Supporting Material

Tracking single particles and elongated filaments with nanometer precision

Felix Ruhnow[§][†], David Zwicker[‡], Stefan Diez[§][†][*]

[§]B CUBE - Center of Innovation Competence, TU Dresden, Germany

[†]Max Planck Institute of Molecular Cell Biology and Genetics, Dresden, Germany

[‡]Max Planck Institute for the Physics of Complex Systems, Dresden, Germany

*Corresponding author. Address: ZIK B CUBE, TU Dresden, Arnoldstrasse 18, 01307 Dresden, Germany, Email: diez@bcube-dresden.de

S.1 Theoretic models

All models are based on the Gaussian distributions (Eqs. 1-4) in the main text. We dismissed the normalization factor and introduced an amplitude variable H instead. To simplify the equations we used the equivalent general bivariate quadratic form (1):

$$Z = \exp[-(AX^2 + BY^2 + CXY + DX + EY + F)] = H \cdot \exp[-(AX^2 + BY^2 + CXY + DX + EY)]$$

In the following we present both the original form $I(x, y)$ and the bivariate quadratic form $\tilde{I}(x, y)$ of the Gaussian distributions. We also show an example MATLAB function to plot the model.

The two-dimensional stretched Gaussian is represented by a bivariate normal distribution. $\hat{x}$ and $\hat{y}$ denote the center position of the peak, σ_x, σ_y the width in x and y direction and ρ the correlation coefficient between the x and y direction.

$$I_1(x, y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \cdot \exp\left[-\frac{1}{2(1-\rho^2)}\left(\left(\frac{x-\hat{x}}{\sigma_x}\right)^2 - 2\rho\left(\frac{x-\hat{x}}{\sigma_x}\right)\left(\frac{y-\hat{y}}{\sigma_y}\right) + \left(\frac{y-\hat{y}}{\sigma_y}\right)^2\right)\right]$$

A, B and C are the parameters of the bivariate quadratic form and H denotes the amplitude.

$$\tilde{I}_1(x, y) = H \cdot \exp[-(A(x - \hat{x})^2 + B(x - \hat{x})(y - \hat{y}) + C(y - \hat{y})^2)]$$

Instead of calculating σ_x, σ_y and the correlation coefficient ρ, we are more interested in σ_{major} and σ_{minor}, the widths along the major and minor axis compared to an ellipse, respectively. Additionally, we calculate the angle θ between the major axis and the x-axis.

$$A = \frac{\cos^2\theta}{2\sigma_{major}^2} + \frac{\sin^2\theta}{2\sigma_{minor}^2} \qquad B = -\frac{\sin 2\theta}{2\sigma_{major}^2} + \frac{\sin 2\theta}{2\sigma_{minor}^2} \qquad C = \frac{\sin^2\theta}{2\sigma_{major}^2} + \frac{\cos^2\theta}{2\sigma_{minor}^2}$$

$$\sigma_{major}, \sigma_{minor} > 0 \qquad \sigma_{major} > \sigma_{minor} \qquad \theta \in [0, \pi)$$

Back transformation:

$$\sigma_{major} = \frac{1}{2}\sqrt{\frac{A+\sqrt{B^2+(A-C)^2}+C}{AC-B^2/4}} \qquad \sigma_{minor} = \frac{1}{2}\sqrt{\frac{A-\sqrt{B^2+(A-C)^2}+C}{AC-B^2/4}} \qquad \theta = \begin{cases} \frac{1}{2}\tan^{-1}\left(\frac{B}{A-C}\right) & C > A \\ symmetric & C = A \\ \frac{1}{2}\tan^{-1}\left(\frac{B}{A-C}\right) + \frac{\pi}{2} & C < A \end{cases}$$

```
function GaussStretched(x,y,sigma_major,sigma_minor,theta,H)
[XG,YG] = meshgrid(-ceil(4*sigma_major):ceil(4*sigma_major));
A = cos(theta)^2/(2*sigma_major^2) + sin(theta)^2/(2*sigma_minor^2);
B = -sin(2*theta)/(2*sigma_major^2) + sin(2*theta)/(2*sigma_minor^2);
C = sin(theta)^2/(2*sigma_major^2) + cos(theta)^2/(2*sigma_minor^2);
I = H*exp( -( A*(XG-x).^2 + B*(XG-x).*(YG-y) + C*(YG-y).^2 ) );
surface(XG,YG,I);
```

The two-dimensional symmetric Gaussian is represented by a symmetric bivariate normal distribution. $\hat{x}$ and $\hat{y}$ denote the center position of the peak and σ the width.

$$I_2(x,y) = \frac{1}{2\pi\sigma^2} \cdot \exp\left[-\frac{(x-\hat{x})^2+(y-\hat{y})^2}{2\sigma^2}\right]$$

$A = 1/2\sigma^2$ is the parameter of the bivariate quadratic form and H denotes the amplitude.

$$\tilde{I}_2(x,y) = H \cdot \exp[-A((x-\hat{x})^2 + (y-\hat{y})^2)]$$

```
function GaussSymmetric(x,y,sigma,H)
[XG,YG] = meshgrid(-ceil(4*sigma):ceil(4*sigma));
A = 1/(2*sigma^2);
I = H*exp( -A*( (XG-x).^2 + (YG-y).^2 ) );
surface(XG,YG,I);
```

The two-dimensional symmetric Gaussian plus Gaussian ring is represented by a two-dimensional symmetric Gaussian with center $\hat{x}$ and $\hat{y}$, and width σ_c plus a symmetric Gaussian ring of radius r with center $\hat{x}$ and $\hat{y}$, and width σ_r. ζ weighs the relative contributions of the center peak to the ring.

