Cytometry PART A
Journal of the International Society for Advancement of Cytometry

Automated Nonparametric Method for Detection of Step-Like Features in Biological Data Sets

C. Tyson,[1,2] C. McAndrew,[2,4†] P. L. Tuma,[3] I. Pegg,[2,4] A. Sarkar[2,4*]

[1]Department of Biomedical Engineering, Catholic University of America, Washington, DC 20064

[2]Vitreous State Laboratory, Catholic University of America, Washington, DC 20064

[3]Department of Biology, Catholic University of America, Washington, DC 20064

[4]Department of Physics, Catholic University of America, Washington, DC 20064

Received 17 June 2014; Revised 19 December 2014; Accepted 28 December 2014

Grant sponsor: Grant sponsor: Vitreous State Laboratory; Grant sponsor: Catholic University of America; Grant sponsor: National Institutes of Health; Grant number: R01 AA0117626 (to P. Tuma)

Additional Supporting Information may be found in the online version of this article.

†Current address: United States Patents and Trademarks Office, Alexandria, Virginia 22314

*Correspondence to: A. Sarkar, 620 Michigan Ave NE, Hannan Hall Room #211, Washington, DC 20064, USA. E-mail: sarkar@cua.edu

Published online 4 February 2015 in Wiley Online Library (wileyonlinelibrary.com)

DOI: 10.1002/cyto.a.22631

• Abstract

Experimental data from single-molecule DNA-protein experiments, such as experiments using optical traps or magnetic tweezers, typically contain steps, plateaus, or dwell regions that are obscured by thermal and other noise sources. We present a nonparametric method for detecting step-like features in noisy biological data sets. Our algorithm does not assume that the steps can be modeled as Heaviside functions or any particular parametric form. No assumptions about the noise source, such as whether the noise is Gaussian or colored, are made either. Instead, for detection of plateaus, the algorithm uses the novel method of analyzing a probability distribution function of the data values. The vast majority of previously published methods for step detection rely on statistical fitting of step functions with the flat segments linked by vertical segments. Our approach is intended for use on data which cannot be modelled as a series of step functions but applies to step functions as a special case. These type of data traces have, so far, been difficult to characterize effectively. We examine the performance of the algorithm through systematic simulation studies and illustrate the use of our algorithm to analyze single molecule DNA-protein micromanipulation experiments carried out by our laboratory. The simulation results and experimental validation suggest that our method is very robust, avoids overfitting, and functions effectively in the presence of noise sources characteristic of single molecule experiments. © 2015 International Society for Advancement of Cytometry

• Key terms

Key terms: step detection; algorithm; single-molecule; biophysics

METHODS to analyze time series or other signals to detect regions in which the signal is constant are useful in many different fields of science and engineering. Single-molecule DNA-protein micromanipulation experiments, for instance, often involve studying the binding and force-induced unbinding of proteins from a single tethered DNA molecule. After the DNA tether is extended under a small force, proteins are introduced into the sample cell. These proteins may bind to the DNA and compact the DNA by stabilizing loops, bends, or kinks. Once the DNA is compacted, the force can be increased until just large enough to drive off the bound proteins. When the end-to-end tether extension is measured, the unbinding events will be seen a jumps between constant-extension regions. This type of step-like structure can also be found in other types of single molecule experiments in biology, and often arise in areas such as electrical engineering or econometrics. In this article, we present a novel method for automatically finding steps in such data series.

Algorithms that perform analyses of this kind are often referred to as "step-finding," "step-fitting," or "step detection" algorithms. The general goal of algorithms in this family is to identify regions of a data trace where the signal maintains a certain level for some time and then changes to another level. This change is frequently abrupt, as in a step described by the Heaviside function. In such a case, the linking segment between successive steps is defined to have a slope of infinity. While it is rare to encounter signals in nature with purely step-function type jumps, it is common to model step data as a series of such functions. Indeed, many step detection algorithms in existence have utilized this principle as a basis for their analysis of the data (1).

A key consideration in using such techniques is how well the signal can be modeled this way and what effect departures from the model have on the results. Another important consideration is preventing over-fitting. For instance, by using suitable number of parameters, it may be possible to detect all steps in a particular type of noisy signal with very high confidence. However, when these settings are applied to other signals, the algorithm's performance may decline rapidly. Thus, a step finding method should be robust enough to be applied to large classes of signals without requiring re-adjustment of parameters to ensure accurate results. Finally, the signals of interest inevitably have large noise components. The noise may or may not be Gaussian and the algorithm should, ideally, perform well in all cases.

A number of previous studies have examined different approaches to the step-finding problem. Kalafut and Visscher (2) describe the application of the Schwarz Information Criterion (SIC) to fitting step functions to a data trace. The SIC is used as a way to prevent over-fitting by penalizing fits that utilize too many parameters.

Carter and Cross (3) propose an algorithm that uses the two sample student's *t*-test to evaluate steps. This algorithm compares a given data point with other individual data points within a certain time range, resulting in the algorithm categorizing each point as belonging to a dwell (plateau), forward step, or downward step.

A method using wavelet transform multiscale products was proposed by Sadler and Swami (4). Multiscale products were shown to be effective in isolating the abrupt step transitions in signals, which in turn allows identification of the steps.

Kerssemakers et al. (5) analyzed step data using a χ^2 reduction principle. By comparing a series of step best-fits to a series of counter-fits, in which the fitted steps are displaced to be in between the best fits, the algorithm is able to approach an optimal number of steps fitted to the data.

Velocity calculation and thresholding is the method described by Levi et al (6). This algorithm calculates local velocities of the data—the change in position over time. If the velocity is below a certain threshold, that region of the data is assumed to be a level step. If the velocity is above the threshold, the data must be subdivided and the analysis is repeated. This method also uses the Akaike Information Criterion to penalize over-fitting of the data.

Arunajadai and Cheng (7) use a combination of Generalized Least Squares and Bayesian Information Criterion to reduce over-fitting of the step functions to the data with success. Their method is notable for its avoidance of assuming a particular type of noise. Rather, it observes and determines the noise correlation throughout the data to more accurately detect the underlying signal.

Pair-wise distribution functions have also been used to map steps in a data trace by Kuo et al. (8) and Block et al. (9), respectively. Such algorithms can be useful when the data contain step sizes that are relatively constant; step-sizes that vary over a wide range are not handled well by such methods. Other groups, including Milescu et al. (10), have based their step detection algorithms on knowledge of underlying principles that generate the data and a Markovian model process to identify steps in the data. These methods are intended for very specific applications, and as such, it is difficult to compare them to more generalized algorithms.

Finally, we refer the method of Herbert et al. (11), which uses an eightfold averaged log-dwell histogram to extract peaks corresponding to dwells of RNA polymerase movement along base pairs of a DNA tether. The first step of this method is similar to ours—calculation of relative probabilities for extensions—but diverges quickly after that to meet the specific needs of the algorithm's application in that situation.

Many of the aforementioned techniques use statistical methods to fit a series of step functions to the data. This means that the transition between steps is instantaneous. For data in which the steps do not feature a linking segment slope of infinity—data for which the linking segments may have a variable slope—such algorithms may not be so effective. Furthermore, several of the methods make assumptions about the underlying noise properties. While it is true that independent Gaussian white noise is the predominant type of noise encountered experimentally, it is not always the case. Optomechanical systems that are used in biology, biological physics, and electrical engineering can introduce frequency-dependent or colored noise. Analyzing a data trace that features correlated noise with an algorithm that assumes uncorrelated noise can lead to problems.

