Published in final edited form as:
Neuroinformatics. 2009 June ; 7(2): 113–130. doi:10.1007/s12021-009-9047-0.

An Automated Pipeline for Dendrite Spine Detection and Tracking of 3D Optical Microscopy Neuron Images of In Vivo Mouse Models

Jing Fan,
Center for Biotechnology and Informatics, Department of Radiology, The Methodist Hospital Research Institute & The Methodist Hospital, Weill Cornell Medical College, Houston, TX 77030, USA, Department of Electrical & Computer Engineering, Northeastern University, Boston, MA 02115, USA

Xiaobo Zhou,
Center for Biotechnology and Informatics, Department of Radiology, The Methodist Hospital Research Institute & The Methodist Hospital, Weill Cornell Medical College, Houston, TX 77030, USA

Jennifer G. Dy,
Department of Electrical & Computer Engineering, Northeastern University, Boston, MA 02115, USA

Yong Zhang, and
Center for Biotechnology and Informatics, Department of Radiology, The Methodist Hospital Research Institute & The Methodist Hospital, Weill Cornell Medical College, Houston, TX 77030, USA

Stephen T. C. Wong
Center for Biotechnology and Informatics, Department of Radiology, The Methodist Hospital Research Institute & The Methodist Hospital, Weill Cornell Medical College, Houston, TX 77030, USA

Abstract

The variations in dendritic branch morphology and spine density provide insightful information about the brain function and possible treatment to neurodegenerative disease, for example investigating structural plasticity during the course of Alzheimer's disease. Most automated image processing methods aiming at analyzing these problems are developed for in vitro data. However, in vivo neuron images provide real time information and direct observation of the dynamics of a disease process in a live animal model. This paper presents an automated approach for detecting spines and tracking spine evolution over time with in vivo image data in an animal model of Alzheimer's disease. We propose an automated pipeline starting with curvilinear structure detection to determine the medial axis of the dendritic backbone and spines connected to the backbone. We, then, propose the adaptive local binary fitting (aLBF) energy level set model to accurately locate the boundary of dendritic structures using the central line of curvilinear structure as initialization. To track the growth or loss of spines, we present a maximum likelihood based technique to find the graph homomorphism between two image graph structures at different time points. We employ dynamic programming to search for the optimum solution. The pipeline enables us to extract dynamically changing information from real time *in vivo* data. We validate our proposed approach by comparing with manual results

X. Zhou, Bioinformatics, TMHRI, Department of Radiology, Cornell University, 6565 Fannin Street, B5-014, Houston, TX 77030-2707, USA, XZhou@tmhs.org.

Information Sharing Statement: The image data used in this work are provided by Drs Tara Spires-Jones and Bradley Hyman at MassGeneral Institute for Neurodegenerative Disease, Massachusetts General Hospital. We are unable to release these data. However, the algorithm used in this work is available from the authors upon request.

generated by neurologists. In addition, we discuss the performance of 3D based segmentation and conclude that our method is more accurate in identifying weak spines. Experiments show that our approach can quickly and accurately detect and quantify spines of in vivo neuron images and is able to identify spine elimination and formation.

Keywords

Dendrite spine detection; Quantification; Tracking; Level set; In vivo image

Introduction

The analysis of dendritic spine morphology is an important endeavor of neurobiology research since over 90% of excitatory synapses in the brain occur on dendritic spines and protrusion (Peters et al. 1991). Studies of dendritic structures are traditionally done in in vitro cell assays. Recent publications indicate that the morphological characteristics of neuronal structure viewed from in vitro microscopy are closely related to neural functions such as learning, memory and attention (Segal 2005; Bourne and Harris 2008). In a live animal model, in vivo imaging describes the actual micro-environment of living organisms under observation as opposed to in vitro assays. In addition, the dynamics of disease processes and treatments can be directly observed, which reveal characteristics undetectable in vitro. This motivates studies of disease mechanisms and therapeutic effects on in vivo bioassays and imaging of disease models. Although in vivo image data are studied in some pathophysiology research and treatment hypothesis of neuron diseases, we rarely find literature that focus on the automated processing of in vivo image data.

The study reported here aims to automate neuron image analysis of dendritic images of live brain. The dataset employed (courtesy of Drs Tara Spires-Jones and Bradley Hyman) is from an animal model of Alzheimer's disease (AD) that develops senile plaques and has plaque-associated dendritic spine loss due to impaired spine stability near plaques (Spires et al. 2005; Spires-Jones et al. 2007). We intend to follow dendritic spines over time to identify whether they are stable, eliminated, or if new spines form providing readout of structural plasticity that could be used to understand the pathophysiology of AD and to investigate therapies, such as antibody treatment. In vivo images allow tracking of morphological changes of the spines before and after treatment. These image data which are collected from a living animal model enable observation of dendritic spine dynamics. We collect neuron images from a number of time points, assess the spine number and view spine formation and elimination along the same dendrite segments. The challenge is to automate the spine quantification and tracking of in vivo neuron image analysis in a robust and accurate manner.

Koh et al. (2001, 2002),Mosaliganti et al. (2006) and Yang et al. (2006) presented various ways to perform image pre-processing, segmentation and tracking of neuron images. However, their images were captured in vitro, where the data was acquired in a controlled environment outside a living organism. Non-invasive real time imaging, in vivo, takes place inside a live organism and allows scientists to witness the process in real-time without distortion which occurs when removed from a live animal during in vitro imaging. To date we found only few works (Mizrahi et al. 2006) on in vivo image analysis and automatic processing due to the poor quality of in vivo image data as compared to in vitro images. The pipeline for the in vitro image data analysis in Koh et al. (2001) and Mosaliganti et al. (2006) started with thresholding-based segmentation to create binary images. Then a morphological thinning operator was applied on binary images to acquire central lines of dendritic structures and a line structure filtering was applied to get the backbone, detached spines and branch points. Cheng et al. (2007) adopted similar strategies but defined more elaborate criteria such as signal to noise ratio (SNR) to differentiate spine

structures from noise, which performed slightly better than the method in Koh et al. (2001). Otsu's thresholding (Otsu 1979) adopted in above papers missed weak spines because the global thresholding based method was unable to solve image inhomogeneity. Marker-controlled watershed is another approach for biomedical image segmentation. But it is very difficult to set the appropriate markers for neuron dendrite images. These image segmentation techniques applied to in vitro images do not work well on in vivo images because in vivo image data are very noisy, blurred and inhomogeneous.

Matching spines for different time points is another task which assists in estimating whether the treatment is effective or not. In Al-Kofahi et al. (2002), there was an assumption of constant number of spines at different times. However in our studies, we do not have any prior knowledge on the number of spines. In addition, the literature on cell and other structure tracking methods usually employed Kalman filtering and mean-shift methods (Yang et al. 2006), both of which required a long time sequence. In our case, there are only two or three time points available. The tracking methods requiring long sequence of time points are not appropriate.

This paper addresses a new pipeline for automated segmentation and tracking of spines from 3D in vivo optical images of neurons. Figure 1 illustrates the whole process of our proposed strategy. Since it is difficult to directly segment foreground spine structures due to poor quality of in vivo data, we directly detect central lines of dendritic backbones and spines. Then, a modified Local Binary Fitting (LBF) energy level set formulation with adaptive variances for the Gaussian kernel of each pixel is proposed to segment the dendritic spines using the central line as level set initialization. This method can detect weak spines and successfully avoid uncorrelated structures. We, then, extract morphological features about the spines, such as area and length, which we utilize later in spine tracking. Using the central lines of the dendritic structure and shape information of spines, we propose a spine tracking method based on maximum likelihood estimation (MLE, Kay, 1993) and dynamic programming (DP, Bertsekas, 2000). Our proposed approach is able to identify changes in spine number caused by spine formation or elimination. The tracking is based on properties of branch points on the central lines as well as their corresponding spine morphological features. A probabilistic model and graph homomorphism are employed to match the spines under the framework of MLE and DP.

The paper is organized as follows. Section 2 describes our approach in detail. Here, we explain acquisition of images, extraction of dendritic structure central lines, detection of spines and dendritic boundaries, extraction of branch point attributes and tracking of spines. The experimental results and validation are presented in the Experimental Results and Validation, Section 3. Algorithm-based issues and scientific generalizations are discussed in Discussion, Section 4. We conclude our work in Conclusion, Section 5.

Materials and Methods

In this section, we first describe how we acquire and pre-process our neuron images. Then we present each step for spine detection and tracking of these images in detail. Figure 1 displays a flow chart of our approach. We start with curvilinear structure detection to get the essential information about points and lines. Then spine boundaries are detected by the proposed aLBF level set model. We extract shape information of the spines, which finally allows us to match spines at different time points and to model the dynamics of the dendritic structures. We also briefly describe a 3D level set method of spine segmentation as a comparison with our proposed spine extraction method at the end of this section.

Image Acquisition

The image data were collected at the MassGeneral Institute for Neurodegenerative Disease, Massachusetts General Hospital by Drs Tara Spires-Jones and Bradley Hyman. Anesthetized mice with a cranial window were positioned on a multi-photon microscope. Surgical details are provided in Spires et al. (2005). Green fluorescent dextrans were injected into primary somatosensory cortex. Emitted light in the ranges of 380 to 480 nm, 500 to 540 nm, and 560 to 650 nm was collected by three photomultiplier tubes. To avoid photo damage, lowest laser power that could detect all spines was used for acquisition. Z-stack images were collected with an inter-slice interval of 0.5 μm and each stack contained 15–20 images. The dendrites were reimaged under the same conditions 1 h later. Dendrites were analyzed in green channel images. The *xy* resolution of the image data was 0.06 μm/pixel. The image sizes are 512 by 512.

Image Pre-processing

In vivo images exhibit noise, blurs and sharp variations in intensity. Image pre-processing is therefore necessary before further analysis. Our pre-processing involves the following steps: We first deblur the 3D image data, project the 3D stack of each time point into 2D, and then register 2D images of different time points in the same data set.

Deblurring—Because of the relative movement between different frames in the same stack and corruption by noise, the projection of the original image is blurred. We deblur 3D images by deconvolving with a Gibson's point spread function (PSF, Gibson and Lanni 1991) to correct artifacts introduced by breathing and heartbeat. In our data, main structures appear at the middle of the stacks. Gibson's PSF considers the non-stationarity of the PSF and asymmetry focal plane. We use this property of Gibson's PSF to obtain optimum deblurring results at the middle of the stack.

Projection—After deblurring, we use maximum intensity projection (MIP) to project 3D images into 2D, which collects the maximum intensity along z-stack.

Registration—Figure 2a displays an overlapped image of two time points prior to registration. Red pixels represent one time point and green pixels represent the other. The yellow pixels in Fig. 2b stand for the overlap of two time points. We observe considerable movement and small deformation between two time points, therefore image registration is necessary. We apply the Iterative Closest Point (ICP) algorithm (Besl and McKay 1992) to align the skeletons. ICP finds the closest point on a geometric entity, which is the foreground object (dendritic structures in our case) in an image to a given point. The ICP algorithm always converges monotonically to the nearest local minimum of a mean-square distance metric, and the rate of convergence is fast (Besl and McKay, 1992). Here, we selecte three key points marked by different signs in Fig. 2a on the dendritic structure at each time point as the control points to guide the registration. Figure 2b shows an overlapping view of both time points with ICP registration applied. Since no severe deformations are observed, we find it sufficient to consider only rigid transformations with the ICP algorithm.

Backbone and Spine Detection

In our algorithm, it is important to first identify the following key structures: backbone, medial axis and branch points. A *backbone* is the centerline of the dendrite structure. A *medial axis* is the connected centerline of spines and dendrite structures. A *branch point* is a pixel which has no less than three branches. In the following, we present our approach to backbone and spine detection: curvilinear structure detection, followed by line linking and branch point detection.