$$I_3(x,y) = \frac{\zeta}{2\pi\sigma_c^2} \cdot \exp\left[-\frac{(x-\hat{x})^2+(y-\hat{y})^2}{2\sigma_c^2}\right] + \frac{1-\zeta}{(2\pi)^{3/2} r\sigma_r} \cdot \exp\left[-\frac{\left(\sqrt{(x-\hat{x})^2+(y-\hat{y})^2}-r\right)^2}{2\sigma_r^2}\right]$$

$A_c = 1/2\sigma_c^2$ and $A_r = 1/2\sigma_r^2$ are the parameters of the bivariate quadratic form. H_c and H_r denote the amplitudes.

$$\tilde{I}_3(x,y) = H_c \cdot \exp[-A_c((x-\hat{x})^2 + (y-\hat{y})^2)] + H_r \cdot \exp\left[-A_r\left(\sqrt{(x-\hat{x})^2 + (y-\hat{y})^2} - r\right)^2\right]$$

```
function GaussPlusRing(x,y,sigma_c,H_c,sigma_r,H_r,r)
[XG,YG] = meshgrid(-ceil(6*sigma_c):ceil(6*sigma_c));
A_c = 1/(2*sigma_c^2);
A_r = 1/(2*sigma_r^2);
I = H_c*exp( -A_c*( (XG-x).^2 + (YG-y).^2 ) ) + …
    H_r*exp( -A_r*( (sqrt((XG-x).^2 + (YG-y).^2) - r ).^2 ));
surface(XG,YG,I);
```

The two-dimensional Gaussian wall is represented by a generalization of the univariate Gaussian distribution of width σ perpendicular to a line with center $\hat{x}$ and $\hat{y}$. l denotes the length of the Gaussian wall and θ the angle between the Gaussian wall and the x-axis.

$$I_4(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \frac{1}{l} \cdot \exp\left[-\frac{\left((x-\hat{x})\cdot\sin\theta + (y-\hat{y})\cdot\cos\theta\right)^2}{2\sigma^2}\right]$$

$A = 1/2\sigma^2$ is the parameter of the bivariate quadratic form and H denotes the amplitude.

$$\tilde{I}_4(x,y) = H \cdot \exp\left[-A\left((x-\hat{x})\cdot\sin\theta + (y-\hat{y})\cdot\cos\theta\right)^2\right]$$

```
function GaussWall(x,y,sigma,H,theta)
[XG,YG] = meshgrid(-ceil(4*sigma):ceil(4*sigma));
A = 1/(2*sigma^2);
I = H*exp( - A * (-( XG - x ) * sin(theta) + ( YG - y ) * cos(theta)).^2 );
surface(XG,YG,I);
```

The two-dimensional curved Gaussian wall is represented by a Gaussian wall with an additional curvature c.

```
function CurvedGaussWall(x,y,sigma,H,theta,c)
[XG,YG] = meshgrid(-ceil(4*sigma):ceil(4*sigma));
A = 1/(2*sigma^2);
I = H*exp( - A * (-( XG  - x ) * sin(theta) + ( YG - y ) * cos(theta) + ...
          c*(-( XG  - x ) * cos(theta) - ( YG - y ) * sin(theta)).^2) .^2);
surface(XG,YG,I);
```

The filament tip consists of two parts: (i) a Gaussian wall starting at $\hat{x}$ and $\hat{y}$ in direction θ with width σ and amplitude H. (ii) Half of a two-dimensional symmetric Gaussian distribution with center $\hat{x}$ and $\hat{y}$, width σ and amplitude H.

```
function FilamentTip(x,y,sigma,H,theta)
[XG,YG] = meshgrid(-ceil(4*sigma):ceil(4*sigma));
f = -( XG - x ) * sin(theta-pi/2) + ( YG - y ) * cos(theta-pi/2) + 0.5;
f( f < 0 ) = 0;
f( f > 1 ) = 1;
A = 1/(2*sigma^2);
I = H*( f .*exp( -A*( ( XG - x ).^2 +         ( YG - y ).^2 ) )+...
     (1-f).*exp( -A*(-( XG - x )*sin(theta) + ( YG - y )*cos(theta)).^2) );
surface(XG,YG,I);
```

The short filament consists of three parts: (i) a Gaussian wall between $\hat{x}_s$, $\hat{y}_s$ and $\hat{x}_e, \hat{y}_e$ with width σ and height H. (ii) Half of a two-dimensional symmetric Gaussian with center $\hat{x}_s$ and $\hat{y}_s$ width σ and height H. (iii) Half of a two-dimensional symmetric Gaussian with center $\hat{x}_e$ and $\hat{y}_e$ width σ and height H.

```
function ShortFilament(x_s,y_s,x_e,y_e,sigma,H)
[XG,YG] = meshgrid(-ceil(4*sigma+sqrt((x_s-x_e)^2+(y_s-y_e)^2)):...
                    ceil(4*sigma+sqrt((x_s-x_e)^2+(y_s-y_e)^2)));
theta = atan2( y_e - y_s, x_e - x_s );
f1 = -( XG - x_s ) * sin(theta+pi/2) + ( YG - y_s ) * cos(theta+pi/2) + 0.5;
f2 = -( XG - x_e ) * sin(theta-pi/2) + ( YG - y_e ) * cos(theta-pi/2) + 0.5;
f1( f1 < 0 ) = 0;
f1( f1 > 1 ) = 1;
f2( f2 < 0 ) = 0;
f2( f2 > 1 ) = 1;
A = 1/(2*sigma^2);
I = H*(f1.*exp( -A*( ( XG - x_s ).^2 + ( YG - y_s ).^2 ) )+...
       f2.*exp( -A*( ( XG - x_e ).^2 + ( YG - y_e ).^2 ) )+...
       (1-f1).*(1-f2).*exp( - A*(-( XG - x_s ) * sin(theta) +...
                                 ( YG - y_s ) * cos(theta)).^2 ) );
surface(XG,YG,I);
```