Here, we present a novel method for detecting steps (or plateaus) in a noisy data trace that does not assume that the signal can be modeled as a series of step functions and does not assume any specific type of noise. Our aim is to develop a generalized method that requires little knowledge of the noise or the underlying mechanism that produced the data. The plan of the article is as follows. In the next section, we describe the algorithm and the simulations used to test it. This is followed by description of the performance metrics used to study the algorithm's performance. We then describe how we obtained experimental data from single molecule experiments that were analyzed using the algorithm. In the following section, we describe the results of our simulation and experimental studies. This is followed by a discussion section where we place our method in the context of previous studies in step detection. We conclude with some observations on future developments of our method.

Materials and Methods

Step-Trace Simulations

The simulated signals consist of a series of individual "steps", which are horizontal line segments or "plateaus" of

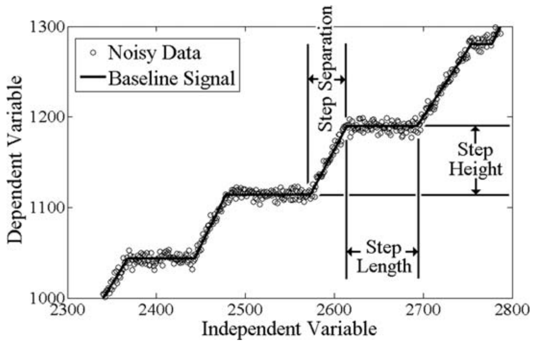


Figure 1. An example of a typical data trace and the various parameters used to describe the steps is shown, with arbitrary units of position and time along the y and x axes, respectively. Step length represents the duration of the horizontal component of the step. Step height refers to the vertical distance between subsequent steps. Step separation is used to describe the horizontal distance between subsequent steps. These three parameters can be used to define an entire data trace of unique steps.

adjustable length, separated by line segments, also of adjustable lengths, with positive slopes. Each step is characterized by three quantities: (a) the step length, which is the length of the plateau; (b) the step height, which is the vertical distance between the end of one plateau and the start of the next, and (c) the plateau separation, which is the linear distance between the end of one plateau and the start of the next one. The slopes can be set from near zero to arbitrarily large values, and are calculated from the ratios of the step heights to plateau separations. The slopes of the connecting segments are not specified explicitly. The goal is to determine the locations of the steps, which are specified in arbitrary units corresponding to the *y*-coordinate value. Figure 1 shows an example data trace.

A MATLAB program was written to generate simulated data traces based on the parameters described above. The input variables consist of: (1) the number of steps to be generated, (2) the step separation, (3) the step height, (4) the step length, (5) whether or not to include normally distributed noise in the final signal, and (6) the noise width, σ, which is the standard deviation of the normal distribution. Input parameters 2, 3, 4, and 6 are specified in arbitrary units.

Similar methodology was used to generate data sets with Poisson-distributed noise. See Supporting Information for details.

From the step separation and step height, a slope is calculated for the link between successive plateaus, which is then used to determine the y-coordinates from the *x*-coordinates for the segments linking the plateaus. Once these segments have been determined, the plateaus are found by extending the last point y-coordinate of the linking segments for the number of points specified by the step length parameter. This generates the solid line data trace of Figure 1. From here, the noisy data trace (dots in Fig. 1) is created by multiplying a normally distributed random number in the interval [-1,1] by σ and adding that to the red base signal. Thus, for $\sigma = 4$, the noise will modify the original base signal by some additive value in [-4,4] in accordance with the normal distribution, for each point in the base signal.

All three step parameters in the simulation program can be assigned randomized values from a user-defined interval within a single simulated signal. This allows us to more closely emulate experimental data. Step lengths, step heights, and step separations may be drawn from uniform or normal distributions, and the user may independently specify the range for each parameter: for instance, [20, 100] for step length, [50, 200] for step height, [40, 80] for step separation. The data traces for such simulations are calculated in essentially the same way as for constant parameter traces, except that for each single step a random parameter is chosen prior to calculating the slope of each linking segment and length of each plateau; the noisy data trace is generated in exactly the same way as in other examples.

Step-Finding Algorithm

Data preprocessing. While the step-finding algorithm is capable of identifying steps in raw data, there are many cases in which data preprocessing can be helpful. In situations where a data set may see benefit from preprocessing, an edge-preserving bilateral filter was adapted from the DIRART Image Processing MATLAB toolbox (12) to best preserve the step-like features in the data. The bilateral filter assigns a new coordinate for each point in a time series based upon a two-dimensional Gaussian-weighted kernel. The basic inputs to the bilateral filter consist of the data, filter width, and filter height. The filter width and height determine the size of the Gaussian-weighted kernels in the horizontal (*x*-coordinate) and vertical (*y*-coordinate) directions, respectively. Increasing the filter width would include more points in the *x*-coordinate range when calculating the new value for each point, while increasing the filter height would use more points in the vertical *y*-coordinate. As a special case, using a width of zero for either parameter would eliminate using any points in that dimension to calculate the new value. A filter height of zero would result in a purely horizontal Gaussian kernel, while a filter width of zero would result in a purely vertical Gaussian kernel. When the data were filtered using this method, filter width and height of 10 and 10 were used. Figure 2 shows an example data trace prefiltering and postfiltering.

Step-detection algorithm. In Figure 3a we plot in blue a portion of a data trace that is to be evaluated by the algorithm, with the base line signal shown as a solid red line in Figure 3b. The raw data shown in 3a have noise $\sigma = 3$ and are unfiltered. From here, the first stage of the step-detection algorithm is determining the discrete probability density function (PDF) of the data trace (Fig. 3c, red). The discrete PDF is a histogram that is calculated by counting the number of data points for discrete bins with each bin defined by an interval of *y*-coordinates. The user may generate the PDF in a completely automated way. In this case, the algorithm will find the mean difference between the y-values of successive points

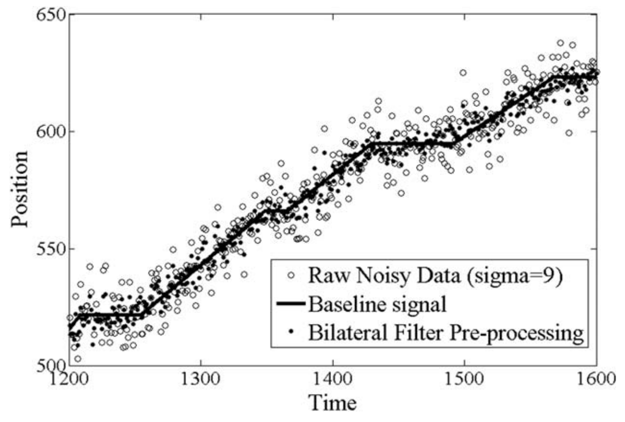


Figure 2. The effects of using the bilateral filter preprocessing on noisy data are shown. The solid black line represents the base signal of randomly chosen step parameters. The empty circles are the signal after normally distributed noise with $\sigma = 9$ is added. The black dots represent the result of the noisy data after the bilateral filter with filter height and width both equal to 10. Position and time are in arbitrary units.

throughout the entire data set, which gives a rough idea of the "width" of the data. This number is then used as the bin size for the data. Alternatively, the user may specify a bin size to be used. The result is the transformation of the data from position-time space to probability-*y*-position space.

