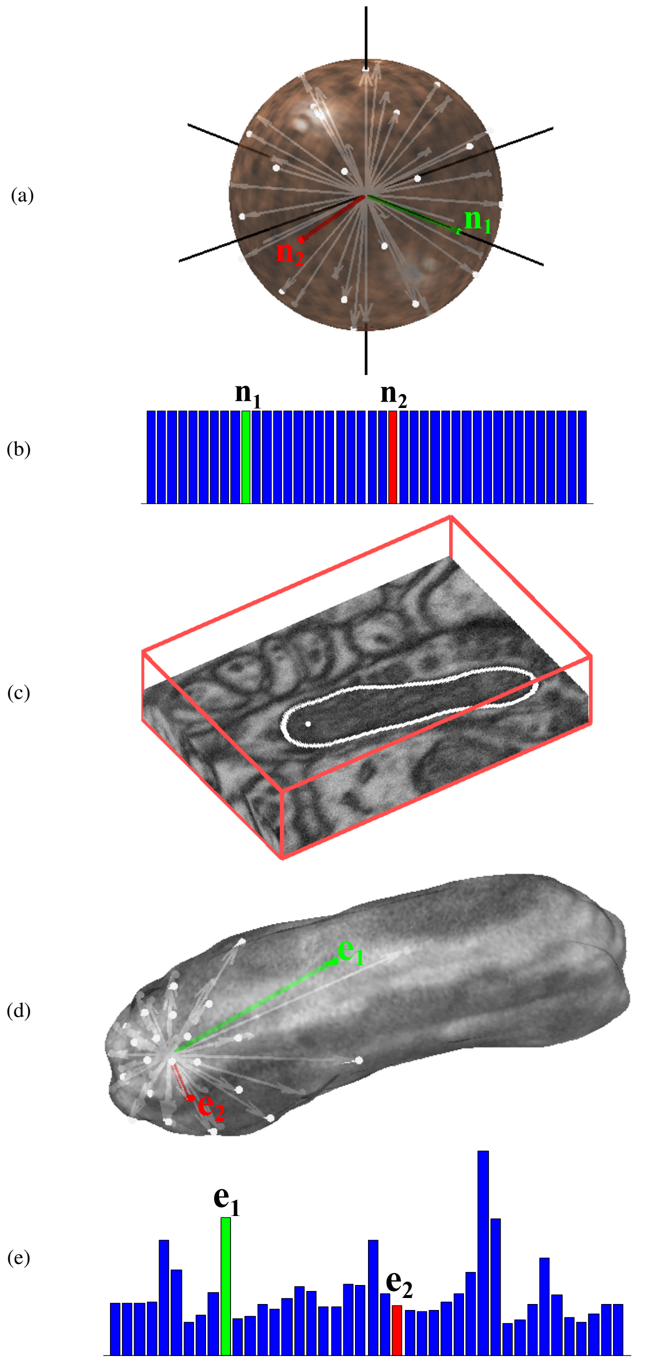

Fig. 4. *3D Ray descriptor*. (a) L Rays are cast from the center of a unit sphere to the closest surface boundary. The two axes defining the orientation of the descriptor $\mathbf{n_1}$ and $\mathbf{n_2}$ are shown in green and red, respectively. (b) The L f_{ndist} features of Eq. 6 are ordered so that the ones corresponding to $\mathbf{n_1}$ and $\mathbf{n_2}$ appear at fixed positions. (c) A cropped image stack containing a mitochondrion is shown, with boundaries highlighted in white. The ray descriptor is cast from the point labeled in white. (d) L Rays are cast from a location c_i to the closest surface boundary. The $\mathbf{e_1}$ and $\mathbf{e_2}$ axes of Section III-B are shown in green and red, respectively. (e) $\mathbf{e_1}$ and $\mathbf{e_2}$ are used to re-order the L features in the same order as in (b).

Ray features by providing low level intensity and texture cues. We tried several types of local texture and intensity features, including local binary patterns [33] and daisy [49], but found that a simple histogram computed from a supervoxel i and its set of neighboring supervoxels $\mathcal{N}$ yields the best results. Histogram features are a concatenation of two b-dimensional histograms. The first one is extracted from the central supervoxel i and the second from all supervoxels belonging to the neighborhood $\mathcal{N}$ of i. We write

$$\mathbf{f}^{\text{Hist}}(\mathbf{I}, i) = \left[h(\mathbf{I}, i, b), \ \frac{1}{|\mathcal{N}|} \sum_{j \in \mathcal{N}_i} h(\mathbf{I}, j, b) \right]^\top \tag{7}$$

where $h(\mathbf{I}, j, b)$ is a histogram extracted from $\mathbf{I}$ over the voxels contained in supervoxel j. Including the neighbors is necessary, because individual supervoxel statistics are nearly uniform by design, which reduces their discriminative power for classification purposes.