S.2 Automated threshold algorithm

Although an automated algorithm would remove any user bias in the thresholding, it could fail if probes of different brightness are present in the experiment. Nonetheless, we included the following MatLab function for optional automatic thresholding.

```
function output = Image2Binary( input )
% convert image to double
input = double( input );
% find edges of objects
automatic_threshold = mean2(input)+std2(input);
% find edges of objects
output = edge(input,'sobel',[],'both','nothinning');
% close small gaps
output = bwmorph(output,'bridge');
% fill one-pixel holes
output = bwmorph( output,'fill');
% fill bigger holes in each object
l = logical(imcomplement(output),4);
l_props = regionprops(l,'Area','Image','BoundingBox','PixelIdxList');
f = find([l_props.Area]<50);
for n = f
  reg = imcrop(input,l_props(n).BoundingBox-[0 0 1 1]) .*l_props(n).Image;
  if mean2(reg) > automatic_threshold
    output(l_props(i).PixelIdxList) = 1;
  end
end
% multiply with low threshold image to rule out very dark areas
output = output.*(input>automatic_threshold);
```

S.3 Error estimation

For a non-linear least square problem with K data points z_i depending on L parameters p_j the Jacobian matrix is defined by $\mathcal{F}_{ij} = \partial z_i/\partial p_j$. The error Δp_j of parameter p_j is estimated by (2):

$$\Delta p_j = \sqrt{C_{jj}\chi_r^2} \qquad j = 1, 2, \dots, k \qquad \text{with} \qquad C = (\mathcal{F}^T \cdot \mathcal{F})^{-1}$$

where C is the variance-covariance matrix and $\chi_r^2 = \frac{\chi^2}{K-L}$ is the reduced chi-squared value computed using the normal χ^2 of the least-square-method divided by the number of data points K minus the number of parameters L. χ^2 is defined as the sum of the squared deviations between the model and the original data.

The length error of a filament is mainly influenced by the error in the position of the filament tips along the centerline and therefore can be estimated by:

$$\Delta l = \sum_{\alpha=s,e} \sqrt{(\Delta x_\alpha \cdot \cos\theta_\alpha)^2 + (\Delta y_\alpha \cdot \sin\theta_\alpha)^2}$$

where α denotes the start (s) and end (e) point of the filament.

S.4 Spline interpolation

Between two consecutive segments, a spline interpolation is used to calculate a continuous and differentiable curve. Here, we describe the interpolation between two points P_1 and P_2 with positions $x_{1,2}$ and $y_{1,2}$ as well as orientations $\theta_{1,2}$. In the coordinate system of the image, P_1 and P_2 are separated by a distance l and the straight line connecting the two points has an angle φ with the x-axis. The interpolation can then be broken down into the following steps:

1. Translate the picture by ($-x_1$,$-y_1$) and rotate it by $-\varphi$, such that point $P_1$is in the origin of a Cartesian coordinate system and point P_2 lies at (l,0). Note, that this also requires transforming the orientations $\theta_{1,2}$

2. Calculate the cubic polynomial that passes through both points and matches the transformed orientations. The four parameters of the polynomial are uniquely defined by the following four conditions: the curve must match the positions and the orientations of the points P_1 and P_2.

3. Apply a backtransformation to determine the curve in the initial coordinate system.

The total filament is then described by a set of cubic functions interpolating between consecutive points. Since we also match the measured orientations at each point, the resulting curve is differentiable. The interpolation procedure is illustrated in Fig. S1.

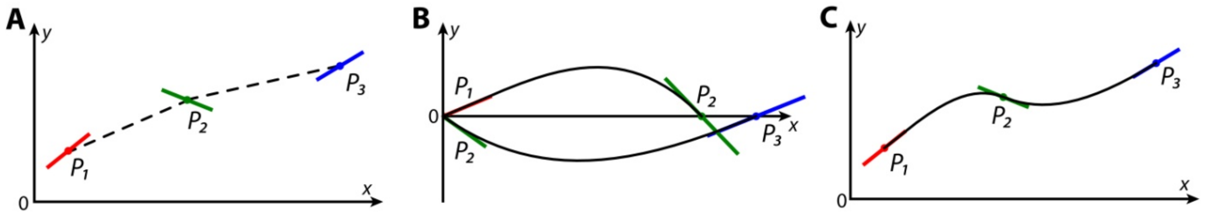

Figure S1: Illustration of the interpolation of the discrete points. (A) Results of the fitting process: distinct points $(\boldsymbol{x_i}, \boldsymbol{y_i})$, each associated with an orientation $\boldsymbol{\theta_i}$. (B) Transformation of the coordinate system: consecutive points are shifted and rotated such that they lie on the abscissa of the coordinate system. Spline Interpolation: a third-order polynomial interpolation (black line) is calculated that passes through both points and matches the orientations. (C) Backtransformation to initial coordinate system: original points are connected by a differentiable curve (black line) defined by the piecewise polynomials.

