Neuroinform (2016) 14:235–250
DOI 10.1007/s12021-015-9288-z

ORIGINAL ARTICLE

A Fast Method for the Segmentation of Synaptic Junctions and Mitochondria in Serial Electron Microscopic Images of the Brain

Pablo Márquez Neila[1] · Luis Baumela[1] · Juncal González-Soriano[2] · Jose-Rodrigo Rodríguez[3] · Javier DeFelipe[3] · Ángel Merchán-Pérez[3,4]

Published online: 16 January 2016

Abstract Recent electron microscopy (EM) imaging techniques permit the automatic acquisition of a large number of serial sections from brain samples. Manual segmentation of these images is tedious, time-consuming and requires a high degree of user expertise. Therefore, there is considerable interest in developing automatic segmentation methods. However, currently available methods are computationally demanding in terms of computer time and memory usage, and to work properly many of them require image stacks to be isotropic, that is, voxels must have the same size in the X, Y and Z axes. We present a method that works with anisotropic voxels and that is computationally efficient allowing the segmentation of large image stacks. Our approach involves anisotropy-aware regularization via conditional random field inference and surface smoothing techniques to improve the segmentation and visualization. We have focused on the segmentation of mitochondria and synaptic junctions in EM stacks from the cerebral cortex, and have compared the results to those obtained by other methods. Our method is faster than other methods with similar segmentation results. Our image regularization procedure introduces high-level knowledge about the structure of labels. We have also reduced memory requirements with the introduction of energy optimization in overlapping partitions, which permits the regularization of very large image stacks. Finally, the surface smoothing step improves the appearance of three-dimensional renderings of the segmented volumes.

Keywords Three-dimensional electron microscopy · Automatic image segmentation · cerebral cortex · Mitochondria · Synapses

Electronic supplementary material The online version of this article (doi:10.1007/s12021-015-9288-z) contains supplementary material, which is available to authorized users.

✉ Ángel Merchán-Pérez
amerchan@fi.upm.es

[1] Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, 28660, Boadilla del Monte, Madrid, Spain

[2] Departamento de Anatomía. Facultad de Veterinaria, Universidad Complutense, Madrid, Spain

[3] Laboratorio Cajal de Circuitos Corticales, Centro de Tecnología Biomédica, Universidad Politécnica de Madrid and Instituto Cajal, CSIC, Madrid, Spain

[4] Departamento de Arquitectura y Tecnología de Sistemas Informáticos, Facultad de Informática, Universidad Politécnica de Madrid, Madrid, Spain

Introduction

The availability of technologies such as combined Focused Ion Beam milling/Scanning Electron Microscopy (FIB/SEM) and Serial Block-Face Scanning Electron Microscopy (SBFSEM) for the study of biological tissues permits the automated acquisition of large numbers of serial sections from brain samples (see for example (Denk and Horstmann 2004; Knott et al. 2008; Merchan-Perez et al. 2009)). These three-dimensional samples contain invaluable structural information that must be extracted from the stack of serial images. Electron micrographs of nervous tissue typically show a large variety of structures, such as neuronal and glial processes with their corresponding cytoplasmic organelles (e.g., vesicles, tubules, filaments and mitochondria) and synapses. From a practical point of view,

manual segmentation of these structures is a difficult and time-consuming task that requires a high degree of expertise. As a consequence, much effort has been devoted to the development of automated algorithms.

Brain images produced by electron microscopy (EM) are very complex and noisy with strong gray-level gradients that do not always correspond to region boundaries. Moreover, different neuronal structures may have similar local image appearance. Hence, it is extremely difficult to develop a fully automated segmentation algorithm. Although automated image processing techniques have addressed the problem of membrane detection and dendrite reconstruction (Turaga et al. 2010), standard computer vision algorithms used for the segmentation of textures (Haindl and Mikes 2008) or natural images (Martin et al. 2001) perform poorly, and standard techniques for the segmentation of biomedical images such as contour evolution (Jurrus et al. 2009) cannot handle the abundant image gradients.

Among the various structures visualized with EM, mitochondria and the synaptic junctions are of particular interest to neuroscience. Indeed, most information in the mammalian nervous system flows though chemical synapses. Thus, the quantification and measurement of synapses is a major goal in the study of brain synaptic organization in both health and disease (DeFelipe 2010). Mitochondria are organelles that produce most of the cell's supply of adenosine triphosphate (ATP) which transports chemical energy within cells for metabolism. In addition to supplying cellular energy, mitochondria are involved in many other crucial cellular physiological tasks (*e.g.*, McBride et al. 2006) and their alterations have been associated with a number of diseases such as Alzheimer's disease (*e.g.*, Santos et al. 2010). Therefore, substantial effort has been put into developing methods for accurate segmentation of synapses and mitochondria in the brain.

Although there are good practical synapse segmentation approaches relying on semi-automated tools (Morales et al. 2011), recent research has focused on machine learning approaches to diminish the degree of user interaction. (Becker et al. 2013) introduced a synaptic junction segmentation approach specifically designed for isotropic resolution image stacks, that is, stacks where voxel dimensions were identical in all X, Y and Z-axes. This method is based on a boosting algorithm that discovers local context cues related to the presence of the synaptic junction. The local context around potential synapse-like regions is also used in (Jagadeesh et al. 2013). However, the approach of Jagadeesh et al. relies on a computationally demanding set of image features that require up to 12 hours of computing time in a 32-node cluster. An alternative way of detecting synapses is by selectively staining them with ethanolic phosphotungstic acid (Navlakha et al. 2013), although this obscures other subcellular details and the tissue preservation is not appropriate for detailed ultrastructural analysis. Finally, (Kreshuk et al. 2011) used the Ilastik toolbox to segment synaptic junctions.

Several algorithms have been specifically designed to segment mitochondria in EM images. A texton-based approach comparing K-NN, SVM and AdaBoost classifiers was proposed (Narasimha et al. 2009). Lucchi and colleagues (Lucchi et al. 2012) later introduced an algorithm using as input 3D supervoxels and assuming almost isotropic image stacks. A different approach has been presented by Giuly and colleagues (Giuly et al. 2012). Their method performs the segmentation of mitochondria in anisotropic stacks of images. However, it is computationally very expensive and requires long processing times.

Consequently, our aim was to develop a method that does not require isotropic voxels and that is computationally efficient to allow the interactive segmentation of large image stacks that are now available. Moreover, our approach also involves image regularization and surface smoothing techniques to improve the segmentation.

Material & Methods

For the development of our segmentation algorithm we have used FIB/SEM image stacks that have been acquired in our laboratory from the rat somatosensory cortex (Merchan-Perez et al. 2009; Anton-Sanchez et al. 2014). The resolution was always the same in the X and Y axes and ranged from 3.7 nm per pixel to 14.7 nm per pixel. Resolution in the Z axis, equivalent to section thickness, was in all cases 20 nm per pixel. The stacks were thus anisotropic, that is, they did not have the same resolution in all three axes. Our segmentation algorithm has been specifically designed to take anisotropy into account, and we define the anisotropy factor as:

$$\rho = \frac{\text{Voxel size in the Z axis}}{\text{Voxel size in the X (or Y) axis}} \tag{1}$$

Thus, our stacks had anisotropy factors ranging from 5.41 to 1.36. To make our method comparable to others, we used an additional stack of SBFSEM images available online with an anisotropy factor $\rho = 5$.

Description of the Segmentation Algorithm

Our algorithm learns to detect and segment potentially any type of structure from the visual information in a stack of FIB/SEM or SBFSEM images, although in this work we have focused on synaptic junctions and mitochondria. To this end, the algorithm must be trained by providing samples of segmented structures. The user provides these samples in an interactive way

using an in-house application we have developed: first, a few voxels are manually labeled as mitochondria, for example, using standard tools such as the two or three-dimensional brush. The system then performs an automatic segmentation based on the training given, thus providing visual feedback. The user then refines the training by labeling new samples in the areas where the automatic segmentation is wrong. This procedure is repeated until the results are satisfactory (see Fig. 1).

Our automatic segmentation algorithm has three steps: feature extraction, voxel-wise classification and regularization. An optional fourth step, smoothing, enhances the visual appearance of the segmentation when it is rendered in 3D.

Feature Extraction

Feature extraction is performed on all voxels in the stack. The features of a voxel are a vector of real numbers that concisely describe the relevant visual information in the vicinity of that voxel. A feature extractor is a function from the space of EM stacks to the space of feature stacks. We have developed two feature extractors, F2D and F3D, which aggregate visual information around each voxel at several scales, and are rotationally invariant and robust to the noise present in EM images. F3D is a feature extractor that takes into account three-dimensional neighborhoods around each voxel. It is adequate for isotropic stacks. F2D, on the other hand, extracts a feature vector for each pixel in an image of the stack considering visual information of a neighborhood of the pixel in that slice and ignoring the information in other slices. F2D is a feature extractor that is suitable for anisotropic stacks. In the paragraphs that follow, we first describe F2D and then introduce F3D as a generalization.

F2D works on each image of the stack separately. Hence we only consider a single image I in the following description. F2D first applies a set of linear operators (zero, first and second order derivatives) to the smoothed image I at several scales. Thus, the set of linear operators at a scale σ is

$$\left\{G_\sigma *, \sigma \cdot G_\sigma * \frac{\partial}{\partial x}, \sigma \cdot G_\sigma * \frac{\partial}{\partial y}, \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial x^2}, \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial xy}, \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial y^2}\right\}, \quad (2)$$

where G_σ is a Gaussian filter of radius σ and $*$ is the convolution operator. The response to these operators will be noted as $s_{00}, s_{10}, s_{01}, s_{20}, s_{11}$ and s_{02} (note that the subscripts denote the order of the derivatives). With these responses to the filters at a scale σ, we can compute a partial feature vector for the pixels of I at that scale. This partial feature vector has four components:

$$\left\{s_{00}, \sqrt{s_{10}^2 + s_{01}^2}, \lambda_1, \lambda_2\right\}, \quad (3)$$

where s_{00} is the smoothed image, $\sqrt{s_{10}^2 + s_{01}^2}$ is the gradient magnitude and λ_1 and λ_2 are the first and second eigenvalues of the Hessian matrix, that are computed as

$$\lambda_1 = \frac{1}{2}\left(s_{20} + s_{02} + \sqrt{(s_{20} + s_{02})^2 + 4s_{11}^2}\right), \quad (4)$$

$$\lambda_2 = \frac{1}{2}\left(s_{20} + s_{02} - \sqrt{(s_{20} + s_{02})^2 + 4s_{11}^2}\right). \quad (5)$$

Figure 2 shows the components of a partial feature vector at a fixed scale for all the pixels in a single image.

