CHAPTER 25

Automated Identification of Microtubules in Cellular Electron Tomography

Daniyar Nurgaliev*[,1], Timur Gatanov*[,1], *and* Daniel J. Needleman†
*Department of Physics, Harvard University, Cambridge, Massachusetts 01238

†Department of Molecular and Cellular Biology, School of Engineering and Applied Sciences, Center for Systems Biology, Harvard University, Cambridge, Massachusetts 01238

Abstract

We describe a method for automatically finding the location and conformations of microtubules in tomograms of high-pressure frozen, freeze substituted cells. Our approach uses two steps: a preprocessing step that finds locations in the tomograms that are likely to lie inside microtubules and a tracking step that uses the preprocessed data to identify the trajectories of individual microtubules. We test this method on a

[1]Daniyar Nurgaliev and Timur Gatanov have contributed equally to this work

reconstruction of a *Caenorhabditis elegans* mitotic spindle and we compare our results with those obtained by a human expert who manually segmented the same data. At present, the method could be used to assist the analysis of large-scale tomography reconstructions. With further improvements, it may be possible to reliably segment cellular tomograms without human intervention.

I. Introduction

Electron microscopy and tomography are increasingly used to study larger and larger systems, including organelles (Marsh, 2005), whole cells (Hoog *et al.*, 2007), and perhaps, in the near future, complete organs (Briggman and Denk, 2006). This work has the potential to transform our understanding of cell and developmental biology by (1) bridging the gap in length scale between the structure of protein complexes—obtained from X-ray, NMR, and electron microscopy—and the structure of entire cells—obtained by light microscopy; (2) allowing massive data sets to be studied so the statistical significance of trends can be analyzed; and (3) enabling investigations of the spatial distribution of features which are too small to resolve by light microscopy.

There has been remarkable progress in sample preparation, data acquisition, and assembling reconstructions for these large-scale electron microscopy studies (Briggman and Denk, 2006; Hoenger and McIntosh, 2009), but the development of methods for analyzing the resulting data have lagged behind. In cellular tomography, most image segmentation—the identification of microtubules, membranes, and other features of interests—is still performed manually. While this approach can be highly accurate and has been very successful to date, the segmentation step is often the most time-consuming part of such work. As larger systems are investigated, purely manual segmentation will no longer be practical. For example, it currently takes a skilled human a few hours to track all of the microtubules in a tomographic reconstruction of a small spindle, such as from yeast. Spindles in human cells have a volume approximately 1000 times larger than spindle from yeast cells, so at this rate, it would take an expert a good part of a year to segment. Spindles from Xenopus egg extracts, another popular model system, have a volume about 10 times larger than spindles in human tissue culture cells; thus the resulting analysis of these structures is expected to take 10 times longer still. Fully automated or computer-assisted approaches have the potential to greatly speed segmentation. There have been previous attempts to automatically segment tomograms of plastic-embedded samples, but this remains a challenging problem (Jiang *et al.*, 2006; Sandberg, 2007).

In this chapter, we describe our efforts to develop an automated method of segmenting microtubules in cellular tomograms of plastic-embedded, freeze substituted samples. We mention a number of unsuccessful approaches we tried; we present an algorithm which gives satisfactory results; we demonstrate the method on a tomographic reconstruction of a *C. elegans* mitotic spindle and compare the automated results with manual segmentation performed by a human expert; finally, we detail ways our method can be improved.

II. Overview

Tomography data is intrinsically three dimensional (3D), consisting of a matrix of voxels with a gray value at each location. Microtubules are low-contrast objects that appear as two thin dark lines (Fig. 1). The resolution of tomograms is asymmetric, being much higher in the *XY* plane than in *Z*, and microtubules are difficult to see if they are not parallel to the viewing axis. As a practice data set, we use a reconstruction of a *C. elegans* mitotic spindle from (O'Toole *et al.*, 2003). The data set is a dual-axis tomogram of two serial sections that have been pasted together. The final reconstruction is a $2880 \times 1024 \times 286$ array of voxels which are 1.7 nm per side and have an intensity value ranging from 0 to 255.

Our method for finding microtubules consists of two steps: First, we identify points that are likely to be part of a microtubule. In this step, which we call preprocessing, we transform the original image into a new image, where the intensity of each voxel is a measure of our confidence that a microtubule exists at that location. Second, we connect these points into lines. We call the algorithm which performs this second task a tracker, because its goal is to track the trajectory of individual microtubules.

Both the preprocessing step and the tracker make use of a number of parameters whose values must be selected. Optimizing the choice of parameters requires some method of measuring the quality of the results. Validation of image recognition tasks, like finding microtubules in tomograms, is a very difficult problem because we do not know what the "ground truth" is—where the microtubules really are. After all, if we

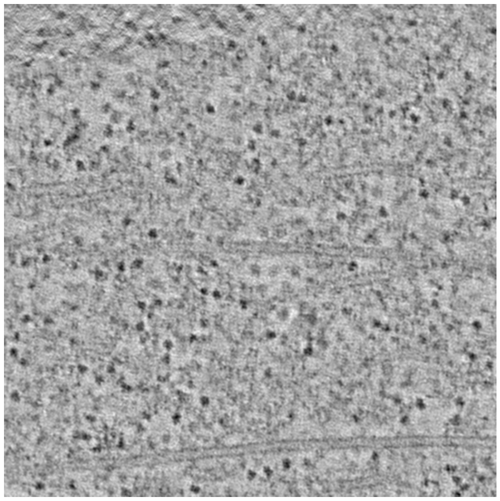

Fig. 1 A typical *XY* slice from a tomogram of a *C. elegans* spindle (O'Toole *et al.*, 2003). The dark long double lines are the walls of microtubules, the black dots are ribosomes, and the smooth region on the top left is a reconstruction artifact from the proximity of the physical edge of the sample. This region is 512×512 pixels. Each pixel is 1.7×1.7 nm.

already had an ideal way of knowing the location of all microtubules in the tomogram, we would use that as our method for finding microtubules instead of trying to develop a new algorithm. Our method of validation is to simply compare our results with those obtained manually by a human expert. We thus attempt to choose parameters that give results which are similar to those obtained by manually tracking.

We will next describe each of these steps in more detail: preprocessing, tracking, validation, and future directions.

III. Preprocessing: Finding Points in Microtubules

We have opted to use two–dimensional (2D) algorithms for our preprocessing step. The tracking algorithm then connects the preprocessed data into trajectories that follow the backbone of the microtubules in 3D. While a 3D processing step would make use of additional information, and thus would presumably be more reliable, we believe that there are a number of advantages associated with employing a simpler 2D preprocessing step. Most importantly, 2D images are much easier to work with and visualize, greatly aiding in testing and debugging the algorithm. Additionally, electron tomography has different resolution and noise structure in the *XY* and *Z* directions, so the data is already intrinsically anisotropic. Finally, 2D preprocessing is very easy to parallelize, as each *XY* slice can be analyzed by an independent computational node.