Curvilinear Structure Detection—A curvilinear structure is a line or a curve with some width, such as dendritic structures in the neuron image and roads in aerial images (Carsten, 1998; Jang and Hong, 2002; Xiong et al. 2006; Zhang et al. 2007). A desired curvilinear structure detector should be able to robustly distinguish lines with a certain range of width and high curvature. In our problem, we would like to precisely detect the central lines of the neuron structures. However, in Carsten (1998) there was a primary restriction on line width because the method were designed to find narrow and long structures with small variation of widths along the line direction. We adopt the strategy described in Xiong et al. (2006) and Zhang et al. (2007) which allows an adaptive width. The steps are as follows:

1. *Local point detection, direction estimation and linking*: To detect the lines, we first apply a Gaussian derivative kernel, which essentially smoothes (de-noises) the image before obtaining its gradient. The variance of the Gaussian kernel should be large enough to detect wide lines. However the thin structures or curves are blurred out. By employing the Hessian matrix calculated for each pixel, we vary the variance of the Gaussian kernel to find the largest eigenvalue and its corresponding eigenvector to determine local direction of the line. The magnitude of the second derivative of the image is considered as the strength map. The points with strengths larger than a given threshold are identified as line points. This method overcomes the drawback of Steger's method of a fixed line width. More importantly, our method works well for the weak lines whose strengths are relatively small. To link the detected line points, the user specifies two line strength thresholds for the starting and end points for a data set. The linking starts at the points with strength values higher than upper threshold and ends at points lower than the lower threshold. Figure 3 demonstrates the line direction detection result. Figure 3a illustrates the relationship between the line direction marked by the red arrow and its normal direction presented by the green line. Figure 3b displays a magnified view of the spine. The blue points are the detected center line points of the curvilinear structures and the cyan arrows denote normal directions.

2. *Backbone linking*: As mentioned above, we classify structures into two classes: backbone and spines. The backbone structures are detected lines with lengths larger than a user specified threshold. It is possible that backbones are disconnected due to noise and inhomogeneity. Therefore connection is necessary. We search for the points which fit the predefined criteria for boundary points described in Zhang et al. (2007) on the strength map. The boundaries are rough estimates of the line segment positions marked as red dots in Fig. 3b. We link two backbone structures if their estimated boundaries overlap in the eight-neighborhood area and linking radius is smaller than the linking threshold. Here we employ the Bresenham line linking algorithm (Bresenham 1965) to connect the backbone segments.

Figure 4a shows the line detection results of the MIP image of the R10XB data sets at time point 1 without any post-processing. The blue lines denote the detected backbone while the red ones indicate the spines. Ideally, according to our previous definition, the backbone should be a smooth line without many trivial branches. However, due to attached spines to the dendrite and the poor quality of the image, as shown in Fig. 4b, some spines are connected to the backbone as branches. We could extract the backbone without branches by line filtering as mentioned in Cheng et al. (2007). However, this is unnecessary because our objective is to extract the medial axis of all the dendritic structure, including backbones and spines, then, more significantly, to locate the backbone branch point. We will describe in the next part that, instead of chopping off the branch, we link detached spines to the backbone.

Linking Detached Spines to the Backbone—Medial axis is a linked line structure of detached spines and backbones. Figure 4b is the linked results of Fig. 4a, which displays the

medial axis of the entire dendritic structure. In the part, we will explain in detail how we link spine segments to the backbone.

We first describe how to identify a candidate end-point on the spine segment to link. For simple cases where the spine segment has no branch, the closest end-point to the backbone is chosen as the candidate for connecting. There are some special cases that are more complicated. The first is that the central lines of the detached spines have more than one branch. In this case, the end-point along the direction of the main structure would be chosen, because usually spines do not curve sharply. Figure 5a illustrates this circumstance where branch 1 follows the true direction. In Fig. 5 the cyan lines represent the backbone and the red line segments indicate spines, while the yellow dots are end-points. Another case is that there are two possible branches that are following the 'right' direction. As shown in Fig. 5b, branch 3 and branch 4 are both smooth extensions of the line before branching. We choose the branch with the smallest distance from the branch end points along its direction to the backbone. As illustrated in Fig. 5b, branch 3 would be our choice since line section 33′ is shorter than 44′, which indicates smaller distance. We formulate the previously described problem as follows:

$$\text{end point } i = \begin{cases} \arg\max_{i=1,2} \left(V_{bp} \times V_{p_i}\right) & if \left|V_{bp} \times V_{p_1} - V_{bp} \times V_{p_2}\right| \geq C \\ \arg\min_{i=1,2} \left[\text{distance}\,(p_i, b)\right] & if \left|V_{bp} \times V_{p_1} - V_{bp} \times V_{p_2}\right| \leq C \end{cases} \tag{1}$$

where p_i represents the end-points on the proximal-end-of-spine branch to the backbone, V_{bp} and V_{pi} denote the direction vectors at the spine branch-point and end-point respectively. C is a threshold which determines whether a distance condition is required for identifying the end point. b represents the backbone. The model can be easily extended to instances with more than two branches.

Secondly, we describe the approach to connect spines to their corresponding backbone. With pre-computed products, we remove spine structures which are away from the backbone based on a distance threshold. After that, the candidate end-point are connected to the backbone according to the end-points' direction with direction vector V_p. We examine each point on the backbone to calculate the distance and the direction vector from the backbone point to spine end-point with direction vector V_b. By taking the inner product, we find the largest magnitude of inner products and define the corresponding point at the backbone as a branch point for the spine. Then we use the Bresenham line linking algorithm to connect two points to get the medial axis of the neuron structure. In the case that there are multiple backbone segments, we avoid cyclic structures by choosing the candidate point which has the smallest distance to the spine end-point. We formulate the problem of locating the branch point j on the backbone for a given spine end-point i as follows:

$$\arg\max_{j=1,2,\ldots} \left(\left|V_{p_i} \times V_{b_j}\right|\right) \tag{2}$$

where V_{pi} denotes the direction vectors of a given end-point on the spine and V_{bj} represents the direction vector of a candidate branch-point on the backbone.

Here, we would like to add one more comment on the direction of the end-points. As shown in Fig. 3b, where cyan arrows present the normal direction, the end-point p is oriented perpendicularly to the other line-points in the same line structure. In this instance, we use the averaged direction vector of neighbors of the end-point. In Fig. 3b, the average of the direction

vectors of two points in the yellow circle would be utilized as the end-point direction vector. We summarize the above description as follows:

$$V_{\text{ep}} = \begin{cases} V_{\text{ep}} & \text{if } V_{\text{ep}} \times V_n < 0.7 \\ V_n & \text{if } V_{\text{ep}} \times V_n > 0.7 \end{cases} \tag{3}$$

where V_{ep} stands for the direction vector of the end-point and V_n is the average of the direction vector of the neighboring points mentioned above. The threshold, which is a measure of similarity of the endpoints' directions, is determined by experiments on several data sets.

Branch Point Detection—In this section, we describe the method for detecting branch points on the central lines of neuron structures. There are two types of branch points on the backbone. One is generated by attached spines and the other appears after connecting detached spines to the backbone. Different strategies are adopted to detect two categories as follows.

A template of *n*-by-*n* matrix of 1 s is applied to the backbone to identify branch points introduced by attached spines. Points with the largest magnitude of a template image on the backbone are considered as branch points. If two or more points have the same value, the neighboring information is used to choose the point with a larger value to its left or right as the branch point. One or 2 pixel uncertainty of the branch points is acceptable because: (1) the probabilistic model for tracking in the next section is robust to 10–20 pixel variance; (2) the registration inaccuracy is larger than 1 pixel; (3) the neighboring attribute for tracking, such as curvature and the direction of the branch point would not fluctuate much in a small neighborhood.

Another kind of branch point is generated by linking detached spines to the backbone. As described in the previous part of detached spine linking, we locate the branch point on the backbone of its matching spine end point by finding the maximal inner product between two vectors: V_p and V_b. The process is illustrated in Fig. 5c. The pink arrow is the direction of the spine end point and the orange ones are the other kind described in the above paragraph. We choose point B at the backbone as the branch point of its corresponding spine.

The central line of the dendritic structures, the branch points of spines and backbones are now available for subsequent processing. Figures 6 and 7 respectively illustrate line detection results of two images, which we will discuss thoroughly in the results section. We can easily measure the spine length on the medial axis by removing the backbone (Cheng et al. 2007).

Boundary Detection with an aLBF Level Set Model

In order to obtain shape information of the spines, we need to accurately locate the boundaries of the dendritic structures. The boundaries can provide information such as area and perimeter which are more explicit to biologists than the medial axis.

The previous extracted medial axis provides a good approximate of dendritic structure and can be used as an initialization for most active contour methods, such as edge based models. The energy function of the piecewise constant (*PC*) model (Chan and Vese 2001) is

$$\begin{aligned} E(c_1, c_2, C) = {} & \mu \text{Length}(C) + \upsilon \text{Area}(\text{in}(C)) \\ & + \lambda_1 \int_{\text{in}(C)} |I(x) - c_1|^2 dx + \lambda_2 \int_{\text{out}(C)} |I(x) - c_2|^2 dx \end{aligned} \tag{4}$$

where $I(x)$ is the image domain. $\mu \geq 0$, $\upsilon \geq 0$, $\lambda_1 \geq 0$, $\lambda_2 \geq 0$ are fixed parameters. in(C) and out (C) stand for regions inside and outside of the contour C. c_1 and c_2 are average intensities in in(C) and out(C). Li et al. (2007) modified the model (4) to solve inhomogeneity by defining a local binary fitting (LBF) energy function for each point given by

$$\begin{aligned}\varepsilon_x\,(C, f_1(x)\,,\, f_2(x)) = \lambda_1 \int_{\text{in}(C)} K_\sigma\,(x-y)\,|I(y) - f_1(x)|^2 dxdy \\ + \lambda_2 \int_{\text{out}(C)} K_\sigma\,(x-y)\,|I(y) - f_2(x)|^2 dxdy\end{aligned} \tag{5}$$

where $I(x)$,λ_1 and λ_2 are the same parameters as those in the *PC* model. $K_\sigma(x)$ is a Gaussian kernel with variance σ, whose value is decreased as $|x|$ increases. Its expression is given by (6). $f_1(x)$ and $f_2(x)$ are the average intensity inside and outside the contour C and with the effect of the local kernel $K_\sigma(x)$, they are actually the average intensities near point x. The local Gaussian kernel is defined as:

$$K_\sigma(t) = \frac{1}{(2\pi)^{1/2}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{6}$$

The LBF model outperforms the *PC* model in LBF's strength in processing inhomogeneous images. However, we need to set λ_1 and λ_2 large to detect all the dendritic structures, including weak spines with low intensities. But large λ_1 and λ_2 can also detect low intensity noise and irrelevant structures. On the other hand, small λ_1 and λ_2 will constrain the zero level set to high intensity regions, which tends to miss some important structures. We solve this problem by adaptively changing σ of the Gaussian kernel while setting λ_1 and λ_2 large enough to detect all the structures. To do this, we first define the following function to calculate σ for each pixel.

$$\sigma_x = \begin{cases} \frac{d_x}{1+g_x}\sigma_s & \text{if point } (x) \notin \{\text{Medial axis}\} \\ \sigma_c & \text{if point } (x) \in \{\text{Medial axis}\} \end{cases} \tag{7}$$

where x represents pixel, σ_x is the variance of the Gaussian kernel at each pixel, d_x is the distance between pixel and the medial axis. g_x is the magnitude of the gradient at each pixel. σ_s is the starting variance value of the Gaussian kernel on each pixel except for the medial axis points whose variance is σ_c. We named our modified LBF model as *adaptive* LBF (aLBF) model. The meaning of Eq. 7 is that when the pixel is far away from the medial axis, the image is smoothed to avoid irrelevant structures and noise. At the same time, since gradients indicate edges, the variance should be small enough to be sensitive to these edges. For points on the medial axis, which is the initialization of the level set formulation, we also employ a small σ to make sure that the weak boundaries are detected. To save calculation cost, if σ_x is larger than a given threshold $\sigma_{\max}$, we set it to $\sigma_{\max}$. In this formulation, the LBF energy for each pixel in Eq. 7 becomes:

$$\begin{aligned}\varepsilon_x\,(C, f_1(x)\,,\, f_2(x)) = \lambda_1 \int_{\text{in}(C)} K_{\sigma_x}\,(x-y)\,|I(y) - f_1(x)|^2 dxdy \\ + \lambda_2 \int_{\text{out}(C)} K_{\sigma_x}\,(x-y)\,|I(y) - f_2(x)|^2 dxdy\end{aligned} \tag{8}$$

Figure 8 shows a comparison between LBF and aLBF segmentation results and details will be provided in the results section. With segmented images we can calculate signal to noise ratio (SNR) which is an important measurement of system robustness. SNR is defined as $SNR = 10\log_{10}(P_s/P_n)$ where P_s is the power spectrum of the signal while P_n is the counterpart of noise. Here the signal is segmented dendritic structures and noise is the background.

