Axon tracking in serial block-face scanning electron microscopy

Elizabeth Jurrus a,*, Melissa Hardy b, Tolga Tasdizen a, P. Thomas Fletcher a, Pavel Koshevoy a, Chi-Bin Chien b, Winfried Denk c, Ross Whitaker a

a Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT, United States
b Department of Neurobiology and Anatomy, University of Utah, Salt Lake City, UT, United States
c Max Planck Institute for Medical Research, Heidelberg, Germany

A R T I C L E   I N   F O
Article history:
Received 16 April 2007
Received in revised form 24 April 2008
Accepted 19 May 2008
Available online xxxx

Keywords:
Image processing
Tracking
Kalman filtering
Active contours
Scanning electron microscopy
Neurobiology

A B S T R A C T
Electron microscopy is an important modality for the analysis of neuronal structures in neurobiology. We address the problem of tracking axons across large distances in volumes acquired by serial block-face scanning electron microscopy (SBFSEM). Tracking, for this application, is defined as the segmentation of an axon that spans a volume using similar features between slices. This is a challenging problem due to the small cross-sectional size of axons and the low signal-to-noise ratio in our SBFSEM images. A carefully engineered algorithm using Kalman-snakes and optical flow computation is presented. Axon tracking is initialized with user clicks or automatically using the watershed segmentation algorithm, which identifies axon centers. Multiple axons are tracked from slice to slice through a volume, updating the positions and velocities in the model and providing constraints to maintain smoothness between slices. Validation results indicate that this algorithm can significantly speed up the task of manual axon tracking.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction
The answers to many biological questions depend on a better understanding of cellular ultrastructure, and microscopic imaging is providing new possibilities for exploring these questions. For instance, an important problem in neurobiology is deciphering the patterns of neuronal connections that govern neural computation and ultimately behavior. However, relatively little is known about the physical organization and connectivities of neurons at the cellular level.

Medical imaging modalities such as MRI provide three-dimensional (3D) measurements of the brain with resolutions on the order of 1 mm (Xiao et al., 2003). This resolution provides macroscopic information about brain organization, but does not allow analysis of individual neurons. Scanning confocal (Minsky, 1961) and two-photon (Denk et al., 1990) light microscopy have several advantages, including the ability to visualize live specimens, but are limited to 200 nm lateral resolution and 500 nm z resolution, which are insufficient to reconstruct connections of individual neurons. Newer light microscopic methods such as 4Pi, STORM, and PALM (Egner and Hell, 2005; Rust et al., 2006; Betzig et al., 2006) promise higher resolution, but 4Pi still cannot resolve closely-bundled axons, while STORM and PALM, at present, are 2D methods requiring very long imaging times. Thus, electron microscopy remains the primary tool for resolving the 3D structure and connectivity of neurons. A number of researchers have undertaken extensive imaging projects in order to create detailed maps of neuronal structure (Fiala et al., 2002) and connectivity (Dacheux et al., 2003; White et al., 1986). At 20 nm resolution, the number of voxels needed to cover a volume sufficient to contain complete dendritic trees is about $10^{12}$ (Denk and Horstmann, 2004), which is beyond any prospect of manual reconstruction. The reconstruction of neural connectivity thus requires better tools for the automated analysis of such large data sets.

A new and promising technique for imaging large arrays of cells at nanometer resolution is serial block-face scanning electron microscopy (SBFSEM) (Denk and Horstmann, 2004), shown in Fig. 1a. In SBFSEM, successive slices are cut away and discarded, and the electron beam is scanned over the remaining block face to produce electron backscattering images. An example image is shown in Fig. 2b. SBFSEM imaging has several advantages over other electron microscopic methods for the analysis of long axonal processes. For instance, because the dimensions of the solid block remain stable after slicing, SBFSEM images have smaller deformations than serial-section transmission electron microscopy (TEM). The resolution and signal-to-noise properties of SBFSEM are generally not as good as those of TEM, but they are sufficient for manual tracking of individual axon paths. Furthermore, unlike TEM, SBFSEM images do not require registration. While 3D data sets can