A 512×512 pixel region from an *XY* slice of a tomogram of a *C. elegans* mitotic spindle is shown in Fig. 1. The set of dark double lines are the walls of microtubules and the black circles are ribosomes. The smooth region in the top left corner is caused by errors in the reconstruction associated with joining together two tomograms from separate physical blocks. We have explored a number of different approaches for identifying which points in images like this are in microtubules. Many of our attempts failed before we developed a method we were satisfied with. We will first discuss various unsuccessful methods that we explored.

A. Unsuccessful Approaches

The walls of microtubules appear as two dark lines, so it is natural to attempt to find them by thresholding the image. Unfortunately, this simple approach produces so many false positives that it is not productive (Fig. 2). This method can be improved by using a local threshold—based on the difference between a pixel's intensity and the average intensity in its vicinity—or by first filtering out ribosomes. However, the performance is quite poor, even with these enhancements.

A common approach for finding features in an image is to use a convolution. A convolution provides a measure of how much a local region in the image matches a mask of interest. First, we need to generate a convolution mask—an average image of a microtubule. We have done this either empirically by averaging together images of microtubules found manually or by conjecturing a simple model for a microtubule, two dark lines spaced an appropriate amount. Both methods produce similar results. If we

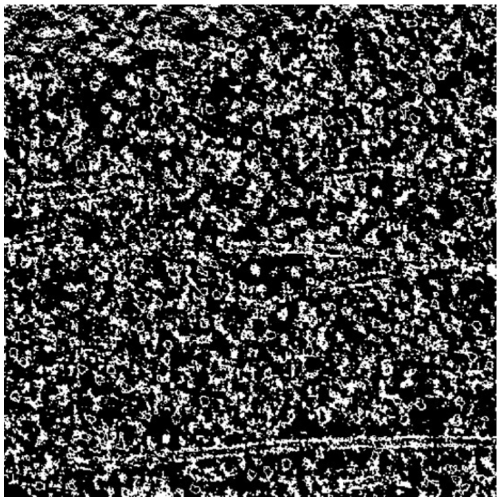

Fig. 2 Intensity thresholding. The region displayed in Fig. 1 was thresholded to highlight pixels with intensities between 100 and 150—the range present along the microtubule walls. While this procedure selects microtubule walls, there are also a very large number of false positives.

simply convolve the data with one of these masks, we are effectively looking for microtubules of a particular orientation. Microtubules in the image could be facing any direction, so we need a more flexible approach. We constructed 60 different masks by rotating one of the original masks in 3 degree increments. Each of these masks can be used to probe the image for a microtubule oriented at the corresponding angle. We created a new image by replacing the intensity of each pixel with the maximum value at the corresponding location obtained from the 60 different convolutions with the 60 different masks. By the logic outlined above, this procedure should provide an estimate of how well a particular region matches the profile of a microtubule, independent of its orientation. The resulting image is visually striking, but, unfortunately, it is not helpful for locating microtubules (Fig. 3). This simple algorithm fails because of the complexity and noise structure of the tomography slices.

An alternative method is to use more sophisticated approaches to search for lines in the image (Lindeberg, T. 1998). This will not be sufficient for finding microtubules because they are composed of pairs of lines, but it might be a reasonable starting point for a more involved procedure. A line is an object that is flat in one direction and has a local extremum in the perpendicular direction—dark lines, like those from the microtubules wall, will be a local minimum. These statements are claims about the derivatives of the image: a region in the image is a line if it has a zero first derivative and a large second derivative along one direction, and a small second derivative in the perpendicular direction.

Computing derivatives in images faces two challenges: (1) The data is not a continuous function, rather, it is composed of discrete pixels. (2) Directly taking differences

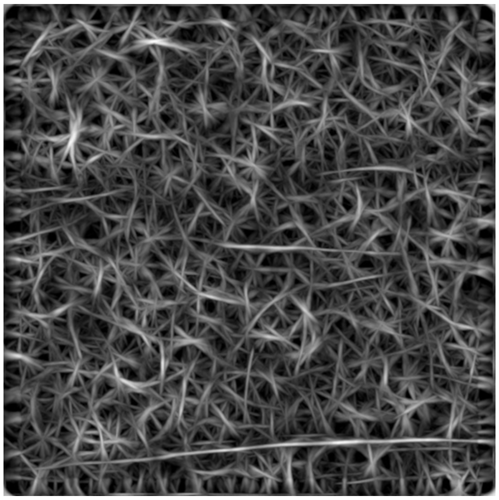

Fig. 3 Direction-independent convolution with a microtubule model. A mask to identify microtubules was empirically constructed by averaging the profile of many microtubules. The mask was oriented in 60 different directions and convolved with the region displayed in Fig. 1. Each pixel in this image corresponds to the maximum value at that location obtained from the different convolutions. Microtubules are visible as long bright lines, but the background structure is very strong and complex.

between neighboring pixels produces poor estimates of the desired derivatives because of the presence of noise. A convenient way to circumvent these difficulties is to first smooth the image by convolving it with a Gaussian kernel:

$$I_s(x,y) = \sum_{i,j} I_{ij} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x-i)^2 + (y-j)^2}{2\sigma^2}\right) = I \times G_\sigma \tag{1}$$

where I_{ij} is the original image, $I_s(x,y)$ is the smoothed image, and σ is the width of the Gaussian kernel. An advantage of this approach is that it easily allows one to search for features at a particular length scale, set by the width of the Gaussian. Each microtubule wall in an *XY* slice of the tomogram appears as a dark line with a width of approximately 3 pixels, so it is appropriate to convolve the image with a Gaussian of width $\sigma = 3$ pixels.

