ELSEVIER

Journal of Neuroscience Methods 167 (2008) 349–357

JOURNAL OF NEUROSCIENCE METHODS

www.elsevier.com/locate/jneumeth

Contour-propagation algorithms for semi-automated reconstruction of neural processes

Jakob H. Macke [a,*,1], Nina Maack [b,*,1,2], Rocky Gupta [c], Winfried Denk [d,3], Bernhard Schölkopf [a,4], Alexander Borst [b,5]

[a] *Max Planck Institute for Biological Cybernetics, Spemannstraße 38, D-72076 Tübingen, Germany*
[b] *Max Planck Institute of Neurobiology, Department of Systems and Computational Neurobiology, Am Klopferspitz 18, D-82152 Martinsried, Germany*
[c] *Indian Institute of Technology Kanpur, Department of Electrical Engineering, Kanpur, India*
[d] *Max-Planck-Institute for Medical Research, Jahnstraße 29, D-69120 Heidelberg, Germany*

Received 6 March 2007; received in revised form 24 July 2007; accepted 26 July 2007

Abstract

A new technique, "serial block face scanning electron microscopy" (SBFSEM), allows for automatic sectioning and imaging of biological tissue with a scanning electron microscope. Image stacks generated with this technology have a resolution sufficient to distinguish different cellular compartments, including synaptic structures, which should make it possible to obtain detailed anatomical knowledge of complete neuronal circuits. Such an image stack contains several thousands of images and is recorded with a minimal voxel size of 10–20 nm in the *x*- and *y*-direction and 30 nm in *z*-direction. Consequently, a tissue block of 1 mm^3 (the approximate volume of the *Calliphora vicina* brain) will produce several hundred terabytes of data. Therefore, highly automated 3D reconstruction algorithms are needed. As a first step in this direction we have developed semi-automated segmentation algorithms for a precise contour tracing of cell membranes. These algorithms were embedded into an easy-to-operate user interface, which allows direct 3D observation of the extracted objects during the segmentation of image stacks. Compared to purely manual tracing, processing time is greatly accelerated.

Keywords: Circuit reconstruction software; Contour detection; Algorithm; Image segmentation; Serial block-face scanning electron microscopy; Neural circuits; Fly visual system

1. Introduction

Serial block-face scanning electron microscopy (SBFSEM) (Denk and Horstmann, 2002) allows for imaging of substantial volumes of biological tissue at high resolution. Using back-scattering contrast in low-vacuum conditions combined with serial sectioning of the specimen inside the vacuum chamber, slice thicknesses down to 30 nm and a resolution of 10–20 nm in the *x*–*y* plane can be achieved (Briggman and Denk, 2006). This resolution is sufficient to trace even thin neuronal processes and visualize sites of synaptic contact, which opens the possibility of reconstructing neuronal circuitry in detail.

We set out to develop algorithms to reconstruct parts of the fly visual ganglia. Although the fly visual system is well described at a resolution accessible with light microscopy (Strausfeld, 1984), knowledge at the ultrastructural level is necessary in order to get further insights into the circuits underlying visual motion processing (Borst and Haag, 2002). The methods described here were developed with this specific goal in mind, but should be applicable to other types of imaging data, too. While it is, in principle, possible to segment three-dimensional image blocks directly, we chose to segment each image in sequence, using the information obtained from previous images.

We first describe the preprocessing steps performed before the actual segmentation. We then introduce the segmentation

* Corresponding author. Tel.: +49 7071 601 1777.
E-mail addresses: jakob@tuebingen.mpg.de (J.H. Macke), maack@neuro.mpg.de (N. Maack), Rocky@iitk.ac.in (R. Gupta), denk@mpimf-heidelberg.mpg.de (W. Denk), bs@tuebingen.mpg.de (B. Schölkopf), borst@neuro.mpg.de (A. Borst).
[1] These authors contributed equally to this work.
[2] Tel.: +49 89 8578 3294; fax: +49 89 8578 3300.
[3] Tel.: +49 6221 486 335; fax: +49 6221 486 325.
[4] Tel.: +49 7071 601 551.
[5] Tel.: +49 89 8578 3251; fax: +49 89 8578 3252.

0165-0270/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.jneumeth.2007.07.021

algorithms. We emphasize the general probabilistic framework for this task and describe two algorithms in detail. We sketch some alternative approaches and possible extensions.

2. Preprocessing steps

First, intensity values within each image are normalized to have the same median and inter-quartile range. Next, each image is spatially filtered. For the filtering, our program gives the user the choice between Gaussian broadening and nonlinear diffusion (Perona and Malik, 1990). Nonlinear diffusion is noise-removing yet edge preserving filtering technique which, at each location, makes the degree of filtering dependent on an estimate of the intensity-gradient at that location. Thereby, strong edges are preserved, but regions that do not have strong edges are smoothed substantially. We used a publicly available matlab filtering toolbox (D'Almeida, 2002). Since the performance of filters can be sensitive to the choice of parameters, it is advisable to optimize parameters on, or even learn the filters from (Vollgraf et al., 2004) manually labeled images.

As a final preprocessing step, the cross-correlation between any two consecutive images is calculated. This allows the detection and elimination of corrupted images, which can result, for example, when debris has gotten onto the block face.

3. Segmentation algorithms

3.1. General approach

To allow segmentation of large stacks of images, as well as to enable user interaction, we segment images sequentially rather than the whole stack at once. We assume that the objects are continuous across adjacent images—an assumption that can be problematic for processes that run at a shallow angle to the slicing plane. To ensure continuity, we need to combine the information from the pixel-intensities of the current image with that from the segmentation of the previous image. Therefore, we use a *prior* that favors such segmentations or, in other words, the segmentation of one image is propagated into the next. For the very first image, however, we have to use a different strategy, which we will described in Section 3.4.