Please cite this article in press as: Jurrus, E. et al., Axon tracking in serial block-face scanning electron microscopy, Med. Image Anal. (2008), doi:10.1016/j.media.2008.05.002
This paper addresses the problem of automatically tracking individual axons in SBFSEM data sets, specifically for the analysis of the optic tract in the embryonic zebrafish. Tracking individual axons is an essential step in analyzing the different organizations of the optic tract in wildtype and mutants (Lee et al., 2004). While, more generally, neurons are composed of dendrites, a cell body, synapses and an axon, here we focus on tracking axons, which are generally more difficult to track than dendrites because of their greater length and smaller diameter.

SBFSEM data presents several challenges for segmentation. Mainly, the axonal cross-sections (see Fig. 2a) are barely discernible by eye, and yet a large number of axons are tightly packed in the optic tract. In addition, the actual axon membranes are difficult to identify by intensity alone as evident in the histogram in Fig. 2b of 20 axons, a very small subset of the data. Also challenging is that the data acquired with SBFSEM does not have isotropic resolution: the out-of-plane resolution is significantly less (50 nm) than the lateral resolution of the slices (26 nm) making segmentation in three dimensions difficult. However, the block is oriented so that the imaged surface is nearly perpendicular to the axon axis so that cross-sections are cut through elongated processes. A single axon will traverse thousands of slices, slowly winding its way around other axons. Axons will rarely branch or terminate, which aids in segmentation.

These SBFSEM data sets of the optic tract present, in some sense, a two-and-a-half-dimensional data processing problem. Thus, the proposed method approaches the problem of segmenting axons from electron microscopy images as a 2D segmentation problem combined with a tracking problem in the third dimension. This avoids the much more difficult full 3D problem of finding thin processes in noisy data amidst a dense packing of similar processes. This also allows an effective interface for user input. When the algorithm fails, the user can, in principle, correct the segmentation and continue tracking on a slice by slice basis. Completed axon pathways can also be viewed in two- and three-dimensional plots.

There is some related work in the literature that applies computer vision and object tracking to medical data. For instance, Vazquez et al. introduced a semi-automatic, differential geometric method for segmenting neurons in two-dimensional EM images (Vazquez et al., 1998). In their method, a user initializes points on the boundary of the neuron, then a minimal length geodesic criterion is used to complete the boundary. Bertalmio et al. propose a slice-to-slice tracking/segmentation approach for electron microscopy images that uses two-dimensional deformable curve models (Bertalmio et al., 1998). This method is similar to ours; however, tracking is not explicit, but is achieved indirectly with coupled partial differential equations. Furthermore, the tracked structures span a much smaller number of slices than axons. Researchers have also proposed segmentation methods for confocal microscopy images (Holmes et al., 2002; Dima et al., 2002; De Solrzano et al., 1999; Wang et al., 2003). Curvilinear structure detection has been studied in various applications, such as detection of blood vessels in magnetic resonance angiography data (Sato et al., 2000; Lorigo et al., 2000). Three-dimensional deformable models for segmentation of tubular objects have also been proven to be effective (Pinho et al., 2007; Behrens et al., 2003; Feng et al., 2004). These methods are tailored to the resolution and specific properties of their application domains and do not readily extend to tracking axons in electron microscopy images.