The derivative of this smoothed image can then be efficiently computed by convolving the original images with the derivatives of a Gaussian. The first derivatives consist of a vector at each location whose components are the derivatives in the x and y direction:

$$I'_s = [I'_{sx}, I'_{sy}] = [I \times G_{\sigma x}, I \times G_{\sigma y}] \tag{2}$$

$$G_{\sigma x} = \frac{1}{2\pi\sigma^2}\frac{-x}{\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{3}$$

$$G_{\sigma y} = \frac{1}{2\pi\sigma^2}\frac{-y}{\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{4}$$

Higher order derivatives can be calculated with additional convolutions. Thus, the second derivatives form a matrix, called the hessian matrix, at each location, which is computed through further convolutions:

$$I''_s = \begin{bmatrix} I''_{sxx} & I''_{sxy} \\ I''_{syx} & I''_{syy} \end{bmatrix} = [\begin{matrix} I * G_{\sigma xx} & I * G_{\sigma xy} \\ I * G_{\sigma yx} & I * G_{\sigma yy} \end{matrix}] \tag{5}$$

$$G_{\sigma xx} = \frac{1}{2\pi\sigma^2}\left(-\frac{1}{\sigma^2}+\frac{x^2}{\sigma^4}\right)\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{6}$$

$$G_{\sigma xx} = \frac{1}{2\pi\sigma^2}\left(-\frac{1}{\sigma^2}+\frac{y^2}{\sigma^4}\right)\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{7}$$

$$G_{\sigma yx} = \frac{1}{2\pi\sigma^2}\frac{xy}{\sigma^4}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{8}$$

Now points on lines can be identified as pixels where the intensity is flat in one direction—one of the eigenvalues of the hessian matrix has a small absolute magnitude—and the intensity has a local minimum in the other direction—the other eigenvalues of the hessian is large and positive, and the first derivative along that direction is zero. One complication is that, because the image is made of finite-size pixels, the first derivative will not be identically zero near the minimum. We thus need to determine if the first derivative is likely to pass through zero inside a pixel, which can be achieved by approximating the pixilated image as a continuous function. Consider the variation in intensity at a pixel along the x axis. This can be approximated as follows:

$$I_s(x,0) = I_s(0,0) + xI'_{sx}(0,0) + \frac{1}{2}x^2 I''_{sxx}(0,0) + O(x^3) \tag{9}$$

The first derivative of this function is zero at $x = -(I'_{sx}/I''_{sxx})$. If this value is less than 0.5 pixels, then the function has a local extreme inside that pixel. Furthermore, if I''_{sxx} is positive, this is a local minimum.

Running this algorithm to search for lines produces many false positives (Fig 4). The problem is that the described algorithm searches for any feature that locally looks like a line of width σ and ends out picking up linear structures present in the background.

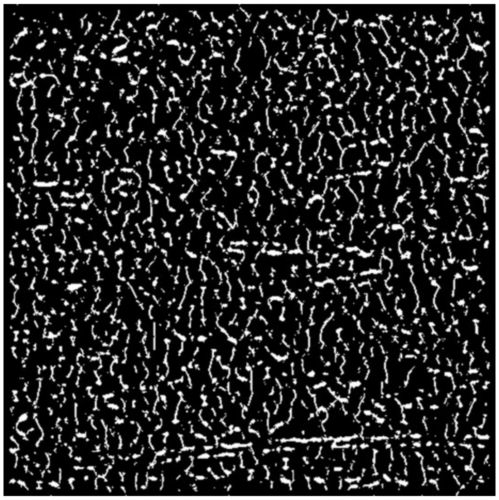

Fig. 4 Ridge detection. Locally linear features present in the region displayed in Fig. 1 where identified as described in the text. While the double lines of the microtubule walls are partially visible, there are a large number of false positives and false negatives.

B. Our Approach

The failure of the local ridge finding algorithm described above inspired us to develop an *ad hoc* method of finding *extended* lines in images. Our method makes use of the fact that microtubule walls are not just linear objects; they are linear objects that point in the same direction for a long distance.

We first smooth the image by convolving it with a Gaussian kernel of width $\sigma = 3$ pixels. Next, we choose a particular direction to interrogate and calculate the first and second derivatives in that direction. We find points which are local minima along this direction as described above: by identifying locations where the second derivative is positive and the absolute value of the ratio of the first and second derivatives is less than 0.5 pixels. Figure 5 shows local minima along a direction –9 degrees to the vertical, obtained by performing this procedure. The displayed auxiliary matrix has a value of 1 at pixels which are local minima along the selected direction and a value of 0 at all other locations. We then searched for extended lines by convolving the auxiliary matrix with a binary mask of length $L = 41$ pixels in the orthogonal direction and thresholding to only select regions with a sufficient number of adjacent local minimum. This operation will incorrectly shorten lines, since their ends will not have enough adjacent minima to register, so we dilate the lines a corresponding amount to correct for this defect. Figure 6 show the regions corresponding to extend lines of the appropriate orientation found in Fig. 5. We then repeat the entire procedure to look for extended lines at 60 different angels, spaced in 3 degree increments.

Next, we combine the output of the searches for extended lines in different directions by creating another auxiliary matrix, which has a value of 1 for pixels where an

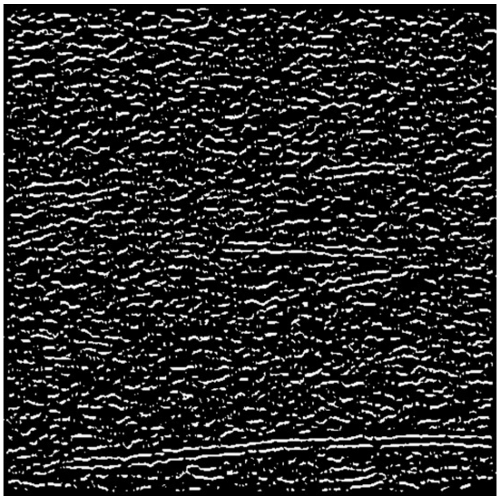

Fig. 5 The auxiliary matrix obtained by finding local minima along a direction –9 degrees from the vertical in the region displayed in Fig. 1. Note that many linear structures are present at different orientations.

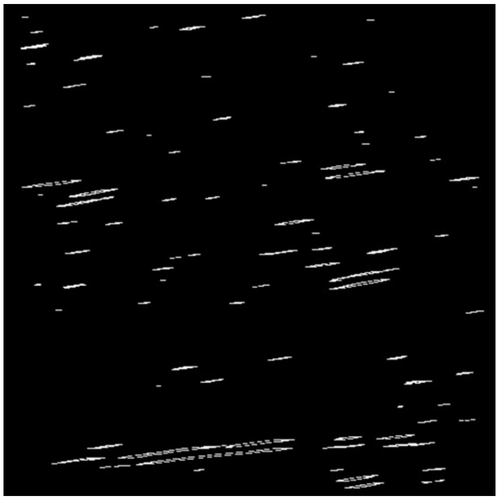

Fig. 6 Extended lines in Fig. 5 were identified by convolving with a linear segment of the appropriate orientation, thresholding to find regions of a large enough length, and dilating to correct for end effects.

extended line was found in any orientation, and a value of zero at all other pixels. The resulting matrix is depicted in Fig. 7. Finally, we probe for double lines—the two walls of a microtubule—in this data by a similar procedure in which we convolve with double lines in all directions and threshold. This preprocessing does an excellent job of locating microtubules (Fig. 8).