C. *Learning the unary and pairwise terms*

We segment the image stack by finding a graph cut that minimizes the energy function of Eq. 1. As previously discussed, we chose to train classifiers to predict both the unary and pairwise terms in Eq. 1. The unary term ψ is taken to be

$$\psi(y_i|x_i) = \frac{1}{1 + P_\psi(y_i|x_i)}, \tag{8}$$

and the pairwise term ϕ is defined as

$$\phi(y_i, y_j|x_i, x_j) = \begin{cases} \frac{1}{1+P_\phi(y_i,y_j|x_i,x_j)} & \text{, if } y_i \neq y_j \text{ ,} \\ 0 & \text{, otherwise,} \end{cases} \tag{9}$$

where $y_i = 0$ indicates background and $y_i = 1$ foreground. P_ψ represents the probability that i is within a mitochondria and P_ϕ the probability that i is within the mitochondria and i's neighbor j is outside, as described below. Since the resulting energy function satisfies the sub-modularity condition [28], the optimal labeling

$$\hat{y} = \underset{y}{\operatorname{argmin}}\, E(y|x, \lambda) \tag{10}$$

can be found by mincut-maxflow algorithm [16].

Because the mitochondria have thick boundaries with their own specific gray-level statistics, we train classifiers using manually annotated data with three labels $\{BG, BD, MI\}$, corresponding to background, boundary, and mitochondria instead of only background and mitochondria. Empirically, we found that introducing an explicit boundary class improved the classifiers' ability to recognize mitochondrial membranes from other membranes in the image stack. P_ψ and P_ϕ are both estimated with three-way one-vs-rest Support Vector Machine (SVM) classifiers using RBF kernels whose parameters were optimized through cross validation to minimize the estimated generalization error.

1) Unary Term: The first classifier is trained using the $\mathbf{f}$ feature vectors of Section III-B extracted from individual supervoxels randomly sampled from the training set. As a result, it returns probabilities of being within a mitochondria $P(MI|x_i)$, within the boundary $P(BD|x_i)$, or outside $P(BG|x_i)$. Since the boundary label separates background regions from mitochondria regions, we write

$$P_\psi(y_i|x_i) = \begin{cases} P(BG|x_i) & \text{, if } y_i = 0 \text{ ,} \\ P(BD|x_i) + P(MI|x_i) & \text{, otherwise .} \end{cases} \tag{11}$$

2) Pairwise Term: As discussed in Section II, while learning the unary term of Eq. 1 is common practice [41], most approaches rely on less sophisticated approaches to computing the pairwise term. It is usually taken to be inversely proportional to image gradients or gradient flux, so as to favor breaks at high-contrast locations. However, in EM imagery containing many distracting contours, this may backfire and result in erroneous cuts either along one of the many membranes found in the data or through a mitochondrial cristae. We address this problem by learning from the data what types of image characteristics indicate a true object boundary. In our specific application, relevant boundaries are characterized by a very dark membrane separating bright cytoplasm on the exterior, and the dark textured interior of the mitochondria on the interior as seen in Fig. 1. We therefore train the second three-way SVM using concatenated feature vectors from neighboring supervoxels i and j

$$\mathbf{f}_{i,j} = [\mathbf{f}_i^\top,\ \mathbf{f}_j^\top]^\top, \tag{12}$$

where $\mathbf{f}_i$ and $\mathbf{f}_j$ are the feature vectors extracted from the individual supervoxels. The resulting classifier assigns probabilities to one of the three classes $y_{ij} = \{0, 1, 2\}$ where class 0 corresponds to BD-BG pairs, class 1 corresponds to BD-BD pairs, and class 2 corresponds to any other combination $**$-$**$ of ground truth labels

$$P_\phi(y_i, y_j|x_i, x_j) = \begin{cases} P(y_{ij}=0|x_i,x_j) & \text{, if } y_i \neq y_j, \\ P(y_{ij}=1|x_i,x_j) + & \\ P(y_{ij}=2|x_i,x_j) & \text{, otherwise.} \end{cases} \tag{13}$$

IV. Experiments

In this section, we first give a background on the FIB-SEM data. We then present mitochondria segmentation results on whole image stacks taken from two different brain regions and list the parameters we used. Finally, we use smaller validation stacks to investigate some of the trade-offs involved and compare our approach to a state-of-the-art method [47].

A. Data and setup

Efforts to map the structural and functional neural connections within the brain promise to offer unique opportunities to better understand basic cognitive functions and related pathologies. An essential step in this process is to identify and map important sub-cellular components such as neural mitochondria, which are dynamic organelles that divide, fuse, and migrate within axons and dendrites. Among other things, they supply neurons with the energy they need and are therefore key to their survival and proper functioning. Furthermore, mitochondrial abnormalities have been linked to neurodegenerative diseases such as Parkinson's, making tools able to quantify such abnormalities of great value.

Mitochondria range from 0.5 to 10 μm in diameter [8] and optical microscopy does not provide sufficient resolution to reveal enough detail. Fortunately, recent EM advances have made it possible to acquire much higher resolution images that have already provided many insights into their structure and function [35]. Specifically, the images we use here were produced using FIB-SEM. It relies on a focused beam of ions that can be operated at low currents for imaging or high currents for milling. It can achieve 5 nm resolution in each of the spatial directions, allowing for the acquisition of nearly isotropic image stacks [25] such as the ones of Fig. 1.