The complete feature vector for each pixel is the concatenation of several partial feature vectors. We apply this procedure at n different scales $\{\sigma_0, \ldots, \sigma_{n-1}\}$, producing a feature vector with $4n$ components for each pixel in I. The set of scales should match the size of the structures that have to be detected in the images. In practice, the user only sets the initial scale σ_0, which we call the base scale, and the rest of scales are given by $\sigma_i = 2^{\frac{1}{2}i}\sigma_0$. For example, if we use $n = 4$ scales and set the smallest scale to $\sigma_0 = 4$ pixels, our feature vectors will have 16 dimensions and they will range from 4 to 11.31 pixels in scale.

F3D is a generalization of F2D for isotropic image stacks. As in F2D, the set of linear operators with the zero, first and second order derivatives at a given scale σ,

$$\left\{\begin{array}{l} G_\sigma *, \sigma \cdot G_\sigma * \frac{\partial}{\partial x}, \sigma \cdot G_\sigma * \frac{\partial}{\partial y}, \sigma \cdot G_\sigma * \frac{\partial}{\partial z}, \\ \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial x^2}, \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial xy}, \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial xz}, \\ \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial y^2}, \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial yz}, \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial z^2} \end{array}\right\}, \quad (6)$$

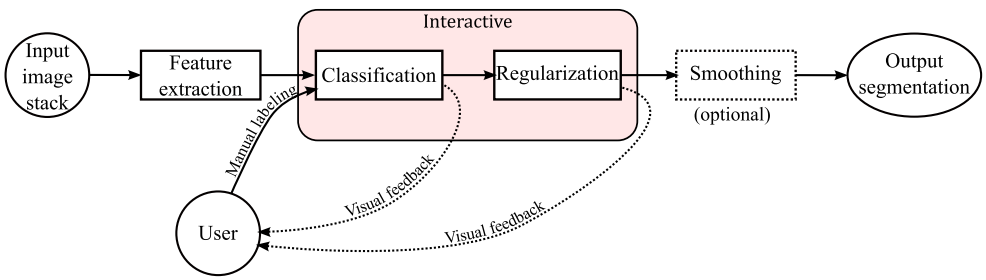

Fig. 1 Workflow of the segmentation algorithm. The input stack of serial images is subjected to four successive steps, two of which can be interactively modified by the user

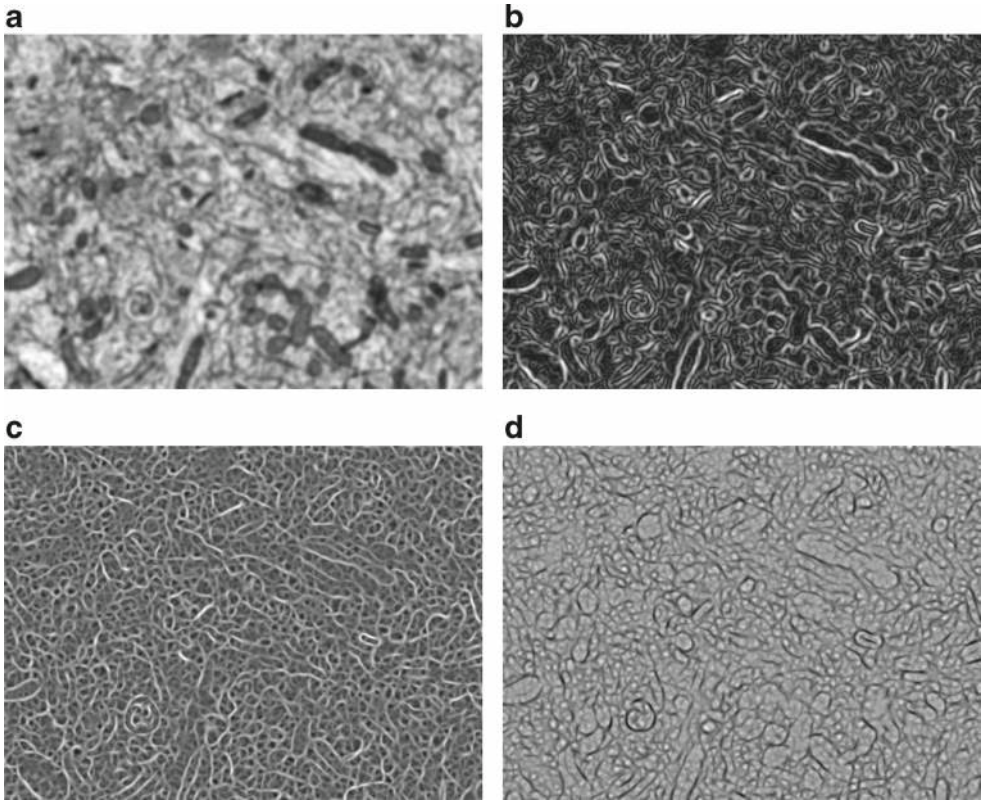

Fig. 2 Feature extraction from a single image. In this example, feature extraction has been performed at a scale $\sigma = 4$ pixels. The resulting partial feature vector has four components: the smoothed image (**a**), the gradient magnitude (**b**), and the first (**c**) and second eigenvalues (**d**) of the Hessian matrices. Feature extraction is also performed at several other scales (not shown) to obtain the complete feature vector

is applied to the stack obtaining the responses $\{s_{ijk} : i + j + k \leq 2\}$, where i, j and k indicate the order of the derivatives in the X, Y and Z axes. The partial feature vector for each voxel of the stack at a given scale is

$$\left\{ s_{000}, \sqrt{s_{100}^2 + s_{010}^2 + s_{001}^2}, \lambda_1, \lambda_2, \lambda_3 \right\}, \tag{7}$$

where the first component is the smoothed image, the second one is the magnitude of the gradient, and λ_1, λ_2 and λ_3 are the eigenvalues of the Hessian matrix

$$\begin{pmatrix} s_{200} & s_{110} & s_{101} \\ s_{110} & s_{020} & s_{011} \\ s_{101} & s_{011} & s_{002} \end{pmatrix}. \tag{8}$$

Again, the complete feature vector for each voxel is the concatenation of the partial feature vectors at several scales.

Classification

A classifier uses the feature vectors to determine the probability that a voxel belongs to each label. This classifier has to be trained with labeled data to learn the relationship between feature vectors and labels. Here we briefly present how the classifier is trained and how a trained classifier can be used with new unclassified data.

Our classifier learns the probability distribution $P(y_i \mid f_i(\mathbf{x}))$ of the label y_i for the pixel i given the observed feature vector $f_i(\mathbf{x})$. We use the Bayes' rule to express this distribution as a product

$$P(y_i \mid f_i(x)) \propto p(f_i(\mathbf{x}) \mid y_i) P(y_i), \tag{9}$$

where the conditional $p(f_i(\mathbf{x}) \mid y_i)$ is the probability density of the feature vector for voxels with label y_i and $P(y_i)$ is the prior probability of a pixel having the label y_i. We model the conditional distribution as a Gaussian,

$$p(f_i(\mathbf{x}) \mid y) = \frac{1}{\sqrt{(2\pi)^k \left|\Sigma_y\right|}} \exp\left(-\frac{1}{2} \left(f_i(\mathbf{x}) - \boldsymbol{\mu}_y\right)^T \Sigma_y^{-1} \left(f_i(\mathbf{x}) - \boldsymbol{\mu}_y\right) \right), \tag{10}$$

where the parameters $\boldsymbol{\mu}_y$ and Σ_y are the mean vector and the covariance matrix of the feature vectors for voxels with the label y, and k is the dimension of the feature vector.

In the training step these parameters are estimated from training data. The user manually labels a few voxels of the stack. During training, the voxels labeled with label y are used to estimate $\boldsymbol{\mu}_y$, i.e., the mean of the feature vectors of voxels labeled as y, and Σ_y, i.e., the covariance matrix of these feature vectors.

When the dimension of the feature vectors is large, the training data often falls in a proper subspace of the complete k-dimensional feature space producing a singular or near singular covariance matrix Σ_y. We avoid this problem by first performing Principal Component Analysis (PCA)-based dimensionality reduction on the cloud of all feature

vectors . The dimensionality after the PCA is established to retain 99 % of variance.

$P(y)$ is the a priori probability of the label y. It is learned from the user-provided data in the training step. In short, the training step consists of estimating the parameters $\boldsymbol{\mu}_y$ and Σ_y of the conditional distribution (after a PCA-based dimensionality reduction) and the prior $P(y)$ from the user-provided data with the interactive tool.

Once the classifier is trained, it processes every voxel in the EM stack. For each voxel i with feature vector $f_i(\mathbf{x})$, the probability $P(y_i \mid f_i(\mathbf{x}))$ is computed for every label y_i. This results in a probability map that maps every voxel to the probabilities of belonging to each label. As an example, see Fig. 3a, b.

In preliminary experiments, we tested other classifiers such as support vector machines. Although these methods improve the results obtained with the Gaussian classifier, their performance is only marginally better at the expense of much higher computational time (in the order of hours vs. seconds), which makes them unsuitable for operation in real time.