3.2. Level set methods

We chose algorithms that do not represent the boundaries of objects explicitly by using, for example, splines, but rather implicitly as the zero-level set of (usually) a signed distance function ϕ. We seek to segment the image $I : \Omega \to \mathbb{R}$, where $I(x)$ is the gray-scale value of pixel x, into foreground and background regions. In the level set framework (Osher and Fedkiw, 2003; Sethian, 1999), one tries to find a function $\phi : \Omega \to \mathbb{R}$, such that the set $\Omega_+ = \{x : \phi(x) > 0\}$ is the foreground, i.e. the set of pixels that are inside a neuron and $\Omega_- = \{x : \phi(x) < 0\}$ is the background. The contour separating objects and background is given by $\Gamma = \{x : \phi(x) = 0\}$. Note that images with multiple objects can be segmented with a single segmentation function, ϕ, by defining objects to be the connected components of the foreground regions.

As this embedding is not unique, ϕ is sometimes constrained to be a signed distance function (SDF), i.e. such that its absolute value gives the distance to the closest boundary. However, this does not have to be the case. For example, given a statistical model for the segmentation, one could interpret $\phi(x) + t$ (where t is a scalar offset) to be the log of the probability that pixel x belongs to the foreground. In this case, $\phi(x)$ indicates not only what region a pixel is assigned to, but also represents a measure of the confidence placed in the assignment.

Compared to an explicit representation, an implicit representation has the advantage that it can easily deal with changes in topology (such as splitting or merging of objects), and can readily be extended to higher dimensions.

We focus on region-based methods, which do not rely on detecting edges in the image, but exploit differences in the distribution of pixel intensities in fore- and background regions. In practice, an (artificial) time variable, t, is often introduced and ϕ_t is evolved to minimize some energy function, $E(\phi, I)$ (Cremers et al., 2006).

Algorithms that have been developed for the analysis of images obtained from confocal- or multi-photon microscopy are often based on the assumption that neuronal structures can be modeled as tube-like structures (Al-Kofahi et al., 2002; Koh et al., 2002; Schmitt et al., 2004) with approximately circular cross-sections. We did not make this assumption, as the processes observed in our data can have shapes very dissimilar to circles.

Level set methods are commonly used for segmentation of three-dimensional objects for bio-medical applications (Whitaker et al., 2001) as well as in other domains (Huang et al., 2005). Our algorithms differ from existing ones in the way that information is propagated from one image to the next, and in the exact form of the energy-functions used. Furthermore, many algorithms segment the three-dimensional data-blocks directly and need time-consuming computations, and are thus not well suited for online user interaction. Our approach is related to the one of (Jurrus et al., 2006), who used a propagation scheme based on kalman filters and explicit contours rather than level sets to represent objects.

3.3. Probabilistic framework

3.3.1. Statistical model

Let I_n be image n, and ϕ_n the corresponding segmentation that is to be found. Also, suppose that we have already segmented the previous image, such that ϕ_{n-1} is known. We want to find the *most probable* segmentation, ϕ_n, given I_n and ϕ_{n-1}, i.e. one such that $P(\phi_n|I_n, \phi_{n-1})$ is maximal (Cremers et al., 2006). By Bayes' Rule, we obtain

$$P(\phi_n|I_n, \phi_{n-1}) \propto P(I_n|\phi_n, \phi_{n-1})P(\phi_n|\phi_{n-1}).$$

$P(\phi_n|\phi_{n-1})$ can be interpreted as a *prior* on the possible segmentations ϕ_n, and can be used to ensure continuity of objects between adjacent images. This helps tracing of structures through multiple images. In addition, other priors can be

used to favor smooth contours, or to incorporate prior knowledge about the likely shapes of segmented objects. Ideally, these priors would be learned from manually labeled images, but for lack of an extensive training set, we had to chose the functional forms for the prior distributions.

For simplicity, we assume that image, I_n, is independent of the segmentation of the previous image, ϕ_{n-1}, given the actual segmentation, ϕ_n, i.e. $P(I_n|\phi_n) = P(I_n|\phi_n, \phi_{n-1})$. For a given (signed) distance, d, $P(I_n(x)|\phi_n(x) = d)$ gives the probability distribution for the intensities of all pixels that are on the outside (or inside, depending on the sign of $\phi(x)$), and have a distance d to the closest boundary. For convenience, we will from now on write ϕ instead of ϕ_n to denote the segmentation function of the current image, n. We use the term ϕ_0 instead of ϕ_{n-1} to denote the signed distance function derived from the segmentation of the previous image.

Maximizing the posterior probability, $P(\phi|I)$, is equivalent to minimizing its negative logarithm, which leads to the energy function

$$E(\phi|I) = -\log(P(I|\phi)) - \log(P(\phi|\phi_0)) = E_I + E_\pi.$$

The total energy is written as a sum of an image-dependent part E_I and a prior-dependent part E_π. In the following, we will specify two possible forms of the probability distributions, $P(I|\phi)$ and $P(\phi|\phi_0)$, in detail and also discuss possible alternatives, such as learning the distributions non-parametrically from data.

3.3.2. A simple algorithm

We assume a normal distribution for the pixel-intensities $I(x)$ with mean $\alpha\phi(x) - \beta$ and variance σ^2, i.e. on average the intensity of pixels is linearly related to their signed distance to the boundary. We also assume that the difference of the signed distance functions of two adjacent images is normally distributed. As mentioned above, this favors segmentations that are similar to the segmentation of the previous images, with the aim of improving the tracking of structures through multiple images.