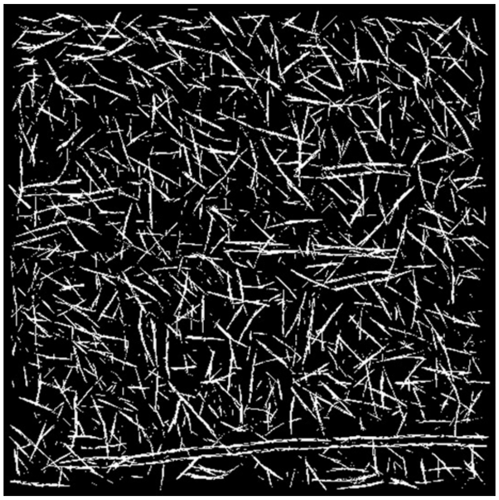

Fig. 7 The location of extended lines from the region depicted in Fig. 1, obtained from 60 different auxiliary matrices such as the one depicted in Fig. 6. See text for details.

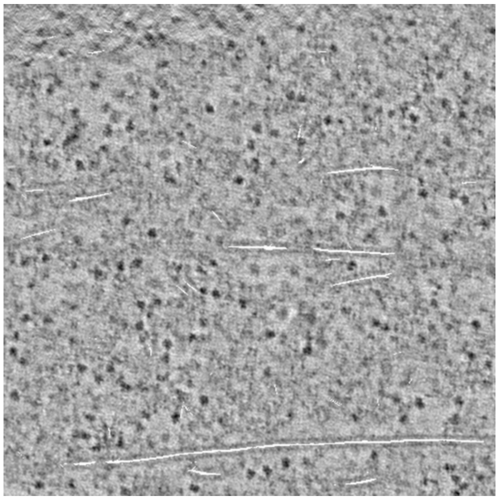

Fig. 8 The final results of the preprocessing step (white) displayed over the original image. Note that while most microtubules are well identified, false positives are also present.

The preprocessing algorithm contains several parameters that need to be set: the width of the Gaussian used for calculating the derivatives, the length L of the mask, and the two thresholds for the convolutions. Adjusting these values can make the procedure more lenient—with fewer false negatives but more false positives—or more stringent—with fewer false positives but more false negatives.

IV. Tracking: Connecting Points into Lines

The preprocessing step of our microtubule detection algorithm finds points that are likely to be inside microtubules. However, at this stage, we have not identified which points are contained in the same microtubule. This data also has errors: some points that we identify as being in microtubules might not really be in microtubules (false positives) and some points that are really in microtubules are not found by our algorithm (false negatives). Moreover, we would like to obtain information on the global properties of microtubules: How many microtubules are present in the tomogram? Where are they located? How long are they? What are their conformations? Ideally, we would like to represent each microtubule as a smooth curve that traces the microtubule's center line, so we can analyze properties of microtubules instead of properties of points that are in microtubules. Therefore, we need to connect the previously identified points into curves. We call the algorithm that performs this task a tracker, because it tracks the trajectory of individual microtubules.

Developing a tracking algorithm is challenging because the preprocessed data has a number of imperfections (Fig. 9):

- Several pixels are identified near the center line of microtubules, making the microtubule's exact position difficult to localize. These regions are only a few pixels wide in the *XY* plane, but their extent in the *Z* direction can be quite variable, and their width is not uniform along microtubules.
- There are gaps in microtubules: regions along their length where no points are found. A naive tracking algorithm might identify these gaps as true breaks, incorrectly finding multiple short microtubules where in reality only one long microtubule is present.
- Microtubules can pass close to each other and a poor tracker might falsely fuse two separate microtubules.
- Points which are clearly not associated with a microtubule are falsely identified as being in microtubules. These false positives can occur at relatively isolated locations or in clusters. Clusters of false positives are particularly difficult to distinguish from short microtubules. These clusters are often adjacent to microtubules.

Thus we need a tracking algorithm which can identify microtubules and find their trajectories despite the flaws that are present in the preprocessing step. One powerful method for tracking lines and other extended objects are "active contour models" (Kass *et al.*, 1988), in which an energy function is defined such that a local minimum corresponds to a tracked object and, ideally, the global minimum corresponds to all objects being located. When implementing active contours one has to choose an appropriate energy function. For tracking lines in high-quality images, such as the ones we obtain after a preprocessing step, active contour methods can be quite robust; thus constructing a suitable energy function is straightforward. The method is actually so powerful that it can even be used to track microtubules in the original, raw data, without the preprocessing step, but the results are less satisfactory. In the next section we will provide an introduction to active contours from a slightly more physical perspective than is standard (Kass *et al.*, 1988).

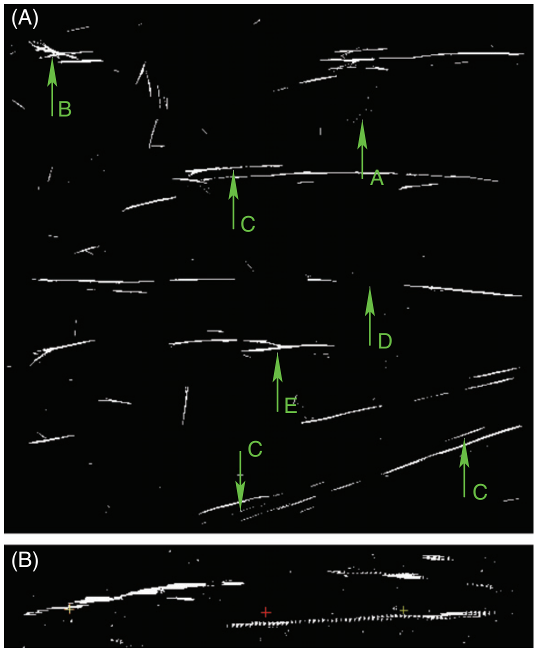

Fig. 9 (A) An *XY* slice of the preprocessing results; different types of errors and difficulties are present: A, isolated noise; B, cluster noise; C, false positives adjacent to a microtubule or between two microtubules; D, gaps (false negatives); E, close approach of two microtubules. (B) An *XZ* slice of the preprocessing results. The thickness of a microtubule in the preprocessing stage varies greatly in *Z*. Finding center the of the microtubule in *Z* is challenging.

A. Active Contour Models and Many-Body Simulation

Active contour models are a class of image analysis methods for finding extended objects—such as lines. In this approach, active contours—one-dimensional (1D) curves—are placed on the image and the energy is calculated based on the location and conformation of the contours. Then the contours are rearranged to search for the local energy minimum, which corresponds to the tracked positions of the lines. To implement this procedure, one needs to choose an appropriate energy function and an energy minimization algorithm. We shall start by considering the choice of energy function.