S.5 Filament simulation

To test our filament tracking algorithm we created artificial images (16 bit, 256x256 pixels, 100 nm pixelsize) containing two straight microtubules. Assuming a length of exactly 12 µm and 13 protofilament, we placed 4875 fluorophores (corresponding to a labeling ratio of 1:3) on the surface of a microtubule with 25 nm diameter. We chose arbitrary tip points for a horizontal microtubule and an angled microtubule (42.3° to the x-axis). The fluorophores were modeled by symmetric two-dimensional Gaussians ($FWHM$=350 nm). The amplitude of each fluorophore was determined using a Poisson distribution of given mean value λ. We also added (i) Poisson noise to every pixel (depending only on the pixel value) to simulate photon shot noise and (ii) Gaussian distributed noise with σ=3.2 to simulate dark noise of the camera. By adjusting the λ of the Poisson distribution of the fluorophore amplitude we could influence the signal to noise ratio (S/N) defined by:

$$S/N = \frac{average\ amplitude\ of\ the\ filament}{standard\ deviation\ of\ the\ background}$$

Whereas the position of each fluorophore of the simulated microtubule will stay fixed, the amplitude will vary in each frame according to the corresponding λ of the Poisson distribution. We created an image stack of 99 frames with a Signal-to-Noise ratio between 4.4 and 3914. Furthermore, we added a 100th frame without any photon shot or dark current noise which will correspond to a theoretical Signal-to-Noise ratio of infinity. These images were used to further characterize the tracking precision (see section S.7 and Fig. S7).

S.6 Filament centerline error

To characterize the filament tracking precision and compare the two-dimensional tracking to alternative methods using one-dimensional functions, we defined a filament centerline error, which is determined by the following steps:

1. The centerline deviation from the theoretical position (filament simulation) or averaged position (validation experiments) of a microtubule in a specific frame of the stack is determined. For each point F_p of the microtubule defined by $F_p = (x_p, y_p)$, $p = (1, 2, \ldots, n)$, the shortest distance to the reference microtubule will be calculated and is denoted as $\delta_{(i,k,p)}$ where i denotes the microtubule index, k the frame number and p the index of the microtubule position (see Fig. S2A).

2. We then define the centerline deviation $\tilde{\delta}_{(i,k)}$ of the complete i-th microtubule in a specific frame k by determining the value that includes at least 68% of centerline deviations $\delta_{(i,k,p)}$. A histogram of all centerline deviations $\delta_{(i,k,p)}$ is shown in Fig. S2B.

3. The centerline error $\bar{\delta}_{(i)}$ of the i-th microtubule for all frames is determined by the value that includes at least 68% of centerline deviations $\tilde{\delta}_{(i,k)}$. A histogram of all centerline deviations $\tilde{\delta}_{(i,k)}$ is shown in Fig. S2C.

4. For a set of microtubules these centerline errors $\bar{\delta}_{(i)}$ can be used to calculate the characteristic filament centerline error. An example using data from our validation experiments is shown in Fig. 2A of the main text.

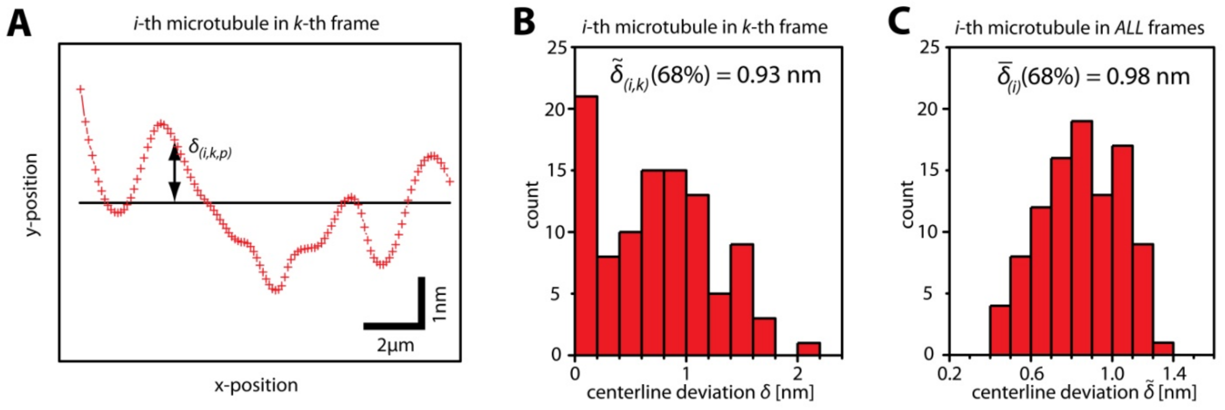

Figure S2: Definition of the filament centerline error: (A) Comparison of the theoretical position of the horizontal microtubule (black) and the tracked positions (red) using the filament tracking algorithm at $S/N = 99$. For each point p of the microtubule (red symbols) the centerline deviation $\delta_{(i,k,p)}$ is calculated. (B) Histogram of the centerline deviation $\delta_{(i,k,p)}$ for each point of the microtubule in a given frame k. $\tilde{\delta}_{(i,k)}$ denotes the value of $\delta_{(i,k,p)}$ that includes at least 68% of centerline deviations $\delta_{(i,k,p)}$ and corresponds to the centerline precision in the k-th frame. (C) Histogram of the centerline deviation $\tilde{\delta}_{(i,k)}$ of the i-th microtubule in all frame. $\bar{\delta}_{(i)}$ denotes the value of $\tilde{\delta}_{(i,k)}$ that includes at least 68% of centerline deviations $\tilde{\delta}_{(i,k)}$ and corresponds to the centerline precision for an entire image series.