Regularization

If voxels are assumed to be independent of each other, it is possible to segment the stack by simply assigning to each voxel i the label y^* with higher probability, i.e., $y^* = argmax_y P(y \mid x)$. However, this offers far from optimal results, since the resulting segmentation is noisy, and it shows many sparse pixels, grainy regions and small holes (Fig. 3c, d).

Therefore, we have to assume some degree of probabilistic dependency between neighboring voxels of the stack. We have modeled this dependency by means of a conditional random field (CRF). A CRF models the distribution $P(\mathbf{Y} \mid \mathbf{x})$ between the set of observed variables x (i.e., the pixel values of the stack of images) and the hidden variables $\mathbf{Y} = \{y_1, \ldots, y_N\}$ (i.e., the segmentation labels) in such a way that $\mathbf{Y}$ conditioned on $\mathbf{x}$ holds the Markov property $P(y_i \mid \mathbf{x}, \mathbf{Y}_{-i})$, where $\mathbf{Y}_{-\mathbf{i}}$ is the set of label variables without y_i, and $N(i)$ are the neighboring voxels of voxel i. The neighborhood function N defines a graph (V, E) with the set of voxels V as nodes and the set of arcs E as given by the pair of voxels related by N. The graph constructed in this way is known as the CRF graph. The Hammersley–Clifford theorem states that the distribution of a CRF can be written as a Gibbs measure,

$$P(\mathbf{Y} \mid \mathbf{x}; \theta) = \frac{1}{Z(\mathbf{x}, \theta)} \prod_{c \in \mathcal{C}} \Psi_c(\mathbf{Y}_c, \mathbf{x}_c) = \frac{1}{Z(\mathbf{x}, \theta)} e^{-\sum_{c \in \mathcal{C}} \theta_c \cdot \phi_c(\mathbf{Y}_c, \mathbf{x}_c)}, \quad (11)$$

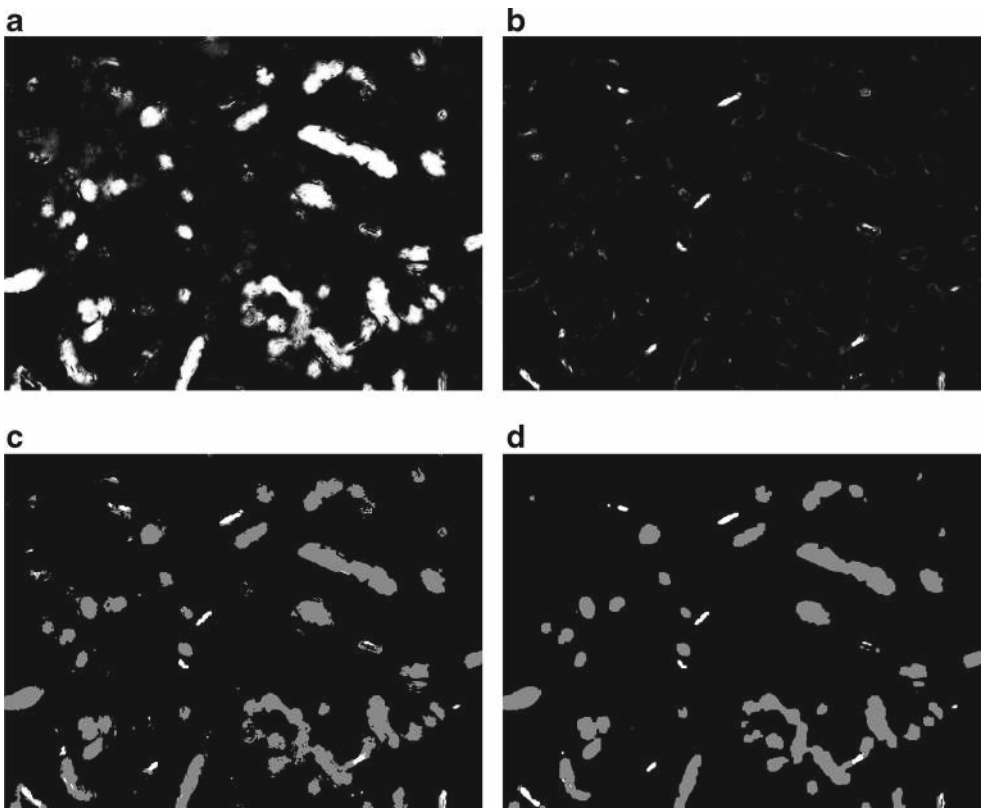

Fig. 3 Classification and regularization of a single image. (**a**) and (**b**): Probability maps obtained by a Gaussian classifier that has been trained with the features extracted from the same image that is shown in Figure 2. The pixel-wise probability of belonging to the label "Mitochondrion" is shown in (**a**) and the pixel-wise probability of belonging to the label "Synaptic junction" is shown in (**b**). (**c**): preliminary segmentation before regularization, where each pixel has simply been given the label with highest probability (Mitochondria, gray; Synaptic junctions, white). Note the sparse pixels scattered throughout the image, the small holes in some of the segmented objects and the jagged edges. (**d**): Final segmentation after regularization via CRF energy minimization. Most sparse pixels have disappeared and edges show a smoother appearance

where $Z(\mathbf{x}, \theta)$ is the partition function, $\mathcal{C}$ is the set of cliques in the CRF graph and $\mathbf{Y}_c$ is the set of variables from $\mathbf{Y}$ related to clique c.

In other words, the Gibbs measure expresses the distribution as the product of potentials Ψ_c (note that the potentials are not required to be probability distributions) depending on subsets of the complete set of random variables. The product of all potentials can be written as a weighted sum of factors using the minus logarithm,

$$-\log \prod_{c \in \mathcal{C}} \Psi_c(\mathbf{Y}_c, \mathbf{x}_c) = \sum_{c \in \mathcal{C}} \theta_c \cdot \phi_c(\mathbf{Y}_c, \mathbf{x}_c). \tag{12}$$

that, when written for a fixed observation $\mathbf{x}$, is known as the energy of the labeling and denoted as $E(\mathbf{Y})$. This energy is a map from the space of labelings to the real numbers. For improbable labelings the energy gives large values, whereas for good, probable labelings it provides lower values.

Finding the best segmentation with the probability distribution of the CRF, that models some degree of dependency among neighboring voxels, requires maximizing the probability $P(\mathbf{Y} \mid \mathbf{x}; \theta)$ for a fixed observation $\mathbf{x}$. This is equivalent to minimizing the energy $E(\mathbf{Y})$.

Figure 4 depicts the factor graph of our CRF. The factor graph is a bipartite graph that shows the random variables of the CRF graph as circles, as well as the potentials as little black squares. A potential depends on the random variables that it is connected to. Therefore, the given factor graph represents the following factorization of our distribution:

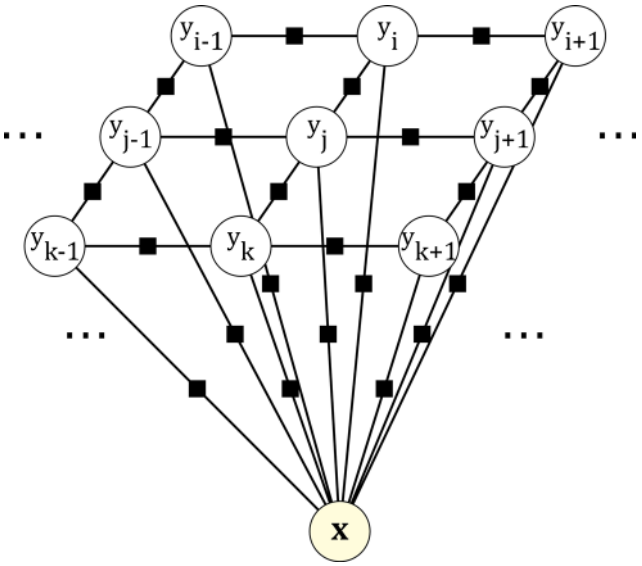

Fig. 4 Factor graph for the CRF. The random variables are inside circle nodes. Black squares represent potentials depending on the random variables they are connected to. The energy of the CRF is the sum of all potentials. This figure shows only a fragment of a 2D CRF, but the generalization to 3D is straightforward

$$P(\mathbf{Y} \mid \mathbf{x}; \theta) = \frac{1}{Z(\mathbf{x}, \theta)} \exp\left(-\sum_{i \in V} \theta_i \cdot \phi_i(y_i, \mathbf{x}) - \sum_{(i,j) \in E} \theta_{ij} \cdot \phi_{ij}(y_i, y_j) \right) \tag{13}$$

There are two kinds of potentials in this graph. The first kind of potential is associated with the terms $\phi_i(y_i, \mathbf{x})$. We will call them unary terms, since they only depend on a single variable y_i for a fixed observation in the energy function. The second kind of potential is related to the terms $\phi_{ij}(y_i, y_j)$. In an analogous way, we will call them pair-wise terms, since they depend on pairs of label variables.