First, consider a single active contour designed to look for weakly bending lines in an image. An intuitive and useful choice is to select an energy functional which corresponds to an elastic string in an external potential:

$$E_{\text{contour}} = E_{\text{potential}} + E_{\text{bending}} + E_{\text{length}} = \underbrace{\int U(\vec{x}(l))dl}_{E_{\text{potential}}} + \underbrace{\int \frac{k}{R(l)^2}dl}_{E_{\text{bending}}} - \underbrace{\mu L}_{E_{\text{length}}} \quad (10)$$

The energy of the active contour consists of three components: the overlap of the contour and the image ($E_{\text{potential}} = \int U(\vec{x}(l)dl)$), how straight the contour is ($E_{\text{bending}} = \int (k/R(l)^2 dl)$), and its length ($E_{\text{length}} = -\mu L$). The first term, $E_{\text{potential}} = \int U(\vec{x}(l))dl$, represents the potential energy of the string. Here $\vec{x}(l)$ is a parametric representation of the contour, where $\vec{x}$ is the arc length along the contour and l is the position of the corresponding point. U, the potential, is a function of the image that determines which features are tracked. If $U = I$ (the intensity of the image), then the energy will be lower in the darker regions, while if $U = -I$ the energy will be lower in the brighter regions. If $U = -|\nabla I|$, the energy will be lower in the regions with high-gradients - edges. More complex functions of the intensity may also be useful. Second-order derivatives can identify local maxima or minima, and we have found that it can be helpful to include nonlocal contributions.

The second term in the energy, $E_{\text{elastic}} = \int (k/R(l)^2)dl$, represents the elastic energy of the contour. Here, $(1/R(l)) = |(d^2\vec{x}(l)/dl^2)|$ is the curvature of the contour and k is a bending modulus which weighs the relative importance of this term. If k is too small then the contour will faithfully follow the minimum of the potential, which might lead to overfitting of the data as even the local noise structure of the image will be tracked. If $k = \infty$ then only perfectly straight contours are allowed. The third term in the energy, $E_{\text{length}} = -\mu L$, favors longer contours. $L = \int dl$ is the length of the contour and μ is a stretch modulus. The role of this term is to grow the active contour until it is as long as the line to be tracked, but not any longer.

We have been discussing the energy of a single active contour, but many images of interest will have multiple lines, and the described approach must be adjusted to account for this situation. One option is to simultaneously model multiple contours and assign an energy to the entire ensemble. Part of the energy will just be the sum of the contribution of the individual contours: $\sum_i E_{\text{contour}}(i)$, where i is an index that denotes the ith contour, the sum extends over the number of contours, N, and $E_{\text{contour}}(i)$ is the energy of the ith contour of the form discussed above. As the number of lines present in the image is not known a priori, the number of contours, N, should be a variable parameter, which will contribute an additional term to the energy ($E_{\text{chempotential}}$). A natural choice is to set $E_{\text{chempotential}} = vN$. Here v is a chemical potential which determines the cost of creating a new contour. This $E_{\text{chempotential}}$ term favors fewer contours and can close gaps by causing two separate contours to fuse into one. It is also undesirable if multiple contours identify the same line in the image. Therefore, the contours should not be too close to each other. This condition can be enforced by including an interaction term to the energy ($E_{\text{interaction}}$) that cause the contours to repel each other. A simple option is to define a pairwise additive potential between strings, U_{int}, that depends on the distance between points on the strings. The interaction energy, $E_{\text{interaction}}$, will be of the form $\sum_{i,j}\int_{\text{string } i}\int_{\text{string } j} U_{\text{int}}(\vec{x}(l_i) - \vec{x}(l_j))dl_i dl_j$, where the sum extends over all pairs of contours, $\vec{x}(l_i) - \vec{x}(l_j)$ is the distance between a point on contour i and a point on contour j, and the integrals cover the lengths of each contour. Different choices for U_{int} are possible, including a simple hard body interaction that prevents contours from approaching closer than a specified distance.

Taken together, these considerations lead to the multicontour energy functional:

$$\begin{aligned} E_{\text{total}} &= \sum_i E_{\text{contour}}(i) + E_{\text{interaction}} + E_{\text{chempotential}} \\ &= \sum_i E_{\text{contour}}(i) + \sum_{i,j}\int_{\text{string } i}\int_{\text{string } j} U_{\text{int}}(\vec{x}(l_i) - \vec{x}(l_j))dl_i dl_j + vN \end{aligned} \tag{11}$$

This model maps the problem of finding lines in an image onto the "physical" problem of finding the minimum potential energy of a many-body system. It would be possible to directly perform a simulation of an ensemble of strings, in which they are created, annihilated, grow and shrink, merge, and split. This very beautiful picture of a dynamic many-body system has only a few parameters: three constants k, μ, v and two energy functions U, U_{int}. Finding the lowest energy state of this system is equivalent to solving a difficult image recognition problem! This analogy allows us to apply powerful methods for finding the minimum energy of complex systems developed in physics. In addition, we can now use our physical intuition to make heuristic simplifications to improve the performance of the tracking algorithim.

B. Tracking in Practice

So far, the discussion of our tracking approach has been very abstract. A number of additional issues must be addressed when using this method to analyze actual data. A real image is not a continuous distribution of intensity values, but rather consists of discrete pixels. Similarly, we must decide how to represent the trajectory of the contours. It is natural, and computationaly efficient, to discretize them as well. A specific energy minimization algorithm must also be selected.

Contours could be represented as splines (piecewise polynomial curves) or polylines (piecewise linear curves). As long as the resulting contours overlap the image to pixel resolution (requiring an accuracy of half a pixel), these two representations will give equivalent results. While the intrinsic smoothness of splines is ascetically pleasing, polylines are more convenient to work with, so we use them instead. Moreover, we can convert polylines to splines afterward to achive subpixel resolution of the microtubule centerlines. While the length of the segments in the polylines may dynamically vary during the simulation, we still need to choose a characteristic default length that is convienient. If the segments are too short, then calulations will be inefficient, but if segments are too long the discrete nature of the polyline will cause artifacts. We would like the polyline to deviate less than half a pixel from the trajectory of the continous curve it is supposed to represent. Thus, if the smallest radius of curvature of a microtubule we expect to encounter is of order 1000 pixels, the characteristic length of the polyline segment should be less than $\sqrt{8(0.5)(1000)} \approx 60$ pixels (where the factor of 8 arises from basic geometric considerations). In practice, we have found that using a segment size of 30 pixels works well.