Next, the second derivative of the PDF is calculated using finite differences (Fig. 3d, green). This helps to emphasize changes in the direction of the signal, which is the core information used to identify values of the *y*-coordinates where there is a high density of data points and therefore a significant probability of the existence of a plateau. Then, the second derivative is added back into the original PDF (Fig. 3e, magenta) and the result is squared (Fig. 3f, blue) to make all values positive. This step of summing the original PDF and its own second derivative is performed because we find that it helps to amplify the portions of the signal at which there are a simultaneously large changes in direction and large number of points while reducing the prominence of the signal where there are fewer points and smaller changes of direction.

After these steps, the data have been transformed into a system of peaks and valleys in which the peaks represent the *y*-coordinate values with a higher probability of data points residing at that value than at the surrounding values and the valleys representing a low probability of points existing at that *y*-coordinate value with respect to the surrounding values.

While identification of peaks is possible using the resulting distribution, nearly all examples will show better results if the PDF is smoothed. A very effective way to smooth this signal is to use our own local extrema interpolation averaging (LEIA). The LEIA method we have created finds all local maxima and local minima separately and then uses a common interpolation method (linear or cubic spline) to determine two smoothed distributions using the local maxima and local minima respectively. Once these two signals are obtained, they are then averaged over the entire data range to obtain a new data distribution (Fig. 3g, black).

Next, a MATLAB routine *extrema.m* is used to find the peaks, which are then analyzed by a peak scoring method based on the arc lengths of all peaks in the data series. For each local maxima, the arc length along the LEIA smoothed function from the minima preceding a maxima to the minima succeeding it is calculated. The standard deviation of the set of arc lengths is calculated. The set of arc lengths is trimmed of the largest value and the standard deviation is calculated again using the trimmed set of arc lengths. This is repeated for the entire set and the percent difference of the standard deviation is calculated after each iteration of the trimming step. The point at which the largest percent difference between standard deviations of trimmed sets occurs is used to differentiate between the significant peaks and noise peaks. The reasoning here is that the standard deviation will show the greatest change in percent difference when the peaks with large arc lengths are trimmed from the set of arc lengths and only small, relatively constant peaks remain. This is generally sufficient to successfully determine which peaks are significant enough to be the result of a plateau in the data.

These significant peaks are then the output of the algorithm, representing the points at which the data signal has a local plateau. Figure 3h plots the original signal (from Fig. 3b) along with the locations (dashed black lines) of the peaks from Figure 3g, showing that the peaks correspond to sections of the signal that contain steps.

Performance Analysis

Determining the performance of the algorithm on the simulated data requires comparing the detected steps from the algorithm against the known steps in the base signal with generated data. This involves the simple process of comparing two lists and finding the closest elements on both. The standard for accuracy of a detected step was whether it fell within half-noise-width of the base signal step. Using this criteria, three performance metrics were computed: detected steps, false positives, and false negatives. Detected steps are those that fall within one-half-noise-width of the known steps. False positives are results that do not correspond with any known step and false negatives are known steps not properly identified. In the event where multiple steps were inferred by the algorithm within one-half-noise-width of a single real step, the inferred step that is closest to the real step was counted as a true detected step and the other(s) as false positive(s).

Experimental Data

We used the algorithm to detect histone complex binding and unbinding events in single-molecule DNA micromanipulation experiments. These experiments are performed on a horizontal magnetic tweezers systems (13). We show a schematic of the instrument in Figure 4a.

The magnetic tweezers system uses a Nikon Dyaphot inverted light microscope to observe fluctuations in beads that are attached to a single 48.5 kbp lambda-DNA molecule. One bead on the DNA is a 2.8 μm super-paramagnetic bead and the other is a 3 μ nonmagnetic polystyrene bead. A 1 mm diameter glass capillary is pulled to form a micropipette with a tip diameter of ~2 μ and is clamped by an aluminum fixture

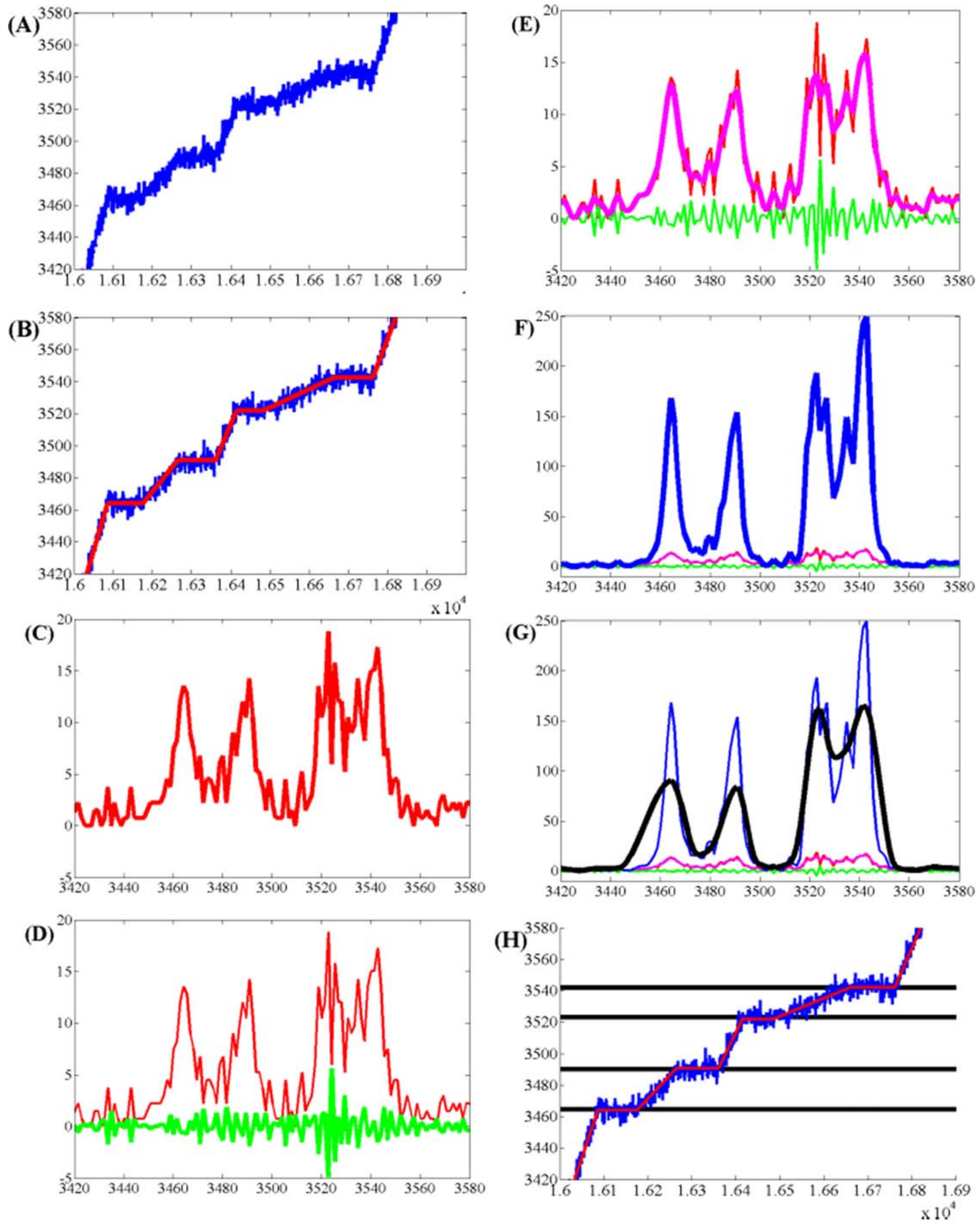
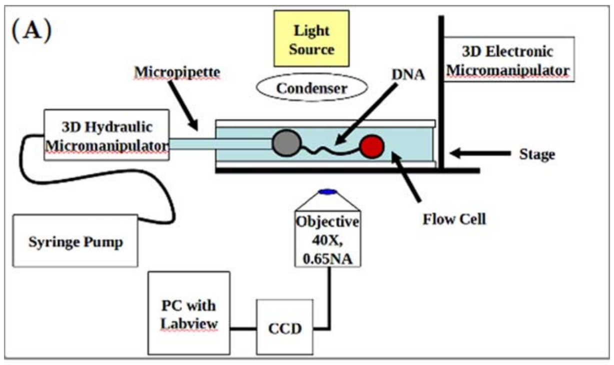