S.7 Comparison to one-dimensional filament tracking

The image stacks created in the filament simulation (see S.5) were used to compare the two-dimensional filament tracking to a one-dimensional approach (3). Therefore, we implemented a MATLAB function that first rotates the image so that the filament orientation corresponds to the x-axis and then fitted a one-dimensional Gaussian function to a column segment (21 pixels) perpendicular to the x-axis.

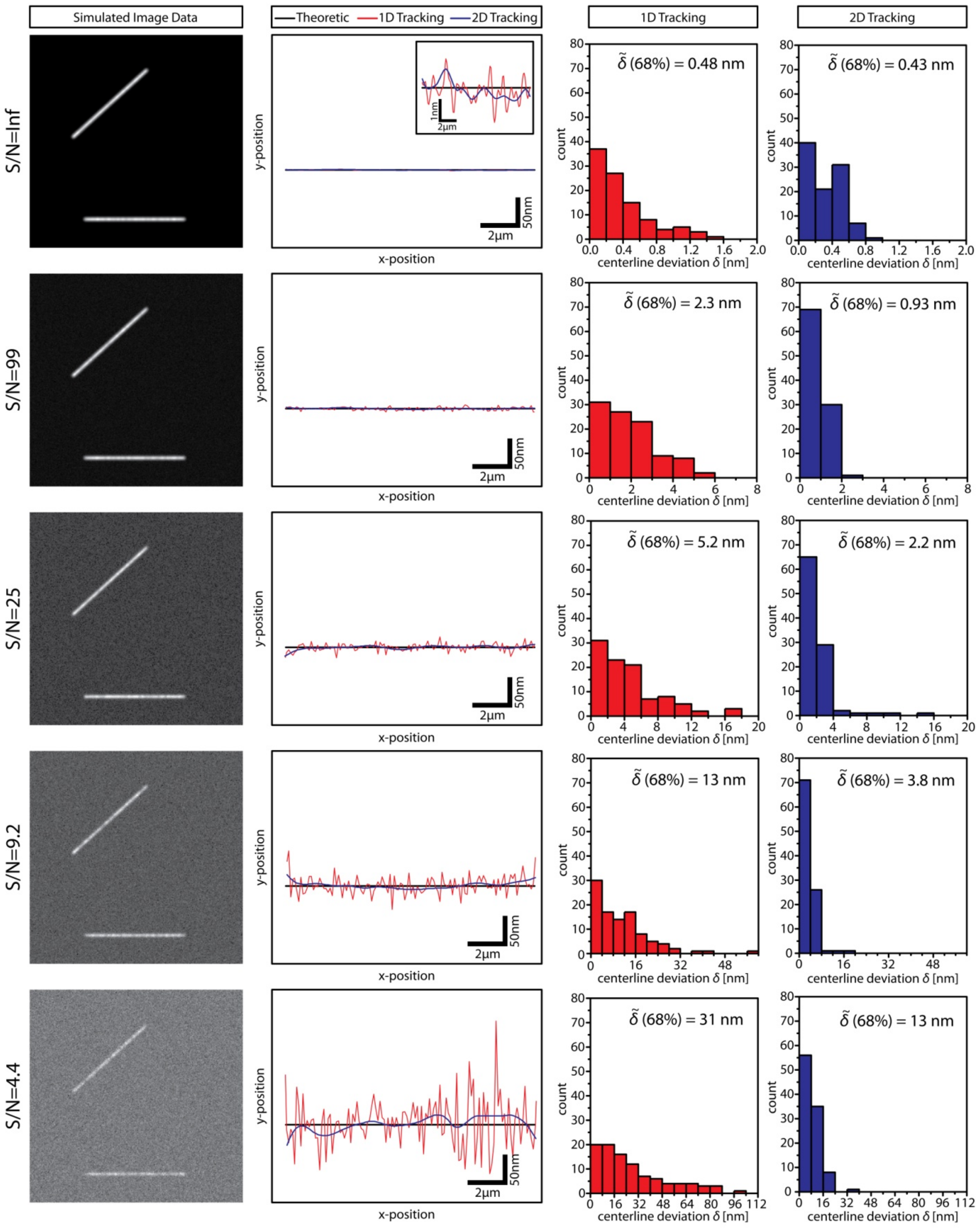

Figure S3: Comparison of two-dimensional filament tracking to a one-dimensional approach.

Although, no feature detection and trajectory linking was applied in the one-dimensional approach, tracking two microtubules over 100 frames took 15x longer than the complete two-dimensional tracking algorithm (one-dimensional tracking 1380 s, two-dimensional 92 s; on the same computer). This is due to optimized implementation of the fitting routines in our algorithm as well as the reduced number of fitting operations compared to the one-dimensional approach. Figure S.3 shows the comparison for five different S/N values. The two-dimensional tracking was found to perform superior than the one-dimensional approach, especially at a low S/N. Expectedly, the centerline deviation $\tilde{\delta}$ decreased with increasing S/N. However, due to the geometry of the microtubules (with random fluorescent labeling) there was still some deviation from the theoretical positions even in the absence of photon shot noise or dark current noise (see inlet of the x-y-position plot at $S/N = Inf$ with different sale bars). We note that the theoretical precision limit of filaments with smaller diameter, such as actin filaments or stretched DNA/RNA segments, is expected to be enhanced.