The corresponding probability distributions are:

$$P(I|\phi) \propto \exp\left(-\int_\Omega \left(\frac{I - \alpha\phi - \beta}{2\sigma}\right)^2 \mathrm{d}x\right),$$

$$P(\phi|\phi_0) \propto \exp\left(\left\|\frac{\phi - \phi_0}{2\eta}\right\|^2\right)$$

$$= \exp\left(-\int_\Omega \left(\frac{\phi(x) - \phi_0(x)}{2\eta}\right)^2 \mathrm{d}x\right).$$

This leads to an energy function of the form:

$$E(\phi|I) = \int_\Omega \left(\frac{I - \alpha\phi - \beta}{\sigma}\right)^2 + \int_\Omega \left(\frac{\phi - \phi_0}{\eta}\right)^2,$$

which attains its minimum for

$$\phi = \left(\frac{\alpha^2}{\sigma} + \frac{1}{\eta}\right)^{-1} \left(\frac{\alpha}{\sigma} I + \frac{1}{\eta}\phi_0 - \frac{\alpha\beta}{\sigma}\right).$$

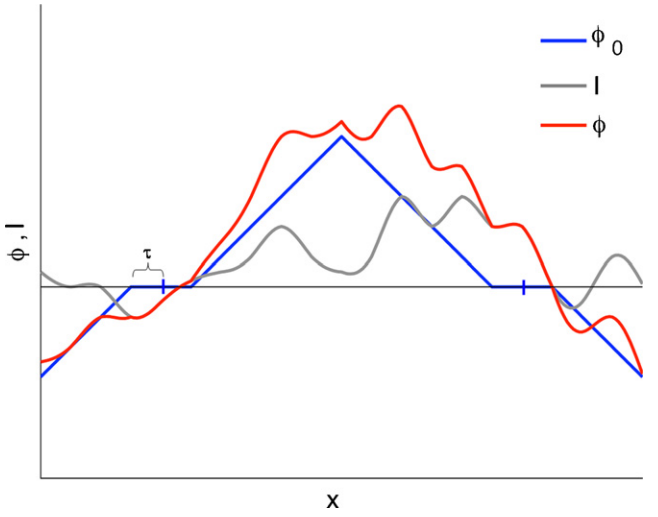

Fig. 1. Schematic illustration of the algorithm: the segmentation function ϕ of the image (red) is the weighted sum of the intensity value of that image (gray) and a term that depends on the segmentation of the previous image (blue).

Therefore, the function ϕ, which characterises the current segmentation, is merely a linear combination of the intensity values of the current image I, the previous segmentation function ϕ_0, and a scalar offset (Fig. 1).

Although this model might be too restrictive to provide a good fit to the actual image, using such a simple form has two advantages: Firstly, the minimum of the energy function, min $E(\phi|I)$, has a simple closed-form solution, which permits rapid calculation of the boundary. Secondly, the parameters α, β, σ and η can be interpreted and adjusted while the effect on the segmentation is visually judged. The parameter α re-scales the distances relative to intensities and β is an offset: If an intensity value of a pixel is larger than β, then (ignoring the prior), it is more likely that the pixel belongs to an object than to the background. The parameters σ and η control the relative weight of the image and the prior-dependent contribution. We also decided to penalize discrepancies between two adjacent segmentations only if they exceed a threshold, τ, as small variations between adjacent images are to be considered as normal. Thus, the term $(\phi(x) - \phi_0(x))^2$ was replaced by $|\phi(x) - \phi_0(x)|_\tau^2$, where

$$|\phi(x) - \phi_0(x)|_\tau^2 = \begin{cases} (|\phi(x) - \phi_0(x)| - \tau)^2 & \text{for } |\phi(x) - \phi_0(x)| \geq \tau, \\ 0 & \text{else.} \end{cases}$$

Then the form of the solution is as before, just with ϕ_0 replaced by $|\phi_0|_\tau \operatorname{sign}(\phi_0)$.

3.3.3. An alternative algorithm

The algorithm described above does not attempt to achieve segmentations with smooth boundaries. In addition, it makes a rather restrictive assumption about the intensity distribution of pixels in the image. We therefore implemented a second segmentation algorithm with a different energy function that also contains a term favoring smooth segmentations.

This algorithm assumes that the pixel-intensity distributions of the background and foreground, P_- and P_+, are different but might overlap. Furthermore, we assume that the probability that a given pixel has a certain intensity only depends on the region that the pixel belongs to, but not on its distance to the closest boundary. We can estimate P_+ and P_-, from manually segmented images. One could use the histograms of intensities in these images directly (Cremers and Rousson, 2006), but we chose to approximate them by Gaussians, with specific means and variances, $\mu_\pm$ and $\sigma_\pm^2$. This has the advantage of specifying the distribution using a small number of parameters, which can be manually optimized by the user. We then get

$$\begin{aligned} E_I &= \int_{\Omega_+} \frac{1}{2} \log(\sigma_+)(I(x)-\mu_+)^2 \,\mathrm{d}x \\ &\quad + \int_{\Omega_-} \frac{1}{2} \log(\sigma_-)(I(x)-\mu_-)^2 \,\mathrm{d}x \\ &= c_+ \int_{\Omega_+} (I(x)-\mu_+)^2 \,\mathrm{d}x + c_- \int_{\Omega_-} (I(x)-\mu_-)^2 \,\mathrm{d}x, \end{aligned}$$

where $c_\pm = (1/2)\log(\sigma_\pm)$. The cost function E_I is low if the pixels inside the objects are close to μ_+, and the pixels outside are close to μ_-.