Extraction of Morphological Information of Spines and Branch Points

The dendritic structure information obtained so far includes medial axes, branch points and boundaries of the dendritic structure. With this information, we can quantify some morphological characteristics, such as spine area and attributes of branch points which include curvatures, positions and branch directions.

When the medial axis image is overlapped with the boundary image, which is illustrated in Fig. 9a, the medial axis passes through the spine region detected by the aLBF method. We calculate the spine area in the following manner: (1) remove aLBF detected regions without intersections with the medial axis and remove the isolated regions where other regions intersect with the same branch. The result is shown in Fig. 9b; (2) extract regions that are detached from the main structures. These are the detached spines, whose areas are calculated as areas of the corresponding detached spines; (3) calculate attached spine areas. Since branch points located on the backbone are available, we follow the attached spine detection method described in Cheng et al. (2007) to identify the attached spines. The results are shown in Fig. 9d. The red regions are the attached spines that are detected by the automated method.

Now the detached spines and attached spines are successfully quantified in spine areas. By representing each branch point as spines, we build an attribute vector V which will be used for spine tracking. For each branch point, six attributes are defined: the Euclidean positions x_i and y_i, the curvature κ_i, the normalized spine area $\bar{a}_i$, and the direction information o_i and ω_i. We represent each set as:

$$V = \left\{ v_1 \cdots v_m | v_i \equiv [x_i, y_i, \kappa_i, \bar{a}_i, o_i, \omega_i] \in \Re^6 \right\} \tag{9}$$

where m is the total number of branch points at each time point.

Next, we describe how to calculate the attribute vector of the i-th branch point. To simplify the notation, we drop the 'i' in $v_i \equiv [x_i, y_i, \kappa_i, \bar{a}_i, o_i, \omega_i]$ temporarily. The curvature of the i-th branch point is given by

$$\kappa(t) = \frac{\dot{x}_\sigma(t)\,\ddot{y}_\sigma(t) - \ddot{x}_\sigma(t)\,\dot{y}_\sigma(t)}{\left[(\dot{x}_\sigma(t))^2 + (\dot{y}_\sigma(t))^2\right]^{3/2}} \tag{10}$$

where the image domain is parameterized by t and $\dot{x}_\sigma(t) = x(t) \times \dot{g}_\sigma(t)$, $\ddot{x}_\sigma(t) = x(t) \times \ddot{g}_\sigma(t)$ where $g_\sigma(t) = \frac{1}{(2\pi)^{1/2}\sigma} \exp\left(-\frac{t^2}{2\sigma^2}\right)$ is the Gaussian kernel with variance σ, which is different from the σ in the aLBF level set part. $\dot{y}_\sigma(t)$ and $\ddot{y}_\sigma(t)$ follow similar formulations as $x(t)$. Here the Gaussian kernel helps smooth the image and makes the algorithm numerically computable.

We normalize the spine area as:

$$\overline{a}_i = (a_i - a_{\min}) / (a_{\max} - a_{\min}) \tag{11}$$

where α_i is the area of each spine and $\alpha_{\min}$ and $\alpha_{\max}$ are the largest and smallest area respectively.

For different time steps, the backbone shape will not change dramatically, hence the curvatures of the corresponding branch points will not fluctuate significantly and neither does the area of the spine region. o_i is the tangent value which points from the branch point to a point several pixels away. We compute o_i as follows:

$$o_i = \left(y_{bp} - y_{ap}\right) / \left(x_{bp} - x_{ap}\right) \tag{12}$$

where x_{bp} and x_{ap} denote the x-axis Euclidean position of the branch point and the point several pixels away respectively and y_{bp} and y_{ap} are the corresponding y-axis Euclidean positions. We usually consider positions 5 pixels away, which is equal to the line length lower threshold. Another reason is that usually the central line of the spine does not have a sharp curvature at the beginning of the structure. However, when two branches point to opposite directions, we cannot differentiate these directions only with o_i. Therefore we add the left or right pointed term ω_i to remove this confusion. The left or right pointed term represents the branch's relative location to the skeleton. We denote the left-pointed branch as −1 and the right-pointed as 1. The term is determined using a technique similar to the one we used to detect the branch point.

Spine Tracking Based on Maximum Likelihood Estimation (MLE)

We represent the dendrite structure that we have extracted at each time point as a dendrite graph $G_i(V)$ of a dendritic structure comprised of the branch points and their attributes. We, then, match two dendrite graphs G_1 and G_2 of time points $T1$ and $T2$ by establishing the graph homomorphism (Hell and Nesetril 2004) between two graphs under with maximum likelihood estimation (MLE).

With dendrite graph, we translate the tracking problem to a graph homomorphism problem by representing each image as a graph with each branch point as a vertex. A graph homomorphism is a mapping $f : V_1 \rightarrow V_2$ from the dendrite graph G_1 to G_2. Most of the existing methods assume constant number of branch points. However in our problem, there exist spine formation and elimination due to the evolution of the neuron and the occurrence of spurious branch points. We employ MLE-based techniques to find the best homomorphism between the two graphs, which can solve spine evolution. Let (o, e) be a pair of vertices at two time points. v_0 is a point in data set V_1 of the first image and v_e is a point in the evolved data set. The objective is to find a mapping which maximizes the posterior $p((o,e)|v_e = f(v_o))$. According to Bayes' law, we get

$$p\left((o,e)\left|v_e = f(v_o)\right.\right) = \frac{p\left(v_e = f(v_o)\left|(o,e)\right.\right) p((o,e))}{p(v_e = f(v_o))} \tag{13}$$

where $p(v_e = f(v_o)|(o,e))$ is the likelihood, $p(v_e = f(v_o))$ is the evidence and $p((o,e))$ is a uniform distributed prior. Therefore, our problem reduces to finding the best e that maximizes the likelihood, which can be expressed as follows:

$$\arg\max_{e} p\left(v_e = f(v_o) \,|\, (o,e)\right) \tag{14}$$

We model this problem as a 6-dimensional Gaussian distribution with a diagonal covariance matrix because the six attributes are uncorrelated with each other. In our experiments, we assume the diagonal terms of the covariance matrix are [10 10 0.02 20 0.05 0.001]. 10 is the variance of the Euclidean position of the branch points which allows 10-pixel movement due to imperfect registration and noise to guarantee the robustness. The typical curvature of the branch points in our problem is in the range of [0.01, 0.20]. Hence we assume the variance of the curvature is 0.02. 20 is the variance for the area. As mentioned above, the mistake that matches a left-pointed branch to a right-pointed one is considered unacceptable, so we assumed a small variance for the left or right pointed term. We also set the variances for the tangent value as 0.05 and the left or right pointed term as 0.001. All these parameters are optimally determined by experiments. The Gaussian distribution $N(0,\Sigma)$ is formed as

$$\begin{aligned} &p\left(v_e = f(v_o) \,|\, (o,e)\right) \\ &= \frac{1}{(2\pi)^{6/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(v_e - v_o)^T \Sigma^{-1}(v_e - v_o)\right) \end{aligned} \tag{15}$$

We developed a dynamic programming (DP, Bertsekas, 2000) solution to find the best e that maximizes the likelihood robability $p(v_e = f(v_o)|(o,e))$ for all the branch points on the two dendritic sections. We build the DP algorithm for our problem as follows,

1. Let $d(ij)$ be the probability of $p(v_j = f(v_i)|(i,j))$. $d(ij)$ serves as the decision in the DP. $S(i, j)$ is the score for the best correspondences of the spines 1 to i in V_1 with spines 1 to j in V_2. The score is considered as a state in DP.
2. The equation of state is

$$S(i,j) = \begin{cases} S(i-1, j-1) + d(i,j) & S(i-1, j-1) + d(i,j) \geq \max[S(i-1, j), S(i, j-1)] \\ S(i-1, j) & S(i-1, j) > S(i, j-1) \\ S(i, j-1) & \text{Otherwise} \end{cases} \tag{16}$$

Equation 16 expresses how we update state when a new decision arrives. The termination condition is to find $S(n, m)$, where n and m are the total numbers of branch points in the two data sets.

Spine Segmentation with 3D Level Set Approach

3D image based strategies segment dendritic structure without image projection. We employ a LoG (Laplacian of Gaussian) based level set approach (Zhou et al. 2008) to segment neuron dendrites, attached spines as well as detached spines. We will compare 3D level set segmentation results with our proposed projected image based strategy to demonstrate the capability of our method in the Discussion section.

The preprocessing starts with a 3D median filter with a kernel size of 3 × 3 × 3 to remove shot noise introduced by imaging device. Then a top-hat operator is used to correct uneven illumination degradation and enhance the images. Considering C as a closed contour and φ as a signed distance function which is negative for points inside C, positive outside C and zero at C, we define the energy function as:

$$F(\varphi)=\lambda E(\varphi)+\alpha L(\varphi)+\beta V(\varphi) \tag{17}$$

where λ, α and β are weighting constants for three energy terms respectively defined as:

$$E(\varphi)=\oint_{\Gamma(\varphi=0)} \nabla(G_{\sigma_1}\times I)\vec{n}\, d\Gamma \tag{18}$$

$$L(\varphi)=\iiint_{\Omega} \delta(\varphi)\,|\nabla\varphi|\, dxdydz \tag{19}$$

$$V(\varphi)=\iiint_{\Omega} g(\nabla I)H(-\varphi)dxdydz \tag{20}$$

$$g(\nabla I)=1/(1+|\nabla(G_{\sigma_2}\times I)|^p) \tag{21}$$

Let Ω denote the image domain and I be an image, $E(\varphi)$ is LoG energy while $L(\varphi)$ and $V(\varphi)$ denote the area of zero level set surface and volume inside zero level set surface respectively. $G_{\sigma 1}$ is Gaussian kernel with standard deviation σ_1 and $\vec{n}$ denotes the outer normal direction of zero level set. δ is Dirac function, H is the Heaviside function and g is a positive non-increasing function defined as: $g(\nabla\ I) = 1/(1 + |\nabla(G_{\sigma 2} \times I)|^p)$, in which $G_{\sigma 2}$ is a Gaussian kernel with standard deviation σ_2. To minimize $F(\varphi)$, the Euler-Lagrange equation for φ can be written as:

$$\frac{\partial F}{\partial \varphi}=-\delta(\varphi)\left[\lambda\Delta G_{\sigma_1}\times I+\alpha\nabla\left(\frac{\nabla\varphi}{|\nabla\varphi|}\right)+\beta g(\nabla I)\right] \tag{22}$$

The first term of Eq. 22 aligns the curve close to the zero crossings along the edge. The second term keeps the surface of the zero level set smooth all along the evolution. The third term is a balloon force, which acts to speed up the evolution where the image gradient is small or slow down the evolution where the gradient is large.