The centerline deviation of both tracking methods was determined for the microtubules in Figure S.3. In both cases, the two-dimensional tracking determines the position of the centerlines more precisely than the one-dimensional approach (Fig. S4 A-B). When investigating the influence of the image rotation, which is necessary in the one-dimensional approach, we found that already small errors in the estimation of the filament orientation further decrease the centerline precision (Fig. S4C). Since this effect already occurs for perfectly straight filaments, it may influence the centerline precision of curved filaments even stronger because different image rotations for different parts of the filament are necessary.

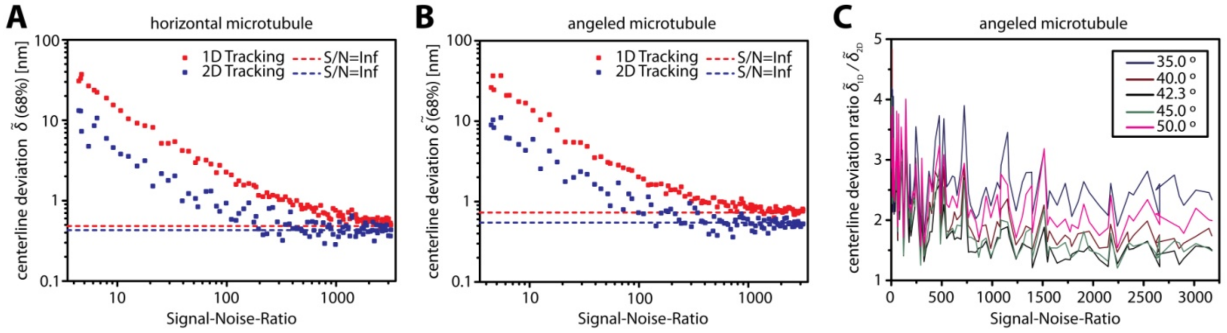

Figure S4: Comparison of two-dimensional filament tracking to the one-dimensional approach: (A) Centerline deviation $\tilde{\delta}$ of the tracking results for the horizontal microtubule as a function of the Signal-Noise-Ratio (S/N). Red points show the results of the one-dimensional method and blue points the results of our two-dimensional filament tracking algorithm, respectively. The dotted lines indicate the centerline deviation for the two methods in the absence of noise. (B) Centerline deviation $\tilde{\delta}$ of the tracking results for the angled microtubule as function of the Signal-Noise-Ratio (S/N). (C) Ratio of the centerline deviation $\tilde{\delta}$ of the one-dimensional approach to the deviation $\tilde{\delta}$ of our two-dimensional algorithm for different angles of image rotation applied to the angled (42.3°) microtubule. This simulates an uncertainty in the estimation of filament orientation in the one-dimensional approach (i.e. not fitting exactly perpendicular to the filament centerline).

S.8 Tracking of kinesin-1 motors on immobilized microtubules

As described in the main text, we tracked the movement of quantum dots (QDs) attached to the motor domain of kinesin-1. We calculated the distance of the QDs to the microtubule centerline as function of QD movement along the microtubule. In addition, we determined the intensity of the QD emission, defined as the integral of the Gaussian intensity distribution. Due to declining excitation intensity of the evanescent field in TIRF, objects that are further away from the surface will appear darker than objects in close proximity to the surface. Correlating data of distance to microtubule centerline with intensity of the QDs may thus allow the characterization of the three-dimensional movement of the motors on the microtubules. We note, that in our case, this characterization is hampered by the intrinsic blinking of the QDs.

The Figures S5A-D show four tracked QD traces suggestive of counter-clockwise movement. In these traces, the motors moved from the right to the left side. The averaged intensity slightly decreased towards the centerline and increased towards the outside. This behavior is expected for kinesin-1 moving on 14-protofilament microtubules polymerized in the presence of GMP-CPP (4). We also found traces of parallel movement with only slight changes in intensity (Fig. S6A-B). Most likely these events corresponded to kinesin-1 walking on one side of a microtubule.

Few traces suggest clockwise movement. Here, the QDs either moved from the left to the right side while the intensity declined (Fig. S6C) or moved from the right to the left side while the intensity increased (Fig. S6D). This indicates that microtubules with a different protofilament number might have been present. We did not see any parallel movement near the centerline of the microtubules, indicating the absence of a large number of 13-protafilament microtubules. Furthermore, we did not detect any QD that crossed the centerline while exhibiting an intensity maximum. This indicates that the kinesin-1 was most likely not able to move in between the microtubules and the surface. Whereas all reported traces represent typical examples, statistical analysis of microtubule supertwist periodicities was impossible due to the short run-length of kinesin-1 (on average 1 μm).