TABLE I
PARAMETERS AND SETTINGS

Parameter	Value(s)	Notes
S	10	Normalized spatial distance. Controls the number of voxels per supervoxel (yields $\approx$ 1000).
m	20	Normalized intensity distance. Controls supervoxel compactness.
n	5	Number of iterations required for supervoxel clustering to converge.
$\lvert\mathcal{N}\rvert$	≈ 8	Typical supervoxel neighborhood size.
L	42	Number of Ray directions. Corresponds to vertices on a geodesic sphere.
ρ	≈ 50	Number of Ray features computed per supervoxel.
σ_G	9	Variance of Gaussian derivative filter used to compute gradient in f_{ori} and f_{norm}.
σ_C	(8,10)	Variance used in 3D Canny edge detection for (CA-1 Hippocampus, Striatum).
t_l	(8,14)	Lower threshold used in 3D Canny edge detection for (CA-1 Hippocampus, Striatum).
t_u	(16,27)	Upper threshold used in 3D Canny edge detection for (CA-1 Hippocampus, Striatum).
b	10	Number of histogram bins. $\mathbf{f}^{\text{Hist}}$ concatenates two b-bin histograms from i and i's neighborhood.

These stacks come from two different locations. The first is a $5 \times 5 \times 3$ μm section taken from the *CA1 hippocampus* and containing $1024 \times 1024 \times 600$ voxels. The second section measures approximately $6 \times 6 \times 4.6$ μm, and was taken from the *striatum*, a subcortical brain region. This image stack contains $1024 \times 1024 \times 400$ voxels.

B. Segmentation results and parameter choices

As shown in Fig. 5, we ran our algorithm on complete stacks in and extracted 3D reconstructions of the mitochondria. We used an 8-core Intel Xeon CPU 2.4 GHz machine with 48 GB RAM. Processing took 13 hours for the hippocampus data set and 9 hours for the striatum. As can be seen in the figure, the concentration of mitochondria is much larger in the striatum stack than in the other one. These results were obtained by setting the parameters introduced in Section III to the values listed in Table I.

The $S = 10$ sampling interval for supervoxel centers introduced in Section III-A was chosen so that the resulting supervoxels fit within the membranes, which are quite thick, thus guaranteeing that they do not straddle boundaries. A strength of our SLIC supervoxel generation scheme is that this S value could be automatically adapted if the image resolution were to be changed. This results in supervoxels containing on average 1000 voxels. As discussed in Section IV-C1, this decreases the computational complexity by several orders of magnitude as compared to what would have been required to operate directly on voxels.