Figure 3. This series of plots shows a step-by-step breakdown of the method by which our algorithm obtains the locations of plateaus in an unfiltered data trace with noise $\sigma = 3$. For each plot, the thick (bold) colored line represents the signal at that specific stage of the algorithm, with other previous steps shown in thinner colored lines for perspective. **A**: [Blue] A segment of a noisy data trace with randomly assigned step parameters. **B**: [Red] The baseline signal for that data set shows four (4) areas where the signal is constant. **C**: [Red] The initial probability distribution function (pdf) result is plotted. **D**: [Green] The second derivative of the pdf is shown. **E**: [Magenta] The second derivative from 3D is added back into the original pdf. **F**: [Blue] The result of 3E is squared. **G**: [Black] The resulting signal from F is modified using LEIA. **H**: After identifying the significant peaks from G and determining the center of those peaks, the results [dashed black] are plotted against the raw data from B.

that is held stationary by a hydraulic micromanipulator over the microscope objective. This aluminum fixture also holds another glass micropipette that is used to inject proteins close to the DNA strand. Both the DNA micropipette and protein micropipette are connected via Tygon tubing to 10 mL syringes that are placed in syringe pumps. An Eppendorf micromanipulator moves an aluminum platform upon which a glass chamber for the experiment is placed.



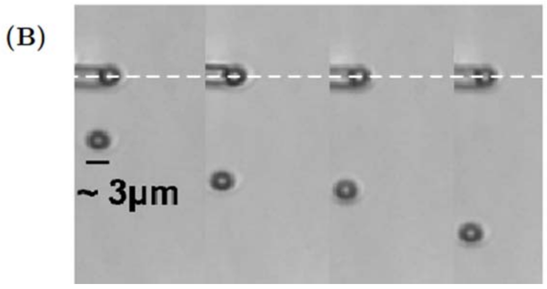


Figure 4. **A**: A schematic of the horizontal magnetic tweezers instrument used for the single-molecule DNA manipulation experiments. **B**: Representative images from a single-molecule DNA-protein extension experiment. Shown are the two beads, which are connected by a single molecule of dsDNA. The force is increased on the molecule in the images from left to right, and the DNA-histone complexes subsequently dissociate and the DNA resumes its normal contour length. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

This glass chamber is built using #1 glass cover slips for the floor and ceiling with 1 mm thick cut pieces of glass microscope slides forming three walls, while the fourth side remains open for movement of the glass micropipette. Also attached within the glass slide chamber is a small permanent magnet. As the chamber moves relative to the stationary micropipette and DNA strand, the magnet's field imparts variable force upon the super-paramagnetic bead and therefore tension along the DNA strand in the range of 0.05–100 pN. The magnet can be moved towards the functionalized DNA strand at rates as low as 0.320 μm/s.

The objective used is a 40×, 0.65 NA bright-field objective with a working distance of 500 μm. A CCD camera collects grayscale AVI format movie files. The movies are analyzed post-experiment using MATLAB software to derive the locations of the beads and the force by way of the fluctuation-dissipation theorem. For experiments in which proteins were injected, bound to the DNA strand, and subsequently mechanically driven-off the tether, the step finding algorithm was used on the interbead distance measurements to deduce the DNA extensions at which protein ruptures occurred.

The general procedure for a DNA-protein binding experiment begins by filling the sample chamber with the desired buffer and then placing 20 μL of bead-conjugated lambda-DNA into the sample chamber. The aspiration pipette is used to produce a negative pressure on the polystyrene bead of a DNA-bead complex. Upon successful acquisition of the DNA strand, it is moved to within 2,000 μm of the magnet. At this distance, the magnetic field strength is such that ~1 pN of force is applied along the DNA molecule. This force is sufficient to observe full extension of the DNA molecule and to note any behavior that may be abnormal. If the DNA complex is fully extended and showing appropriate behavior, the proteins are injected at a rate of roughly 1 μL per minute toward the DNA molecule. The DNA molecule is then observed as the proteins bind to it. When the proteins are fully bound and there is no longer any change in the extension of the DNA molecule, the distance between the DNA and magnet is reduced by moving the magnet towards the DNA molecule that is being held steady over the objective. As the magnet moves closer, the field strength increases and yields a greater force on the DNA molecule. In Figure 4b, we show several images from an experiment in which the force is increased and the DNA length is recovered. Proteins that are bound to the DNA molecule will then rupture; different proteins will dissociate from the molecule at different forces. In between protein ruptures, the DNA molecule will be at a constant extension. Locating these regions of constant extension is the task of the step-finding algorithm and the results of those locations is used to infer the binding and unbinding characteristics of the proteins.

Results

Simulation Results

Simulated data were generated as described in the previous section to test the performance of the algorithm in response to changes in each individual step parameter: length, height, and separation. We refer again to Figure 1 to define these parameters, where the step length describes the duration of the dwell at a certain position or y-value, the step height describes the vertical distance between successive dwells, and the step separation describes the horizontal distance between successive dwells. Additionally, for all permutations of these parameters tested, the noise width was changed between 1 and 9. Tests were carried out by holding two parameters constant at 50 (arbitrary units as discussed in the methods section) and varying the other parameter from 0 or 1 to 90. Since step length or height of zero eliminates any plateaus in the data, those parameters start at 1. However, a step separation of 0 is possible as it corresponds to a canonical step function, and so for step separation the simulated data begins at zero. While holding two parameters constant at 50 and varying the other from 0 or 1 to 90, each parameter triplet is repeated 100 times by drawing from a distribution with a given noise width.

For example, the first parameter set of [1,50,50] for [height, length, separation] is generated 100 times by drawing the noise component from a normal distribution of width 1. This parameter set is then repeated 100 times with noise of width 3; then 100 times with width 5; and so on until noise width of 9.

Next, the independent parameter (in this case height) would be increased to 10, and so a parameter set of [10,50,50] is simulated 100 times, first with noise width 1; then with noise-width 3, and so on until noise width 9. This process is repeated until finally a parameter set of [90,50,50] is generated. This process is similarly repeated for step length by varying the parameter triplet from [50,1,50] to [50,90,50], with each parameter triplet used to generate 100 traces for each of 5 different noise widths. Therefore, each single parameter is tested for 10 different values, at 5 different noise widths, and for 100 replications (up to noise-induced fluctuations); in all, this led to 5,000 runs for each parameter, and 15,000 data sets in total. For all of these trials, the data was analyzed without any preprocessing or filtering.

To test the algorithm in more realistic cases, it was used on both randomized simulated data and experimental data. The randomized simulated data were generated as described in the methods section, with input parameters of [20 200] for all three parameters of length, height, and separation. Thus, each step is randomly assigned a unique linking segment slope and step length. With parameters for step separation and step height ranging from 20 to 200, linking segment slopes between 10 and 1/10 are possible for instance. These ranges were chosen as they are typical of the parameters observed in the DNA-protein experiments. Normally distributed noise was added with noise width σ ranging from 1 to 5. These trials were analyzed both without and with the bilateral filter preprocessing step. Additionally, experimental data were used from DNA-histone binding and unbinding tests described in the methods section.