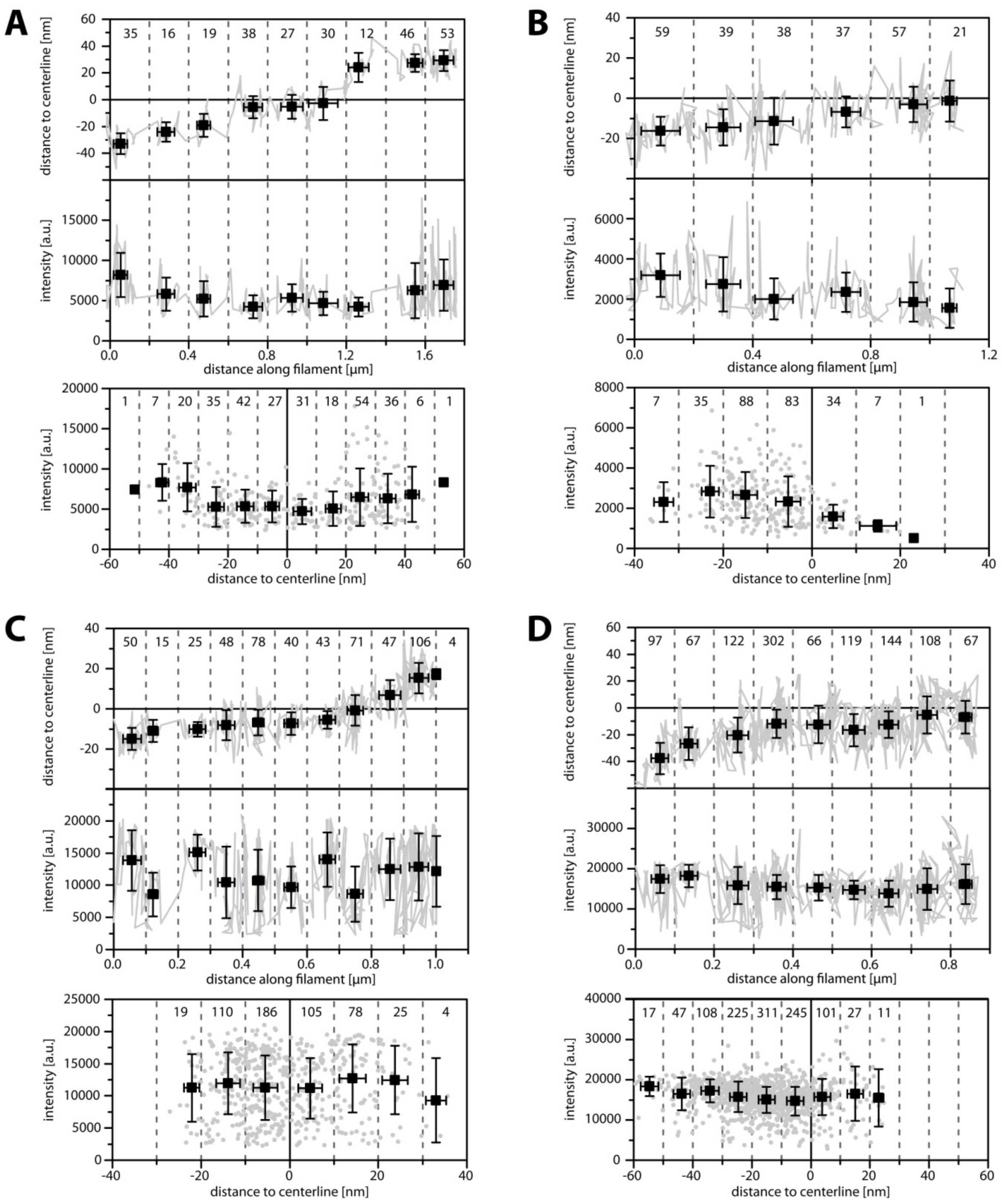

Figure S5: Quantum dot traces that suggest counter-clockwise movement (A-D) of kinesin-1 motors on the microtubule. The grey points denote the original data points for every frame. The black squares denote the averaged data with the respective errors (SD). The dotted lines mark the bins for averaging and the number of the original data points is given for each bin.

