Video-microscopy-based automated trajectory determination

Christopher Tyson,[1,2] Santosh Gaire,[1,3,*] Ian Pegg,[1,3] and Abhijit Sarkar[1,3]
[1]Vitreous State Laboratory, Washington, District of Columbia; [2]Department of Biomedical Engineering, The Catholic University of America, Washington, District of Columbia; and [3]Department of Physics, The Catholic University of America, Washington, District of Columbia

ABSTRACT We present a method for tracking densely clustered, high-velocity, indistinguishable objects being spawned at a high rate and moving in a directed force field using only object centroids as inputs and no other image information. The algorithm places minimal restrictions on the velocities or accelerations of the objects being tracked and uses a methodology based on a scoring function and a backtracking refinement process. This combination leads to successful tracking of hundreds of particles in challenging environments even when the displacement of the individual objects at successive times approaches the separation between neighboring objects in any one frame. We note that these cases can be particularly difficult to handle by existing methods. The performance of the algorithm is methodically examined by comparison to simulated trajectories, which vary the temporal and spatial densities, velocities, and accelerations of the objects in motion, as well as the signal/noise ratio. Also, we demonstrate its capability by analyzing data from experiments with superparamagnetic microspheres moving in an inhomogeneous magnetic field in aqueous buffer at room temperature. Our method should be widely applicable since trajectory determination problems are ubiquitous in video microscopy applications in biology, materials science, physics, and engineering.

WHY IT MATTERS Many scientific problems require automated methods for tracking objects in video microscopy data. New algorithms that can perform well in challenging contexts are required. Here, we describe a versatile method suitable for diverse applications spanning biology, materials science, physics, and engineering. Our algorithm is designed for situations involving densely packed, indistinguishable, and fast-moving objects in directed force fields and uses only their centroids as inputs. It imposes minimal constraints on object speeds or accelerations and incorporates a scoring system with refinement via backtracking. We thoroughly evaluate the algorithm's performance using a combination of simulations and experimental tests and show that it accurately follows many particles in challenging scenarios, for example when they are packed together closely and moving at high speeds.

INTRODUCTION

Automatic tracking of objects in images acquired through video microscopy is critical in many fields of engineering and science. The track estimation task in general involves two distinct steps (1). First, objects of interest—for example, particles, microspheres, cells, fluorophores—must be found and annotated in each image frame. Second, by comparing successive frames, trajectories for the annotated objects must be built up. The second task becomes easier if the objects are distinguishable, but the more challenging case is when the objects are indistinguishable. In this latter case, it is a priori not apparent how to map an object in one frame to itself in the next frame. If the objects are closely spaced, further complications arise since trajectory assignments for two neighboring objects may be switched by an algorithm without the altered trajectories deviating significantly from the true ones, making such errors especially hard to detect computationally.

A number of track determination algorithms have been developed and have performed with varying degrees of success in the most challenging case of densely clustered, identical particles moving at high speed—see (2), (3), and (4) and references (32–57) therein. These algorithms may be divided into two

Submitted November 2, 2023, and accepted for publication February 22, 2024.

*Correspondence: gaire@cua.edu

Editor: Jorg Enderlein.

https://doi.org/10.1016/j.bpr.2024.100148

categories: predictive tracking and measurement-assignment tracking. In predictive tracking, image data are used to estimate an ensemble of kinematic models, and based on this ensemble, the algorithms determine probabilities for the current observations conditional on each possible trajectory in the ensemble. These probabilities can also be augmented with additional data—for example, Anderson et al. use image intensity information (5), while other methods have considered distributions of errors for observed parameters and kinetic model fits (6). For densely clustered particles or significant noise, predictive tracking can offer an advantage. However, if the exact kinematic model is unknown or not accurately estimated—which can often be the case—then the effectiveness of these algorithms will suffer.