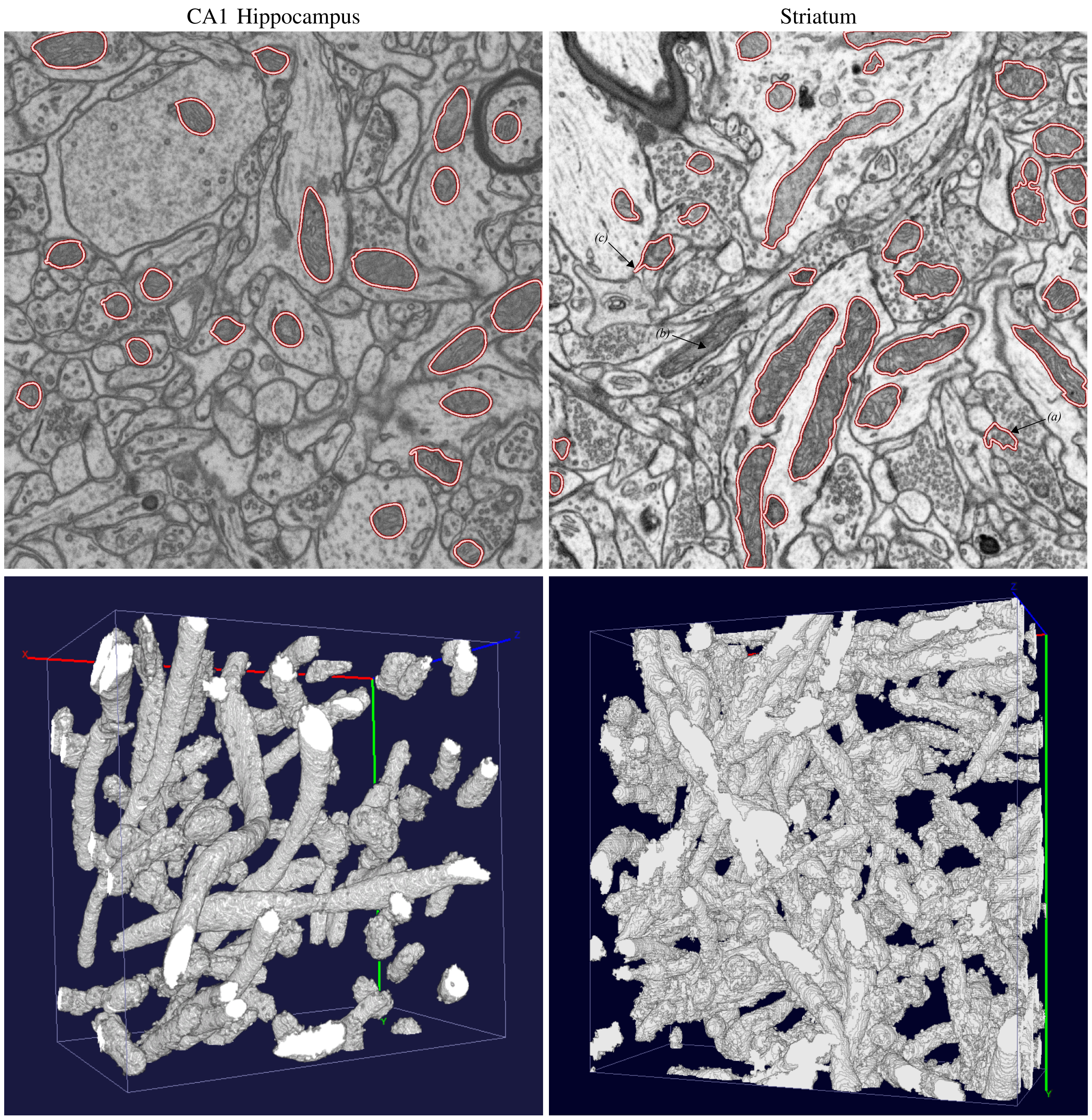

Fig. 5. *Segmentation of mitochondria from FIB-SEM image stacks and 3D reconstructions.* We applied our approach to two FIB-SEM image stacks acquired from different brain regions. The top row shows segmentation results on individual image slices taken from the image stack. Automatically segmented mitochondria are marked by red contours. The bottom row shows the 3D reconstructions of mitochondria extracted from the image stacks. The mitochondria are mostly correctly segmented, but some mistakes remain. They are highlighted by black arrows in the striatum slice in the upper right image. *(a)* A cluster of vesicles is mistaken for a mitochondria. The texture of vesicle packs can appear deceptively similar to that of mitochondria. *(b)* Our algorithm fails to segment some long and thin mitochondria due to the well-known "pinching-off" problem associated with graph cuts. *(c)* Dendritic or axonal membranes in close proximity to mitochondria can confuse our algorithm, causing it to include part of the nearby membrane with the mitochondria.

The ray descriptors $\mathbf{f}^{\text{Ray}}$ of Eq. 5 are 126 dimensional vectors, consisting of $3 \times L$ Ray feature responses. We have found $L = 42$ to be a good trade-off between computational complexity and angular resolution for the rotational alignment discussed at the end of Section III-B1. Rays terminate when they encounter edges found in a 3D Canny edge map [17]. Because the Canny edge detector can easily miss edges or add spurious ones, we increase robustness by shooting rays from 5% of the voxels within each supervoxel—50 in our case—for each direction and average the results. It is those averages that we use for classification purposes.

The intensity histogram features $\mathbf{f}^{\text{Hist}}$ of Eq. 7 are built by concatenating two 10-element histograms.

C. Training and validation

Ideally, the entire image stacks segmentations of Fig. 5 should be validated using annotated ground truth for the whole volume. However, because of the forbiddingly large amount of labor that would have been involved in generating

TABLE II
SEGMENTATION RESULTS

	Method				
	Ilastik	Standard $\mathbf{f}^{Hist}$	Learned Cube $\mathbf{f}$	Standard $\mathbf{f}$	Learned $\mathbf{f}$
VOC Score [11]	61%	63%	68%	81%	84%

such ground truth, we annotated smaller sub-stacks and used some for training and others for validation purposes. For the hippocampus data, our labeled training volume contained 200 slices of size 1024×1536. The validation set contained 165 similarly sized slices. For the striatum data, we annotated a smaller $850 \times 880 \times 70$ annotated volume, which was used for training only.

We summarize our validation results using the hippocampus data set in Table II. They are expressed in term of the so-called *Jaccard index*, or *VOC score* [11], which has become a *de facto* standard in the Computer Vision community to evaluate segmentation quality when ground-truth data is available. It is computed as

$$\text{VOC} = \frac{\text{True Pos}}{\text{True Pos} + \text{False Pos} + \text{False Neg}}, \qquad (14)$$

which is the ratio of the areas of the intersection between what has been segmented and the ground truth, and of their union. The VOC is reported for a fixed value of λ for each experiment, which was determined by cross-validation on the training data set. These λ values range from 0.07 to 0.13.

We can also vary the value of λ to explore the trade-off between the unary and pairwise terms in the energy function. Doing so changes the influence of the two terms in the final segmentation, which causes variations in the segmentation's true positive rate (TPR) and false positive rate (FPR), albeit in a non-linear fashion. Varying λ from 0 to 2 allows us to construct the ROC-like curves appearing in Figures 6, 7 and 8. Note that true ROC curves are obtained by varying a classification decision threshold when individual elements, supervoxels in our case, are classified *independently*. They do not apply where the entire graph of supervoxels is labeled *jointly* with a label $y_i \in \{0, 1\}$ using graph cuts Hence, we provide the ROC-like curves appearing in Figs. 6, 7 and 8 which still provide valuable insight into how consistently our algorithm performs over various choices for λ.

In the remainder of this subsection, we will use the VOC score and ROC-like curves to investigate the effects of our proposed contributions including the supervoxels, the Ray descriptors, and the learned pairwise term.

1) Benefits of using supervoxels: The computational bottlenecks in our approach are obtaining classifier probabilities and applying graph-cuts. The complexity of extracting the features and obtaining classifier probabilities is $O(N)$, that is, linear in the number of graph nodes. We use the graph cut approach of [6], which has a worst case complexity of $O(eN^2)$, where e is the number of edges. Using supervoxels instead of voxels reduces N by a factor of 1000 given the parameters described in IV-B and therefore very significantly speeds things up. It is also worth noting that, due to memory limitations, performing graph-cuts on a graph of voxels of this dimension is not possible with commercially available computers.