Algorithm Performance

The relationship between algorithm effectiveness and step height is shown in Figure 5a. This figure plots the percent of steps found against the step height for various noise width σ. The data sets for these trials were not filtered prior to analysis. It is observed that at very low step heights—in this case, step height of 1 is equal to or less than the noise width—the algorithm has difficulty identifying steps, with only about 10% of steps found for all σ. A general trend is evident in Figure 5a that the larger the σ, the larger the step height needs to be for the algorithm to correctly identify the steps. For $\sigma = 9$, the algorithm has difficulty identifying >60% of steps for any step height value. Correctly identifying >90% of steps for noise widths of 1, 3, and 5 requires step heights of 20, 50, and 80, respectively. This relationship between step height and noise is expected since noise introduces variability of the signal in the vertical direction, and it follows that greater noise width values require greater vertical step heights in order for the algorithm to discriminate between neighboring steps.

In Figures 5b and 5c, we plot the false positive and false negative rates for these trials. For $\sigma = 1$, both rates fall rapidly to <5% by a step height of 20, and eventually reach zero false step identifications. When $\sigma = 3$, the algorithm requires a step height of 50 before it has false positive and false negative rates of less than 5%. In general, we observe an inverse relationship between the percent steps found and the rate of false positives

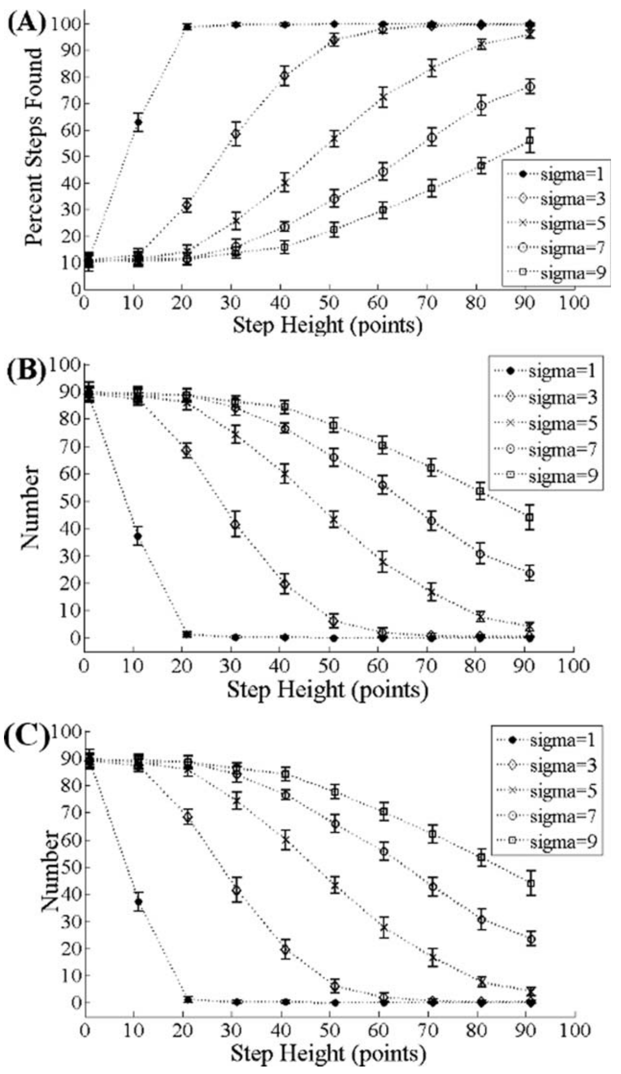


Figure 5. **A**: The average percent of steps found for trials with variable step height is plotted for several noise width σ. The unfiltered data were analyzed for these simulations. In these trials, step length and step separation were held at 50 points each while step height was assigned values of 1, 10, 20, 30, 40, 50, 60, 70, 80, and 90. Each step parameter tuple, that is, [40,50,50], was tested with 100 different sets of normally distributed noise. **B**: The average number of false positive step identifications are plotted for the same trials. **C**: The average number of false negatives for the same trials are plotted. Step height in arbitrary units.

and false negatives. We also find that the rates of false positives and negative are essentially equal to each other for these trials.

In Figure 6a, the performance of the algorithm as a function of step length and noise is plotted. Again, the raw data was analyzed without any filtering. Similarly to the step height plot, we found that a longer step length—and therefore more points at that y-coordinate value—increased the likelihood that the algorithm would identify steps correctly. For step length of 1, it is extremely difficult for the algorithm to extract steps since the "gap" between successive sloped linking segments is minimal and steps are nearly nonexistent. For each σ, there comes a point where an increase in step length does not

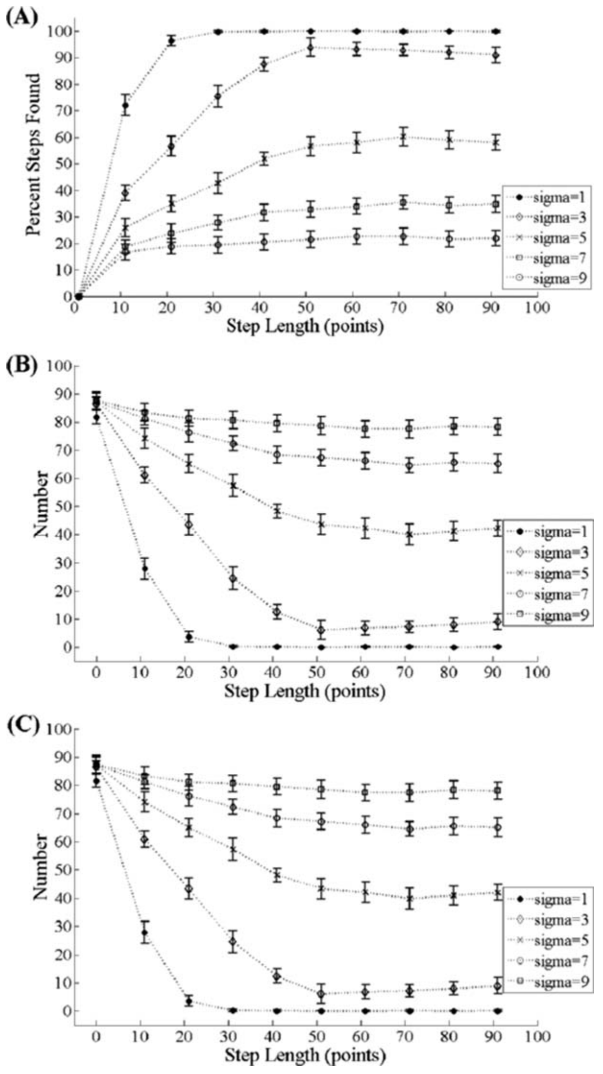


Figure 6. A: The average percent of steps found for trials with variable step length is plotted for several noise width σ. The unfiltered data were analyzed for these simulations. In these trials, step height and step separation were held at 50 points each while step length was assigned values of 1, 10, 20, 30, 40, 50, 60, 70, 80, and 90. Each step parameter tuple, that is, [40,50,50], was tested with 100 different sets of normally distributed noise. **B**: The average number of false positive step identifications are plotted for the same trials. **C**: The average number of false negatives for the same trials are plotted. Step length in arbitrary units.

have much impact on the ability of the algorithm to correctly identify the steps. With $\sigma = 1$, the algorithm reaches 100% effectiveness at step length of 30. For $\sigma = 3$, the algorithm reaches maximum effectiveness of ~92% at step length of 50 and remains close to that for the rest of the step lengths. Similarly, increasing σ to 5 shows a maximum effectiveness at step length of 70, with higher step lengths remaining close to ~60% effectiveness.