Training a CRF consists of determining its parameters θ. This tends to be a complex task, especially if the CRF has many parameters, as in our case. We therefore need to simplify it further. A very common and reasonable assumption is that the CRF is translation and orientation invariant, and as a consequence all of the parameters for a kind of term (unary or pairwise) share the same value. This would lead to the energy function:

$$\theta_1 \sum_{i \in V} \phi_i(y_i, \mathbf{x}) + \theta_2 \sum_{(i,j) \in E} \phi_{ij}(y_i, y_j). \tag{14}$$

Unfortunately, in non-isotropic stacks we cannot assume orientation invariance. Usually, the stack has a lower resolution in the Z axis than in the X and Y axes. Therefore, we must treat the pair-wise terms that are oriented in the Z axis in a different way. We divide the set E into two disjoint subsets, E_{XY} and E_Z, for the edges oriented along the X and the Y axes and for the edges oriented along the Z axis, respectively. The energy is now:

$$\theta_1 \sum_{i \in V} \phi_i(y_i, \mathbf{x}) + \theta'_{XY} \sum_{(i,j) \in E_{XY}} \phi_{ij}(y_i, y_j) + \theta'_Z \sum_{(i,j) \in E_Z} \phi_{ij}(y_i, y_j). \tag{15}$$

Finally, since we are interested in the minimum energy, we can multiply the energy by $\frac{1}{\theta_1}$ and the solution remains unchanged:

$$\sum_{i \in V} \phi_i(y_i, \mathbf{x}) + \theta_{XY} \sum_{(i,j) \in E_{XY}} \phi_{ij}(y_i, y_j) + \theta_Z \sum_{(i,j) \in E_Z} \phi_{ij}(y_i, y_j). \tag{16}$$

This energy has only two parameters, θ_{XY} and θ_Z, which control the strength of the regularization in the XY plane and in the Z axis. In an anisotropic stack we can assume that θ_{XY} and θ_Z are related by the expression $\theta_Z = \frac{\theta_{XY}}{\rho}$, and

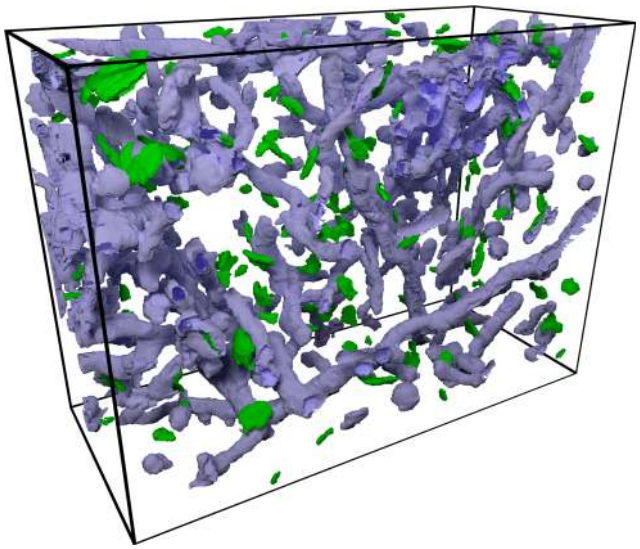

Fig. 5 Segmentation of a whole stack of serial images after the regularization step. Mitochondria are shown in purple and synaptic junctions are shown in green

only one of them needs to be estimated by cross-validation or manually by the user. The manual estimation is further facilitated by the fact that the CRF offers good results for a large range of parameter values.

The unary and pair-wise terms have to be defined in such a way that they provide lower values for good, probable inputs. The unary terms $\phi_i(y_i, \mathbf{x})$ are responsible for introducing the observed data into the energy value. It is customary to define the unary terms as the minus logarithm of the probability that our trained classifier provides: $\phi_i(y_i, \mathbf{x}) = -\log P(y_i \mid f_i(\mathbf{x}))$. This definition is justified since, in the absence of the pair-wise terms, the CRF would lead to the segmentation given by the classifier acting on each voxel separately.

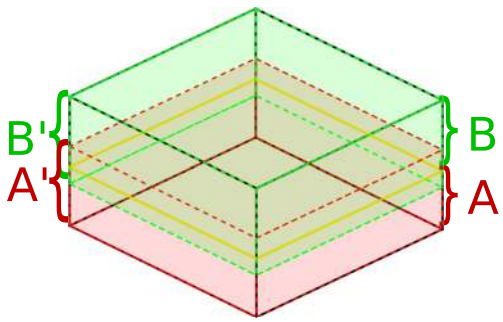

Fig. 6 Regularization in overlapping partitions. Example of partition of a full stack into two disjoint substacks **A** and **B** and two overlapping substacks **A**' and **B**'. Once the regularization has been performed in **A**', only voxels belonging to **A** are updated. Then, regularization is performed in **B**' and only voxels belonging to **B** are updated. This procedure prevents the appearance of regularization artifacts in the boundary between **A** and **B**

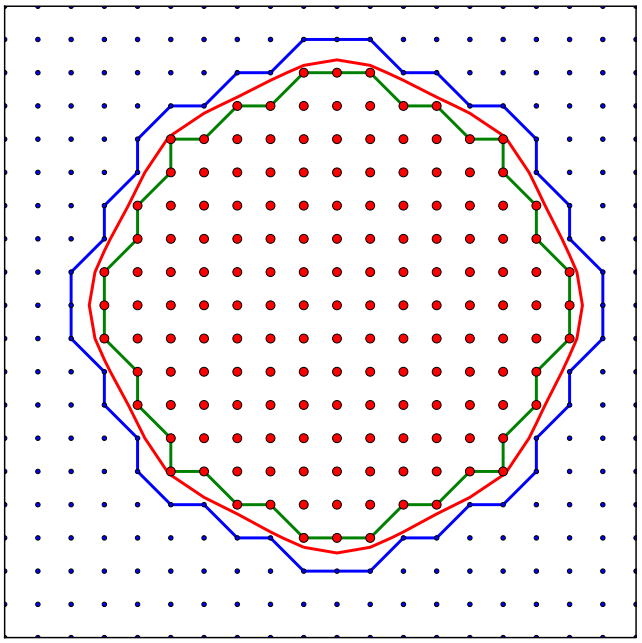

Fig. 7 Example of a smoothed surface that meets the constraints imposed by the segmentation. The red and blue dots indicate the pixels that are inside and outside the object, respectively. The green and blue contours are the maximum and minimum area contours, respectively. Although there are infinite contours that would lie between them, we look for the smoothed minimum curvature contour depicted in red

The role of the pair-wise terms $\phi_{ij}(y_i, y_j)$ is twofold. First, they regularize the segmentation results by penalizing

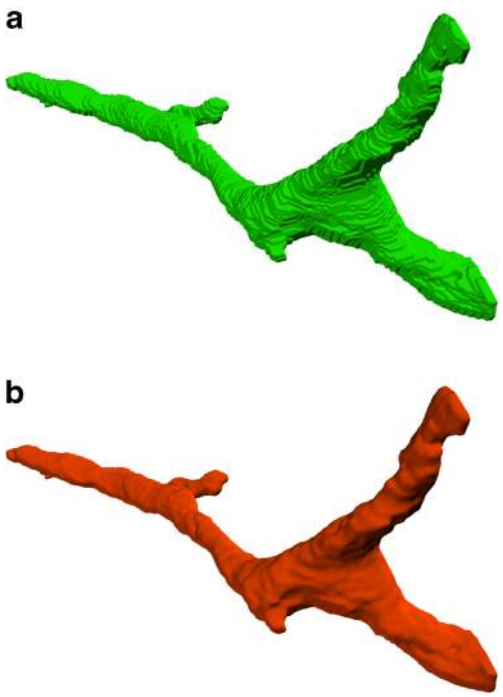

Fig. 8 Example of a branching mitochondrion. A large branching mitochondrion is shown before (**a**) and after smoothing (**b**)

the change of labels between neighboring voxels. This prevents the occurrence of isolated pixels and small holes that could appear (Fig. 3c, d). Second, they serve to introduce some extent of high-order knowledge about the structure of the stack. For example, we could impose the condition that synaptic junctions and mitochondria cannot touch each other, by setting a very large penalty to that label change in our experiments.

Therefore, the pair-wise terms are assigned as follows. A low penalty 1 is given to pairs of labels that are allowed to be neighbors. Second, a very high penalty ∞ is assigned to pairs of labels that cannot be adjacent (e.g., synaptic junctions and mitochondria). Third, no penalty is given for pairs of neighboring pixels with the same label. The distance matrix between labels *background* (bg), *synaptic junction* (syn) and *mitochondria* (mit) is therefore:

$$\begin{array}{c} \\ bg \\ syn \\ mit \end{array} \begin{array}{c} \begin{array}{ccc} bg & syn & mit \end{array} \\ \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & \infty \\ 1 & \infty & 0 \end{pmatrix} \end{array}.$$

Once we have defined our energy, we need to find the segmentation that minimizes the energy function, $\mathbf{Y}^* = argmin_{\mathbf{Y}} E(\mathbf{Y})$. This is in general an NP-hard optimization problem. However, it is known that when the terms are up to order two, i.e., there are only pair-wise (order 2) and unary (order 1) terms, the number of labels is two