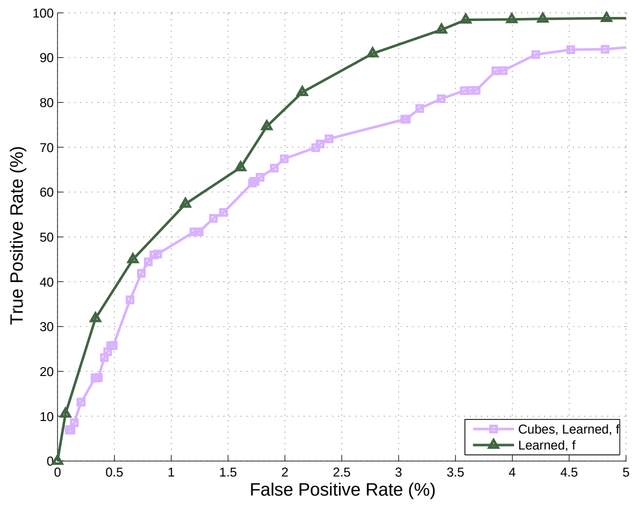

Fig. 6. *Supervoxels vs. regular cubes.* We compare segmentations obtained with our method using either SLIC supervoxels of size $S = 10$ or simple $10 \times 10 \times 10$ cubes. Supervoxels, which respect boundaries in the image stack, significantly outperform the cubes while yielding a similar computational complexity. As noted in Section IV-C, this ROC-like plot was generated by varying the weight λ, thus changing the influence of the unary and pairwise terms in the Energy function of Eq. 1.

To demonstrate the quality of the SLIC supervoxels and the importance of adhering to the object boundaries, we replace them by regularly spaced $10 \times 10 \times 10$ cubes, which have roughly the same size but ignore boundaries. As denoted by Cubes, Learned f in Fig. 6, the resulting segmentation is significantly worse than using SLIC supervoxels. Consequently, this results in a 16% drop in the VOC score of Eq. 14 when compared to using SLIC supervoxels, as shown in Table II.

2) Ray descriptors and learned pairwise term: Using the Ray features of Eq. 4 and learning the pairwise term of Eq. 9 are two critical components of our approach. To demonstrate their importance, we compare the performance of our system when using them as described in Section III as opposed to simpler baseline versions.

Fig. 7 shows that using the full feature vector $\mathbf{f}$ of Eq. 4 yields much better results than using only $\mathbf{f}^{\text{Hist}}$ intensity histograms of Eq. 7. Not using the Ray features translates in a 18% drop in the VOC score Eq. 14, as can be seen in Table II. Fig. 7 also shows a decrease in performance when replacing the pairwise term of Eq. 9 by a standard gradient-based term [5] of the form given in Eq. 2. For the purpose of this experiment, we set $\sigma = \frac{1}{2E[\hat{I}_i - \hat{I}_j]^2}$ in Eq. 2, where $\hat{I}_i$ is the average intensity within supervoxel i and $E[.]$ denotes the expectation over supervoxels. The corresponding drop in the VOC score is on the order of 3%.

D. Comparing against a state-of-the-art method

The Interactive Learning and Segmentation Tool Kit (Ilastik) is a software package for image classification and segmentation [47]. It allows for interactive labeling of an arbitrary number of classes in data sets of various dimensionality. Similar

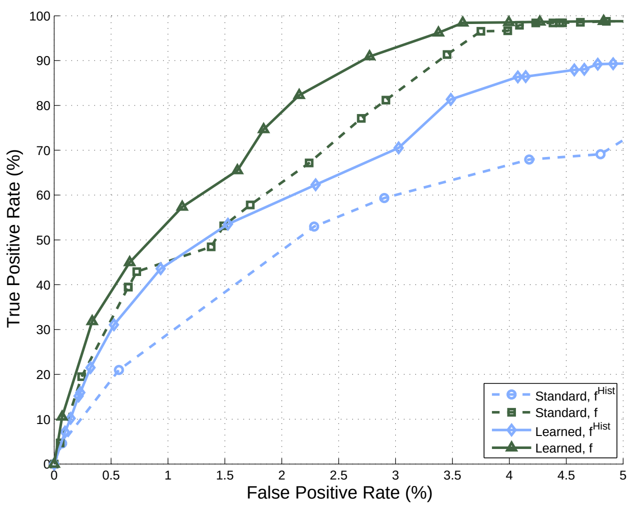

Fig. 7. *Contributions of the various elements of our approach.* The dashed blue line labeled "Standard, $\mathbf{f}^{\text{Hist}}$" represents a baseline result obtained by using a unary term that only depends on the histogram features of Eq. 7 and a contrast-based pairwise term given in Eq. 2. Replacing this pairwise term by the learned one of Eq. 9 results in the improved solid blue curve labeled "Learned, $\mathbf{f}^{\text{Hist}}$." An even larger improvement is obtained by introducing the Ray features of Eq. 4, producing the green dashed curve labeled "Standard, $\mathbf{f}$." Finally, combining the learned pairwise term and the Ray features yield the high quality result denoted by the solid green curve labeled "Learned, $\mathbf{f}$". This ROC-like curve is generated by varying λ, as explained in Section IV-C.