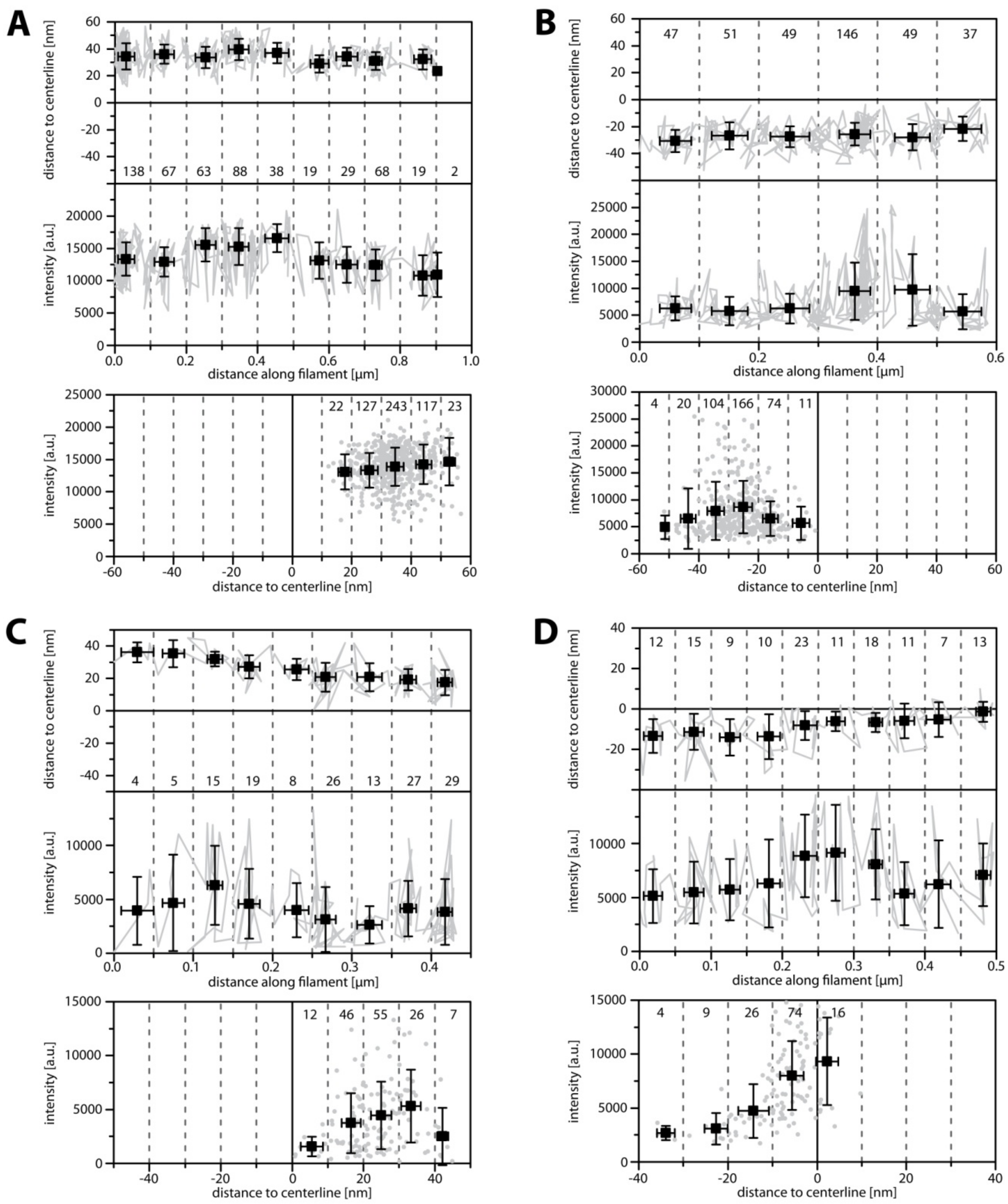

Figure S6: Quantum dot traces that suggest parallel (A-B) and clockwise (C-D) movement of kinesin-1 motors on the microtubule. The grey points denote the original data points for every frame. The black squares denote the averaged data with the respective error (SD). The dotted lines mark the bins for averaging and the number of the original data points is given for each bin.

S.9 Path Statistics

Two methods for calculating the traveled distance of particles or filaments with a trajectory $T_k = (X_k, Y_k),\ k = (1, 2, \dots, N)$ are commonly used. The 'distance to origin' $\widetilde{D}_k$ is defined by the displacement of an object between the first frame and any other frame: $\widetilde{D}_k = \sqrt{(X_k - X_1)^2 + (Y_k - Y_1)^2}$. Whereas the value $\widetilde{D}_k$ will only be influenced by the spatial error in frame 1 and frame k, it would underestimate the real traveled distance if the movement is not straight. The 'cumulative distance' $\widehat{D}_k$ is defined by the sum of the displacements between consecutive frames: $\widehat{D}_k = \sum_{l=2}^{k} \sqrt{(X_l - X_{l-1})^2 + (Y_l - Y_{l-1})^2}$. This method would correct for a non-linear trajectory, but is influenced by the spatial error in every frame. Therefore, we developed a 'walked distance' algorithm that smoothes the trajectory and accounts for turns on a user-defined scale. The first and last points of the trajectory are the starting and end point of the path, respectively. We then calculate the centroid $(\bar{X}, \bar{Y})$ of all consecutive points that lie within a certain user-defined radius r (scale) of the first point. Afterwards, the same procedure will be applied to the next point of the trajectory outside of the radius r until the last point of the trajectory is reached. The results will be a smoothened trajectory $\bar{T}_i = (\bar{X}_i, \bar{Y}_i),\ i = (1, 2, \dots, M)$ and we calculate an orientation θ by the angle to both neighbors:

$$\theta_1 = \tan^{-1}\left(\frac{\bar{Y}_2 - \bar{Y}_1}{\bar{X}_2 - \bar{X}_1}\right) \quad \theta_i = \tan^{-1}\left(\frac{\bar{Y}_{i+1} - \bar{Y}_{i-1}}{\bar{X}_{i+1} - \bar{X}_{i-1}}\right) \quad \theta_M = \tan^{-1}\left(\frac{\bar{Y}_M - \bar{Y}_{M-1}}{\bar{X}_M - \bar{X}_{M-1}}\right); \quad i = (2, \dots, M-1)$$

The continuous path is then determined using a spline interpolation (see S.4). Each point of the trajectory $T_k = (X_k, Y_k),\ k = (1, 2, \dots, N)$ will be projected on the path and the distance D_k along the path as well as the sideways deviation from the path are calculated. Additionally, the instantaneous velocity V_k can be determined by the displacement between the two neighboring frames:

$$V_1 = \frac{D_2 - D_1}{t_2 - t_1} \qquad V_k = \frac{D_{k+1} - D_{k-1}}{t_{k+1} - t_{k-1}} \qquad V_N = \frac{D_N - D_{N-1}}{t_N - t_{N-1}}; \quad k = (2, 3, \dots, N-1)$$

S.10 References

1. Paulitz, T. C., P. Dutilleul, S. H. Yamasaki, W. G. Fernando, and W. L. Seaman. 1999. A Generalized Two-Dimensional Gaussian Model of Disease Foci of Head Blight of Wheat Caused by Gibberella zeae. Phytopathology 89:74-83.
2. Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 1992. Numerical recipes in C (2nd ed.): the art of scientific computing. Cambridge University Press.
3. Brangwynne, C. P., G. H. Koenderink, E. Barry, Z. Dogic, F. C. MacKintosh, and D. A. Weitz. 2007. Bending dynamics of fluctuating biopolymers probed by automated high-resolution filament tracking. Biophys J 93:346-359.
4. Nitzsche, B., F. Ruhnow, and S. Diez. 2008. Quantum-dot-assisted characterization of microtubule rotations during cargo transport. Nat Nanotechnol 3:552-556.

S.11 MATLAB Source Code

The MATLAB source code of FIESTA is available at
http://www.bcube-dresden.de/fiesta/uploads/FIESTA(source).zip