Experimental Results and Validation

Results for Dendritic Structure Detection

Figures 6 and 7 show the dendritic structure detection results. Figures 6a and 7a are raw images without processing. Figures 6b and 7b illustrate the line detection results without post-processing. The cyan lines denote the backbone and the attached spine structures, while the red lines represent the detached spines. There are some line segment far away from the main dendritic structures and they are considered as noise which can be easily filtered. Figures 6c and 7c show the dendritic structures after linking detached spines to the backbone. The manual spine detection results are displayed in Figs. 6d and 7d.

In Fig. 6, by comparing c and d, we could tell that using our algorithm, we can detect all the spines marked manually by the biologist. The manual detection is performed in 3D without knowing the automated detection results. An expert observes several neighboring slices before marking each spine. The expert can zoom in every region during the observation. In addition to the manually marked spines, our method detected spines marked by capital letters, which are considered as false positives. The expert cannot confirm whether spine A is a positive one because it is overlapped with some uncorrelated structures even in the 3D domain. For spine B, we noticed that the expert first marked it and then deleted it based on the 3D information.

Figure 7 presents the dendritic structure detection result for the same data set at time point 2. From the result shown in (b), we could tell that we successfully detected the central line of this spine (Spine A in Fig. 7c). By comparing (c) with (d), we can see that our method finds all of the spines except the one manually marked as number 6. Note that spine number 6 is parallel to the backbone. The error appeared in the line linking process because of the parallelism. Another false positive happened at spine B which was ignored by the expert since it was too small.

Results for Dendritic Structure Boundary Extraction

Figure 8 shows the results of the standard LBF model and proposed modified aLBF model using the same parameter settings. The red contours in both images depict boundaries. Here, we set λ_1 and λ_2 to the smallest value that can identify the weak spines A and B marked by yellow circles in Fig. 8a. These two spines have relatively lower average intensities than the other spines. In Fig. 8b, with the same λ_1 and λ_2 setting, our method can identify spine A and B more accurately. The accuracy is important for spine tracking. We can see that in Fig. 8a there are more irrelevant structures and noises detected at the northwestern part than that at the same position in Fig. 8b, as well as in some other locations. Furthermore, the boundaries identified by aLBF were tighter and more precise than its counterparts in LBF. This desired property is an outcome of our adaptive setting for σ. Therefore, the aLBF model outperforms the LBF level set method in detecting weak spines.

Figure 10 displays the value of σ in the whole image domain. As mentioned above, we set the σ as σ_{max} if it is larger than σ_{max}. From Fig. 10a, a 3-D view of σ, we can tell that when the pixel is far away from the main structures, σ is set to σ_{max}. The value of σ near the medial axis varies according to Eq. 7. In Fig. 10, the red color denotes the large value while the blue one represents small.

Results for Spine Tracking

In this section, we present the spine tracking results with our MLE based method. Figure 11 shows the tracking result of two time points in data set R10XB. As the cyan two-way arrows illustrate, we find 14 pairs of the matching spines at two time points. Their branch points are circled by green rings. Yellow squares in the left image, which present the first time point, denote the spines for which we cannot find their evolved versions. They are considered as eliminated spines or false detected spines of our method. On the other hand, in the right image, the pink triangles at the second time point highlight the spine branch points without matching spines with its previous time point, which are newly formed ones. By comparing with the manual results, we notice that we successfully match 14 of the 17 spine pairs and locate one disappeared spine at the first time point and one newly grown spine at the second time point.

Validation

To evaluate our proposed algorithm for dendritic backbone and spine detection, we compare our results with manual labeling by visual inspection done by an expert. The protocol of derivation of manual results is described in early part of this section. The validation process is

designed as follows: we randomly select eight neuron images from a total of 14 images available to us. Eight images are four pairs of two time point data sets. Each pair of images was taken from different regions or different dendrites of mouse brain. For each image we randomly select one dendrite region to lighten the burden of the manual process. We manually label spines in each selected dendrite and use the results as the ground truth. Our proposed algorithm is then applied to each dendrite and results are compared with the manual results.

The manual method detected 107 spines while the automated method detected 115 spines. Among these spines, a total of 103 spines were detected by both methods; four additional spines were detected only by the manual method and they were false negatives which we fail to identify using the automatic method; 12 spines were detected only by the automated method and they were found to be false positives. As mentioned above, some of the false positives are introduced by noise and some are too small for the expert to mark. Table 1 quantitatively summarizes the results.

Figure 12 shows manual spine labeling results for two time points in data set R10XB. The manual labeling of the spines strictly follows the shape of the spines until the edge becomes too weak. Figure 13 illustrates the spine length comparison between the manual and automatic methods. From Fig. 13c we notice that in most cases, the spine lengths of the automatic method are shorter than the manual measurement. However, a small part of the automatic results are far larger than the manual ones because of noise and uncorrelated structures linked to the spines. From Fig. 13a, b, respectively the CDF (Cumulative Distribution Function) comparison and the quantile–quantile plot, we conclude that the manual and automatic results follow the same uniform distribution. The spine areas calculated from the automated and manual method are consistent. For attached spines, results are similar in both methods. For detached spines, manual results are larger than automated ones because our neurologist included the area between detached spines and dendrites. The comparison of R10XB time point 2 is illustrated by Fig. 14. KS (Kolmogorov–Smirnov) test confirms that automated and manual area measurements are drawn from the same distribution.

To validate the tracking results, we compare our results with manual results. We use the same data sets as those used in the spine detection validation part. The results are listed in Table 2. From Table 2 we observe that the total number of matched spines is 51 and our method identifies 43 of them. Compared with the manual results of two eliminations and three formations, our results are 15 and 14 respectively, much larger than the manual ones. The errors mainly introduced by the false positives in the spine detection as well as imperfect registration results. The errors in elimination are induced by false negatives in the first time point or by wrong matching results with time point 2. The errors in formation are caused by false positive detection of spines and also due to errors in tracking. Dynamic programming is a forward optimization technique, assuming that the solution previous to the current decision is optimal and it would not resort to the former decision to update current state. Therefore although it is a global optimization technique, matching errors may occasionally occur during the optimization process. A possible solution to improve the tracking performance is to incorporate more time points and integrate all information to eliminate false positives with “future” time point information. The false negatives can be corrected if the same spine appears in the previous time and the subsequent time but not in the current time point. Once we derive the accurate spine detection result with multiple time points instead of only two, dynamic programming will be able to find the global optimal solution.

Parameter Selection

Table 3 lists typical parameter values used in this work. The standard deviation for the Gaussian kernel is determined automatically in the curvilinear structure detection part. Other parameters for the curvilinear structure detection, such as the upper threshold, the minimum line length,

and the linking distance, can be fixed loosely using a large range and do not affect the final results significantly. This conclusion is derived by a large number of test experiments. The parameters for the backbone and spine detection include the weakness threshold and the minimum backbone length. These two parameters may vary for different images, but can be fixed for a set of images from the same biological experiment, or images with similar intensity distributions. In addition to the detection part, the parameter settings for the aLBF level set are also listed in Table 3. The other unmentioned parameters are the same as the original LBF model in Li et al. (2007). The average running time is only in minutes for processing a 512*512*10 neuron image with a Pentium IV 2.8G Hz processor for the selected parameters.

Discussion

3D based segmentation methods are employed in in vitro neuron spine detection. We also employ a 3D level set method to segment dendritic structures in in vivo data. A rotated version of our 3D segmentation result of R10XB time point 2 is shown in Fig. 15a. The uncorrelated structures far away from the dendrite were filtered. We could tell that the structure surfaces were smooth. However, this method failed in detecting the weak spines, which are marked by yellow circle in Fig. 15b. We marked the corresponding region in Fig. 15a where we could not see the spine. In 3D space, signals of this spine in each slice are very weak and they are contaminated by noise. However, in MIP projection, the largest intensities of each column of voxels are recorded in the projection image, thus the weak signal can be detected in the 2D MIP image. We then quantify the spines to get volumes (Zhou et al., 2008) and add this feature to the branch point feature vector. The variance of the Gaussian assumption for volumes in MLE model is set to 20. The tracking result is the same as previously reported results for data set R10XB and in other data sets.

This result indicates that if in vivo images are directly segmented and quantified in 3D, due to the low SNR, weak signals can be missed. Thus far, volume information does not improve the tracking result. Therefore, only 3D based methods may not be the optimal solution and in some case they are inferior to 2D based methods. Possibly a hybrid method including both 2D and 3D could improve spine detection as well as spine tracking. In addition, when 3D curvilinear structure detection is mature enough to detect lines with different widths in 3D space in a low SNR environment, we might generalize our whole pipeline to 3D.

Another issue is time series of images. The interval of our in vivo image data is 1 h. There are two time points in every data set. The details about these data sets are provided in Validation part in Section IV. If we keep this interval, the algorithm will be able to process data of multiple time points two by two and a summarization of the time series is necessary to incorporate time series information together. The time series information helps to eliminate false positives and false negatives. If the interval is larger, such as one day, the deformation and shifting make the problem much more difficult. A more complex non-rigid registration strategy should be necessary, as well as segmentation methods which are robust to changing and lower SNRs. Moreover, unlike in in vitro imaging, it is very challenging to keep the object stable for days to record the same regions of interest using in vivo imaging. Therefore, although time series images might be a potential improvement to derive better results, the difficulties of in vivo imaging should be addressed first.

The last issue we discuss is the robustness of our system. Signal to noise ratio (SNR) is an important criterion to quantify the quality of an image as well as a measurement of system robustness. In order to find a SNR threshold which approximately indicates how bad an image our system is able to process robustly, we tested the SNR of our validation data sets and images contaminated by additive Gaussian noise.

A box plot of SNRs of eight images used as validation data is presented in Fig. 16a with average SNR 24.2dB. We noticed that L10XBT1 had the worst SNR of 19.58dB. For image R10XBT1 and R10XDT2, the SNR are as low as 22.34dB and 21.92dB respectively, which was coincidental with the relatively large false detection error in spine detection. In addition, Gaussian additive noise with zero mean and different variance was added to R10XDT1, an image with relatively higher SNR (26.68) in our data set and results were illustrated by Fig. 16b. The SNR dramatically dropped when the variance increased from 1e-5 to 1e-3. When contaminated images were processed by our system, the false detection rate rises in accordance with the increase of Gaussian noise variance. We noticed that when SNR was larger than 10dB, the detection rate was higher than 75%. Therefore, we adopted 10dB as a SNR cut-off value for our system.

Conclusion

In this paper, we have presented a novel strategy for spine detection and tracking for in vivo neuron images. There is little work in in vivo neuron image processing and quantification available in the literature according to our knowledge. The pipeline presented here will help neurologists automatically label and quantify small spines and find matching spine pairs at different time points. It overcomes the difficulties of poor image quality and extracts neuron dynamic changes from the real-time in vivo data. The algorithm detects the dendritic backbones and the detached spines from noisy and low contrast in vivo image data of live mice of an AD model. It is able to quantify the spine length and area, with the proposed adaptive local binary fitting (aLBF) level set method. The proposed aLBF model can accurately locate the boundary of dendritic structures using the central line of curvilinear structure as the initialization. Then, we present a maximum likelihood estimation based tracking algorithm to track the evolution of spines at different time points and employ dynamic programming to find the solution. Our tracking algorithm is able to detect spine elimination and formation. The algorithm is highly automated and fast. It requires minimum human interaction. With the results derived from the proposed pipeline, biologists can study the pathology of certain types of neuron diseases and sketch the spine dynamics without manually labeling and pairing spine structures.

Acknowledgments

The authors would like to thank members of the Center for Biomedical Informatics, The Methodist Hospital Research Institute, especially Zheng Xia and former members of the HCNR Center for Bioinformatics, Harvard Medical School. We also thank Dr Tara Spires-Jones and Dr Bradley Hyman for providing sample in vivo image data. This research is partially funded by NIH R01 AG028928 and NIH R01 LM009161 and by HCNR, Harvard Medical School.