Measurement-assignment tracking algorithms typically involve "scoring" each potential trajectory assignment and then using the scores to determine the best assignment. Velocities, trajectories, smoothness, shape, and size of objects of interest are often used to compute scores and must be estimated from the data. A number of techniques use a Kalman filter (7) for this purpose (8). Constraints on the acceleration/deceleration, radius of turn, or inertia can be used to isolate only the objects of interest, increasing computational efficiency (9–12). However, one limitation is that these methods are primarily intended for tracking single objects under low noise conditions, although modifications exist to remove these constraints (13–15). An alternative to Kalman-filter-based methods is the multiple-hypothesis tracking algorithm (16–18), in which all measurements prior to the current observations are compared against a predefined kinematic model to generate a set of parent hypotheses. Measurements in the current observation, as well as dummy measurements for noise, new objects, and false positives, are assigned a hypothesis with respect to the parent hypotheses, and Bayes theorem is used to calculate the probabilities of each current measurement based on the prior measurements.

These techniques, although powerful, tend not to perform well for densely clustered, indistinguishable particles moving unidirectionally at high speed and spawned at high rates, and, thus, new algorithms are called for. Here, we present an algorithm that fills the gap. Our method can be thought of as a hybrid. Its core is a measurement-assignment tracking method—we do not use the data to fit an explicit kinematic model. However, it takes advantage of the underlying dynamics of objects moving in a unidirectional force field, much like a predictive tracking algorithm would, in order to provide an initial set of trajectories for the measurement-assignment-inspired step.

In the next section, we describe the algorithm in detail, as well as provide the procedures—simulation and experimental—used to validate it. This is followed by a detailed description of our results and a discussion of what implications they have for the accuracy and efficiency of our technique.

MATERIALS AND METHODS

Algorithm

We wish to analyze scenes consisting of densely clustered, indistinguishable objects moving at high speed. These objects may be spawned at high rates, and their number frame to frame is not conserved, as newly born objects add to the object population in successive frames and others exit the scene before traveling the entirety of the sensor's field of view. Further, objects may have a variety of entrance points, may exit the field of view at any point for any reason, and may have widely distributed velocities and other trajectory parameters.

The tracking algorithm involves two conceptually distinct but mutually supporting computations. The first involves a scoring function, which generates trajectory assignments for the particles, and the second involves the backtracking method, which takes as its input the scoring function output and further refines or modifies those trajectories. (The method presented here is independent of how particle centroid localization is performed.)

We illustrate the tracking algorithm concept in Fig. 1 with a simple example consisting of two objects, drawn as a square and a circle for clarity but considered identical by the algorithm, that have been observed at three different times; the times have been color

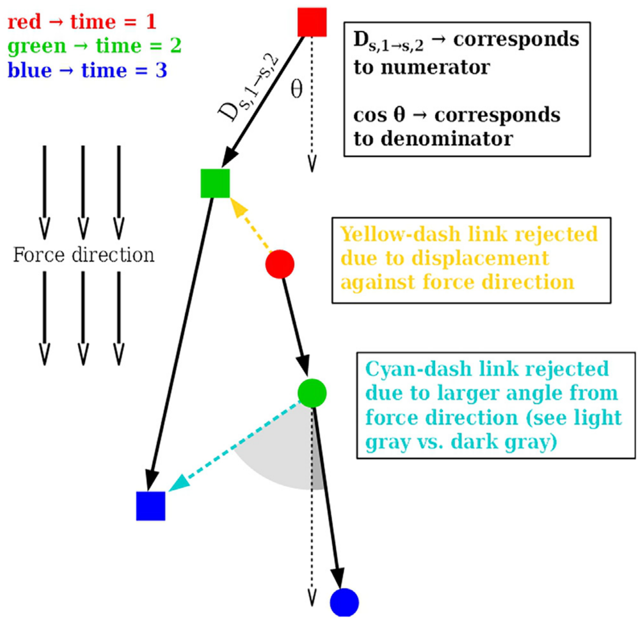
red → time = 1
green → time = 2
blue → time = 3
$D_{s,1\to s,2}$
θ
$D_{s,1\to s,2}$ → corresponds to numerator
cos θ → corresponds to denominator
Force direction
Yellow-dash link rejected due to displacement against force direction
Cyan-dash link rejected due to larger angle from force direction (see light gray vs. dark gray)