Figure 6b shows the rates of false positives for these trials, and Figure 6c plots the false negatives. These plots show a similar behavior to that of the response to step height. For $\sigma = 1$, the rates decline to near zero as the step length approaches 20. For $\sigma = 3$, the false positive rate reaches 5% at a step length of 50 and then remains relatively constant. Similarly, when $\sigma = 5$, the algorithm maintains an essentially constant rate of 50% false positives after step lengths of 60. For high values of noise, such as $\sigma = 9$, the rates of false positives and false negatives both see very little fluctuation over all ranges of step length at around 80%. Again, as with step height, these plots for false positives and false negatives essentially mirror the plots for the percent of steps found.

The algorithm performance as a function of step separation is shown in Figure 7a. As in Figures 5 and 6, the data in these trials were processed without any filtering. In opposition to the other two parameters, algorithmic effectiveness here is inversely proportional to step separation. For very low noise ($\sigma = 1$), the algorithm can identify nearly all steps for each step separation value in the plot. As the noise width σ increases, the decrease in algorithm effectiveness occurs at lower step separation values as expected. For noise width $\sigma = 3$, the algorithm shows reasonable performance of >95% recognition until the step separation reaches 50. Step recognition for $\sigma = 5$ drops below 95% when step separation is >20, and at $\sigma = 7$ the recognition rate is above 95% only for a step separation of 0. A larger σ inhibits the algorithm's ability to successfully identify >90% of steps for any step separation.

Figures 7b and 7c plot the false positive and false negative rates, respectively, for these trials. At $\sigma = 1$, the algorithm is able to limit false identifications of both kind to under 5% for all step separation values. For $\sigma = 3$, only after a step separation of 50 does the algorithm exceed 5% false identifications for both positives and negatives. Increasing the noise width to $\sigma = 5$, we find that beyond step separation of 30, the algorithm exceeds 5% false identification rate. Akin to the rates for false identifications in trials, which vary the step height and length, the rates for false identifications as step separation varies is essentially inverse to the rates of steps found as plotted in Figure 7a.

We also analyzed data sets with Poisson-distributed noise and found that the algorithm's performance was generally unaffected by the choice of the noise distribution. See Supporting Information for details.

Typical data from an experiment as discussed in Methods and Materials, and as shown in Figure 4b, is a plot of bead separation (DNA extension) over time. Figure 8a shows a subset of this information along with the results of applying the step-finding algorithm to experimental data. The black lines were automatically identified by the step-detection algorithm and serve to identify the areas of constant DNA extension. In between the black lines are regions of increasing extension, which is the result of a histone complex dissociating from the DNA molecule. Once these extensions are determined for an experiment, the change in extension between subsequent steps is calculated. A histogram of the step jumps for several experiments is plotted in Figure 8b. These results show peaks in the number of rupture events at 50, 90, and 130 nm. These data agree with the known model for DNA-histone octamer binding which loops about 45–50 nm of DNA in approximately

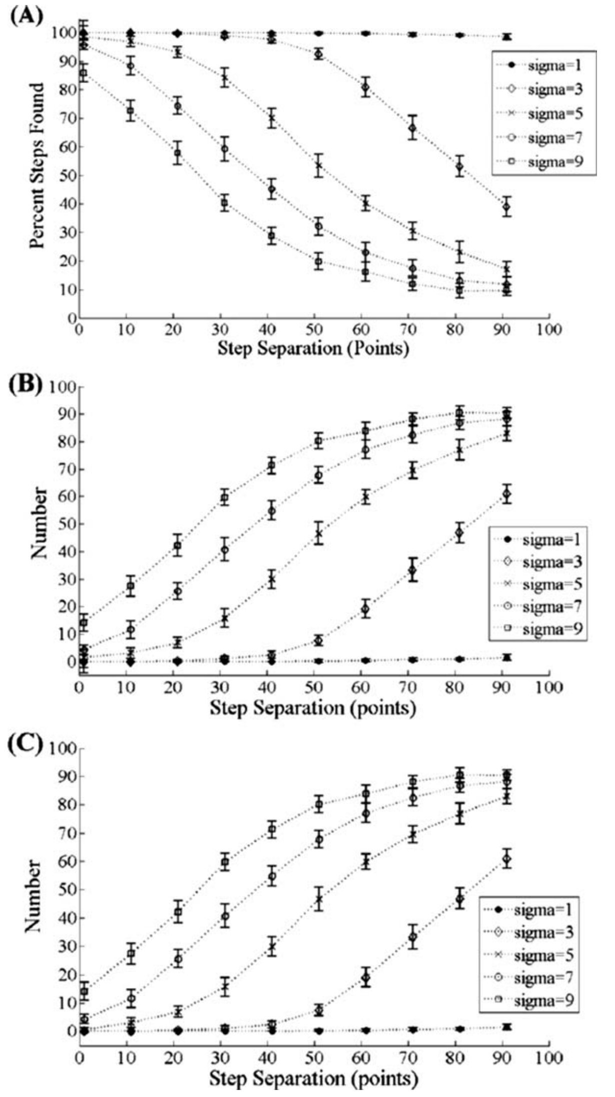


Figure 7. **A**: The average percent of steps found for trials with variable step separation is plotted for several noise width σ. The unfiltered data were analyzed for these simulations. In these trials, step length and step height were held at 50 points each while step separation was assigned values of 1, 10, 20, 30, 40, 50, 60, 70, 80, and 90. Each step parameter tuple, that is, [40,50,50], was tested with 100 different sets of normally distributed noise. **B**: The average number of false positive step identifications are plotted for the same trials. **C**: The average number of false negatives for the same trials are plotted. Step separation in arbitrary units.

one and three quarter turns around each core particle (14). Thus, jumps of 90 nm should represent two histone octamers dissociating at the same time, 130 nm jumps three histone octamers, and so on. Additionally, the force and step jump distance can be used to calculate the binding energy of the histone octamers. These calculations result in energies of about 20 k_bT (data not shown) which agrees with estimates in the literature (15,16).

In Figure 9, the results of randomized step parameter testing are plotted, trials which most closely represent actual data from single-molecule DNA extension experiments. The solid line represents the percent steps found when the raw unfiltered data are analyzed using the algorithm, while the dashed lines utilize bilateral filter preprocessing. Each point represents the average percent steps found for 100 replications at each noise width σ. The unfiltered analysis shows $\sim$90% steps found for $\sigma = 1$ to $\sigma = 4$, dropping down to 80% at σ equal to 5. However, with preprocessing, performance improves to >90% steps found at all noise widths. The exception is for $\sigma \leq 2$ where using the unfiltered data results in better performance. Also, note that the unfiltered data have increasing standard error in the percent steps found as noise width grows, while the filtered data sets maintain a relatively constant standard error for all σ. Also in Figure 9, we show the false positive results from these trials. The dotted line shows the false positives from the trials on raw data. We see that false positives number less than on average for noise widths of <2, growing in number to 27 at a noise width of 5. Conversely, the false positives for the trials in which bilateral filter preprocessing was used are plotted with the dash-dot line. Interestingly, the number of false positives decrease over the range of noise widths tested here, going from about 12 false positives at $\sigma = 1$ to 8 false positives at $\sigma = 5$.