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 Transactions on Information Technology in Biomedicine 2002;6:171–187.10.1109/TITB.2002.1006304 [PubMed: 12075671]

Bertsekas, DP. Dynamic programming and optimal control. 2nd. Vol. 1 & 2. Athena Scientific; 2000.

Besl P, McKay N. A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence 1992;14:239–256.10.1109/34.121791

Bourne JN, Harris KM. Balancing structure and function at hippocampal dendritic spines. Annual Review of Neuroscience 2008;31:47–67.10.1146/annurev.neuro.31.060407.125646

Bresenham JE. Algorithm for computer control of a digital plotter. IBM Syst 1965;4:25–30.

Carsten S. An unbiased detector of curvilinear structures. IEEE Transactions on Pattern Analysis and Machine Intelligence 1998;20:113–125.10.1109/34.659930

Chan TF, Vese L. Active contours without edges. IEEE Transactions on Image Processing 2001;10:266–277.10.1109/ 83.902291 [PubMed: 18249617]

Cheng J, Zhou X, Miller E, Witt MR, Zhu J, Sabatini BL, et al. A novel computational approach for automatic dendrite spines detection in two-photon laser scan microscopy. Journal of Neuroscience Methods 2007;165:122–134.10.1016/j. jneumeth.2007.05.020 [PubMed: 17629570]

Gibson SF, Lanni F. Experimental test of an analytical model of aberration in an oil-immersion objective lens used in three-dimensional light microscopy. J Opt Soc Am A 1991;8:1601–1613.10.1364/JOSAA.8.001601

Hell, P.; Nesetril, J. Oxford Lecture Series in Mathematics and its Applications. Vol. 28. Oxford University Press; 2004. Graphs and homeomorphisms.

Jang JH, Hong KS. Detection of curvilinear structures and reconstruction of their regions in gray-scale images. Pattern Recognition 2002;35:807–824.10.1016/S0031-3203(01)00073-5

Kay, SM. Fundamentals of Statistical Signal Processing: Estimation Theory. Prentice Hall; 1993.

Koh IYY, Lindquist WB, Zito K, Nimchinsky EA, Svoboda K. An image analysis algorithm for dendritic spines. Neural Computation 2002;14:1283–1310.10.1162/089976602753712945 [PubMed: 12020447]

Li, C.; Kao, CY.; Gore, JC.; Ding, Z. Implicit active contour driven by local binary fitting energy. MN: IEEE Conf. Comput. Vis Pattern Recognition; Minneapolis, Mn. 2007. p. 2007

Mizrahi A, Lu J, Irving R, Feng G, Katz LC. In vivo imaging of juxtaglomerular neuron turnover in the mouse olfactory bulb. Proceedings of the National Academy of Sciences of the United States of America 2006;103:1912–1917.10.1073/ pnas.0506297103 [PubMed: 16446451]

Mosaliganti, KP.; Janoos, F.; Xu, X.; Machiraju, R.; Huang, K.; Wong, STC. Temporal matching of dendritic spines in confocal Microscopy images of neuronal tissue sections. MICCAI 2006 Proceedings; Copenhagen, Denmark. 2006. p. 106-113.

Otsu N. A threshold selection method from gray-level histogram. IEEE Transactions on Systems, Man, and Cybernetics 1979;9:62–66.10.1109/TSMC.1979.4310076

Peters, A.; Palay, SL.; Webster, H. The fine structure of the nervous system: neurons and their supporting cells. Oxford University Press; 1991.

Segal M. Dendritic spines and long-term plasticity. Nature Reviews Neuroscience 2005;6:277–284.10.1038/nrn1649

Spires-Jones TL, Meyer-Luehmann M, Osetek JD, Jones PB, Stern EA, Bacskai BJ, et al. Impaired spine stability underlies plaque-related spine loss in an Alzheimer's disease mouse model. American Journal of Pathology 2007;171:1304–1311.10.2353/ajpath.2007.070055 [PubMed: 17717139]

Spires TL, Meyer-Luehmann M, Stern EA, McLean PJ, Skoch J, Nguyen PT, et al. Dendritic spine abnormalities in amyloid precursor protein transgenic mice demonstrated by gene transfer and intravital multiphoton microscopy. The Journal of Neuroscience 2005;25:7278–7287.10.1523/JNEUROSCI.1879-05.2005 [PubMed: 16079410]

Xiong G, Zhou X, Degterev A, Ji L, Wong STC. Automated neurite labeling and analysis in fluorescence microscopy images. Cytometry Part A 2006;69A:494–505.

Yang X, Li H, Zhou X. Nuclei segmentation using marker-controlled watershed, tracking using mean-shift and Kalman filter in time-lapse microscopy. IEEE Trans on Circuits and Systems I 2006;53:2405–2414.10.1109/TCSI.2006.884469

Zhang Y, Zhou X, Witt R, Sabatini B, Adjeroh D, Wong STC. Dendritic spine detection using curvilinear structure detection and LDA classifier. NeuroImage 2007;36:346–360.10.1016/j.neuroimage.2007.02.044 [PubMed: 17448688]

Zhou, W.; Li, H.; Zhou, X. 3D Dendrite Reconstruction and Spine Identification. MICCAI; New York, NY. 2008. p. 2008

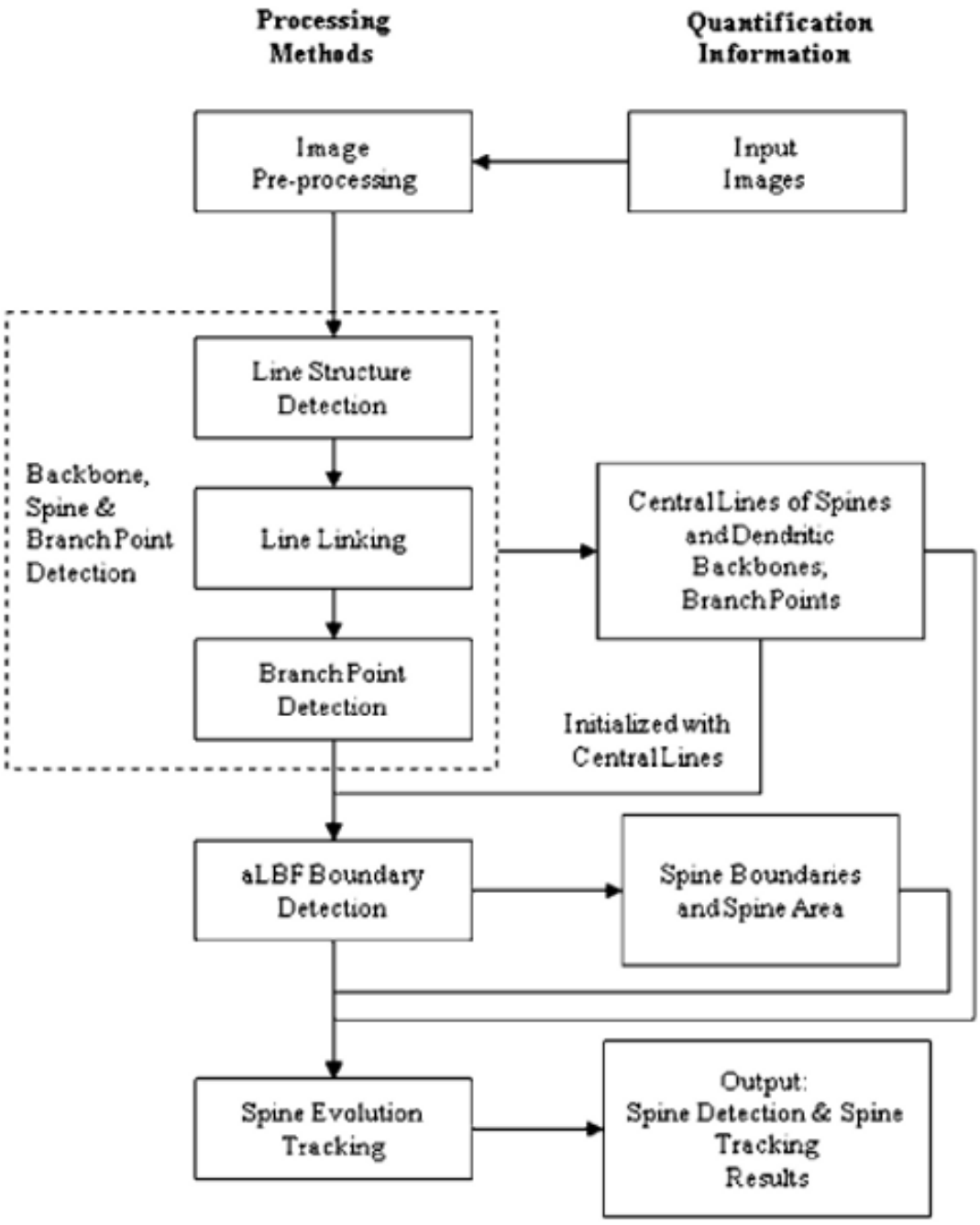

Fig. 1.
The pipeline of the proposed spine detection and tracking strategy

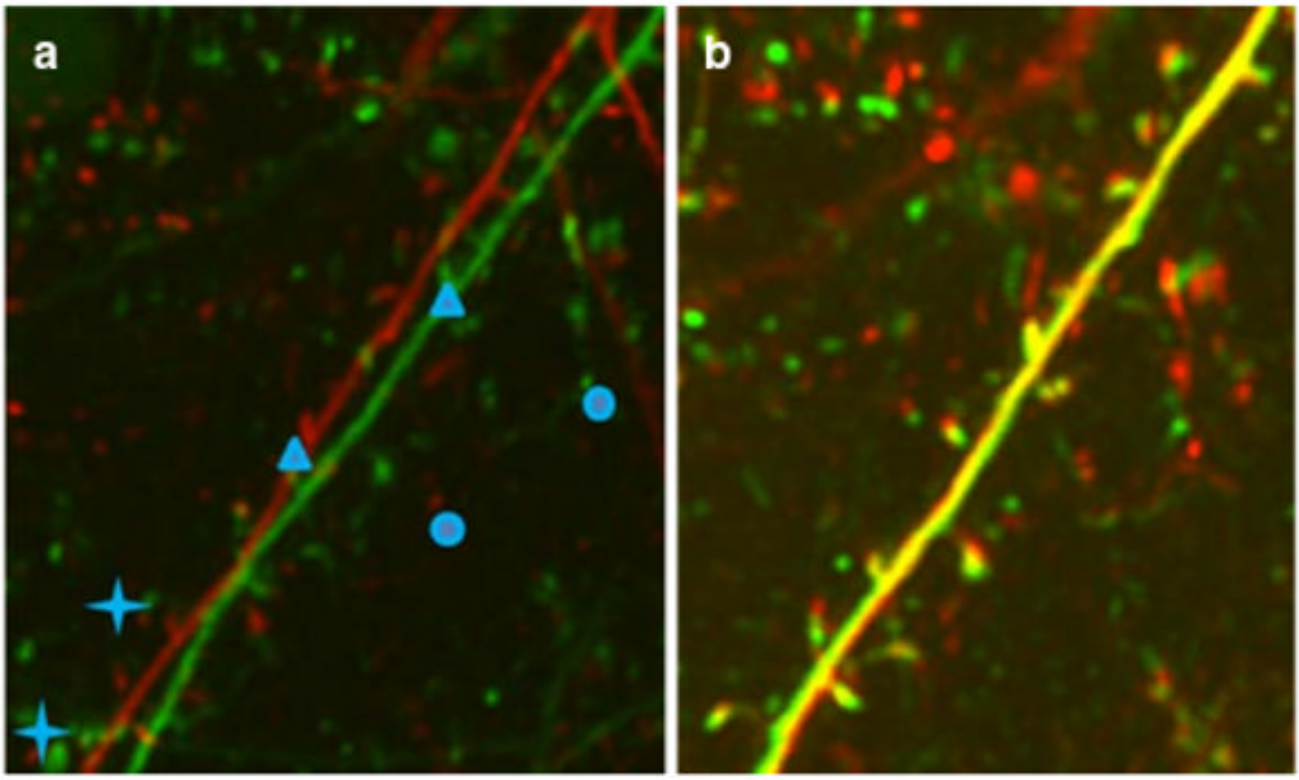

Fig. 2.
Illustration of registration. **a** Before registration. The *two colors* represent two time points. Control points are paired with different marks; **b** After registration. *Yellow* denotes the overlap of green and red