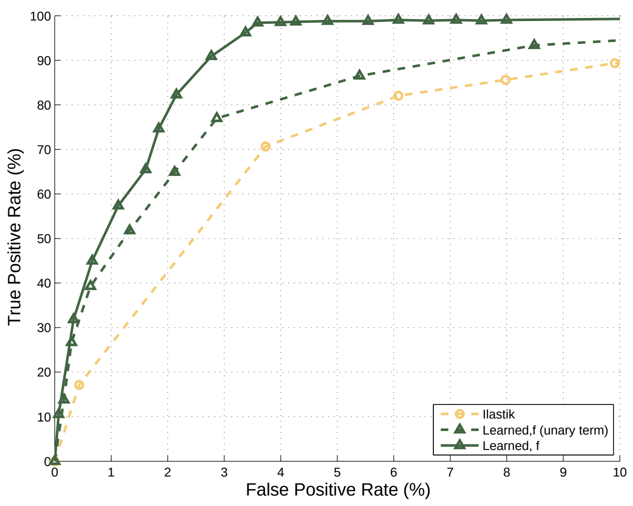

Fig. 8. *Comparing our approach to Ilastik* [47]. We trained Ilastik on the same data we used to train our SVMs and tested both our algorithms on the same validation stack. The solid green ROC-like curve was generated for our algorithm by varying λ, as explained in Section IV-C. Because Ilastik includes neither smoothing nor regularization, for a more fair comparison we plot as dotted lines the results obtained from Ilastik, and our results obtained by simply thresholding the unary term of Eq. 8. Note that thresholding our unary term does not perform as well as our full approach but still better than Ilastik, indicating that the features we use are better adapted to the task at hand. Note that the two dotted curves are conventional ROCs.

to the work of [42] which also segments mitochondria, Ilastik uses texture cues as well as color and edge orientation in a machine learning framework to perform segmentation. Ilastik's Random Forest classifier can provide real-time feedback of the current classifier predictions, allowing it to perform interactive or fully automatic classification and segmentation.

We provided Ilastik with the same training data used to train our approach, and compare its output to ours in Fig. 8. In addition to comparing Ilastik to our full approach, we also plot results obtained by simply thresholding probabilities of Eq. 8 that define the unary term in the energy function. We do this to compare the features we use against those of Ilastik, which does not include a smoothing or regularization step.

While Ilastik achieves a reasonable segmentation, our approach consistently outperforms it, even when using only the unary term, as shown in Fig. 8. As shown in Table II, our full approach outperforms Ilastik by a margin of 23% as measured by the VOC score. As can be seen in Fig. 9, Ilastik mistakenly labels vesicles as mitochondria and has trouble with other various membranes and synapses. Without the global shape information provided by the Ray features such mistakes are difficult to avoid.

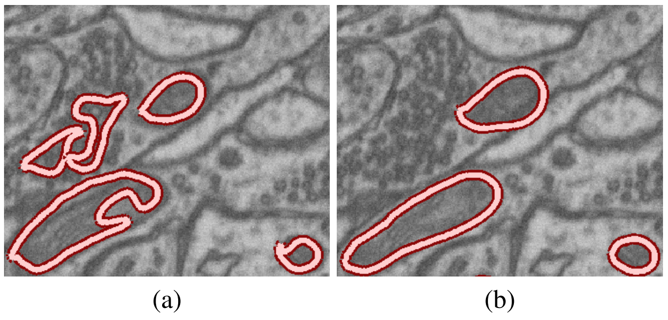

Fig. 9. *Visualizing the difference between Ilastik results and ours.* (a) The voxels of a particular slice that are labeled as being within mitochondria by Ilastik are marked by a red contour. These include a number of voxels that belong to vesicles instead of mitochondria. (b) These mistakes disappear when using our approach.

E. Failure modes

Qualitatively our segmentation results are very promising. Note that the 84% VOC score achieved by our algorithm is outstanding in terms of numbers reported in the VOC challenge [11]. However, this number should be taken with a grain of salt as the VOC Challenge contains 21 categories of objects, while we only deal with 2 – the mitochondria and the background. Nevertheless, there is still room for improvement. Two common failure modes are depicted in the upper right of Figure 5. Clusters of vesicles can mimic the texture of mitochondria, thereby confusing our classifiers. Also, some mitochondria have an uncharacteristically light or dark appearance, which can also cause problems for our algorithm. Very thin, elongated objects are known to get "pinched off" by graph cuts, causing our algorithm to miss some long, thin mitochondria. Finally, axonal or dendritic membranes occasionally in close proximity to mitochondria are occasionally included in the segmentation.

Increasing the amount of training data to account for these appearance variations would help. Furthermore, it would be relatively simple to exploit the fact that graph-cut minimization allows for efficient user interaction [5]. This means that, given an adequate interface, remaining errors could be quickly corrected by the user.

V. Conclusion

In this work, we proposed a fully automatic method to segment mitochondria from FIB-SEM image stacks. Our approach overcomes the limitations of standard graph-partitioning approaches to segmentation by: operating on supervoxels instead of voxels for computational efficiency, by using 3D Ray descriptors to model shape in the unary term, and by using a learning approach to model the appearance of the boundary in the pairwise term. We have demonstrated the computational efficiency of using supervoxels, and experimentally shown the increases in segmentation quality attributed with using 3D Ray descriptors and learning to model boundaries in the pairwise term. Our experiments have also demonstrated that our approach outperforms a state-of-the-art 3D segmentation method, and that our segmentation closely matches the performance of human annotators.

While the focus of our work is on the segmentation of mitochondria in FIB-SEM image stacks, the proposed techniques should have wide applicability to other cellular structures in EM as well as in other forms of microscopy. Future work will focus on learning boundaries using higher-order cliques, exploring the use of other features, and applying our technique to additional types of data.