2. Methods

The field of computer vision provides numerous methods for tracking features through a set of images. By treating the 3D volume as a sequence of 2D images in time, feature tracking methods...
ods can be applied to the volume. This tracking application is built on the Kalman-snakes framework (Terzopoulos and Szeliski, 1992; Peterfreund, 1999). The processing pipeline begins with a denoising of the input volume. Next, using an active contour model, initial contours are computed in the first 2D slice at user defined or automatically detected locations using a watershed filter (Beucher and Meyer, 1992; Ibanez et al., 2003). A simple smoothing constraint is used in the fitting of the contour to the axon, maintaining the shape for each slice. Each successive contour is then tracked through the 3D volume using a Kalman filter that predicts axon locations in upcoming slices. Each contour contains a series of points with a position in the image, and a velocity, which is the direction a point moves between slices. These points are weighted according to the strength of the underlying data to produce a new axon location at each slice. When tracking is completed, users can scroll through slices in the volume, inspecting the tracking for errors and re-initializing the tracking if necessary.

2.1. Image preprocessing

The SBFSEM data set used in the experiments has resolution $26 \times 26 \times 50$ nm per voxel and has a relatively poor signal-to-noise ratio, partly due to nonoptimized specimen preparation. Given this resolution, and orienting the block such that the main axon bundles are roughly perpendicular to the imaging plane, axons range from four to six pixels in width in each 2D slice. Tracking axon bundles are roughly perpendicular to the imaging plane, even at this resolution, and orienting the block such that the main breast enhancement and reduces noise and enhances structure by reducing randomness. In this sense UINTA is particularly well suited for the highly repetitive (texture-like) structure of the block-face images of the optic tract. There are faster image denoising methods that predict axon locations in upcoming slices. Each contour is represented by a series of contours which consist of a set of points. Each point is associated with its own Kalman Filter that updates the state, $w_k = [x_k, y_k, u_k, v_k]^T$, where $[u_k, v_k]$ is the velocity at contour position $[x_k, y_k]$. Every iteration of the Kalman filter consists of three computations: a prediction, $\hat{w}_k$, measurement, $z_k$, and correction, $w_k$. A linear update using the previous state estimate, $w_{k-1}$, gives the prediction state, $\hat{w}_k = A w_{k-1}$.

2.2. Kalman filter based axon tracking

Axons have a tendency to “drift” at a slowly changing velocity through the image stack. They also change shape between sections despite the near perpendicular arrangement of the cells to the cutting plane. For this reason, we implement a tracking algorithm that takes into account the slowly varying velocity and change in shape to predict the location of the axon in each slice. The framework in the Kalman filter allows us to follow an axon through several slices, with simple updates to position and velocity estimates.

2.2.1. Kalman filter

Kalman filtering (Blake et al., 1995) provides a feedback control loop for predicting the location of the axon at each slice, sampling the image, and correcting the estimate. Each axon is represented by a series of contours which consist of a set of points. Each point is associated with its own Kalman Filter that updates the state, $w_k = [x_k, y_k, u_k, v_k]^T$, where $[u_k, v_k]$ is the velocity at contour position $[x_k, y_k]$. Every iteration of the Kalman filter consists of three computations: a prediction, $\hat{w}_k$, measurement, $z_k$, and correction, $w_k$. A linear update using the previous state estimate, $w_{k-1}$, gives the prediction state, $\hat{w}_k = A w_{k-1}$.

This prediction assumes constant velocity and adds the current velocity to the current position to predict the next position. The measurement state, $z_k$, is a combination of positions from active contour measurements (see Section 2.2.2) and velocities from optical flow (see Section 2.2.3). The filter combines the predicted and measured state to produce the corrected state estimate, $w_k = \hat{w}_k + K_k (z_k - H \hat{w}_k)$, where $K_k$ is the Kalman gain matrix, given by $K_k = P_k H^T (H P_k H^T + R)^{-1}$, and $P_k$ is the a posteriori error covariance of the current state estimate, given by $P_k = A P_k \cdot A^T Q$. The Kalman gain matrix blends the measurement and predicted states so as to minimize $P_k$. After each estimate, $P_k$ is updated by $P_k = (I - K_k H) P_k$.  