FIGURE 1 The key concepts of the algorithm. This figure highlights the method by which the scoring function assigns trajectories. The two identical objects, depicted as a square and a circle for clarity, are observed at three times, represented as red at time t = 1, green at time t = 2, and blue at time t = 3. The objects move in a directional force field as shown. The black arrows that connect the objects symbolize the trajectory links that must be determined. The various parameters for the scoring function are displayed.

coded as follows: t = 1 by red objects, t = 2 by green, and t = 3 by blue. The force field is pointing downwards. Arrows connecting the objects at two consecutive times are characterized by their magnitude and the angle θ that they make with the force direction. Starting with the red circle at t = 1, the algorithm must decide at t = 2 whether to assign the green circle or the green square to the red circle. This is done by computing the score for all vectors from the red circle to the green objects. As drawn, the link from the red circle to the green square (the dashed yellow vector) has a *y* component that points against the direction of the force. Thus, even though the green square is closer to the red circle than the green circle is, the green square link is rejected due to the "backward" movement, and the green circle is assigned to the red circle's trajectory. Accordingly, the green square would be linked to the red square's trajectory. In the next iteration, the blue objects (at t = 3) have to be linked to the green objects. Again, the blue square is slightly closer to the green circle then the blue circle is, and furthermore, in this case, the blue square does not show movement against the direction of the force. Yet, the angle that is formed between the vector connecting the green circle and the blue square is considerably larger than the angle for the vector connecting the circles, and thus the score for the blue square to green circle would be appreciably larger than the score for the blue circle to the green circle. Thus, the algorithm would assign the blue circle to the trajectory of the green circle (and therefor also the red circle) while assigning the blue square to the trajectory defined by the previous squares.

The scoring function, $W_{p,1;q,2}$, calculates a score for each object p with coordinates $(x_{p,1}, y_{p,1})$ and at time 1 with respect to objects q at time 2 with coordinates $(x_{q,2}, y_{q,2})$, i.e., it assigns a real number to every possible pairing of objects in two consecutive frames. The score is not bounded, can take negative, positive, and zero values, and is defined as follows:

$$W_{p,1;q,2} = \mathrm{sgn}(y_{p,1} - y_{q,2}) \times \frac{\sqrt{(x_{p,1} - x_{q,2})^2 + (y_{p,1} - y_{q,2})^2}}{\left|\cos\left(\arctan\left(\frac{x_{p,1} - x_{q,2}}{y_{p,1} - y_{q,2}}\right)\right)\right|} - DB_y \qquad \text{(Equation 1)}$$

This can be simplified for computational purposes to read

$$W_{p,1;q,2} = \mathrm{sgn}(y_{p,1} - y_{q,2}) \times \frac{(x_{p,1} - x_{q,2})^2 + (y_{p,1} - y_{q,2})^2}{|y_{p,1} - y_{q,2}|} - DB_y \qquad \text{(Equation 2)}$$

Referring to Eq. 1, the first term in the scoring function is the signum function, which gives the score a positive value for movements in the direction of the force (which we remind the reader is taken to be in the negative *y* direction) and a negative value for movements in the opposite direction. The numerator in Eq. 1 is the Euclidean distance between the two observations, while the denominator is the cosine of the angle formed between the direction of the force and the vector that connects the two objects. The final term, DB_y, is calculated by finding the difference in the mean *y* coordinates for all objects in two successive frames. However, since the location of a particle in the next frame has not been definitively assigned at this stage, it is calculated for different particle pairings, and the minimum value is used. This term serves to penalize observation pairs that do not show substantial movement in the direction of the force and approximates the minimum distance an object is expected to move between successive observations based on the gross behavior of all objects.

The scoring function is constructed such that small, positive scores are preferred, as these indicate movement in the direction of the force; large, negative scores, on the other hand, imply displacements in opposite or orthogonal directions.