Acknowledgments

The authors would like to thank Christopher Sommer for his help with Ilastik, Yunpeng Li for his help with the manuscript, and the developers of V3D which was used to render our 3D reconstructions.

References

[1] A. Ali, A. Farag, and A. El-Baz. Graph Cuts Framework for Kidney Segmentation With Prior Shape Constraints. In *MICCAI*, 2007.
[2] A. Delong and Y. Boykov. Globally Optimal Segmentation of Multi-Region Objects. In *International Conference on Computer Vision*, 2009.
[3] A. Levin and Y. Weiss. Learning to Combine Bottom-Up and Top-Down Segmentation. In *European Conference on Computer Vision*, 2006.
[4] R. Achanta. *Finding Objects of Interest in Images using Saliency and Superpixels*. PhD thesis, EPFL, Lausanne, 2010.
[5] Y. Boykov and M. Jolly. Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images. In *International Conference on Computer Vision*, 2001.
[6] Y. Boykov and V. Kolmogorov. An Experimental Comparison of Min-Cut/max-Flow Algorithms for Energy Minimization in Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.
[7] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut" - Interactive Foreground Extraction Using Iterated Graph Cuts. In *SIGGRAPH*, 2004.
[8] N. Campbell, B. Williamson, and R. Heyden. *Biology: Exploring Life*. Pearson Prentice Hall, first edition, 2006.
[9] J. Carreira and C. Sminchisescu. Constrained Parametric Min-Cuts for Automatic Object Segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2010.
[10] D.Padfield, J. Rittscher, N. Thomas, and B. Roysam. Spatio-temporal cell cycle phase analysis using level sets and fast marching methods. *Medical Image Analysis*, 13(1):143 – 155, 2009.
[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results.
[12] D. Freedman and T. Zhang. Interactive Graph-Cut Based Segmentation With Shape Priors. In *Conference on Computer Vision and Pattern Recognition*, 2005.
[13] P. Fua. Model-Based Optimization: Accurate and Consistent Site Modeling. In *International Society for Photogrammetry and Remote Sensing*, July 1996.
[14] B. Fulkerson, A. Vedaldi, and S. Soatto. Class Segmentation and Object Localization With Superpixel Neighborhoods. In *International Conference on Computer Vision*, 2009.
[15] J. Gonfaus, X. Boix, J. Weijer, A. Bagdanov, J. Serrat, and J. Gonzalez. Harmony Potentials for Joint Classification and Segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2010.
[16] D. Greig, B. Porteous, and A. Seheult. Exact Maximum a Posteriori Estimation for Binary Images. *Journal of the Roayl Statistical Society*, 51:271–279, 1989.
[17] L. Ibanez, W. Schroeder, L. Ng, and J. Cates. *The ITK Software Guide*. Kitware, Inc. ISBN 1-930934-15-7, http://www.itk.org/ItkSoftwareGuide.pdf, second edition, 2005.
[18] V. Jain, B. Bollmann, M. Richardson, D. Berger, M. Helmstaedter, K. Briggman, W. Denk, J. Mendenhall, W. Abraham, K. harris, N. Kasthuri, K. Hayworth, R. Schalek, J. Tapia, J. Lichtman, and H. Seung. Boundary Learning by Optimization with Topological Constraints. In *Conference on Computer Vision and Pattern Recognition*, 2010.
[19] V. Jain, J. Murray, F. Roth, S. Turaga, V. Zhigulin, K. Briggman, M. Helmstaedter, W. Denk, and H. Seung. Supervised Learning of Image Restoration with Convolutional Networks. In *International Conference on Computer Vision*, 2007.
[20] V. Jain, H. S. Seung, and S. Turaga. Machines that Learn to Segment Images: A Crucial Technology for Connectomics. *Current Opinion in Neurobiology*, 20:1–14, 2010.
[21] E. Jurrus, M. Hardy, T. Tasdizen, P. Fletcher, P. Koshevoy, C. Chien, W. Denk, and R. Whitaker. Axon Tracking in Serial Block-Face Scanning Electron Microscopy. In *MICCAI*, 2006.
[22] E. Jurrus, A. Paiva, S. Watanabe, J. Anderson, R. Whitaker, B. Jones, R. Marc, and T. Tasdizen. Detection of Neuron Membranes in Electron Microscopy Images using a Serial Neural Network Architecture. *Medical Image Analysis*, 2010.
[23] E. Jurrus, R. Whitaker, B. Jones, R. Marc, and T. Tasdizen. An Optimal-Path Approach for Neural Circuit Reconstruction. In *International Symposium on Biomedical Imaging*, May 2008.
[24] V. Kaynig, T. Fuchs, and J. Buhmann. Neuron Geometry Extraction by Perceptual Grouping in ssTEM Images. In *Conference on Computer Vision and Pattern Recognition*, 2010.
[25] G. Knott, H. Marchman, D. Wall, and B. Lich. Serial Section Scanning Electron Microscopy of Adult Brain Tissue Using Focused Ion Beam Milling. *Journal of Neuroscience*, 28(12):2959–64, 2008.
[26] V. Kolmogorov and Y. Boykov. What Metrics cnd be Approximated by Geo-Cuts, or Global Optimization of Length/Area and Flux. In *International Conference on Computer Vision*, 2005.
[27] V. Kolmogorov and C. Rother. Minimizing Nonsubmodular Functions With Graph Cuts-A Review. *PAMI*, 29(7):1274–1279, 2007.
[28] V. Kolmogorov and R. Zabih. What Energy Functions Can Be Minimized Via Graph Cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.
[29] N. Komodakis, G. Tziritas, and N. Paragios. Fast, Approximately Optimal Solutions for Single and Dynamic MRFs. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.
[30] R. Kumar, A. Vazquez-Reina, and H. Pfister. Radon-like Features and their Application to Connectomics. In *Workshop on Mathematical Methods in Biomedical Image Analysis*, 2010.
[31] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.
[32] M. Kumar, P. Torr, and A. Zisserman. OBJ CUT. In *Conference on Computer Vision and Pattern Recognition*, 2005.
[33] T. Mäenpää. *The Local Binary Pattern Approach to Texture Analysis-Extensions and Applications*. University of Oulu, Oulu Finland, 2003.
[34] F. Malmberg, C. Ostlund, and G. Borgefors. Binarization of Phase Contrast Volume Images of Fibrous Materials: A Case Study. In *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2009.
[35] C. Mannella, M. Marko, and K. Buttle. Reconsidering Mitochondrial Structure: New Views of an Old Organelle. *Trends Biochem. Sci.*, 15:37–38, 1997.
[36] B. Marsh, D. Mastronarde, K. Buttle, K. Howell, and J. McIntosh. Organellar Relationships in the Golgi Region of the Pancreateic Beta Cell Line, HIT-T15, Visualized by High Resolution Electron Tomography. *Proc. Nat. Acad. Sci.*, 98:2399–2406, 2001.
[37] Y. 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, 2008.
[38] N. Vu and B. Manjunath. Graph Cut Segmentation of Neuronal Structures from Transmission Electron Micrographs. In *International Conference on Image Processing*, 2008.