The prior-dependent energy E_π is a sum of the two terms E_{cont} and E_{smooth}:

$$E_\pi = E_{\mathrm{cont}} + E_{\mathrm{smooth}}.$$

The first term, E_{cont}, ensures that segmentations of adjacent images are similar, and is identical to the corresponding term in the algorithm described in Section 3.3.2:

$$E_{\mathrm{cont}} = \lambda_{\mathrm{cont}} \int_\Omega |\phi(x) - \phi_0(x)|_\tau^2 \,\mathrm{d}x.$$

We also include a second term, E_{smooth}, which penalizes the length of the boundary between foreground and background, favoring smooth boundaries:

$$E_{\mathrm{smooth}} = \lambda_{\mathrm{smooth}} \int_\Omega \delta(\phi(x))|\nabla\phi| \,\mathrm{d}x.$$

The choice of the parameters λ_{cont} and $\lambda_{\mathrm{smooth}}$ determines the relative importance of the two energy terms. In particular, the smoothness term can be switched off by choosing the weight $\lambda_{\mathrm{smooth}}$ to be 0.

If $\lambda_{\mathrm{cont}} = 0$, this model is similar to the (piecewise continuous) version of the Mumford-Shah functional (Chan and Vese, 2001; Mumford and Shah, 1989). In our model, the means, $\mu_\pm$, are known a priori and do not have to be optimized further.

We are primarily interested in the regions where the actual segmentation ϕ is close to 0 rather than in calculating its exact value for all x. Therefore, we also add the constraint that, for all pixels x:

$$|\phi(x) - \phi_0(x)| < \hat{\tau},$$

where $\hat{\tau}$ is some positive scalar value. This constraint is computationally convenient, as it allows us to restrict all calculations to the region of the image where $|\phi_0| < \hat{\tau}$. The parameter $\hat{\tau}$ is chosen such that it exceeds the largest difference expected between successive images.

3.3.4. Solution by gradient descent

We can find a (local) minimum of the energy function E by an iterative procedure: Starting with an initial guess, $\phi^{(0)}$, we repeatedly update the segmentation function ϕ via the evolution equation:

$$\phi^{(k+1)} = \phi^{(k)} + t_{\mathrm{step}} \frac{\mathrm{d}\phi}{\mathrm{d}t},$$

where t_{step} is the step-size. The update-direction $\mathrm{d}\phi/\mathrm{d}t$ can be found by calculating the gradient of the function E with respect to the segmentation function ϕ (Chan and Vese, 2001):

$$\begin{aligned} \frac{\partial E}{\partial \phi} &= \frac{\mathrm{d}\phi}{\mathrm{d}t} \\ &= -\delta(\phi)(\log(P_+(I)) - \log(P_-(I))) - \frac{\partial}{\partial\phi} \log(P(\phi|\phi_0)). \end{aligned}$$

With the particular distributions $P_\pm$ and our energy function $E = E_I + E_{\mathrm{cont}} + E_{\mathrm{smooth}}$ we obtain

$$\begin{aligned} \frac{\mathrm{d}\phi}{\mathrm{d}t} &= \delta(\phi)(c_+(I-\mu_+)^2 \\ &\quad -c_-(I-\mu_-)^2) + 2\lambda_{\mathrm{cont}}|\phi-\phi_0|_\tau \,\mathrm{sign}(\phi-\phi_0) \\ &\quad +\lambda_{\mathrm{smooth}}\delta(\phi)\nabla\cdot\left(\frac{\nabla\phi}{\|\nabla\phi\|}\right). \end{aligned}$$

The gradient $\mathrm{d}\phi/\mathrm{d}t$ is, except for the last two terms in the sum, identical to the one used in (Chan and Vese, 2001). We employ the commonly used smooth approximations, H_ϵ and δ_ϵ, for the Heaviside- and δ-functions:

$$H_\epsilon(s) = \frac{1}{2}\left(1 + \frac{2}{\pi}\tan^{-1}\left(\frac{s}{\epsilon}\right)\right),$$

$$\delta_\epsilon(s) = \frac{\mathrm{d}}{\mathrm{d}s}H_\epsilon(s) = \frac{\epsilon}{\pi(\epsilon^2+s^2)},$$

where ϵ is a positive parameter.

We initialize the algorithm with the segmentation function of the previous image, i.e. by $\phi^{(0)} = \phi_0$. Since this initial value is usually reasonably close to the desired solution, the algorithm converges very quickly. In addition, we only have to update ϕ close to the boundary Γ, and not on the whole domain Ω.

Even if the segmentation ϕ is initialized to be a signed distance function, this property is not preserved by the update step. Therefore, ϕ has to be *reinitialized* every few iterations to ensure stability of the algorithm. Reinitialization creates a SDF function that is consistent with the given segmentation. For details, see Osher and Fedkiw (2003) and Sethian (1999).

The algorithm described in this section is more flexible than the one in Section 3.3.2, but also computationally more demanding, as each iteration takes as long as one run of the simple algorithm.

3.4. Segmentation of first image

For the segmentation of the very first image in a stack, we can not rely on information from the preceeding slice, so a slightly different strategy has to be used. We segment the first image with one of the two algorithms above (ignoring the term E_π), and define objects to be the connected components of foreground regions Ω_+, which we call $o_1 \dots o_n$. For each object o_i, we have to determine whether it actually corresponds to a neuronal process. We extract three coefficients for each object (mean intensity, area, and "roundness", i.e. area divided by square of circumference), and perform classification in this three-dimensional space with logistic regression, using the manually labeled data as a training set. We also allow adjustment of the weights of the regression via sliding-bars. With this procedure, a reasonable initialization is obtained quickly, which can subsequently be fine-tuned and corrected manually.