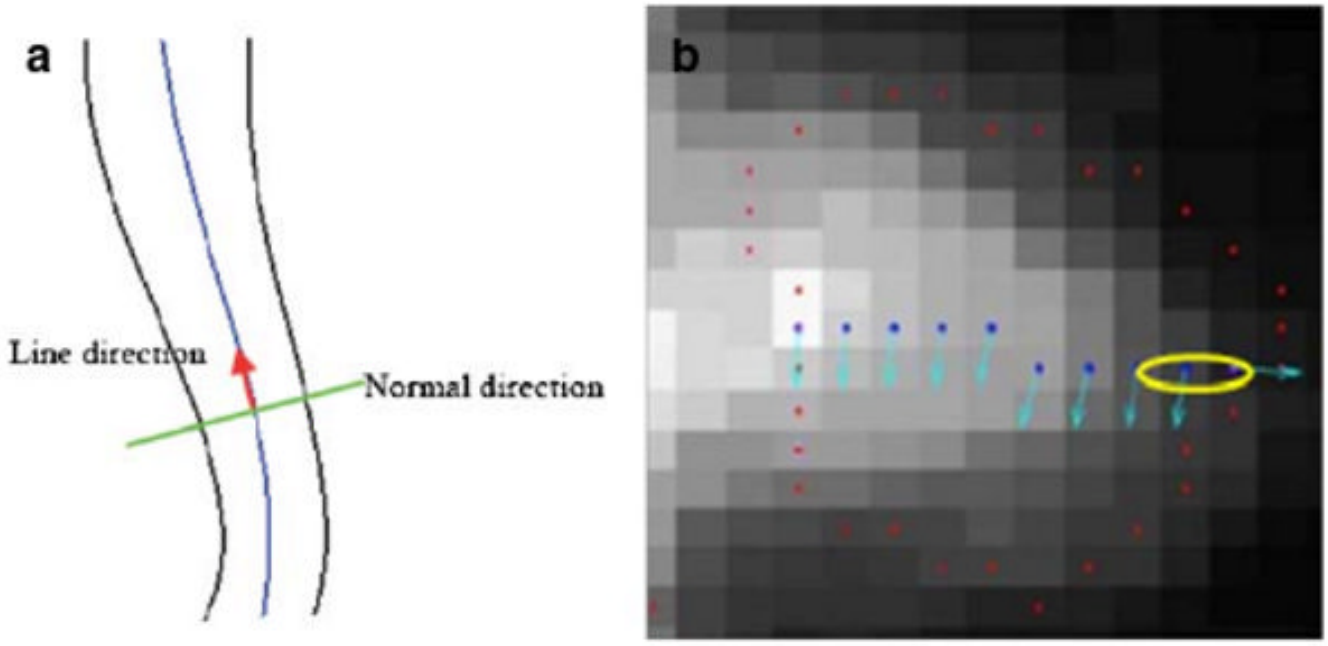

Fig. 3.

a Illustration of line direction and normal direction; **b** illustration of line detection algorithm. The *cyan arrows* denote the normal direction. *Red dots* are the rough estimates of the boundary

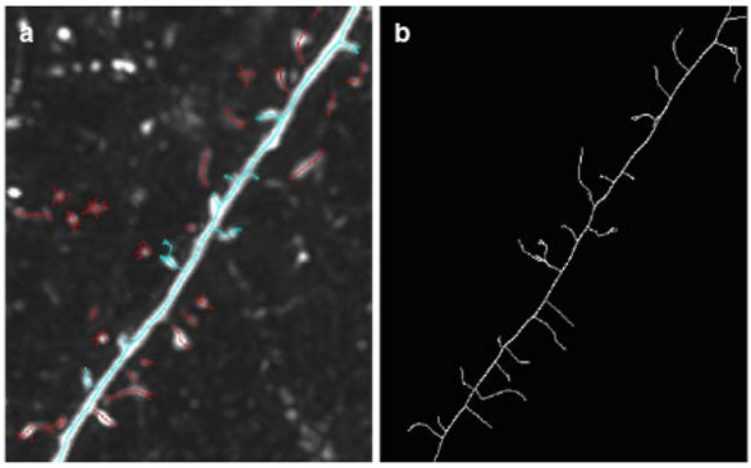

Fig. 4.
a Original spine and backbone detection results of R10XB time point one without post processing. *Cyan* backbone; *red* spines; **b** line linking results of **a**

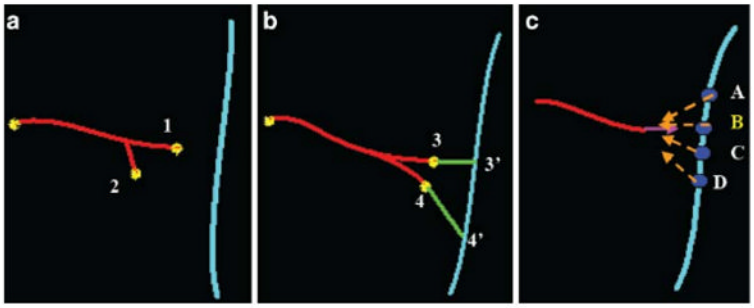

Fig. 5.

Some possible situations in line linking: **a** two branches with obvious orientation differences; **b** both branches are smoothly connected with the main structure; **c** illustration of branch point selection on the backbone; our algorithm will choose the *yellow marked point B. Purple arrow* is the direction of v_p and *orange arrows* are the direction of v_b

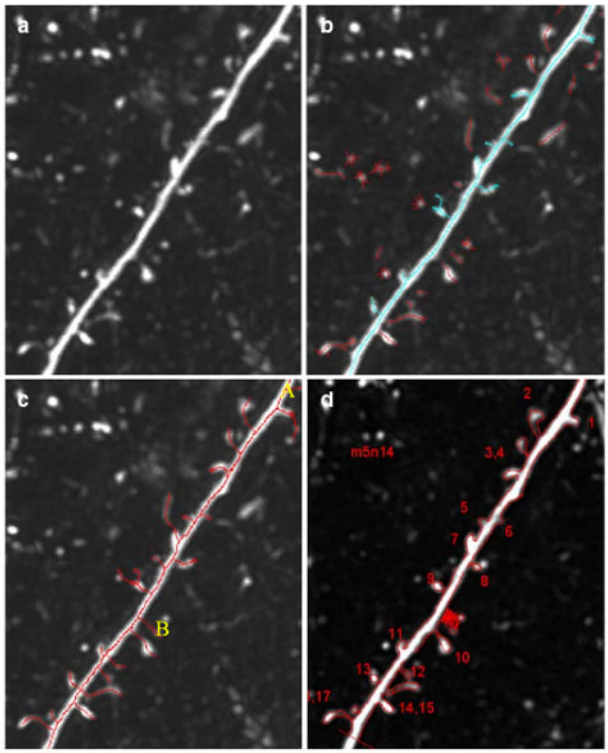

Fig. 6.

Line structure detection results of data set R10XB, time point one; **a** original image; **b** the line detection result without post processing; **c** medial axis derived by line linking. Branch A and B are false positive detected branches; **d** manual labeling result

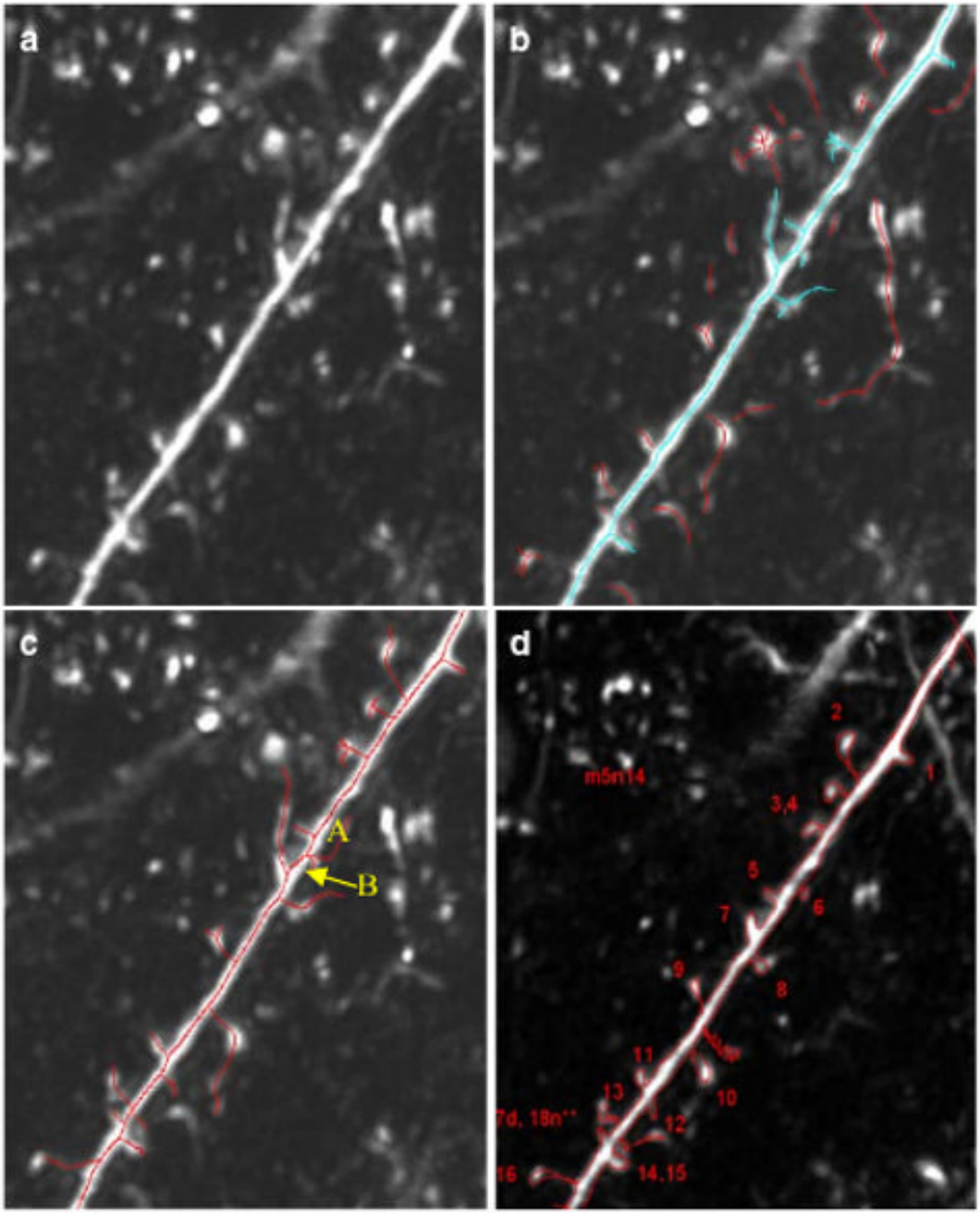

Fig. 7.
Line structure detection results of data set R10XB, time point two; **a** original image; **b** the line detection result without post processing; **c** medial axis derived by line linking; **d** manual labeling result