The backtracking approach provides a quality check for the score-based track assignments. It builds on the idea that if an object is observed at (x_1, y_1) at time 1 and subsequently at (x_3, y_3) at 3, then it is likely that for an observation at an intermediate time 2, the coordinates (x_2, y_2) should fall close to the straight line connecting (x_1, y_1) and (x_3, y_3). This is accomplished by calculating the following quantity:

$$D_{1,2,3} = \left[\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} + \sqrt{(x_2 - x_3)^2 + (y_2 - y_3)^2}\right] - \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2}. \qquad \text{(Equation 3)}$$

$D_{1,2,3}$ is a measure associated with a frame triplet. It calculates the sum of distances from an object's intermediate position (at time 2) to its positions at times 1 and 3 and, from this, subtracts the distance between its positions at times 1 and 3. Larger D values indicate a significant deviation from the direct path between times 1 and 3, while smaller values represent closer alignment with the path; the limit of D = 0 represents the case of three collinear points.

An important advantage of the backtracking method is that it can provide a means to determine the extent of nonphysical trajectory assignments, i.e., imputed trajectories that involve movements against the direction of the force. It also checks for significant lateral movements. Close agreement between the two methods suggests that there are few, if any, unphysical trajectories in the set of computed trajectories. Further discussion of the various aspects of the algorithm and its implementation, including a simplified illustrative example, can be found in Fig. S1.

Particle trajectory simulations

Performance of the algorithm was tested against simulated trajectories generated using the following inputs: 1) the total number of objects generated over the course of a simulation n_p; 2) the size of the "image" in the *x* and *y* dimensions; 3) the spawn rate, the typical time between successive objects entering the field of view; 4) the initial position (x_0, y_0); 5) the initial velocity (v_{x0}, v_{y0}); 6) the acceleration (a_y) of each object and noise components drawn independently for *x*- and *y*-object coordinates from a normal distribution with mean zero; and 7) the variance σ, a measure of the signal/noise ratio. x_0 is defined as the row coordinate of the pixel where a particle enters the simulated field of view, and y_0 is the column coordinate. The inputs can be single valued or defined from an interval from which they may be drawn with uniform probability. The latter allows us to assess how well the algorithm can deal with objects with nonuniform behavior and also perform sensitivity analyses. The simulations continue until the final object exits the field of view of the "image." Object coordinates are determined using the basic kinematical equations.

$$x_{t2} = x_{t1} + (t_2 - t_1)v_{x0} \qquad \text{(Equation 4)}$$

$$y_{t2} = y_{t1} + (t_2 - t_1)v_{y0} + \frac{a_y(t_2 - t_1)^2}{2} \quad \text{(Equation 5)}$$

Here, (x_{t1}, y_{t1}) is the position of the particle at time t_1 and (x_{t2}, y_{t2}) is its position at t_2. Position is measured in pixels, velocity in pixels/frame, and acceleration in pixels/frame^2. Given a specific pixel size and frame rate, these values can be converted to physical units. As the time slices are uniformly separated, if an object spawns at a time between two time steps, its motion to the next time step is determined using the kinematical equations. Subsequently, it is simply a matter of stepping through time to calculate the spatial coordinates of the trajectory for each object. Noise is then added, and the final output is a list of observations (x, y, t) and trajectory ID numbers for each observation.

We selected simulation parameters to cover a range of cases that might exist in experiments or data processing tasks. Our choices for simulation parameters were motivated by simulations performed to recapitulate bead drop experiments (described under experimental validation and in the supporting material) These typically capture around 100 particles, so we set $n_p = 100$. Since the particles are injected from the micropipette in the manner of a vertical line source, we set $x_0 = [100, 100]$ and $y_0 = [1, 100]$. The magnetic beads have a horizontal velocity (x direction) component; however, we found minimal initial velocity in the y direction: $v_{x0} = [-0.5, 2.5]$ and $v_{y0} = [0, 0]$. For the spawn rate, there are a few instances in which as many as six particles appear simultaneously out of the pipette, which would imply a rate of six. However, most often particles may be separated by dozens of frames, which would be a rate of 1/12. Since we wanted to challenge the algorithm, we drew spawn rates from [6, 1/2]. Variations in the magnetic moments of the superparamagnetic particles imply variations in the magnetic forces on them. Thus, we expect small fluctuations around the mean acceleration. Moreover, the mean acceleration depends on the distance from the magnet. Thus, simulations were run with $a_y = [1, 2]$; [5, 6]; [9, 10]; [13, 14]; [17, 18]; [21, 22]; [25, 26]. Simulations were also performed at the four different noise strengths described previously. For further discussion on the rationale for the selection of simulation parameters, especially for a detailed discussion of the rationale behind the choice of the spawn rate $n_{p,}$ please see the supporting material.