4. Segmentation software

4.1. Graphical user interface

The graphical user interface (GUI) was implemented using the object-oriented programming language Java. The preprocessing steps (Section 2) and the segmentation algorithms (Section 3) were implemented in MATLAB (Mathworks), and the matlab-engine is called within the Java-program. Our software includes an easy-to-operate GUI and provides three image-processing functionalities:

- *Image Viewer*. The image viewer displays image stacks in single-image or movie mode and can optionally calculate the intensity distribution for each image.
- *Image Preprocessing*. The GUI allows selection of the filter (Section 2), and the setting of user-specified parameters.
- *Image Segmentation*. The GUI allows to choose between the segmentation algorithms (Section 3) and to set additional parameters. After the segmentation of the first image, the resulting boundaries are superimposed on the gray-level image. The user can now change parameters and select regions of interest manually. Only the selected regions are used to calculate the prior for the following image. It is also possible to interact manually during the segmentation process of later images by inserting new regions or by adjusting contour lines. In a last step, contour line information of each image is stored in an Extensible Markup Language (xml) file that can later be used for reconstruction and visualization of 3D surfaces.

Regions can be adjusted or added manually by inserting points along the desired contour using mouse-clicks, which are subsequently interpolated by B-splines. B-splines can easily be edited and quickly recalculated, because changes in one control point only affect the local shape of the curve (de Boor, 1978). For the implementation we used functions from Matlab's Spline Toolbox.

The following figure (Fig. 2) displays screenshots of the GUI, e.g. segmentation of the first image (segmentation movies of 100 and 215 images available as supplementary material).

4.2. Visualization

The software was developed to visualize 3D structures of neuronal circuits. The reconstruction of neuronal surfaces from contour lines is implemented using a reconstruction method similar to that described by Keppel (1975). The software provides 3D visualization during the segmentation process (Fig. 3).

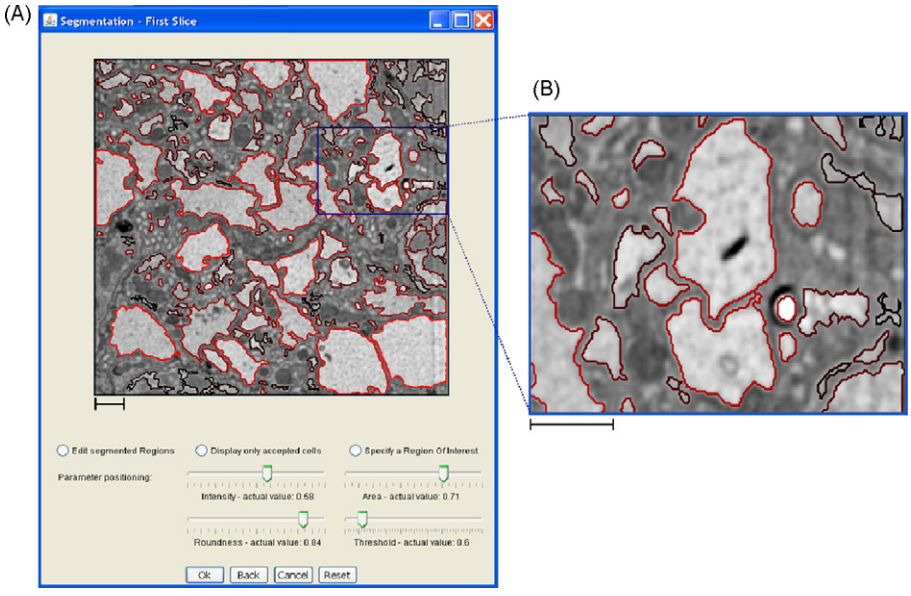

Fig. 2. Segmentation of the first image (screen shots). The overview (A) shows the image and the parameter adjustment GUI. In the zoomed image (B) it can be seen how even small structures can be detected. For each object of the first segmented image the mean intensity, area, and "roundness", i.e. area divided by square of circumference, are extracted (see Section 3.4). The weighted sum of these three coefficients is represented by the color of the contour line: a continuous color scale is used from black for very small values to red for very high values of the weighted sum. Scale bars 1 μm.

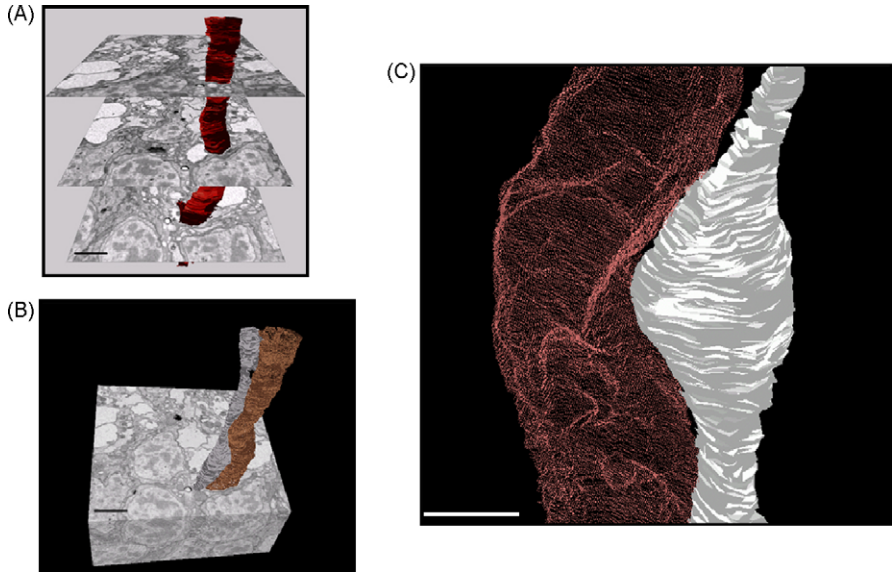

Fig. 3. Visualization of axonal structures (size of image stack: 14 μm × 14 μm × 26 μm, slice thickness 50 nm, average diameter of process shown in brown is 1.53 μm and in grey 1.13 μm). (A) Single axonal structure with SBFSEM-images. (B) Two axonal structures lying close to each other, embedded in a SBFSEM-image block. (C) Zoomed and rotated view of (B). Scale bars: 3 μm (A and B); 1 μm (C).