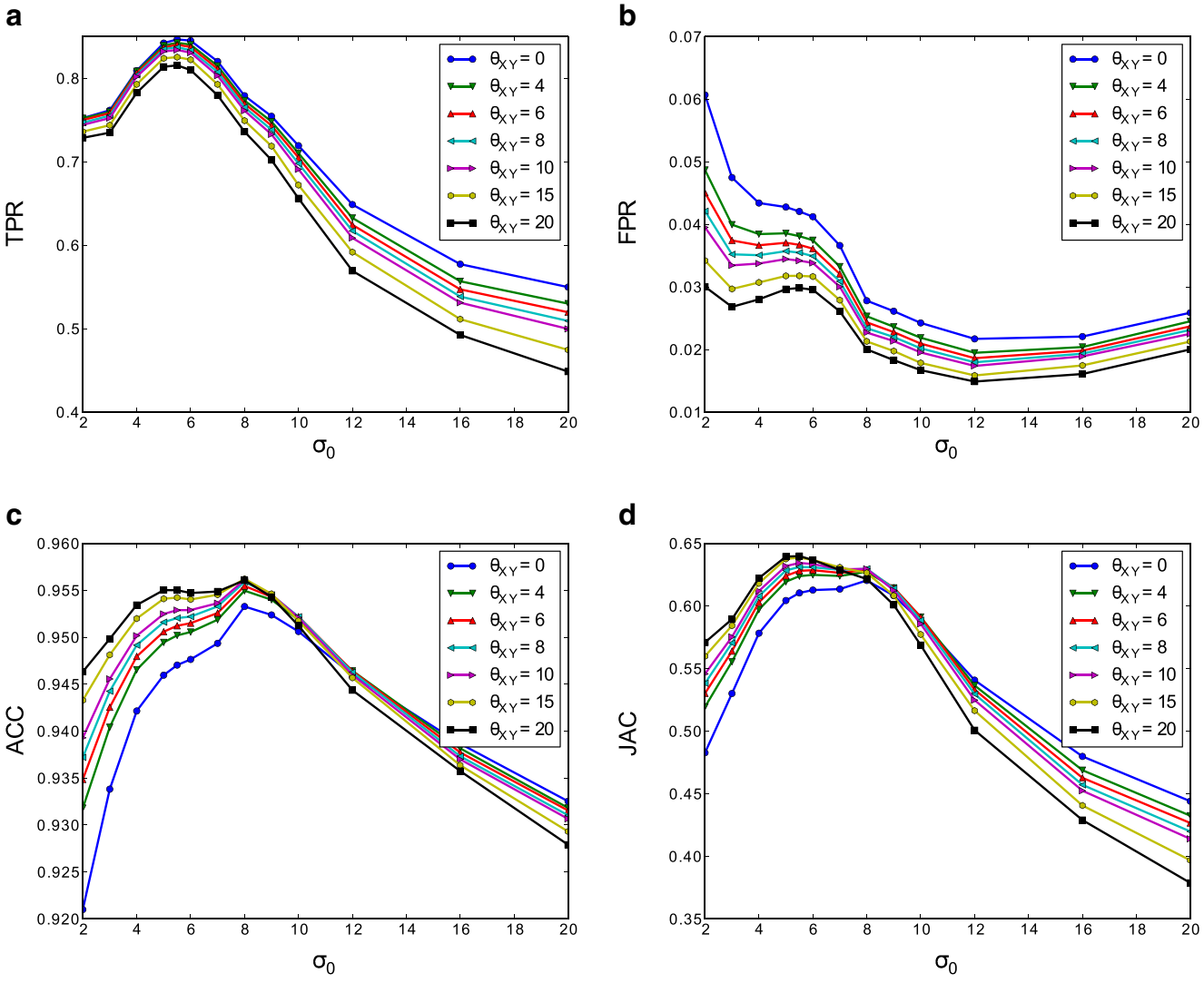

Fig. 9 Metrics obtained by cross-validation for several values of the parameters σ_0 and θ_{XY}. (**a**) TPR vs. σ_0 for different values of θ_{XY}. (**b**) FPR, (**c**) ACC and (**d**) JAC

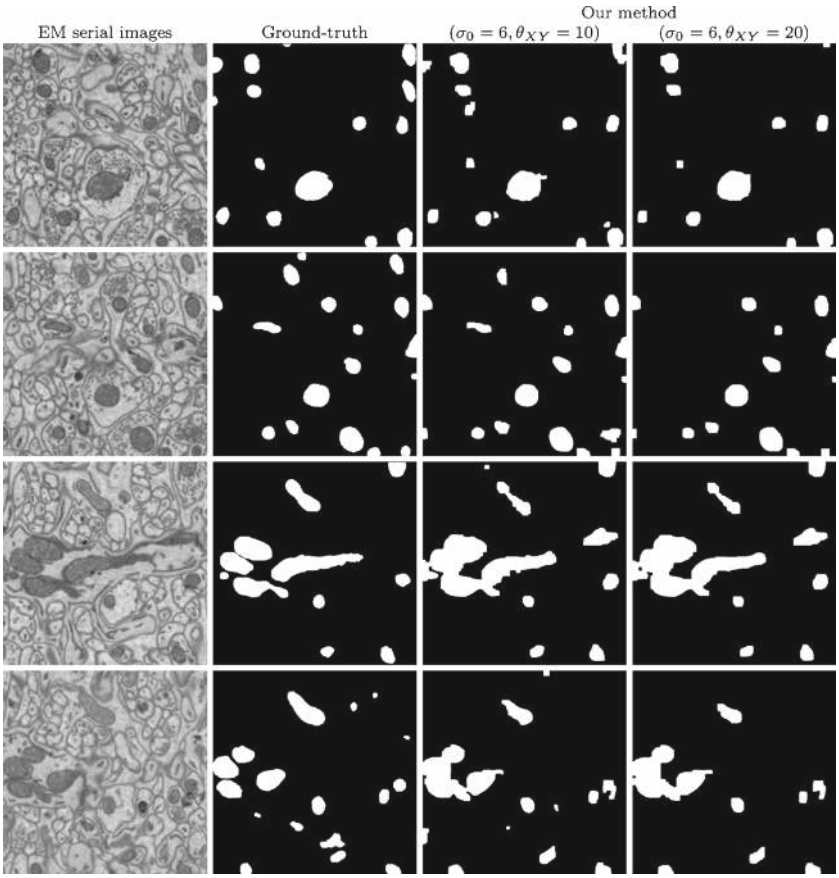

Fig. 10 Segmentation of mitochondria in a $700 \times 700 \times 50$ stack of EM serial images. The left column shows four individual, non-consecutive images from the stack. The second column shows ground truth data, manually segmented by an expert. The two rightmost columns show the results obtained with our algorithm using a base scale $\sigma_0 = 6$ and two different sets of regularization parameters θ_{XY} and θ_Z.

and the pair-wise terms are submodular, i.e., $\phi(y_i, y_i) + \phi(y_j, y_j) \leq \phi(y_i, y_j) + \phi(y_j, y_i)$, then a max-flow/min-cut algorithm finds the global optimum of the energy in polynomial time.

Unfortunately, when there are more than two labels, the max-flow algorithm is no longer applicable. Instead, we have to rely on approximate energy minimization using the $\alpha\beta$-swap algorithm from (Boykov et al. 2001). Figure 3c, d shows the segmentation of a single image after the $\alpha\beta$-swap regularization. Figure 5 shows the segmentation of a whole stack of serial EM images.

The graph-cut techniques needed for regularization require a considerable amount of computer memory. For a reasonably sized stack, the required memory usage usually becomes too big. Therefore we need to regularize parts of the full stack separately and merge them together at the end.

A simple approach is to divide the stack into disjoint, i.e., non-overlapping substacks and regularize them separately. This method works well for the inner areas of each substack, but it looks jumpy in their boundaries, since the CRF does not have enough context to determine the correct labels. This is visually noticeable as abrupt terminations of mitochondria and synaptic junctions at these boundaries.

The solution to this problem consists of extending each substack with a margin, effectively making the new

Table 1 Comparative results for mitochondria detection performed by our algorithm with two different sets of regularization parameters, by Ilastik (Sommer et al. 2011), and by the Cytoseg process, according to (Giuly et al. 2012)

Method	TPR	FPR	ACC	JAC	VOE
Ours, $\sigma_0 = 6; \theta_{XY} = 10; \theta_Z = 2$	0.78	0.018	0.96	0.68	8.26 %
Ours, $\sigma_0 = 6; \theta_{XY} = 20; \theta_Z = 4$	0.81	0.024	0.96	0.67	2.51 %
Cytoseg	0.80	0.02	0.97	N/A	N/A
Ilastik	0.77	0.02	0.96	0.66	5.45 %

extended substacks overlap with their neighbors. The regularization is then applied to the extended substacks, but only the results obtained in the original substack volume are preserved in the final segmentation (Fig. 6).

Determining the optimal size of the margin is a problem beyond the scope of this paper. However, we have found that a margin of 10 voxels in each direction offers very good results in practice.