[39] H. Nguyen and Q. Ji. Shape-Driven Three-Dimensional Watersnake Segmentation of Biological Membranes in Electron Tomography. *IEEE Transactions on Medical Imaging*, 27(5):616–628, 2008.

[40] P. Felzenszwalb and D. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[41] M. Prosad, A. Zisserman, A. Fitzgibbon, M. Kumar, and P. Torr. Learning Class-Specific Edges for Object Detection and Segmentation. In *ICVGIP*, 2006.

[42] R. Narasimha, H. Ouyang, A. Gray, S. McLaughlin, and S. Subramaniam. Automatic Joint Classification and Segmentation of Whole Cell 3D Images. *Pattern Recognition*, 42(2009):1067–1079, 2007.

[43] A. Radhakrishna, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC Superpixels. Technical report, EPFL, June 2010.

[44] W. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, pages 846–850, 1971.

[45] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[46] K. Smith, A. Carleton, and V. Lepetit. Fast Ray Features for Learning Irregular Shapes. In *International Conference on Computer Vision*, 2009.

[47] C. Sommer, C. Straehle, U. Koethe, and F. Hamprecht. Interactive Learning and Segmentation Tool Kit. In *Systems Biology of Human Disease*, 2010.

[48] T. Tasdizen, R. Whitaker, R. Marc, and B. Jones. Enhancement of Cell Boundaries in Transmission Electron Microscopy Images. In *International Conference on Image Processing*, 2005.

[49] E. Tola, V. Lepetit, and P. Fua. Daisy: An Efficient Dense Descriptor Applied to Wide Baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):815–830, 2010.

[50] S. Turaga, J. Murray, V. Jain, F. Roth, M. Helmstaedter, K. Briggman, W. Denk, and H. Seung. Convolutional Networks can Learn to Generate Affinity Graphs for Image Segmentation. *Neural Computation*, 22:511–538, 2010.

[51] L. Vazquez, G. Sapiro, and G. Randall. Segmenting Neurons in Electronic Microscopy via Geometric Tracing. In *International Conference on Image Processing*, 1998.

[52] A. Vazquez-Reina, E. Miller, and H. Pfister. Multiphase Geometric Couplings for the Segmentation of Neuronal Processes. In *Conference on Computer Vision and Pattern Recognition*, 2009.

[53] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In *European Conference on Computer Vision*, 2008.

[54] O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and supervoxels in an energy optimization framework. In *European Conference on Computer Vision*, volume 6315, pages 211–224. Springer Berlin / Heidelberg, 2010.

[55] K. Venkataraju, A. Paiva, E. Jurrus, and T. Tasdizen. Automatic Markup of Neural Cell Membranes using Boosted Decision Stumps. In *IEEE Symposium on Biomedical Imaging: From Nano to Macro*, 2009.

[56] S. Vitaladevuni and R. Basri. Co-clustering of Image Segments Using Convex Optimization Applied to EM Neuronal Reconstruction. In *Conference on Computer Vision and Pattern Recognition*, 2010.

[57] S. Vitaladevuni, Y. Mishchenko, A. Genkin, D. Chklovskii, and K. Harris. Mitochondria detection in electron microscopy images. In *Workshop on Microscopic Image Analysis with Applications in Biology*, 2008.

[58] H. Yang and Y. Choe. 3D Volume Extraction of Densely Packed Cells in EM Data Stack by Forward and Backward Graph Cuts. In *International Symposium on Biomedical Imaging*, 2009.

[59] J. Yedidia, W. Freeman, and Y. Weiss. *Understanding belief propagation and its generalizations*, pages 239–269. Morgan Kaufmann Publishers Inc., 2003.