The reconstructed anatomical data can be exported to different 3D visualization programs, such as Amira (2006). To give an impression of the final visualization of neuronal structures, Fig. 3 displays 3D surfaces of axonal structures in the outer chiasm of the blowfly *Calliphora vicina*.

5. Quantifying the performance of the algorithms

The performance of the described algorithms depends on the properties of the original image stack, e.g. the image contrast and number of disrupted images. In particular, we assumed that objects are separated from the background by their pixel intensities, and that the objects do not vary to quickly between slices. These assumptions have to hold in order for the algorithms to work.

To quantify performance we used an image stack from the *Calliphora vicina* outer chiasm. Osmium tetroxide and uranyl acetate were used as contrast agents, which made membranes appear darker. We filtered the image stack with the Gaussian filter, standard deviation $\sigma = 1$ pixel (Section 2) and extracted an image cube of 512 × 512 × 250 pixels in x-, y- and z-direction. The extracted image cube covers a tissue volume of 14 μm × 14 μm × 13 μm. To demonstrate the performance of the algorithm for processes with different properties, we selected regions with area size ranging from 55 up to 7800 pixels in the initial slice, and different "roundness factors" (see Fig. 4).

We analyzed this data set with both methods described in Section3:

- Using the simple algorithm (Section 3.3.2), it took, on average, 3 s to segment one image. User interaction consisted of eye-inspection of the segmentation and, if necessary, manual correction. In the image stack displayed above, the user had to correct 2% of 2000 regions manually.
- *Alternative algorithm.* Average processing time per image amounted to 5 s plus user interaction. Out of 2140 regions, 1.5% regions had to be corrected manually.

For both segmentation methods the quality of the segmentation, and hence the need for user interaction depends mainly on the contrast between the object and the surround background as

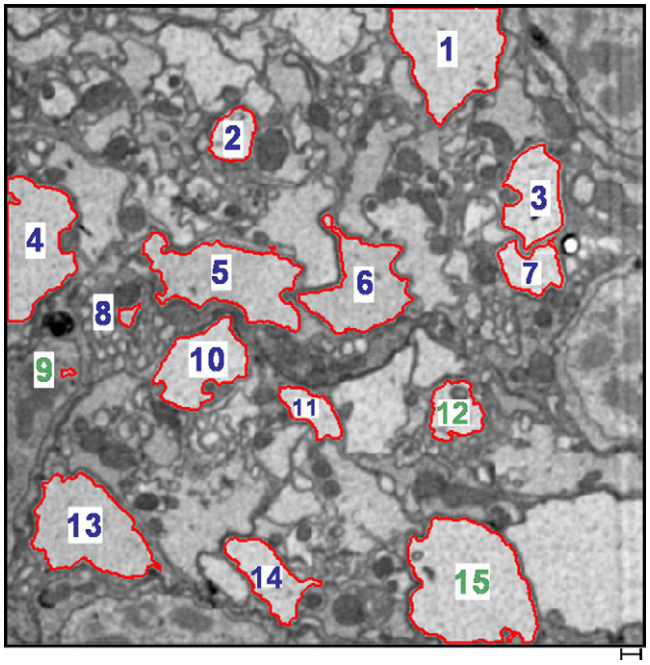

Fig. 4. Quantification of performance. Fifteen regions (labeled in red) were segmented for the performance test. The number of pixels per region ranges from 55 pixels (region 9, labeled in green) up to 7800 pixels (region 15, labeled in green). The roundness factors range from 0.3217 (region 12, labeled in green) up to 0.9069 (region 9, labeled in green). Scale bar: 20 pixels (540 nm).