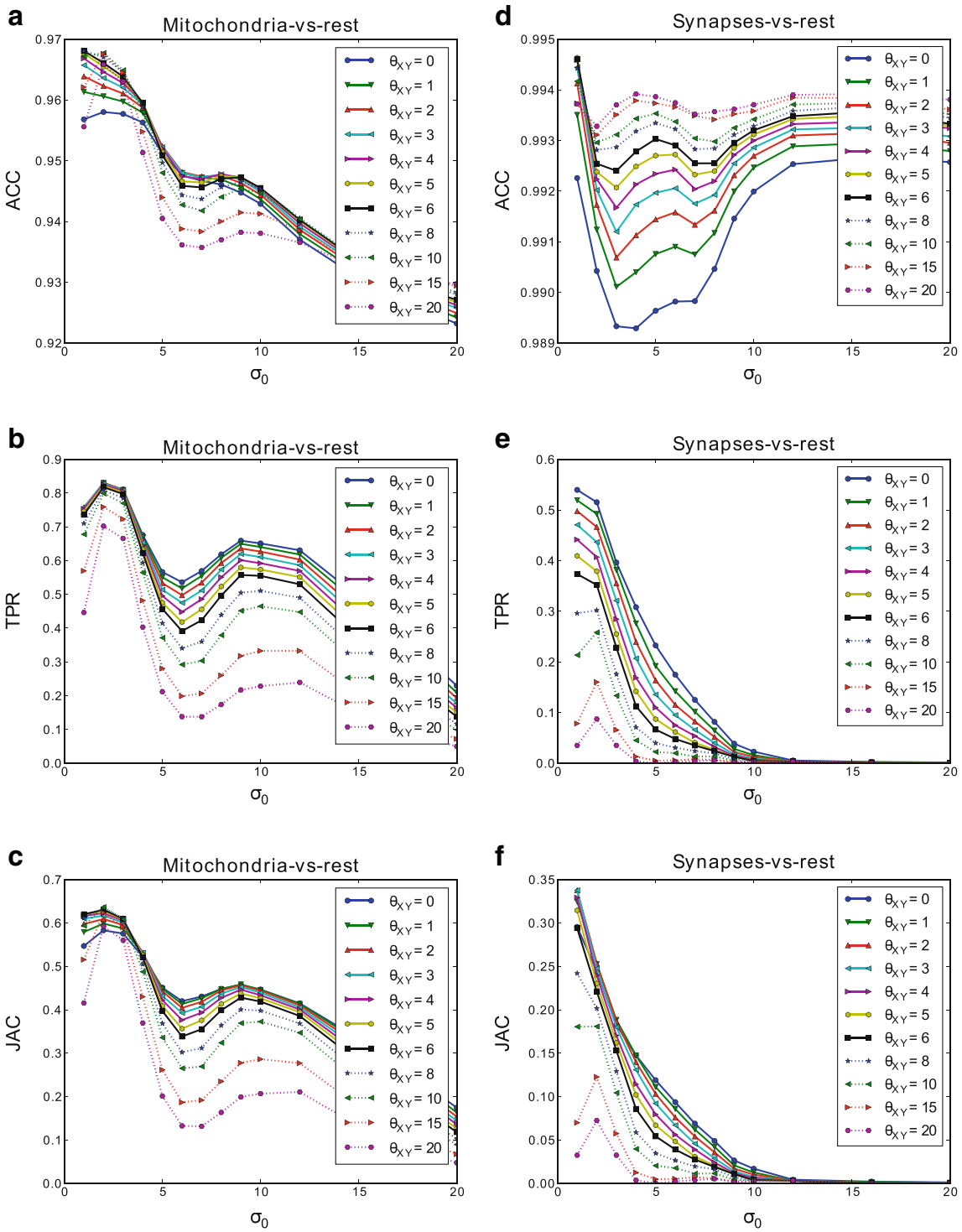

Fig. 11 Simultaneous segmentation of mitochondria and synaptic junctions. Metrics obtained by cross-validation for several values of the parameters σ_0 and θ_{XY}. ACC, TPR, and JAC are shown for mitochondria (**a**–**c**) and synaptic junctions (**d**–**f**)

Finally, the size of the substacks is limited by the available memory. As a rule of thumb, the regularization process takes 5 or 6 times the memory used by the original substack being regularized.

Segmentation Smoothing

The segmentation we obtain consists of hard label assignments to each voxel of the stack. This is suitable for several tasks such as counting of labeled objects or the estimation of the volume of the segmented objects, but presents disadvantages concerning the visualization of their surfaces. We use the marching cubes algorithm to extract the surfaces of the objects from the labels in the segmentation volume. This process not only produces unpleasant and unnatural renderings, but also biases the area estimates. Therefore we need to smooth the surfaces, but at the same time we want to preserve the constraints imposed by the labels in the segmentation volume. Among the infinite surfaces that meet these constraints, we compute the surface that minimizes curvature (see Fig. 7). To this end, we have adapted the method described by (Lempitsky 2010) to anisotropic stacks.

First, we build a vector of constraints $\{v_i\}_{i=1}^N \in \{-1, 1\}^N$ such as $v_i = 1$ if the voxel i is inside an object and $v_i = -1$ if it is outside an object, i.e., if it has the background label. We find a real-valued vector $\{f_i\}_{i=1}^N \in \mathbb{R}^N$ such that $f_i > 0$ if $v_i = 1$ and $f_i < 0$ if $v_i = -1$ and its zero-levelset is smooth. In a continuous setting we would minimize the functional

$$f^* = \arg\min_f \int_\Omega \left(\frac{\partial^2 f}{\partial x^2}\right)^2 + \left(\frac{\partial^2 f}{\partial y^2}\right)^2 + \left(\frac{\partial^2 f}{\partial z^2}\right)^2 \mathrm{d}V. \quad (17)$$

The minimization of the above functional smooths the segmentation result. In an anisotropic discrete setting, the smoothing problem becomes

$$\begin{aligned} f^* = \min_f \sum_i & \left(f_{N_x(i)} + f_{N_{-x}(i)} - 2f_i\right)^2 \\ & + \left(f_{N_y(i)} + f_{N_{-y}(i)} - 2f_i\right)^2 \\ & + \frac{\left(f_{N_z(i)} + f_{N_{-z}(i)} - 2f_i\right)^2}{\rho^4}, \end{aligned} \quad (18)$$

subject to

$$v_i \cdot f_i \geq m_i \quad \forall i, \quad (19)$$

where $N_{[-]d}(i)$ is the neighbor of i in the direction $(-)d$, and $\{m_i\}_{i=1}^N$ is a vector of margins imposing the deviations from zero of f at each point. m_i is 0 if i is at the boundary of an object (i.e., if v_i and v_j for any of the neighbors j of i have different values) and 1 everywhere else. Here we have included factor ρ in the second derivative along the Z-axis to account for the anisotropy of the stack.

The Eqs. (18) and (19) constitute a convex quadratic program that can be solved with the Jacobi method with a slight modification to include the constraints in the algorithm (Lempitsky 2010). Figure 8 shows a mitochondrion before and after smoothing with this method.

Note that this smoothing only affects the estimated surfaces and, therefore, the rendering of these surfaces. The segmentation volume and the numerical results extracted from it are not affected by this procedure. Therefore, the quantitative comparisons offered in the Section "Results" are computed with no smoothing.

Results

As explained in the previous section, we conceived our algorithm to be used interactively. However, to evaluate its performance we have used two datasets that have been fully segmented manually. We need these manual segmentations as ground-truth data to validate our results and compare them to others. Moreover, given that we have enough training data, we use them to find optimum values for the base scale σ_0 and the regularization term σ_{XY}.

We have used several voxel-based metrics to evaluate the quality of the segmentations. Voxel-based metrics measure the error rates in voxel classification taking into account true positive (TP), true negative (TN), false positive (FP) and false negative (FN) classifications.

- True positive rate (TPR):

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \quad (20)$$

- False positive rate (FPR):

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \quad (21)$$

- Accuracy (ACC):

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \quad (22)$$

- Jaccard index (JAC):

$$\mathrm{JAC} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \quad (23)$$

- Volume error (VOE):

$$\mathrm{VOE} = \frac{|\mathrm{FP} - \mathrm{FN}|}{\mathrm{TP} + \mathrm{FN}} \quad (24)$$

Unless otherwise stated, all running times were obtained on a Linux system with an Intel Xeon at 2.40GHz with no GPU processing. Our algorithm is mostly implemented in Python using the NumPy library. The inner parts of the CRF regularization are written in C++. Our implementation runs in a single thread.

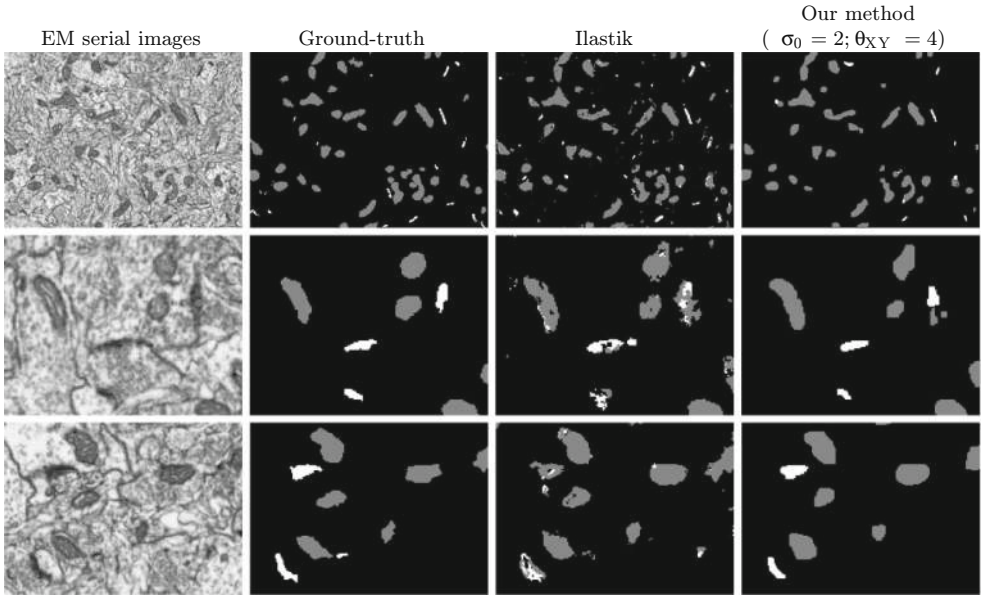

Fig. 12 Simultaneous segmentation of mitochondria and synaptic junctions. The left column shows the original images; the center left column is the ground-truth; the center right column is the result obtained with Ilastik; the right column is the result of our algorithm with parameters $\sigma_0 = 2$; $\theta_{XY} = 4$; $\theta_Z = 2.94$. The first row shows a full slice. The second and third rows show zoomed regions of different slices for detail

Mitochondria Segmentation

We used a stack of serial EM images from the mouse cerebellum to test our method for the segmentation of mitochondria. This stack is available online in the Cell Centered Database[1] with ID 8192. We have selected this stack to make our method comparable with Cytoseg, an automatic segmentation tool proposed by (Giuly et al. 2012). These researchers provide the raw images as well as a manual segmentation of mitochondria at the Cytoseg web page.[2] The stack has a size of $700 \times 700 \times 50$ voxels and a voxel size of $10 \times 10 \times 50$ nm (anisotropy factor $\rho = 5$). We have applied our method to automatically detect the mitochondria in this stack.

The stack was divided into 5 sets of 10 images and we used 5-fold cross-validation to estimate the quality of the segmentation for different pairs of values of σ_0 and θ_{XY}. Values of σ_0 ranged from 2 to 20 and values of θ_{XY} ranged from 0 to 20. Figure 9 plots the TPR, FPR, ACC and JAC metrics obtained. The curves show that the base scale for feature extraction σ_0 is much more critical for the quality of the segmentations than the regularization penalty θ_{XY}. In fact, the regularization penalty is almost irrelevant for the quantitative measurements, but it is much more important in the visual or qualitative results (see Fig. 10).