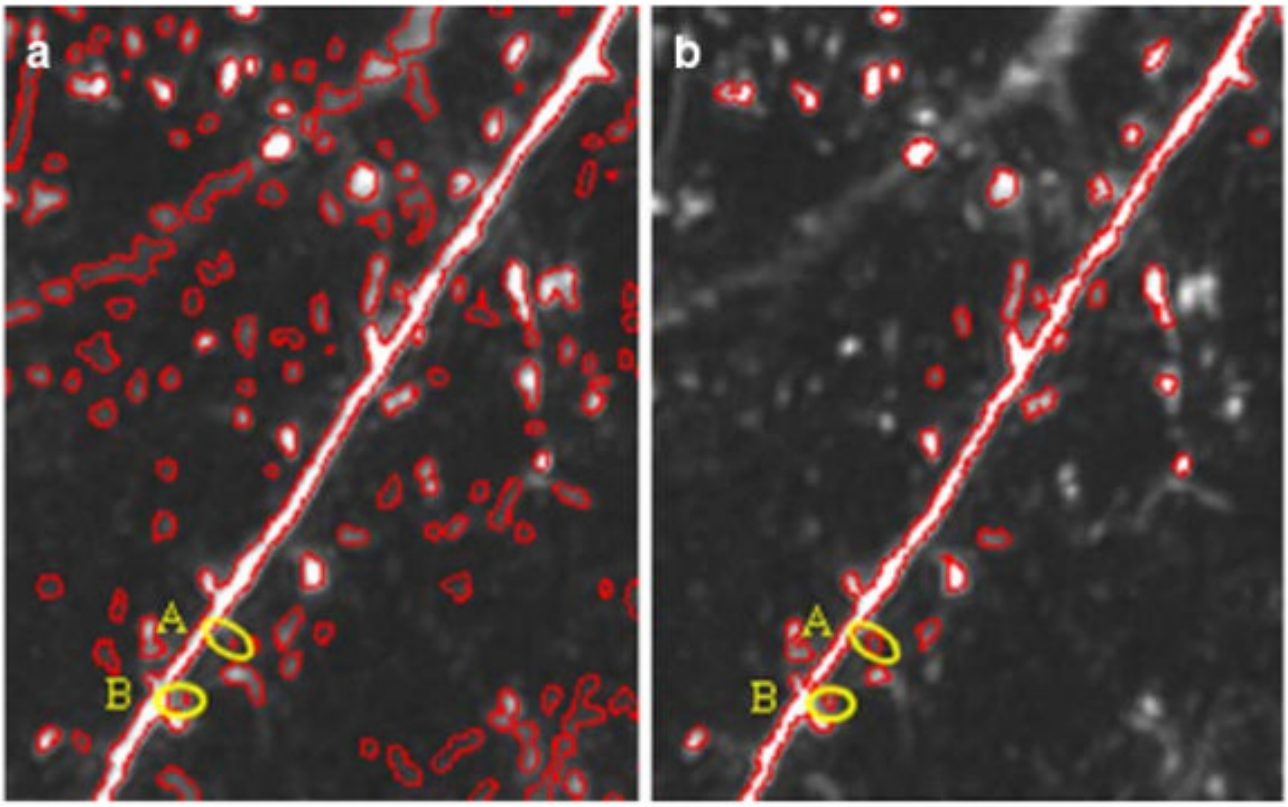

Fig. 8.
Comparison of **a** LBF level set model and **b** aLBF level set model. The *yellow circled regions* are weak spines

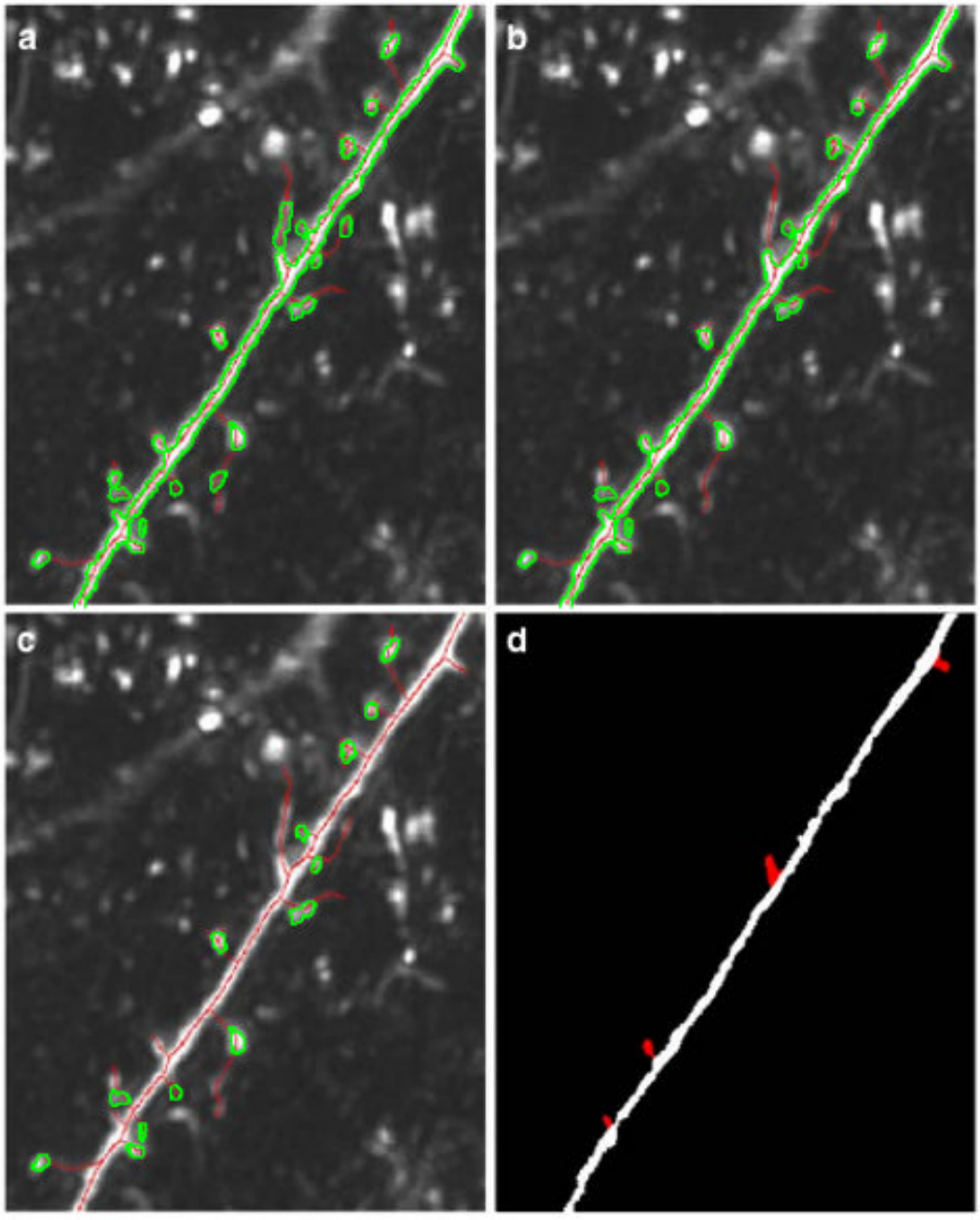

Fig. 9.

a Overlapped view of the medial axis and boundaries detected by aLBF; **b** result after filtering the un-intersected region and redundant region; **c** a view of detached spine regions; **d** a view of attached spine regions; the spines are presented in *red*

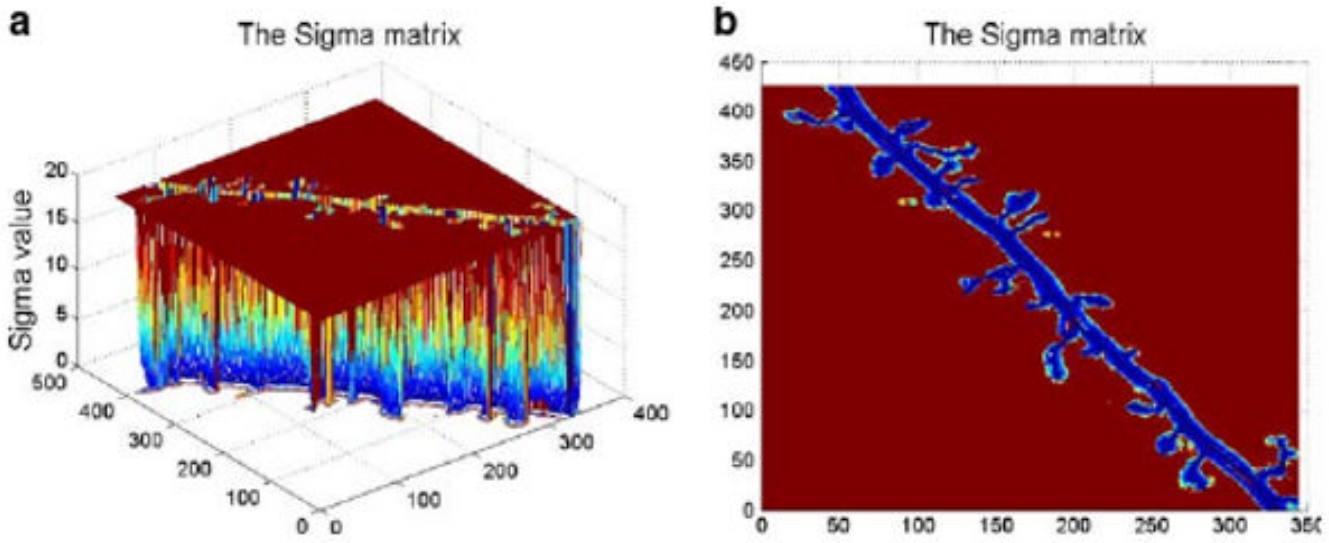

Fig. 10.
The value of sigma. **a** 3-D view; **b** 2-D view

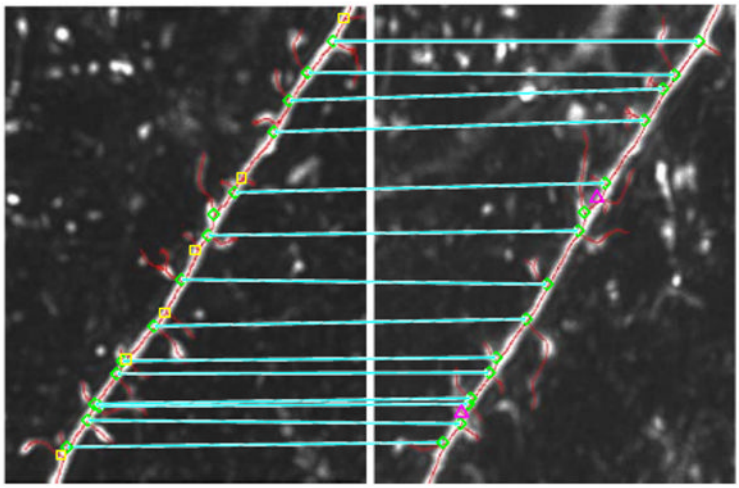

Fig. 11.