There are many methods for computationaly finding the minimum of a function, but our analogy with a physical system suggests that we use either Molecular Dynamics or Monte Carlo simulations. Molecular Dynamics simulations define equations of motions based on "forces" arising from the prescribed energy functional and use them to evolve the system forward from its intial configuration. For complex energy landscapes, Molecular Dynamics simulations can get stuck in local minima, making it difficult to find the true minimum of interest. Molecular Dynamics simulations have been used with active contours to track actin filaments from fluorescence microscopy (Li *et al.*, 2009a, b), but the corresponding images have a much higher signal to noise than tomograms, so the resulting energy landscape is simpler. Monte Carlo simulations, in which trial moves are made and probablistically either accepted or rejected depending on the resulting change in energy, are more efficient for finding the global minimum in high-dimensional spaces with complex energy landscapes (Metropolis *et al.*, 1953). One powerful form of Monte Carlo simulation, called Simulated Annealing (Kirkpatrick, 1983), starts by frequently accepting energetically unfavorable moves and gradually lowers the "temperature," becoming more and more stringent over time. It is difficult to know what Simulated Anealing protocol is best to use—how fast to lower the temperature and when to stop the simulation—but the technique is still very useful. We next describe an implementation that can track microtubules in tomograms, even without any preprocessing.

C. Tracking in the Original Images

In this section we will illustrate how the presented tracking methods can be used to find microtubules in tomograms, even without the benefit of a preprocessing step. Since this is only intended to demonstrate our approach, we will limit the discussion to the task of finding a single microtubule in a 2D slice. In the next section we will show how preprocessing improves the quality of the data so greatly that the tracker can be further simplified and we expand the approach to handle the full 3D, multicontour problem.

We need to specify a potential, U that will be helpful for finding microtubules. A microtubule in a 2D slice of a tomogram appears as two dark, parralel lines (Fig. 1), so it is natural to choose $U = I(\vec{x}(l) + \frac{w}{2}\vec{n}(l)) + I(\vec{x}(l) - \frac{w}{2}\vec{n}(l))$, where $\vec{n}(l)$ is a vector normal to the contour at point $\vec{x}(l)$ and w is the width of a microtubule. This results in an energy function for a contour:

$$\begin{aligned} E_{\text{contour}} &= E_{\text{potential}} + E_{\text{bending}} + E_{\text{length}} \\ &= \int \left[I(\vec{x}(l) + \frac{w}{2}\vec{x}(l)) + I(\vec{x}(l) - \frac{w}{2}\vec{n}(l)) \right] dl + \int \frac{k}{R(l)^2} dl - \mu L \end{aligned} \tag{12}$$

We next need to determine appropriate values for the three parameters w, μ, and k. By inspection, the width of a microtubule is 10 pixels, so we set $w=10$. We want to select a value of μ such that the E_{length} term in the energy will compete with the $E_{\text{potential}}$ term, so a contour grows if it is on a microtubule and shrink if it is off one. The average image intensity is 165 and the average intensity along the walls of a microtubule is 140–150. Therefore, choosing $\mu=155$ will have the desired effect of making $E_{\text{potential}}$ dominate if the contour is on a microtubule and E_{length} be larger if the contour is off a microtubule. As long as the bending modulus k is not too large its exact value does not seem to effect the results and we select $k=5\times10^6$.

To find the global minimum we use a Monte Carlo algorithm. We start a contour in a random configuration and evolve its position through a series of steps. A trial move is made by displacing, growing, or shrinking the contour and the move is either accepted or rejected depending on how it changes the system's energy. If the new conformation lowers the energy, then the move is accepted. If the new conformation increases the energy, the move is accepted with probability $\exp(-\Delta E/T)$), where ΔE is the change in the contour's energy and T is the "temperature"—a parameter that determines how frequently energetically unfavorable moves are allowed. This process is repeated until the simulation ends. To avoid having the contour be stuck in unwanted local minimuma, we use a Simulated Anealing protocol in which we start with a high temperature, $T=150$, and gradually decrease it, by 2% every 200 steps. When the temperature becomes very low, the contour "freezes" in its current local minimum.

The initial state of the contour is a single segment with a random position and orientation. The following types of moves were implemented in the simulation:

- Growing by one segment (at one end or another).
- Shrinking by one segment (at one end or another, if the polyline is longer than 1 segment).
- Random displacement of a single vertex of the polyline.

When the simulation starts, the segment travels in the image until it overlaps a microtubule. The contour rapidly grows along the microtubule, but will occasionally reverse direction, quickly shrinking back to the original one-segment length. Amusingly, the process is highly reminiscent of the polymerization of microtubules by dynamic instability! After annealing is complete, the contour settles in on the position of a microtubule. While the found trajectory is often very accurate, errors are present. Figure 10 shows an example where the contour suddenly shifts to the side, following one edge of the microtubule and noise instead of tracing the true microtubule backbone. This type of error occurs because the energy landscape is highly complex, and the efficiency with which local minimum are circumvented is sensitive to the annealing protocol. Another drawback of the algorithm is that it is very slow. It is therefore highly advantageous to use the preprocessed images discussed earlier—resulting in a simpler energy landscape—and to streamline the Monte Carlo simulation, greatly speeding up the algorithim.

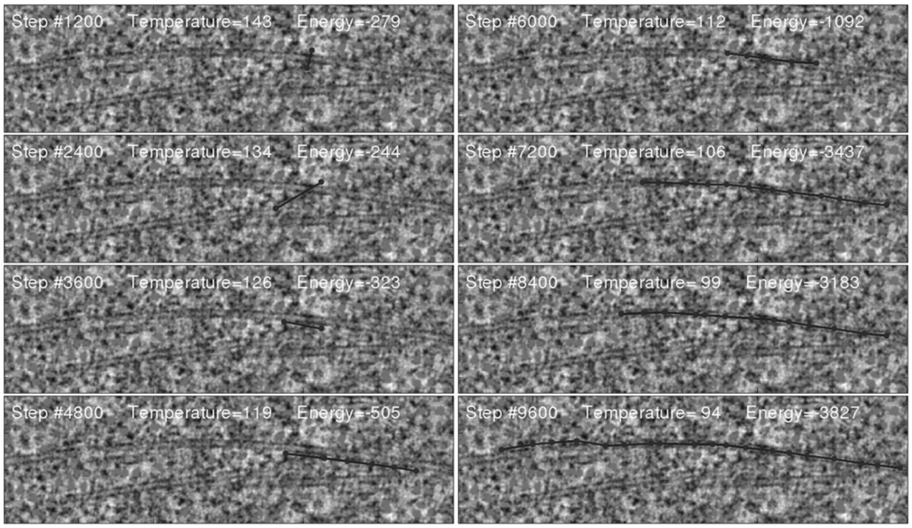
Step #1200 Temperature=143 Energy=-279
Step #2400 Temperature=134 Energy=-244
Step #3600 Temperature=126 Energy=-323
Step #4800 Temperature=119 Energy=-505
Step #6000 Temperature=112 Energy=-1092
Step #7200 Temperature=106 Energy=-3437
Step #8400 Temperature= 99 Energy=-3183
Step #9600 Temperature= 94 Energy=-3827

Fig. 10 Simulated annealing for tracking in the original images. Steps 1200–2400: a single segment moves randomly in the image. It does not grow when it is not on a microtubule. Step 3600: the segment overlaps with the microtubule and aligns with it. Growing from this position can lower the energy. Step 4800: the segment does grow from the position found on step 3600. However, it is not well aligned with the microtubule backbone. Step 6000: the contour undergoes a displacement causing it to more accurately follow the microtubule backbone, and the energy is substantially reduced. Steps 7200–8400: the polyline continues to grow inside the found microtubule. Step 9600: the contour has grown nearly the entire length of the microtubule. Note the region on the left: the contour jerks to the side and follows one microtubule wall and noise, instead of the true microtubule center line. This problem could be solved by increasing the bending rigidity, *k*, or by finding different parts of microtubules and fusing them in many-body simulation.