From the results of the cross-validation process, we choose $\sigma_0 = 6$ as it offers a good trade-off between the considered metrics. For the regularization parameter θ_{XY}, we select two different values: 10 and 20. The parameter $\theta_Z = \frac{\theta_{XY}}{\rho}$ is set to 2 and 4, respectively.

After choosing the parameters, we apply our segmentation algorithm to the full stack. We used the last 10 slices of the stack for training. The results obtained with our method are similar to those obtained with the Cytoseg process (see Table 1). However, Cytoseg (Giuly et al. 2012) required 80 minutes of processing time for a stack of $350 \times 350 \times 30$ according to their paper, while our algorithm took 53.8 seconds for the segmentation of a $700 \times 700 \times 50$ stack (training takes 9.6 seconds and the complete stack labeling, including CRF regularization, 44.2 seconds).

We also applied the software Ilastik ((Sommer et al. 2011), www.ilastik.org) to segment mitochondria in this dataset. The quantitative results obtained in Ilastik (see Table 1) are comparable to those of the other methods. However, Ilastik took 56.5 minutes for processing the full stack using 8 threads, resulting in a total of 452 minutes of CPU. This is about 500 times slower than our method.

Other methods for mitochondria segmentation are even less suitable for large anisotropies. Supervoxel segmentation with learned shape features (Lucchi et al. 2012) aims to learn non-local shapes of the target objects to segment. They use a combination of supervoxels, 3D ray features and structured prediction. 3D ray features are specially affected by anisotropy since both the edge detector and the length of the rays are highly dependent on the orientation. The achievable segmentation accuracy —i.e., the highest accuracy that can be achieved using supervoxels assuming perfect classification of each supervoxel— drops significantly with anisotropy. Moreover, the structured prediction requires training with a large portion of the stack fully labeled in order to infer the terms of the pairwise

[1] http://ccdb.ucsd.edu

[2] https://code.google.com/p/cytoseg/

Table 2 Quantitative results for the simultaneous segmentation of mitochondria and synaptic junctions

Method	TPR	FPR	ACC	JAC	VOE
Ours, mit-vs-rest,$\sigma_0 = 1, \theta_{XZ} = 4, \theta_Z = 2.94$	0.84	0.02	0.97	0.60	15.72 %
Ours, syn-vs-rest,$\sigma_0 = 1, \theta_{XZ} = 4, \theta_Z = 2.94$	0.60	0.004	0.99	0.26	87.17 %
Ours, mit-vs-rest,$\sigma_0 = 2, \theta_{XZ} = 4, \theta_Z = 2.94$	0.78	0.014	0.97	0.63	0.01 %
Ours, syn-vs-rest,$\sigma_0 = 2, \theta_{XZ} = 4, \theta_Z = 2.94$	0.42	0.004	0.99	0.23	24.00 %
Ilastik, mit-vs-rest	0.68	0.009	0.97	0.61	21.58 %
Ilastik, syn-vs-rest	0.36	0.002	0.99	0.29	41.43 %

interaction. As a consequence of these factors, the method from (Lucchi et al. 2012) required more training data (half of the stack) to work properly, and provided rather unsatisfactory results with low Jaccard indices (< 0.48). The running times were also higher (> 21 minutes) due mainly to the cost of extraction of the ray features.

Mitochondria and Synaptic Junctions Segmentation

For this test we used a stack of $366 \times 494 \times 213$ voxels and a voxel size of $14.7 \times 14.7 \times 20$ nm ($\rho = 1.36$) acquired from the rat somatosensory cortex. We used the first 100 slices of the stack to estimate the parameters of the algorithm with 5-fold cross-validation. The results of the cross-validation are plotted in Fig. 11. Again, the base scale σ_0 had a critical impact on performance, whereas the regularization penalty θ_{XY} only caused subtle variations. Note that we were segmenting three different classes (background, mitochondria and synapses) and therefore we measured the quality of segmentations with mitochondria-vs-rest and synapses-vs-rest metrics, i.e., considering one of the classes as foreground and the rest as background.

From the results of cross-validation, we chose $\sigma_0 = 1$ and $\sigma_0 = 2$ as a trade-off value that worked reasonably well for both mitochondria and synaptic junctions. We set $\theta_{XY} = 4$ and $\theta_Z = \frac{\theta_{XY}}{\rho} = 2.94$. The training was performed using 11 evenly distributed slices of the stack and it took 3.27 seconds to complete. The segmentation of the full stack (213 serial images) took 10.15 minutes (48.31 seconds for the classification step and the rest for regularization). Figure 12 shows the results of our algorithm and Ilastik. Table 2 compares the quantitative performance of both algorithms. The Ilastik performance results were obtained using a very similar manual segmentation to the one used with our algorithm. The results obtained with both methods were similar when considering the numerical performance, with ours being marginally better. However, Ilastik took 56 minutes with 8 threads to train and segment the full stack, making our method 45 times faster. Visual appearance of the final segments were also much better in our case thanks to the regularization procedure (see Fig. 12).

Running Time Comparison

Table 3 summarizes running times for our experiments in both datasets. Our method runs much faster compared to the others with similar or better performance. Cytoseg and learned shape features are specialized in mitochondria segmentation; thus, we only report results in the first dataset for those methods.

There is an important difference in running times for our method in both datasets (2.15 vs. 15.9). This large difference is due to the regularization with > 2 labels, where a single graph-cut is inviable and iterative, slower algorithms such as $\alpha\beta$-swap are required.

Counting Structures

Estimating the number of structures from the results of an automatic segmentation process is still an open problem with plenty of ongoing research. As an approximate, simple solution, it is commonly assumed that each connected component of the segmentation is one structure. This is the approach we use. Despite its simplicity, it has

Table 3 Absolute and normalized running times for different methods. Absolute times are given in seconds of CPU (s·CPU), and normalized times (in parentheses) are given in seconds of CPU per megavoxel $\left(\frac{\text{s·CPU}}{\text{Megavoxel}}\right)$

Method	CCDB-8192	Mit&Syn
Cytoseg (Giuly et al. 2012)	4800 (192.08)	N/A
Ilastik (Sommer et al. 2011)	14216 (568.87)	27112 (704)
Learned shape features (Lucchi et al. 2012)	1380 (55.22)	N/A
Ours	53.8 (2.15)	612.27 (15.9)

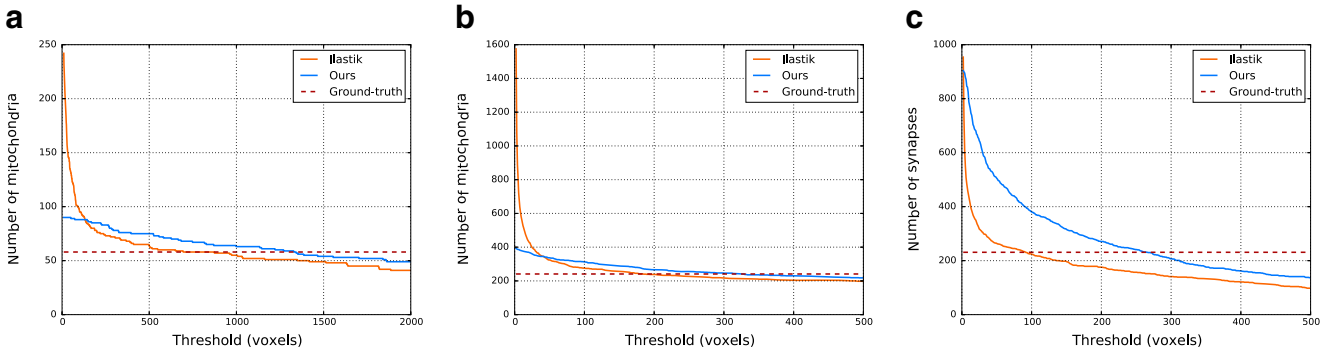

Fig. 13 Count estimations for varying thresholds. (**a**) shows estimations in the number of mitochondria for the CCDB-8192 dataset. (**b**) and (**c**) show estimations in the number of mitochondria and synaptic junctions respectively in the Mit&Syn dataset

several drawbacks, namely, a group of structures close to each other often merge in a single connected component, and large structures are sometimes split into two or more connected components. Also, when spatial regularization is not present, false positive detections result in many small connected components that bias the counting estimations. To alleviate these problems, we discard the connected components smaller than a given threshold. Setting the threshold is not trivial, as it might greatly affect the counts depending on the quality of the segmentation. A good segmentation is expected to be more robust to different thresholds than a bad one, i.e., estimations should be close to the real value and should be stable for a large range of thresholds.

Figure 13 shows count estimations for different thresholds with both Ilastik and our method. The regularization makes our method more robust: it reduces the number of small components and the estimations of our method are closer to the ground-truth for a wider range of thresholds. Table 4 gives numerical assessment of this idea. It shows the absolute value of the deviations of the estimations from the ground-truth averaged over all thresholds in the range $T = [10, 2000] \subset \mathbb{Z}$:

$$\frac{1}{|T|} \sum_{t \in T} \left| \#\mathrm{CC}_{[\mathrm{size} \geq t]} - \mathrm{GT} \right|, \tag{25}$$

where $\#\mathrm{CC}_{[\mathrm{size} \geq t]}$ is the number of connected components with size $\geq t$, and GT is the real count. Table 4 shows that our method has smaller errors than Ilastik for all datasets and considered structures.

Discussion

Concerning the segmentation of mitochondria, Lucchi and colleagues (Lucchi et al. 2012) have recently used ray descriptors and the gray-level histogram as the key features to classify 3D image supervoxels. The result of this classification is further regularized using graph cuts to minimize an energy function involving learned potentials. They used stacks of FIB/SEM images from the hippocampus and striatum that had isotropic resolution. In their method, isotropy is an essential requirement for the computation of the 3D supervoxel over-segmentation. Alternatively, Giuly and colleagues (Giuly et al. 2012) segment mitochondria in anisotropic stacks of images obtained by SBFSEM. They use a random forest classifier to label 2D image patches. The result of this initial segmentation is further refined using 2D contour classification across images and 3D level-set surface evolution. Their method, however, is computationally intensive, requiring long processing times

Regarding synapses, the popular Ilastik toolbox (Sommer et al. 2011) used by (Kreshuk et al. 2011) to segment synaptic junctions uses a random forest classifier with a set of differential image features. They use a simple regularization strategy based on Gaussian smoothing. Overall, the resulting algorithm is also very demanding in terms of computing power.