Quantifying the algorithm's performance

Two criteria were used to compare the results of the algorithm to experimental data and the ground-truth trajectories known from simulations. First, the correct number of individual observation links is determined by comparing the simulation particle coordinates at each time step (ground truth) to the coordinates assigned to the same particles by the algorithm. From this, we determine if a particle has been correctly linked to itself at the next time step, and the total number of correctly identified links is recorded as a percentage of the known links.

The second approach utilizes the variation of information (VI) metric. This compares two partitions F and G of a set A, where each partition consists of disjoint subsets $F = \{f_1, f_2,..., f_e\}$ and $G = \{g_1, g_2,..., g_f\}$, by computing the following quantity:

$$VI(F; G) = -\sum_{i,j} r_{i,j}\left[\log\left(\frac{r_{i,j}}{p_i}\right) + \log\left(\frac{r_{i,j}}{q_j}\right)\right] \quad \text{(Equation 6)}$$

where $p_i = \frac{|F_i|}{n}$, $q_j = \frac{|G_j|}{n}$, $r_{i,j} = \frac{|F_i \cap G_j|}{n}$, and $n = \sum_i |F_i| = \sum_j |G_j| = |A|$. In our case, the set A consists of all observations in the simulations. The partition F is the correct trajectory assignments from the simulations and the partition G is the algorithm output trajectories. The disjoint subsets within F and G are the individual object trajectories. As defined, VI is zero if the two partitions are identical. As the difference between the partitions grows, VI grows as well; thus, low values of VI are desirable.

Experimental validation

We performed experimental validation of our method by finding the magnetic force on magnetic microparticles moving in a buffer in a magnetic field by using to inferred trajectories of magnetic microparticles to compute the drag force and by a second independent method and comparing the two results. See the supporting material for more details.

RESULTS

Performance and sensitivity analyses on simulation data

Figs. 2, 3, 4, 5, and 6 display the algorithm's performance when varying each simulation parameter (excluding total particles) individually. Each figure has two columns with eight plots, illustrating the algorithm's performance variation with changing parameters. The left column shows the percentage of correct trajectory links (averaged over 50 simulations), while the right column presents VI scores, computed by grouping all 50 replications into a superset. Each column includes four plots, arranged by noise strength (0–3). The x axis is spawn rate, and each plot represents specific spawn rates with grouped bars of