The algorithm's processing times were also evaluated as a function of the number of steps and the number of points in a data set. The relevant methods and results are described in the Supporting Information.

Discussion

The step-detection algorithm we present provides a tool for unbiased automatic detection of plateaus in a data signal. We have demonstrated that it is able to accurately determine plateaus in a data set for certain ranges of step height, length, separation, and noise. Because we approach the step detection problem in a novel way, we believe that our algorithm provides a robust alternative to existing step-detection algorithms.

An important feature of our algorithm is that it does not rely on statistical fitting of canonical step functions to the data. Most of the step-detection algorithms discussed in the introduction uses such an approach to fit a piecewise series of canonical step functions, such as the Heaviside function, to the data. For a data set that consists of strict step functions, such an algorithm would be appropriate. However, for data sets in which the plateaus of the signal are not canonical, that is, two adjacent plateaus are connected by a line with a finite slope, such fitting algorithms will not be ideal. Our method, however, relies on relative probabilities within the data to determine regions expected to have a plateau. As shown in Figure 9, even when the slopes of lines connecting plateaus change from step to step, the algorithm correctly identifies plateaus in the data set.

The trials plotted in Figure 9 most closely resemble the data traces for single-molecule DNA experiments and the ability of the algorithm—when used with preprocessed data—to correctly identify and locate >90% of steps in these trials is critical to a faithful analysis of experimental data.

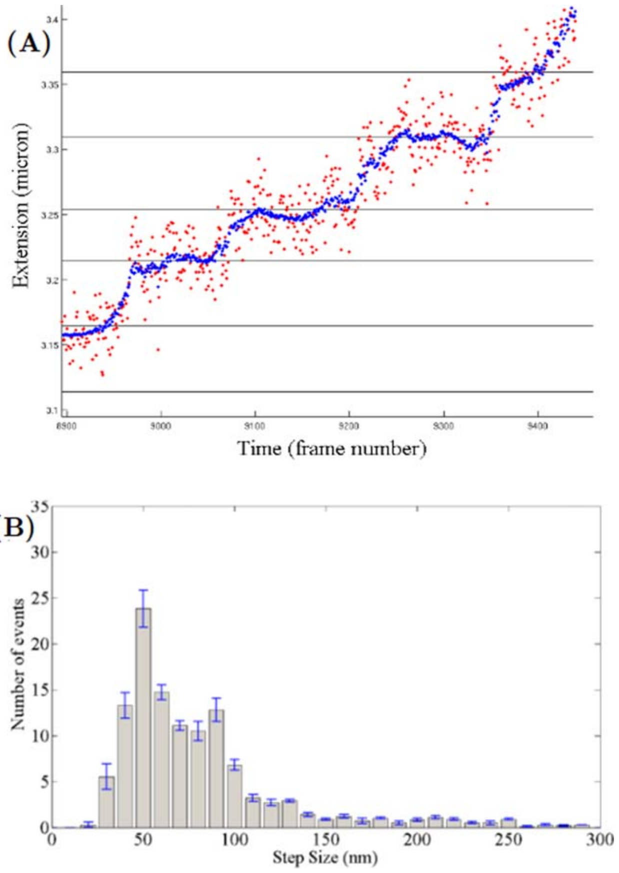


Figure 8. **A**: An example of steps in a single-molecule DNA-protein extension experiment that have been automatically identified by our algorithm. The red points are raw data and the blue points are data after bilateral filtering with height 12 and width 12. The black lines are the steps as identified by our algorithm. **B**: The average step size distribution after analysis of 10 different DNA-protein experiments by the algorithm. The results are plotted as a histogram with bins of 10 nm. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

When we applied our method to analyze data from single molecule micromanipulation experiments, we found that the results were consistent with parameters reported in the literature. For instance, the distribution of lengths of DNA wrapped around each histone octameric core particle was found to be centered at the correct values: ~50 nm and multiples of the same. In addition, the number of such particles detected was consistent with the observed change in DNA tether extension and the estimated maximum number of octamers that could populate the tether. The free energies of binding calculated, ~20 k_BT were also consistent with the values reported in ensemble and other single molecule experiments. These results strongly suggest that when our algorithm is applied to experimental data, the constant-extension regions are correctly enumerated and quantified.

Another aspect of many step-detection algorithms is that they rely on numerous user-defined parameters to achieve a good fit of the step-functions to the data. Aside from making such algorithms complicated to implement for people unfamiliar with the definitions of the parameters, manipulation of these parameters can lead to over-fitting of the data. From the beginning, our approach was designed to minimize the difficulty of using the algorithm and to reduce the chance of over-fitting of the data. Our method can be implemented easily and quickly since any user input or modification is optional. The algorithm we have presented also does not involve fitting of a specific function to the data, which eliminates any chance of over-fitting. Rather, we base our approach on finding the regions of the data more likely to represent a plateau and, in this sense, our approach may be deemed non-parametric.

We sought to minimize user input so as to reduce user bias. In searching for certain features of the data, a user may be able to adjust algorithm parameters to confirm a specific result that they would like to find. It is all too easy to desire that a small step in the data is attributable to noise when it should not be, or vice versa, and to adjust the parameters to either include or exclude those steps. By eliminating most user input and relying on the mathematical properties of the data itself, our step-detection algorithm offers a substantial reduction in the possibility for user bias to alter results.

With regard to user-defined parameters, the primary values in our algorithm that can be adjusted are the bin size for the discrete probability distribution function and the tolerance for determining which peaks are to be taken as identifying steps in the data. Our method for automatically determining the bin size for the distribution function is sufficient for many applications, but in situations where the steps are closely spaced or the noise is substantial (and no filtering is desired), it can be useful to decrease or increase the bin size parameter. The peak tolerance parameter can be adjusted or the value determined by comparison to the median path lengths of all peaks (see Materials and Methods section) can be used as default. It can be advantageous for the user to set this value when automatic determination of the plateaus fails to recognize data plateaus that are significantly shorter than the majority of the plateaus in the data. A user may wish to modify this tolerance parameter to expand the number of peaks that are classified as representing a step in the data, but adjusting this too far can lead to false positives.

While our algorithm is designed to detect data "plateaus," which may or may not be true step functions, the algorithms discussed in the introduction are specifically tailored to fit a series of step functions to the data. As we have shown, our algorithm performs very well when analyzing data, which are not a sequence of pure step-functions. To the best of our knowledge, there are no other well-characterized methods capable of doing so. This lack of other methods was a primary driver behind developing our algorithm, but it also makes direct performance comparisons to existing algorithms difficult.