Our method does not require isotropic voxels so it can be applied to image stacks that have been acquired with different resolution in the X, Y and Z axes. The results obtained with our method were similar or better than those

Table 4 Average absolute error of estimations of the number of mitochondria and synaptic junctions over all thresholds in the range [10, 2000] voxels

Method	CCDB-8192	Mit&Syn, Mit	Mit&Syn, Syn
Ilastik (Sommer et al. 2011)	12.12	68.44	166.17
Ours	**10.71**	**54.97**	**161.76**

obtained with the Cytoseg process (Giuly et al. 2012) for mitochondria only, and to those obtained with Ilastik for both mitochondria only and simultaneous mitochondria and synaptic junctions. Other approaches such as the one from (Lucchi et al. 2012) are not ready to work with anisotropic stacks and therefore our method outperforms them. Unlike Cytoseg, that focuses on mitochondria segmentation, our method is not tied to a specific type of cellular structure but can be used to segment a variety of structures. When compared to Ilastik we obtained better visual results thanks to the regularization and surface smoothing techniques described above.

Moreover, our method is much faster than any other approach we have tried. The speed up comes from the Gaussian classifier, that can be trained in $O(Nk^2 + k^3)$, being N the number of data points and k the dimension of the feature space. For comparison, the complexity of training random forests is $O(MNkd)$, being M the number of trees and d the average depth of the trees. We found in our experiments that the classifier was the main bottleneck of the Ilastik approach. In our approach the most expensive computation was the regularization step, which Ilastik omits. On the other hand, we found no significant difference in speed for feature extraction, taking only a small fraction of the total processing time in all compared methods.

For the case of segmentation of 2 labels, a speed of 2.15 seconds per megavoxel in a single thread is fast enough to enable interactive segmentation of the large image stacks that are now available, providing real-time feedback to the user. Of course, parallelization of the proposed approach is straightforward, and it would make it even faster. To our knowledge, no other previous work provides state-of-the-art performance while running in an interactive setting.

Conclusions

We have presented an algorithm that can be trained to segment a variety of structures in anisotropic EM stacks. In this work we have focused on its capabilities for the segmentation of synaptic junctions and mitochondria. It features some important properties that are not available in other methods in the literature. It uses a graph cut-based image regularization procedure that not only provides better segmentations, but also introduces high level knowledge about the structure of labels. We have solved the limitation of graph cuts in terms of memory requirements with the introduction of energy optimization in overlapping partitions. This allows the regularization of very large stacks. The surface smoothing step introduces smoothness priors on the segmentation that improves the appearance of three-dimensional renderings of the segmented volumes. Finally, and most importantly, we have also shown that our approach is much faster than any other competing method with a state-of-the-art quantitative segmentation performance.

Information Sharing Statement

The automatic segmentation method described in this paper is available as a plugin of the imaging processing software Espina. The software and instructions for installing it can be found at http://cajalbbp.cesvima.upm.es/espina.

This software provides an efficient multi-thread implementation of the presented algorithm together with an intuitive user interface. After activating the Automatic Segmentation plugin, the user has to segment a few voxels of the target objects manually and receives almost real-time feedback of the results. Additional manual segmentations can be performed until the user is satisfied with the final results. Quantitative data regarding the segmented objects are then obtained with standard Espina tools.

Acknowledgements This work was supported by funding from the Spanish Ministry of Economy and Competitiveness (grants TIN2013-47630-C2-2-R to L.B. and BFU2012-34963 to J.DF.), CIBERNED (CB06/05/0066 to J.DF.), the Cajal Blue Brain Project, Spanish partner of the Blue Brain Project initiative from EPFL (to J.DF. and L.B.) and the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 604102 (Human Brain Project) to J.DF.

References

Anton-Sanchez, L., Bielza, C., Merchán-Pérez, A., Rodríguez, J.-R., DeFelipe, J., & Larrañaga, P. (2014). Three-dimensional distribution of cortical synapses: A replicated point pattern-based analysis. *Frontiers in Neuroanaty*, *8*, 85. doi:10.3389/fnana.2014.00085.

Becker, C.J., Ali, K., Knott, G., & Fua, P. (2013). Learning Context Cues for Synapse Segmentation. *IEEE Transactions on Medical Imaging*, *32*(10), 1864–1877. doi:10.1109/Tmi.2013.2267747.

Boykov, Y., Veksler, O., & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *23*(11), 1222–1239.

DeFelipe, J. (2010). From the connectome to the synaptome: An epic love story. *Science*, *330*(6008), 1198–1201.

Denk, W., & Horstmann, H. (2004). Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. *PLoS biology*, *2*(11), e329. doi:10.1371/journal.pbio.0020329.

Giuly, R., Martone, M., & Ellisman, M. (2012). Method: Automatic segmentation of mitochondria utilizing patch classification, contour pair classification, and automatically seeded level sets. *BMC Bioinformatics*, *13*(1), 29+. doi:10.1186/1471-2105-13-29.

Haindl, M., & Mikes, S. (2008). Texture segmentation benchmark. In *ICPR, IEEE* (pp. 1–4). http://dblp.uni-trier.de/db/conf/icpr/icpr2008.html#HaindlM08a.

Jagadeesh, V., Anderson, J., Jones, B., Marc, R., Fisher, S., & Manjunath, B. (2013). Synapse classification and localization in electron micrographs. Pattern Recognition Letters. doi:10.1016/j.patrec.2013.06.001.

Jurrus, E., Hardy, M., Tasdizen, T., Fletcher, P.T., Koshevoy, P., Chien, C.B., Denk, W., & Whitaker, R. (2009). Axon tracking in serial block-face scanning electron microscopy. *Medical Image Analysis*, *13*(1), 180–188. doi:10.1016/j.media.2008.05.002. Includes Special Section on Medical Image Analysis on the 2006 Workshop Microscopic Image Analysis with Applications in Biology.

Knott, G., Marchman, H., Wall, D., & Lich, B. (2008). Serial section scanning electron microscopy of adult brain tissue using focused ion beam milling. *Journal of Neuroscience*, *28*(12), 2959–2964. doi:10.1523/JNEUROSCI.3189-07.2008.

Kreshuk, A., Straehle, C., Sommer, C., Koethe, U., Cantoni, M., Knott, G., & Hamprecht, F. (2011). Automated detection and segmentation of synaptic contacts in nearly isotropic serial electron microscopy images. PLoS ONE, *6*, e24899. doi:10.1371/journal.pone.0024899.

Lempitsky, V.S. (2010). Surface extraction from binary volumes with higher-order smoothness. In *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR), IEEE* (pp. 1197–1204). http://dblp.uni-trier.de/db/conf/cvpr/cvpr2010.html#Lempitsky10.

Lucchi, A., Smith, K., Achanta, R., Knott, G., & Fua, P. (2012). Supervoxel-based segmentation of mitochondria in em image stacks with learned shape features. *IEEE Transactions on Medical Imaging*, *31*(2), 474–486. http://dblp.uni-trier.de/db/journals/tmi/tmi31.html#LucchiSAKF12.

Martin, D., Fowlkes, C., Tal, D., & Malik, J. (2001). *A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics* (Vol. 2, pp. 416–423).

McBride, H.M., Neuspiel, M., & Wasiak, S. (2006). Mitochondria: More than just a powerhouse. *Current Biology*, *16*(14), R551–R560. doi:10.1016/j.cub.2006.06.054.

Merchan-Perez, A., Rodriguez, J.R., Alonso-Nanclares, L., Schertel, A., & DeFelipe, J. (2009). Counting synapses using FIB/SEM microscopy: a true revolution for ultrastructural volume reconstruction. *Frontiers in Neuroanatomy*, *3*(18). doi:10.3389/neuro.05.018.2009. http://www.frontiersin.org/neuroanatomy/10.3389/neuro.05.018.2009/abstract.

Morales, J., Alonso-Nanclares, L., Rodriguez, J.R., Defelipe, J., Rodriguez, A., & Merchan-Perez, A. (2011). Espina: a tool for the automated segmentation and counting of synapses in large stacks of electron microscopy images. *Frontiers in Neuroanatomy*, *5*(18).

Narasimha, R., Ouyang, H., Gray, A., McLaughlin, S.W., & Subramaniam, S. (2009). Automatic joint classification and segmentation of whole cell 3d images. *Pattern Recognition*, *42*(6), 1067–1079. doi:10.1016/j.patcog.2008.08.009.

Navlakha, S., Suhan, J., Barth, A.L., & Bar-Joseph, Z. (2013). A high-throughput framework to detect synapses in electron microscopy images. *Bioinformatics 29*, *13*, i9–i17. doi:10.1093/bioinformatics/btt222.

Santos, R.X., Correia, S.C., Wang, X., Perry, G., Smith, M.A., Moreira, P.I., & Zhu, X. (2010). Alzheimer's disease: diverse aspects of mitochondrial malfunctioning. *International Journal of Clinical and Experimental Pathology*, *3*, 570–581.

Sommer, C., Straehle, C., Kothe, U., & Hamprecht, F. (2011). Ilastik: Interactive learning and segmentation toolkit. In *IEEE International Symposium on Biomedical Imaging: From Nano to Macro* (pp. 230–233). doi:10.1109/ISBI.2011.5872394.

Turaga, S.C., Murray, J.F., Jain, V., Roth, F., Helmstaedter, M., Briggman, K., Denk, W., & Seung, H.S. (2010). Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Computation*, *22*, 511–538.