Tracking results for the two time points in data set R10XB. *Cyan arrows* denote the matching relationship. *Green circles* denote branch points that can find its matching point. *Yellow squares* represent the eliminated spines or noise while *purple triangles* are the newly formulated spines

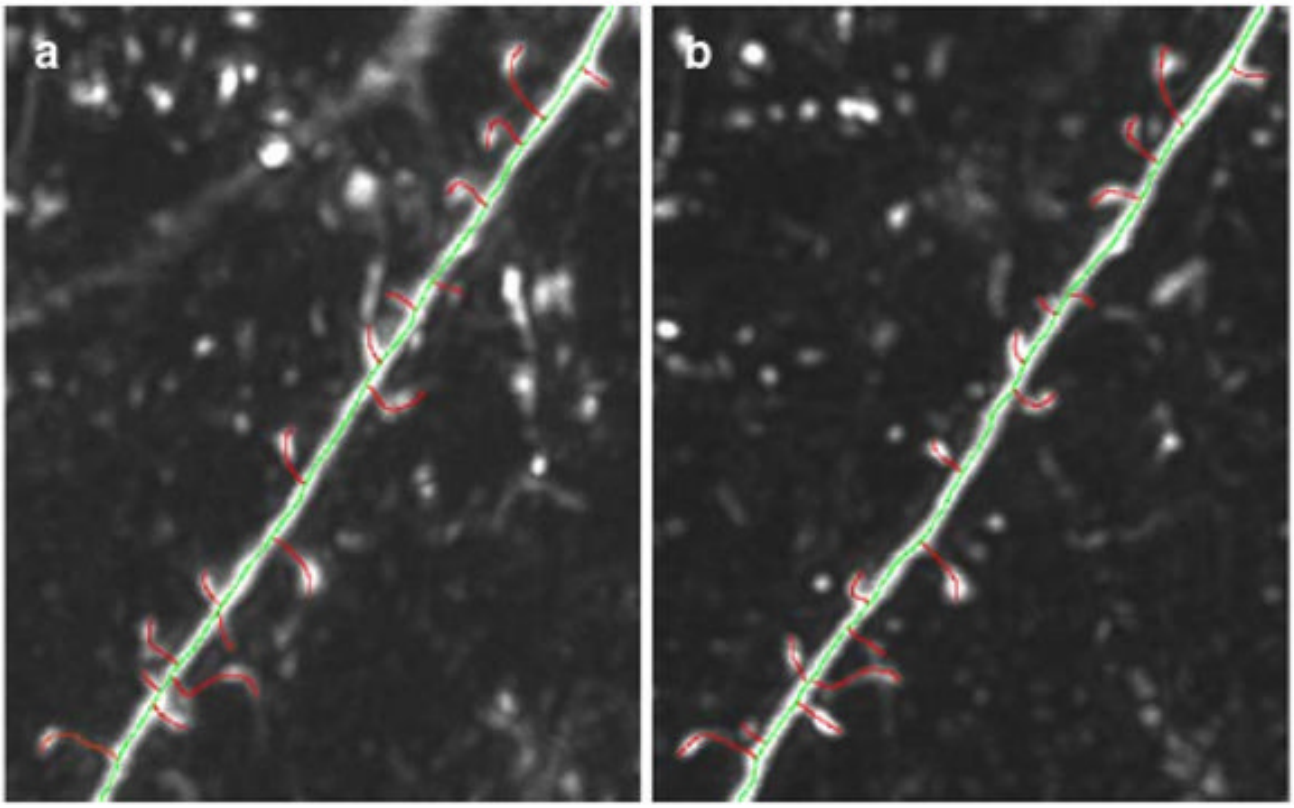

Fig. 12.
Manually labeled results for two time points in data set R10XB

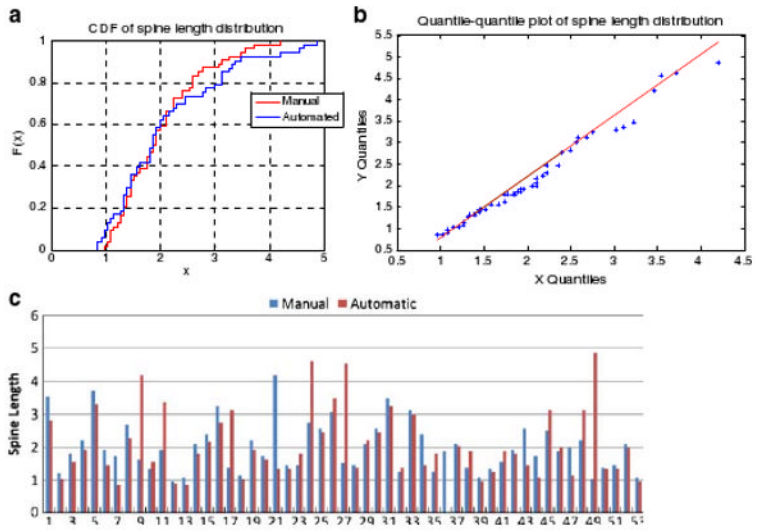

Fig. 13.

a CDF of spine length distribution; **b** quantile–quantile plot of spine length; **c** spine length comparison between manual and automatic results

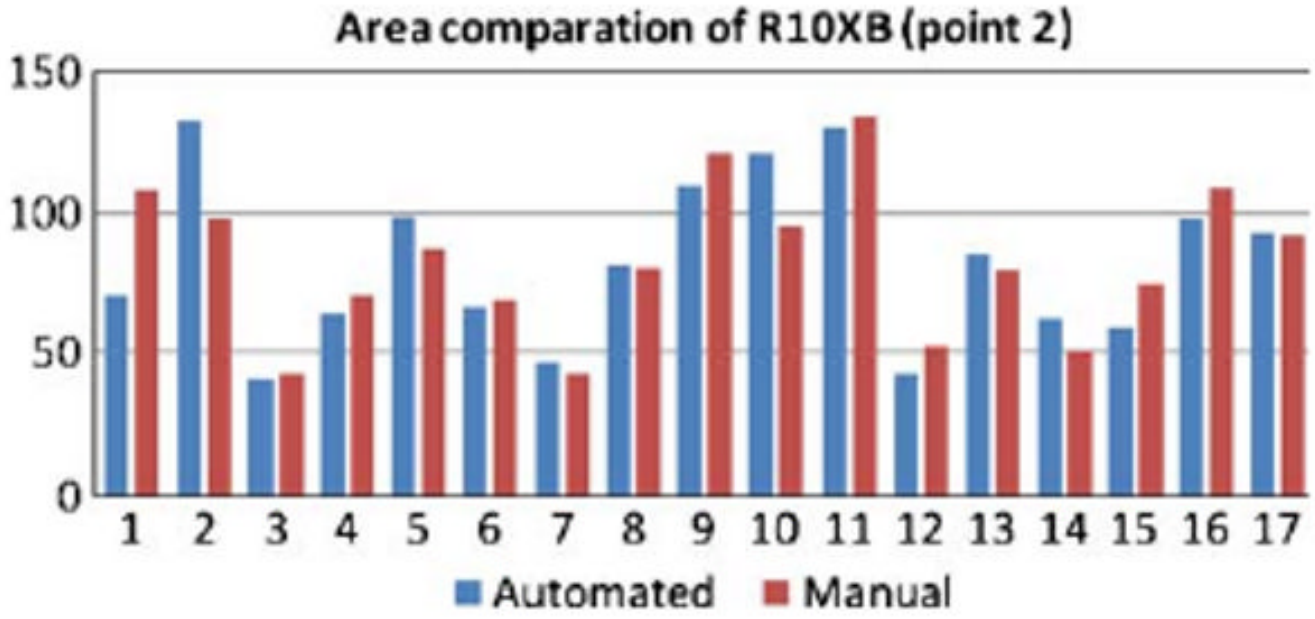

Fig. 14.
Area comparison between automated and manual method on data R10XB, time point 2

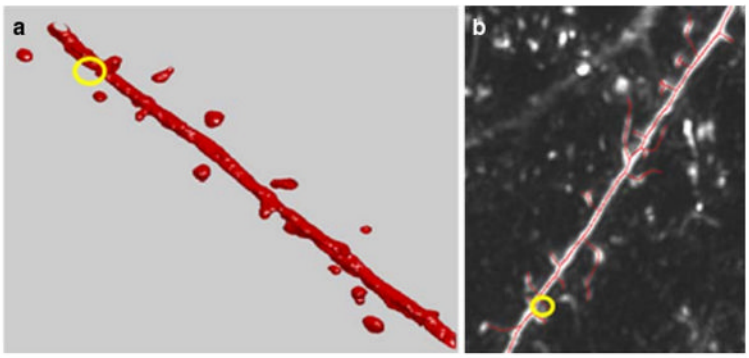

Fig. 15.

a A Rotated view of LoG 3D level set segmentation of R10XB time point 2; **b** image R10XB time point 2 overlapped with medial axis. The *yellow circles* indicate the corresponding position in the two images

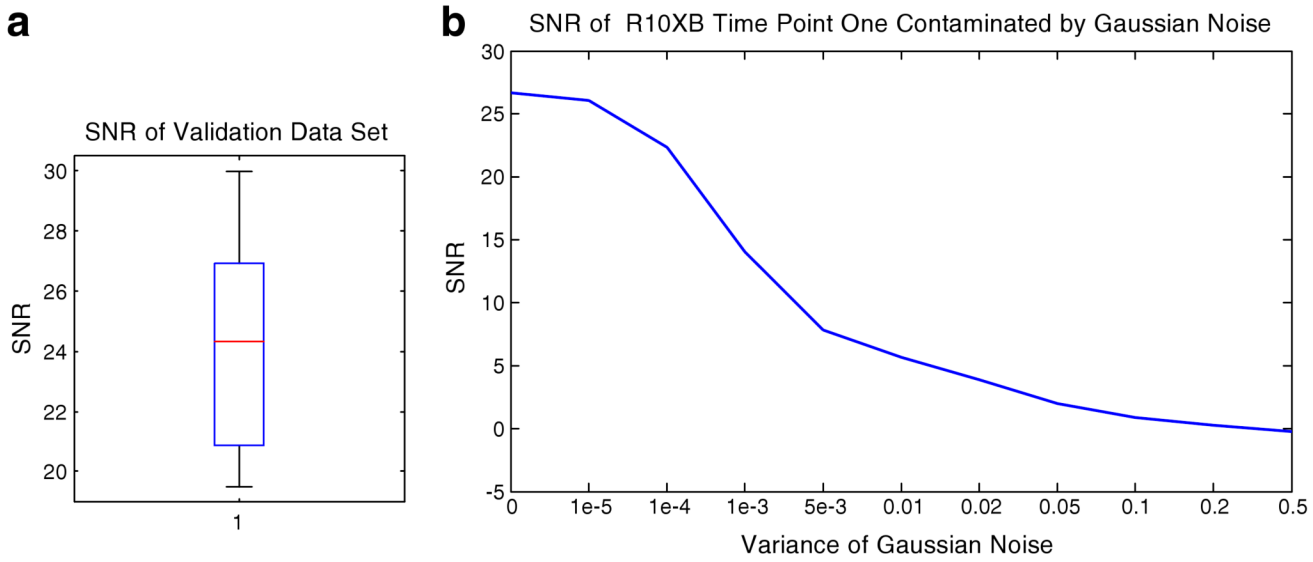

Fig. 16.

a Box plot of SNR of the validation data set; **b** SNR changes with additive white noise with different variance

Table 1

Comparison of spine detection results between manual and the proposed automatic method

Image name	Spine number (manual)	Spine number (our method)		
		Total number	False positive (wrong detection)	False negative (missing)
R10XAT1	6	7	1	0
R10XAT2	6	6	0	0
R10XBT1	17	20	3	0
R10XBT2	17	18	2	1
R10XDT1	10	10	0	0
R10XDT2	10	12	3	1
L10XBT1	20	21	2	1
L10XBT2	21	21	1	1
Total	107	115	12	4

Table 2

Comparison of tracking results between the manual and the proposed automated method

Data sets names	Spines number (manual)			Spine number (our method)		
	Matched	Elimination	Formation	Matched	Elimination	Formation
R10XA	6	0	0	5	2	1
R10XB	16	1	1	14	6	4
R10XD	10	0	0	8	2	4
L10XA	19	1	2	16	5	5
Total	51	2	3	43	15	14

Table 3

Parameter settings used in our method

Parameter	Typical Value	Description	Parameter	Typical Value	Description
Upper/lower threshold	3/1	Curvilinear structure detection	Lambda1/ Lambda2	3/3	aLBF
Linking distance/ Line length	5/5	Curvilinear structure detection	Maximal value of sigma	20	aLBF
Minimum backbone length	80	Curvilinear structure detection	Starting sigma	10	aLBF
Vector inner product difference	0.1	Spine Linking			