Fig. 3. (a) A portion of an SBFSEM image and (b) after denoising.
H defines the relationship between the measurement and the model. For this model, H is the identity, while Q and R are $4 \times 4$ diagonal matrices defining the process and measurement noise covariance. We assume the covariance process noise, represented by $Q$, is constant. However, we can model the measurement noise at a contour point using a membrane strength metric. The strength of a membrane can be defined as the second derivative in the direction perpendicular to the membrane:

$$
\mu = \frac{d^2}{d\ell^2} I',
$$

where

$$
I' = I \circ B_3.
$$

$I'$ is the intensity along the vector $\vec{n}$, normal to the contour point, $B_3$ is a box filter used to smooth any remaining noise and is oriented along the vector $\vec{d}$, perpendicular to $\vec{n}$. As the membrane strength approaches zero, the Kalman gain matrix will favor the input from $z_k$ more strongly in calculating $w_k$. If the membrane strength is large (closer to one), the Kalman gain will favor $w_k$ more. To scale $\mu$ to a range between zero and one, we calculate,

$$
\mu' = \exp \left( -\frac{\mu^2}{c} \right),
$$

where $c$ is a constant representing the value of a strong edge weight. In order to maintain continuity between the weights of neighboring contour points, preventing jagged contours, we smooth the weights across the sequence of points using a 1D Gaussian filter. This maintains a smooth transition between points on the contour.

This system, with input from the positional and velocity measurements, provides a set of steps for predicting and finding the location of axons at each slice in the volume.

### 2.2.2. Positional measurement – active contour models

Active contour models (Kass et al., 1988), or snakes, are often used in image segmentation and feature tracking (Terzopoulos and Szeliski, 1992; Peterfreund, 1999). Provided some user input or initialization, active contour models can lock onto and identify local features in an image. The Kalman filter uses the contour control points, $[c_k, v_k]$, as part of its state model (described in Section 2.2.1).

There are two main energies, $E_{\text{int}}$ and $E_{\text{image}}$, that control the placement of the snake:

$$
E_{\text{snake}} = \int_{s=0}^{1} W(s) (E_{\text{int}}(v(s)) + E_{\text{image}}(v(s))) \, ds.
$$

The internal snake energy,

$$
E_{\text{int}} = \alpha \|v(s)\|^2 + \beta \|v_n(s)\|^2,
$$

serves as a smoothness constraint. $v(s)$, $v_n(s)$, and $v_m(s)$ are the parameterized contour model and its first and second derivatives with respect to arclength, respectively. $E_{\text{int}}$ uses $\alpha$ and $\beta$ to control how elastic and stiff the final snake will be with respect to the surrounding data points. This maintains the circular shape of the axon as it may change in size between slices.

$E_{\text{image}}$ is computed by sub-sampling the image along a ray $\vec{R}$, as shown in Fig. 4a. An axon edge is defined to be along

$$
\vec{R} = P_t + t\vec{v}.
$$

$C$ is the center of the axon and $t$ is the sampling interval along $\vec{v}$, the normalized vector from $C$ to $P_t$. An edge is defined to be at the maximum of $\frac{d}{ds} I_{\text{edge}}$ on the interval $[-m, m]$, where $m$ is the size of the axon membrane and

$$
I_{\text{edge}}(t) = I( R(t) ) + B_y.
$$

$B_y$ is a box filter operating over the vector perpendicular to $\vec{v}$. This allows for contribution from neighboring pixels and smoothing of any remaining noise. The external image energy,

$$
E_{\text{image}} = \left| \frac{d}{dt} x \right|^2
$$

represents the edge information needed to fit the snake to the axon membranes. Finally, the strength of the edge, $w(s)$, is used as a weight to constrain the contour more tightly to points with strong edges.

The set of contours used to define the axon through the volume is found using an iterative sampling process driven, in part, by the Kalman filter. The Kalman filter provides an initial set of predicted contour points, as in Fig. 4b. The contour location constraints provided by the Kalman filter enables contours to maintain their shape and location even when the data in a particular slice is not sufficient for axon detection.