D. Tracking in the Preprocessed Images

In the preprocessed data, the location of microtubules are given by bright lines, so we can simply set $U(x) = -I(x)$. The resulting energy is as follows:

$$E_{\text{contour}} = E_{\text{potential}} + E_{\text{bending}} + E_{\text{length}} = -\int I(\vec{x}(l))dl + \int \frac{k}{R(l)^2}dl - \mu L \qquad (13)$$

This potential has two parameters, k and μ. As before, the end result is relatively insentive to the value of k, and we set it to $k = 5\times10^6$. μ should be intermediate between the intensity of points on microtubules and the intensity of points not on microtubules. One convenient approach to determine this level is to plot a histogram of intensity values in the preprocessed data. Then, after making an estimate of the volume fraction of microtubules, we can choose a value of μ which an appropriate fraction of intensity values are above—corresponding to points on microtubules.

The preprocessed data is of very high quality, allowing substantial simplifications of the Monte Carlo simulation. First, because most microtubules are well separated from each other, we do not explicitly perform the full many-body simulation. We simulate a single contour at a time, identifying the trajectory of each microtubule in the image independently. Second, we do not start the contours at random locations. Initial segments are added at points likely to be on microtubules—high-intensity pixels in the preprocessed images. Third, the preprocessed data is very smooth and has little noise. Therefore, the energy landscape has few unwanted local minima, and we can perform the Monte Carlo simulation at zero temperature—only accepting moves that decrease the contour's energy. With these considerations in mind, our resulting algorithm is as follows:

1. Find a high-intensity pixel in the image, this point is likely to be on a microtubule. Start a new, one-segment contour at this location at an orientation that minimizes the energy.
2. Grow the contour one segment at a time with a conformation that minimizes the energy.
3. If adding the segment results in an increase in energy, then this indicates that that portion of the contour is unlikely to be on a microtubule. This could be for two reasons: the contour might have grown past the end of the microtubule or there could be a gap in the microtubule caused by false negatives from the preprocessing step. We first check for a gap by attempting to grow the microtubule even farther to see if it can find another region that is likely to be part of the same microtubule. If a gap is not identified, then we continuosly shrink the contour until removing length no longer decreases the energy, at which point the microtubule end has been identified.
4. Refine the trajectory of the contour by repeatedly moving each point of the polyline by a random vector and keeping the move if the contour's energy is lowered.
5. Remove the found microtubule from the image by setting $I(x)$ to zero at all points within 5 pixels of the contour's final position.

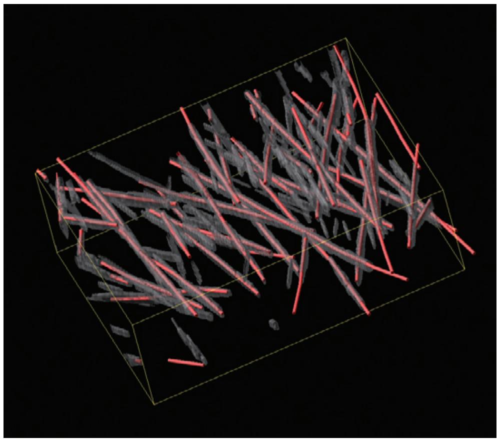

Fig. 11 Tracker's performance. The gray regions are points likely to be inside microtubules generated from the preprocessing step. The tubes mark the continuous lines found by the tracking algorithm.

6. Start again at step 1 in the new updated image and repeat until all points that are likely to be in a microtubule—those above a predetermined intensity value—have been interrogated.

Contours that are too short are most likely caused by noise and are discarded. The remaining contours give the locations and conformations of the tracked microtubules. This tracking algorithim is rapid and gives results that appear to be very good (Fig. 11).

V. Validation and Future Work

Simple visual inspection indicates that our algorithm accurately finds many microtubules in the tomogram. However, it is highly desirable to have a more systematic means of testing the success of our approach for automated microtubule identification. Having an objective validation scheme is important for optimizing the parameters used in the algorithm, discovering what type of errors are present so we can introduce new steps to minimize them, and understanding how reliable the results are. We therefore compared the output of our method with previously obtained results from a human expert who manually identified microtubules in this data set (O'Toole *et al.*, 2003) (see Fig. 12)

While many microtubules manually identified were found with our automated approach, we discovered a number of errors. The most common mistakes are as follows:

1. Some microtubules that were found manually were entirely missed by our algorithm. Many of these missed microtubules were located near the centrosome, where the noise structure is different than in other parts of the tomogram. Other missed microtubules pass through the *XY* planes at very steep angles.
2. Our automatically identified microtubules tend to be shorter than the manually identified ones: our algorithm has difficulty accurately finding the end of microtubules.
3. Long microtubules are sometimes incorrectly identified as multiple, short microtubules by our method. These errors arise because our tracking algorithm does not sufficiently correct for gaps in microtubules present in the preprocessing step.

We are in the process of developing ways to avoid these errors and we are attempting to use automated approaches to optimize the values of the parameters in our algorithm. One difficulty is that there are also errors in the manual data: some microtubules found by our algorithm were missed by the human expert and our automated method more accurately identifies the precise location of microtubule center lines.