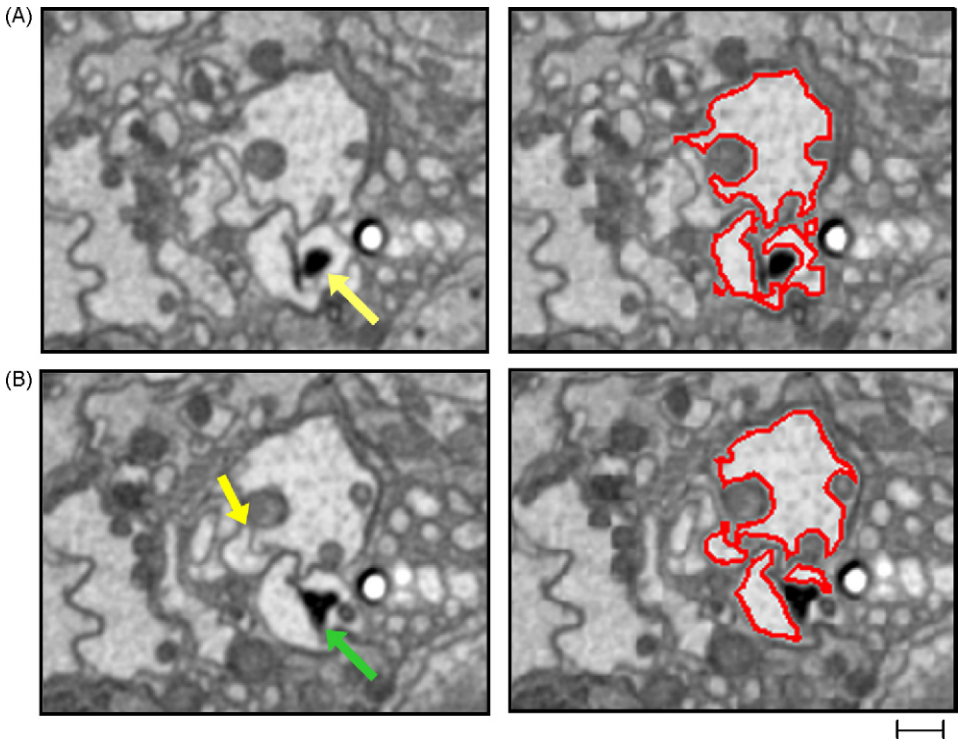

Fig. 5. Typical errors made by the algorithm. Two typical situations in which the algorithm failed to give a satisfactory segmentation, and had to be corrected manually. (Left side) Original image section; (right side) segmentation results, labeled in red. Scale bar: 28 pixels (750 nm). (A) Image 120: the structure surrounding the black dot (see yellow arrow) has a ragged contour, and is not successfully segmented by the algorithm, which splits it up into two regions. (B) Image 129: the region at the lower left of the region we want to segment is separated by only a thin and light line (yellow arrow), and thus wrongly joined to the large region. As before, the region indicated by the green line is separated into two regions.

well as the complexity of the contour lines. For example, region 15 (in Fig. 4), which has a high contrast and does not deviate from one image to the next, could be segmented and traced through all slices without any user interaction. Regions 3 and 7 in Fig. 4 have complicated contour lines which are very variable between images. Much more user interaction was required for these two objects than for others, but they could be reconstructed using a combination of algorithmic user-interaction and manual corrections (see Fig. 3 for the three-dimensional reconstruction, and Figs. 5 and 6).

Independently of the method used, the total time taken per image (algorithmic segmentation plus user interaction) was a about 15 s for 15 selected regions. A purely manual tracing of the selected regions (Fig. 4) takes about 12 min per image. The user had to trace the boundaries for each region manually by marking corner-points with mouse-clicks. We, thus, have a speed-up by a factor of about 50 by algorithmic over manual segmentation.

All performance tests were executed on a 64-bit workstation (64-Bit-Dual-Core Intel Xeon Processors, operating system: Windows XP64).

6. Possible extensions

6.1. Including texture information

Instead of using the difference in distributions of single pixel intensities, additional information can be obtained by looking at *texture* differences between fore- and background regions.

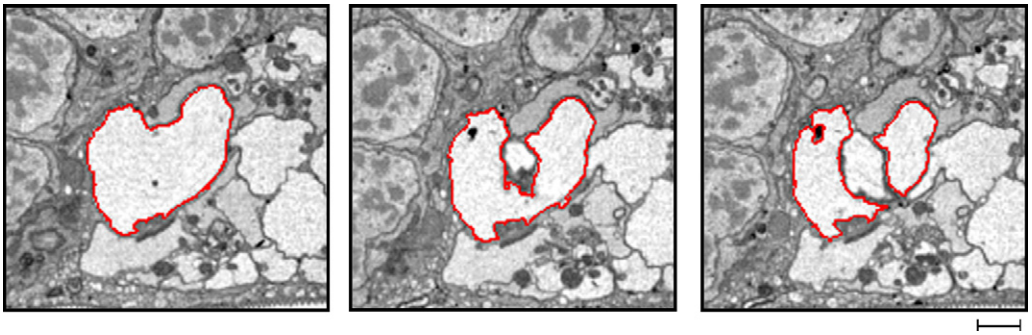

Fig. 6. Splitting of objects. The use of a level set approach makes it possible to segment splitting objects without reparametrization. In this example, the object is split into two constituents, each of which is tracked in subsequent images. Scale bar: 55 pixels (1.47 μm).

Texture information could be, for example, useful for detection of mitochondria. To use texture information, we can look at the neighborhood, $\mathcal{N}(x)$, of each pixel x, and estimate the distributions of these patches inside and outside objects. The neighborhood could either be taken to be two-dimensional, i.e. by looking at nearby pixels, or three-dimensional by also including pixels from adjacent images. We can train a classification algorithm on a training set of patches sampled from labeled images, and set $L(x)$ to be the output of the classifier to input $\mathcal{N}(x)$. The value of $L(x)$ will then be positive if the neighborhood $\mathcal{N}(x)$ is more likely to belong to the foreground than the background, and negative otherwise. This newly constructed image, $L(x)$, can be segmented using the algorithms described above. This approach can be implemented with any classifier that produces real-valued rather than binary output, such as support-vector machines (Kim et al., 2002).

6.2. Incorporating edge information

Region-based methods are only appropriate if the distributions of pixels (or textures) in the fore- and background regions are sufficiently different. In some situations, it might be advantageous to use information about the edges themselves, for example when different regions have similar statistical structure, but are separated by strong edges.