This sampling method prevents self-intersecting contours through the use of non-overlapping rays. The algorithm settles on a fit that minimizes the energy function $E_{\text{snake}}$. The Kalman filter uses the positions on the contour and the strength of the edge at those points to compute the final contour.

---

**Fig. 4.** (a) The contour is refined by iteratively sampling along the rays and recomputing the new edge location. C is the center of the axon. (b) The final location of the contour is dependent on a weighted combination of the predicted contour, $a$, and measured contour, $b$. The corrected estimate is the final contour $c$. New control points for each slice are repeatedly sampled, computing new contours until they converge.
Axon tracking is initialized with a user defined point at the approximate center of the axon. The area immediately around this point is sampled for edges and the Kalman filter is initialized. The algorithm continues to track the axon in each slice, iteratively sampling the image data and updating the state estimate in the Kalman filter.

2.2.3. Velocity measurement – optical flow

The shape and size of axonal cross-sections remain relatively constant as we move from one slice to the next, but the position of the image will change unless an axon runs exactly perpendicular to the imaging plane. The change in position is proportional to the angle between the axonal axis and the imaging plane normal. This is used as the velocity component, \[ |u| = |v|, \]

where \[ |u| \] and \[ |v| \] are the magnitudes of the velocity components in the \[ x \] and \[ z \] directions, respectively. Due to the alignment of the imaging plane perpendicular to the main running direction of the optic tract, individual axons are never parallel to the imaging plane; hence, the division by \[ e_{1,z} \] does not pose a practical problem.

2.3. Multiple axon initialization

For tracking initialization, users select individual axons with a single click, marking the center of the axon they want to track. Selecting multiple axons with this method can be time consuming. For this reason, we use a watershed filter to automatically segment and select axons. The user selects a point in the data and all axons within an \( n \times n \) area of the click will be identified for tracking.

The watershed algorithm treats the image intensities as a height function, so that high intensities correspond to boundaries. The boundaries form regions in the image; water poured from above would tend to pool in those regions, creating segmentations. Each image is thresholded as a percentage of the maximum depth to remove shallow regions and help prevent over-segmentation. Then, using a top-down steepest descent algorithm, regions are segmented by following each maximum pixel to its local minimum. The top-down approach makes access to different levels of the segmentation straightforward, allowing users to customize their segmentation.

For this axon tracking application, we invert the region image so the high intensities represent axon boundaries and apply an edge preserving anisotropic diffusion filter to smooth out any remaining noise. The watershed filter threshold is set to 20% and the user is allowed to choose the depth of the segmentation, allowing for an optimal distribution of axon initialization points.

3. Results

Manual tracking of axons through a volume is tedious, requiring hours of careful labeling and correction, while automatic tracking allows for much faster annotation of axon locations. To demonstrate, results on the reliability and expected tracking distance of a series of axons tracked through a \( 900 \times 500 \times 500 \) voxel volume are presented. Fig. 5 shows three different slices through the volume. The closed curves are automatically detected contours and the points are tracking annotations by an expert. In this section, we also demonstrate how large selections of axons can be tracked using a watershed initialization approach. Finally, a three-dimensional rendering using the contours generated in the volume (Fig. 11) is examined.

Detecting membranes automatically requires parameters tuned for small distances and changes in the data. For this reason, the

---

Fig. 5. Sequence of 56 axons tracked through 21 slices in a volume. Images are at slice 1, 11, and 21. Points inside the contour mark axons that are tracking correctly. Points not inside an axon contour are those for which tracking failed.