For the trials plotted in Figures 5 through 7, in which a single step parameter was altered, we find that false positives equal the false negatives. (The number of false negatives is simply the difference between the total number of steps and the number of steps found.) It should be noted that in principle the false positives can be larger or smaller in number than the false negatives. That they happen to be equal in number to

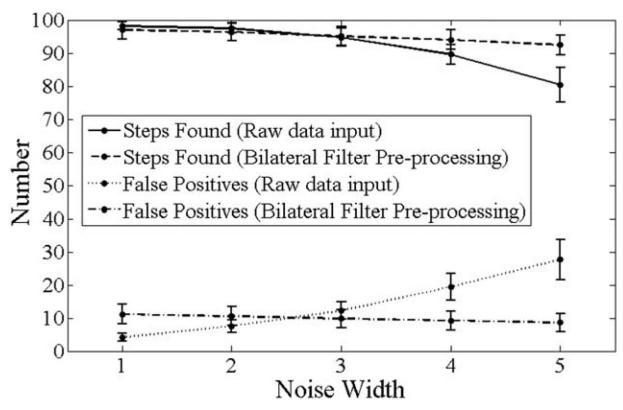


Figure 9. Results for randomized step parameter simulations are plotted. For these trials, the step parameters were assigned values randomly in the interval [20,200]. Thus, each step has a step height between 20 and 200 points, step length between 20 and 200 points, and step separation between 20 and 200 points. These parameters closely mimic the values seen in the single-molecule DNA-protein experimental data. For each noise width σ, the trials are repeated with 100 different sets of normally distributed noise. We show in Figure (9), the average percent steps found for these trials for raw data (solid line) and data that has been preprocessed with the bilateral filter (dashed line) using a filter height of 10 and a filter width of 10. Also plotted are the average false positives for the raw data analysis (dotted line) and the average false positives for the preprocessed data (dash-dot line). Noise width in arbitrary units.

each other strongly suggests that the algorithm is finding only those steps present in the signal—not more or less—but for some of the steps is not assigning them accurately enough to the correct locations. This result is likely due to the criterion that we use to declare that a step has been correctly detected, namely, the step must lie within one-half σ of the true location of the signal.

The relationship between false positives and negatives is most likely unique to the type of signals used in Figures 5 through 7—signals in which all step parameters are the same for all steps within an individual signal. Specifically, if the calculated arc lengths for all significant peaks are relatively equal—as would be the case for a data trace in which all steps have the same configuration—it is trivial to classify which peaks correspond to actual steps as opposed to which peaks result from fluctuations due to noise. Indeed, the results plotted in Figure 9 show that when step parameters are not uniform throughout the data trace, the number of false positives is larger than the number of missed steps. This indicates that the algorithm is inferring steps where there are none, rather than being unable to accurately locate the actual steps as in Figures 5 through 7.

Part of our method that may seem counterintuitive is the addition of the second derivative of the PDF back into the original PDF. This is opposite to image processing in which the second derivative is frequently subtracted from the image to enhance edge transitions of the image as in unsharp masking (17). It is important to note that we are applying the second derivative to the PDF and not the data (or image) itself. With that clarified, our justification for this step is primarily empirical in that we have run multiple trials using subtraction of the second derivative, as well as skipping the second derivative step altogether, and find optimal performance when the second derivative of the PDF is added back to the signal. Figure 3e illustrates this effect, where the magenta signal is the sum of the PDF (red) and second derivative (green), resulting in better isolation of the significant peaks in the signal. This is an effect we see repeatedly and is the reason we include this step in the algorithm. A theoretical justification of this process can be realized by examining individual peaks in Figure 3e. It is evident that peaks in the PDF correspond to large negative values in the second derivative. The sharpness of the peaks will generally be uncorrelated to the size of the peaks. Therefore, when the second derivative is added back to the PDF, the peaks in the PDF will see moderate reduction in magnitude. However, since most peaks in the PDF are about equally sharp but differ in height, addition of the second derivative amounts to subtracting a relatively constant quantity from the peaks. This clearly effects the shallower peaks more profoundly than the taller peaks, helping to reduce the identification of nonsignificant peaks.

An aspect of our method that must be acknowledged is that we use it primarily on monotonic data sets and demonstrated it on monotonic data sets. The algorithm was designed to evaluate the changes in the extension of a DNA molecule under increasing tension as proteins dissociate, which is often a monotonic function of time. Although the algorithm is capable of finding plateaus in nonmonotonic data sets, such use is not ideal. Developing an extension of this method to improve the ability to analyze such nonmonotonic data is a primary goal of future work on this algorithm. Furthermore, we hope to be able to expand the basic framework of this algorithm to multiple dimensions as well.

In summary, we have shown in detail the ability for our novel algorithm to identify steps, or plateaus, in a data trace. We have examined the limits of the algorithm's performance for several characteristics of the data. Furthermore, our method has proven useful in actual single-molecule biophysical experiments, obtaining results that agree with theoretical models and other experimental results.

Future modifications of the algorithm will focus on three aspects. Specifically, we suspect that a local adjustment of the bin size for calculation of the probability distribution function would be more effective than the current global definition of bin size for the distribution function, and a peak scoring method that takes into account the transient segment characteristics, instead of simply the plateau segment characteristics, could more accurately classify which peaks represent plateaus in the data. Finally, taking into account the breadth of the peaks that result from the analysis of the data trace, either globally or in a subset of peaks, should help to increase the accuracy of locating the steps in the data trace.

Literature Cited

1. Neuman K, Saleh O, Lionnet T, Lia G, Allem J, Bensimon D, Croquette V. Statistical determination of the step size of molecular motors. J Phys Condens Matter 2005;17: S3811–S3820.
2. Kalafut B, Visscher K. An objective, model-independent method for detection of non-uniform steps in noisy signals. Comput Phys Commun 2008;179:716–723.
3. Carter N, Cross R. Mechanics of the kinesin step. Nature 2005;435:308–312.
4. Sadler BM, Swami A. Analysis of multiscale products for step detection and estimation. IEEE Trans Inf Theory 1999;45:1043–1051.
5. Kerssemakers JWJ, Munteanu EL, Laan L, Noetzel TL, Janson ME, Dogterom M. Assembly dynamics of microtubules at molecular resolution. Nature 2006;442:709–712.
6. Levi V, Gelf V, Serpinskaya A, Gratton E. Melanosomes transported by myosin-V in xenopus melanophores perform slow 35 nm steps. Biophys J 2006;90:L7–L9.
7. Arunajadai SG, Cheng W. Step detection in single-molecule real time trajectories embedded in correlated noise. Plos One 2013;8:e59279
8. Kuo SC, Gelles J, Steuer E, Sheetz MP. A model for kinesin movement from nanometer-level movements of kinesin and cytoplasmic dynein and force measurements. J Cell Sci (Suppl) 1991;14:135–138.
9. Block S, Asbury C, Shaevitz J, Lang M. Probing the kinesin reaction cycle with a 2D optical force clamp. Proc Natl Acad Sci USA 2003;100:2351–2356.
10. Milescu LS, Yildiz A, Selvin PR, Sachs F. Maximum likelihood estimation of molecular motor kinetics from staircase dwell-time sequences. Biophys J 2006;91: 1156–1168.
11. Herbert KM, La Porta A, Wong BJ, Mooney RA, Neuman KC, Landick R, Block SM. Sequence-resolved detection of pausing by single RNA polymerase molecules. Cell 2006;125:1083–94.
12. Yang D, El Naqa I, Aditya A, Wu Y, Goddu M, Mutic S, Deasy JO, Low DA. DIRART—A software suite for deformable image registration and adaptive radiotherapy Research. World Congr Med Phys Biomed Eng 2009;25:844–847.
13. McAndrew C. Studies of Single DNA-Histone Binding Events Using a Novel Magnetic Tweezer. Thesis, Catholic University of America, Washington, DC; 2011.
14. Luger K, Mader AW, Richmond RK, Sargent DF, Richmond TJ. Crystal structure of the nucleosome core particle at 2.8 angstrom resolution. Nature 1997;389:251–260.
15. Marko J, Siggia ED. Driving proteins off DNA using applied tension. Biophys J 1997; 73:2173–2178.
16. Brower-Tol B, Walker DA, Fulbright RM, Lis JT, Kraus WL, Wang MD. Specific contributions of histone tails and their acetylation to the mechanical stability of nucleosomes. J Mol Biol 2005;346:135–146.
17. , Levi L. Unsharp masking and related image enhancement techniques. Comput Graph Image Process 1974;3:163–177.