Edge-based algorithms could be incorporated by including additional terms into the evolution-equation for the segmentation ϕ, yielding a *level set equation* of the form:

$$\frac{\mathrm{d}\phi}{\mathrm{d}t} + \mathbf{V}_E \cdot \nabla\phi + (V_n - V_\kappa \kappa)|\nabla\phi| = 0.$$

The vector field $\mathbf{V}_E$ and the scalar field V_n are chosen such that the contour Γ moves towards edges of the image, with the aim of making the segmentation-boundary, Γ, consistent with edges in the image. The coefficient for the curvature-dependent force, V_κ (which has to be non-negative for the method to be stable), can be used to ensure smooth boundaries.

6.3. Flexible priors for ϕ

Our prior for the segmentation ϕ_n of image n is peaked at the segmentation ϕ_{n-1} of the previous image independent of the segmentation of slices $n-k$, with $k > 1$. Given sufficient training data, one could drop this rather restrictive assumption, and use more than one image in the calculation of the prior (Cremers, 2006). For example, one could assume ϕ_n to be similar to $2\phi_{n-1} - \phi_{n-2}$, which is the linear prediction based on the two previous images. More generally, one could learn the best prediction of ϕ_n (given previous slices), and use both the prediction and the estimated uncertainty of the prediction for the segmentation process.

7. Conclusion

The described segmentation algorithms in combination with the developed software allows a comfortable analysis of SBF-SEM image stacks. Images are analyzed in sequence to allow processing of large amounts of data, and to facilitate user interaction at every stage. Continuity of segmentation, which is important especially for the tracking of fine structure across images, is enforced by our algorithms. In addition, simultaneous reconstruction of the resulting 3D structure enables the user to locate errors and to interact during the segmentation step. While, ultimately, algorithms that do not require any or very little user-interaction are desirable, the algorithms described here present a first step in this direction. In particular, they can be applied if an extensively labeled data-set is not available, or as a means of quickly obtaining such data-sets.

Acknowledgements

We would like to thank Daniel Cremers for useful discussions and Matthias Bethge and Vinzenz Schönfelder for proofreading the manuscript.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.jneumeth.2007.07.021.

References

Al-Kofahi KA, Lasek S, Szarowski DH, Pace CJ, Nagy G, Turner JN, et al. Rapid automated three-dimensional tracing of neurons from confocal image stacks. IEEE Trans Inf Technol Biomed 2002;6:171–87.

Amira. Amira—advanced 3D visualization and volume modeling. http://www.amiravis.com/; 2006.

Borst A, Haag J. Neural networks in the cockpit of the fly. J Comp Physiol 2002;188:419–37.

Briggman K, Denk W. Towards neural circuit reconstruction with volume electron microscopy techniques. Curr Opin Neurobiol 2006;16:562–70.

Chan TF, Vese L. Active contours without edges. IEEE Trans Image Process 2001;10:266–77.

Cremers D. Dynamical statistical shape priors for level set based tracking. IEEE Trans Pattern Anal Mach Intell 2006;28:1262–73.

Cremers D, Rousson M. Efficient kernel density estimation of shape and intensity priors for level set segmentation. In: Suri JS, Farag A, editors. Parametric and geometric deformable models: an application in biomaterials and medical imagery. Springer; 2006.

Cremers D, Rousson M, Deriche R. A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape. Int J Comput Vision 2006:67–81.

D'Almeida F. Nonlinear diffusion toolbox for matlab. http://www.mathworks.com/matlabcentral/fileexchange/loadAuthor.do?objectType=author&objectId=938351; 2002.

de Boor C. A practical guide to splines. Springer Verlag; 1978.

Denk W, Horstmann H. Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. PLoS Biol 2002;2:e329.

Huang B, Li H, Huang X. A level set method for oil slick segmentation in SAR images. Int J Remote Sens 2005;26:1145–56.

Jurrus E, Tasdizen T, Koshevoy P, Fletcher PT, Hardy M, Chien CB, et al. Axon tracking in serial block-face scanning electron microscopy. In: Workshop on microscopic image analysis with application in biology; 2006.

Keppel E. Approximating complex surfaces by triangulation of contour lines. IBM J Res Dev 1975;19:2–11.

Kim K, Jung K, Park S, Kim H. Support vector machines for texture classification. IEEE Trans Pattern Anal Mach Intell 2002;24:1542–50.

Koh I, Lindquist W, Zito K, Nimchinsky EA, Svoboda K. An image analysis algorithm for dendritic spines. Neural Comput 2002;14:1283–310.

Mumford D, Shah J. Optimal approximations by piecewise smooth functions and associated variational problems. Commun Pur Appl Math 1989;42:577–684.

Osher S, Fedkiw R. Level set methods and dynamic implicit surfaces, vol. 153 of Applied mathematical sciences. New York: Springer Verlag; 2003.

Perona P, Malik J. Scale-space and edge detection using anisotropic diffusion. IEEE Trans Pattern Anal Mach Intell 1990;12:629–39.

Schmitt S, Evers J, Duch C, Scholz M, Obermayer K. New methods for the computer-assisted 3D reconstruction of neurons from confocal image stacks. Neuroimage 2004;23:1283–98.

Sethian JA. Level set methods and fast marching methods. In: Cambridge monograph on applied and computational mathematics. Cambridge University Press; 1999.

Strausfeld NJ. Functional neuroanatomy of the blowfly's visual system. In: Ali MA, editor. Photoreception and vision in invertebrates. Plenum Publishing Corporation; 1984. p. 483–522.

Vollgraf R, Scholz M, Meinertzhagen I, Obermayer K. Nonlinear filtering of electron micrographs by means of support vector regression. In: Thrun S, Saul L, Scholkopf B, editors. Adv Neural Inf Process Syst, 16. Cambridge, MA: MIT Press; 2004.

Whitaker R, Breen D, Museth K, Soni N. Segmentation of biological volume datasets using a level set framework. Vol Graphics 2001:249–63.