Please cite this article in press as: Jurrus, E. et al., Axon tracking in serial block-face scanning electron microscopy, Med. Image Anal. (2008), doi:10.1016/j.media.2008.05.002
length of \( n \) in Eq. (8) is 4.0 and \( d \) in Eq. (9) is 2.5. This represents the number of pixels the algorithm will use to compute the best location of the membrane. In addition, \( c \), in Eq. (8) equals 4.0. The noise model, \( Q \), in Eq. (5) is set to 0.4. The active contour model uses \( \alpha \) and \( \beta \) parameters which are set to 0.4 and 2.0, respectively. Finally, when we sample the ray \( R \) in Eq. (13), \( r \) equals 0.1 and \( m \) equals 2.5.

The computational costs of axon tracking are relatively low. Image denoising takes approximately 20 minutes per slice and the computation of structure tensors takes 1 minute per slice on a standard desktop PC. These steps are the computational bottlenecks; however, both can be computed offline. Tracking 56 axons takes about 10 seconds per slice. Alternatively, the tensor calculation can be performed locally while tracking, taking up to 5 seconds more per slice, depending on the size of the image buffer used. In comparison, the time it took for an expert to track the same 56 axons over 500 slices is approximately 14 hours, averaging 1.7 minutes per slice. The automated method is 10 times faster if the tensors are precomputed and 7 times faster if the tensor calculation is buffered.

We performed several validation experiments using human expert segmentations as ground truth. Fifty six axons were tracked through 500 slices by a human expert to provide ground truth; additionally, another three axons were tracked through 700 slices. The human expert placed markers at the pixel locations closest to the centers of the 56 tracked axons in each of the 500 slices and the centers of the three axons in the 700 slices. The expert was provided with a graphical user interface which allowed her to place colored markers and scroll through the slices. She was asked to use a unique color for each axon; hence, the markers are uniquely identifiable with these axons. To quantitatively assess the performance of the algorithm several metrics are defined. Let \( m_k(i) \) denote the position of the marker for the \( i \)th axon at the \( k \)th slice.

**Definition 1.** The segmentation for axon \( i \) at slice \( k \) is defined as correct if \( m_k(i) \) falls inside the region defined by the final contour for axon \( i \) at slice \( k \) given by the segmentation algorithm; otherwise, it is defined as incorrect.

In the 700 slice volume, the human expert selected axons that were visually easy to track. Our algorithm tracked one axon through 608 slices, and the other two through 657 slices. In the 500 slice/56 axon volume, a more diverse population of axons was used, including many that were visually more challenging to track. Fig. 6 plots the number of correct axon segmentations, according to the above definition, as a function of the slice number.

As expected, the number of correct segmentations starts at 56, and declines as the slice number grows. A less intuitive observation is that this number does not decrease monotonically but can also increase. However, this observation fits well with the expectations of the algorithm. It is expected that segmentations that miss the axons they are tracking due to bad data slices will recover to the correct segmentation, due to the correction by the Kalman filter, as long as the number of consecutive bad slices is not too large. Due to this robustness, it can be more meaningful to ignore intermediate errors from which the segmentation recovers in assessing the performance of the algorithm. The following definition addresses this property.

**Metric A:** The segmentation for axon \( i \) is defined to have failed at slice \( n \) if, for all slice numbers larger than or equal to \( n \), the segmentation for axon \( i \) according to **Definition 1** is incorrect.

It can be argued that **Metric A** is overly optimistic: if an axon segmentation recovers after a large number of consecutive failed slices, is the recovery due to the Kalman filter, or due to chance? To address this question another definition of “last correctly tracked slice number” can be made.

**Metric B:** The segmentation for axon \( i \) is defined to have failed at slice \( n \) if, for all slices in the range \([n, n + k - 1]\), the segmentation for axon \( i \) according to **Definition 1** is incorrect.

We chose \( k = 10 \) for the above definition in this paper. Fig. 7 compares the two metrics by plotting the number of correctly tracked axons as a function of the slice number. Notice that the performance reflected by **Metric B** is lower than **Metric A**, as expected. The curve for **Metric A** demonstrates an approximately linear decline whereas the curve for **Metric B** appears roughly exponential. Using these curves, the expected number of slices after which a certain fraction of axons will be mistracked can be computed. For instance, according to **Metric A** approximately 90% and 50% of the axons will still be correctly tracked after 30 and 250 slices, respectively.