Despite the mistakes that are present and the need for further optimization, our approach performs quite well for the tomogram under study. One concern for future applications is that the parameters in the algorithm may have to be adjusted for

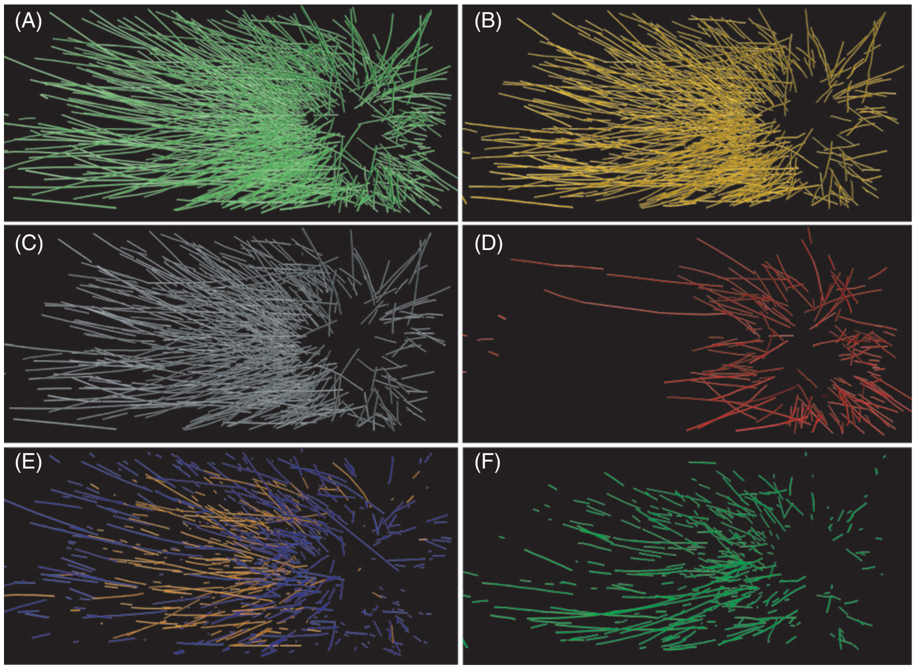

Fig. 12 Comparison of manual and automatic tracking: (A) All microtubules found by a human expert (O'Toole *et al.*, 2003). (B) All microtubules found by our algorithm. (C) Parts of microtubules found both by manual and by automatic tracking. (D) Microtubules missed completely in automatic tracking. Many are concentrated near the centrosome where the noise structure is different. Some are oriented at steep angles to *XY* plane. (E) Missing ends and gaps in the automatically identified microtubules. (F) Parts of microtubules found by our automatic algorithm and absent in the manual tracking result. Some of these are false positives, but others are true microtubules that were missed by the human expert.

different biological samples and different microscope settings. A hint that this procedure might not be straightforward is that the success of the preprocessing step can even vary within a sample: our algorithm performed worse near centrosomes, which exhibit a different background than other regions in the reconstruction. We are currently exploring the use of machine learning techniques to overcome this difficulty with preprocessing. Our preliminary work using Support Vector Machines (SVMs) is extremely promising. We create a training set by having a user click on several regions containing microtubules and several regions where microtubules are absent. With enough examples, SVMs are excellent at recognizing microtubules. Furthermore, the process of training is interactive: a user can improve the performance of the SVM by noting areas where it makes mistakes.

At the present stage of development, our automated method is not sufficiently reliable to replace manual tracking by a human expert. It could be used to provide an initial guess for the location of microtubules, which can then be corrected by a human. While this computer-assisted approach would still be cumbersome, it should be much faster than currently used, entirely manual methods. As our method is improved and new approaches are developed, it may eventually become possible to segment tomograms without human intervention. Such a development would greatly facilitate the analysis of large-scale tomographic reconstructions of cells.

Acknowledgments

We thank Eileen O'Toole for supplying the *C. elegans* tomography data and the manual microtubule identification that were used to test our automated segmentation. Figures 11 and 12 were generated with VMD (Humphrey *et al.*, 1996).

References

Briggman, K. L., and Denk, W. (2006). Towards neural circuit reconstruction with volume electron microscopy techniques. *Curr. Opin. Neurobiol.* **16**, 562–570.

Gonzalez, R. C., and Woods, R. E. (2007). "Digital Image Processing." Pearson Prentice Hall, New Jersey.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **57**(**1**), 97–109.

Hoenger, A., and McIntosh, J. R. (2009). Probing the macromolecular organization of cells by electron tomography. *Curr. Opin. Cell Biol.* **21**, 89–96.

Hoog, J. L., Schwartz, C., Noon, A. T., O'Toole, E. T., Mastronarde, D. N., McIntosh, J. R., and Antony, C. (2007). Organization of interphase microtubules in fission yeast analyzed by electron tomography. *Dev. Cell* **12**, 349–361.

Humphrey, W., Dalke, A., and Schulten, K. (1996). "VMD—visual molecular dynamics". *J. Mol. Graph.* **14**, 33–38.

Jiang, M., Qiang, J., and McEwen, B. F. (2006). Model-base automated extraction of microtubules from electron tomography volume. *IEEE Trans. Inf. Technol. Biomed.* **10**(**3**), 608–617.

Jiang, M., Qiang, J., and McEwen, B. F. (2006). Automated extraction of fine features of kinetochore microtubules and plus-ends from electron tomography volume. *IEEE Trans. Image Process.* **15**(**7**), 2035–2048.

Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *Int. J. Comput. Vis.* **1**(**4**), 321–331.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science. New Series.* **220**(**4598**), 671–680.

Kremer, J. R., Mastronarde, D. N., and McIntosh, J. R. (1996). Computer visualization of three-dimensional image data using IMOD. *J. Struct. Biol.* **116**(**1**), 71–76.

Li, H., Shen, T., Smith, M. B., Fujiwara, I., Vavylonis, D., and Huang, X. (2009a). Automated actin filament segmentation, tracking and tip elongation measurements based on open active contour models. *In* "ISBI'09: Proceedings of the Sixth IEEE international conference on Symposium on Biomedical Imaging," IEEE Press. (http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2804988/)

Li, H., Shen, T., Vavylonis, D., and Huang, X. (2009b). Actin filaments tracking based on particle filters and stretching open active contour models. *Med. Image Comput. Comput. Assist. Interv.* **12**(Pt 2), 673–681.

Lindeberg, T. (1998). Edge detection and ridge detection with automatic scale selection. *Int. J. Comput. Vis.* **30**(**2**), 117–154.

Marsh, B. J. (2005). Lessons from tomographic studies of the mammalian Golgi. *Biochim. Biophys. Acta* **1744**(**3**), 273–292.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., Teller, E., *et al*. (1953). Equation of state calculations by fast computing machines. *J. Chem. Phys.* **21**(**6**), 1087–1092.

O'Toole, E. T., McDonald, K. L., Mantler, J., McIntosh, J. R., Hyman, A. A., and Muller-Reichert, T. (2003). Morphologically distinct microtubule ends in the mitotic centrosome of *Caenorhabditis elegans*. *J. Cell Biol.* **163**(**3**), 451–456.

Sandberg, K. (2007). Methods for image segmentation in cellular tomography. (McIntosh Ed, J. R., ed.), *Methods in Cell Biology.* **79**, 769–798.