Most failures occur when an axon disappears from view for too many slices. Fig. 8 shows how the tracking fails when an edge appears in the middle of a feature separating the tracked axon from its actual path. It is also not unusual for a tracked axon to “latch onto” a neighboring axon and then find its way back to the correct axon within 3 or 4 slices. Parameters affecting the tracking include
the $\alpha$ and $\beta$ terms of the active contour and the measurement and noise covariance values in the $Q$ and $R$ matrices of the Kalman filter. Small changes to the Gaussian standard deviations in the structure tensor computation do not have an effect. Computing membrane weights for the positional measurement, however, provides a better fit for the data.

Fig. 9 is a close-up view of the axon contours with their respective predicted and measured contours. Images (a) and (b) are examples of how a weighted noise covariance in the Kalman filter helps the contour stay on track of the correct axon when the measured axon has a weak membrane. The Kalman filter used on the points of the contour in image (a) is weighting the measured (red) and predicted (yellow) contour more evenly to produce the corrected (blue) contour. In doing this, the contour misses the actual location of the contour, as indicated by the blue point just outside of the corrected contour in image (a). In contrast, the corrected contour in image (b) conforms more to the predicted contour, maintaining the correct position. Images (c) and (d) have similar outcomes except the membranes are much stronger causing the filter to fit more closely to the measured contour. In this case, the edge strength of the measured (red) contour is very strong, forcing the final corrected (blue) contours to fit the measured data.

In order to speed up the initialization process, a watershed filter is used to automatically find axon centers, allowing many axons to be initialized at once. Fig. 10 shows the results of this initialization compared to the axon centers identified by the expert. In some cases the axon initialization is very close to the expert's initialization, while in other cases, the watershed segmentation places a boundary where an axon center should lie. Poorly initialized axons can be easily identified by the algorithm during tracking and removed. The user can adjust the level of the watershed to find the best fit and reinitialize axons when the watershed initialization fails.

Examine the complex 3D nature of the data is possible with 3D renderings of the tracked axons. A 3D representation is formed by connecting contours from each traced axon. Fig. 11 shows a three-dimensional rendering of axons from the 900 $\times$ 500 $\times$ 500 volume and Fig. 12 shows similar tracking results for two different sets of axons in the same volume. All axons were compared against expert tracking, including Fig. 11a, which required manual reinitialization.

4. Conclusions

The described system can successfully track axons through a series of slices in a volume. Given the noisy nature of the data and the small axon sizes, denoising provides a cleaner view of the data in which to perform tracking. Tracking from slice to slice is made possible with the Kalman filter, which predicts and corrects the placement of the axon in each slice using optical flow and active contours as velocity and position estimates. Initialization and correction of the algorithm is performed with user
interaction. We have also presented a validation study that effectively tracked multiple axons through slices of an SBFSEM volume. Future work will include using the entire set of contour points as a single state vector in the Kalman Filter, accounting for the entire contour rather than individual points on the contour. It is worth exploring other methods, such as particle filters (Smal et al., 2007), which track multiple axons with similar displacements and have the potential of more accurately estimating axons’ locations in larger volumes. We also want to validate our tracking with more axons through larger volumes (of at least 1000 slices), and compare the axon organization of wildtypes with mutants. In addition, more advanced volume visualization methods are being developed to more easily examine the contours within the data, helping to detect errors in tracking and restart tracking at those locations.

The software to aid in the image processing was written using the Insight Segmentation and Registration Toolkit (ITK) (ITK, XXX). The 3D axon renderings were created using the Visualization Toolkit (VTK) (VTK, XXX).

Acknowledgements

This work was supported in part by NIH RO1 EB005832, NIH RO1 EY12873, and the Max-Planck Society.

References