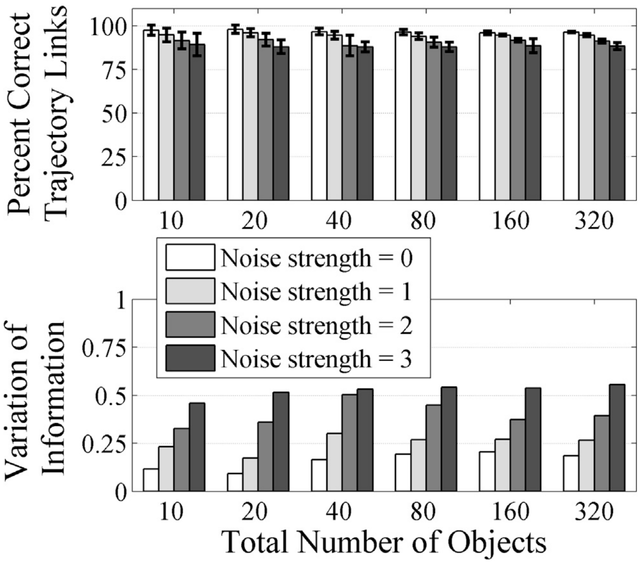

FIGURE 2 Results of simulations in which the number of total objects simulated was varied. All other simulation parameters remained in the baseline configuration with $x_0 = [1, 400]$, $y_0 = [1, 1]$, $v_{x0} = [0, 0]$, $v_{y0} = [0, 0]$, $a_y = [9, 9]$, and a spawn rate of 3/10, corresponding to 10 new objects every 3 frames. The simulations were run with added noise of strengths zero, one, two, and three as indicated by the legend.

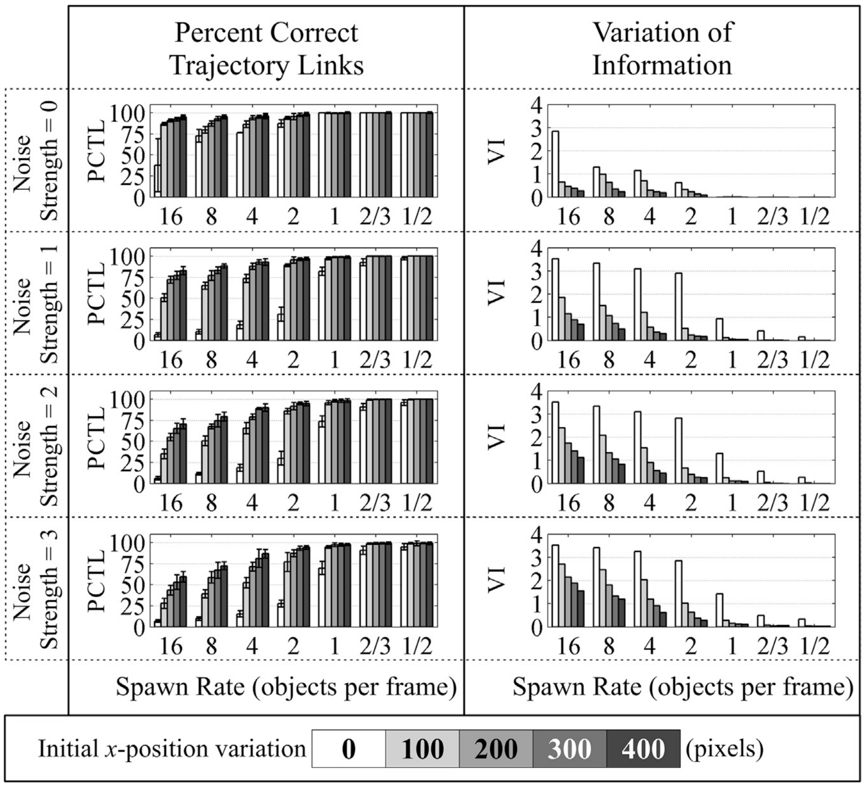

FIGURE 3 Results from simulations examining the initial *x*-position variation. While initial *x*-position variation was tested with values of 0, 100, 200, 300, and 400 (units of pixels), the remaining simulation parameters were held in the baseline configuration with $n_p = 30$, $y_0 = [1, 1]$, $v_{x0} = [0, 0]$, $v_{y0} = [0, 0]$, and $a_y = [9, 9]$. The two columns show the two performance metrics: percentage of correct trajectory links in the left column and variation of information in the right column. Four rows correspond to the four different noise strengths: zero, one, two, and three. The *x* axis for all plots is the spawn rate in units of objects/frame. Each spawn rate grouping contains five bars of different shades of gray, which represent the different initial *x*-position variations as shown in the legend at the bottom.

different shades of gray, indicating the varied parameter. See the legend for an explanation of the gray-shade values.

We varied the total number of particles tracked (10, 20, 40, 80, 160, 320) in simulations with a spawn rate of 10/3 and other parameters at baseline values (see

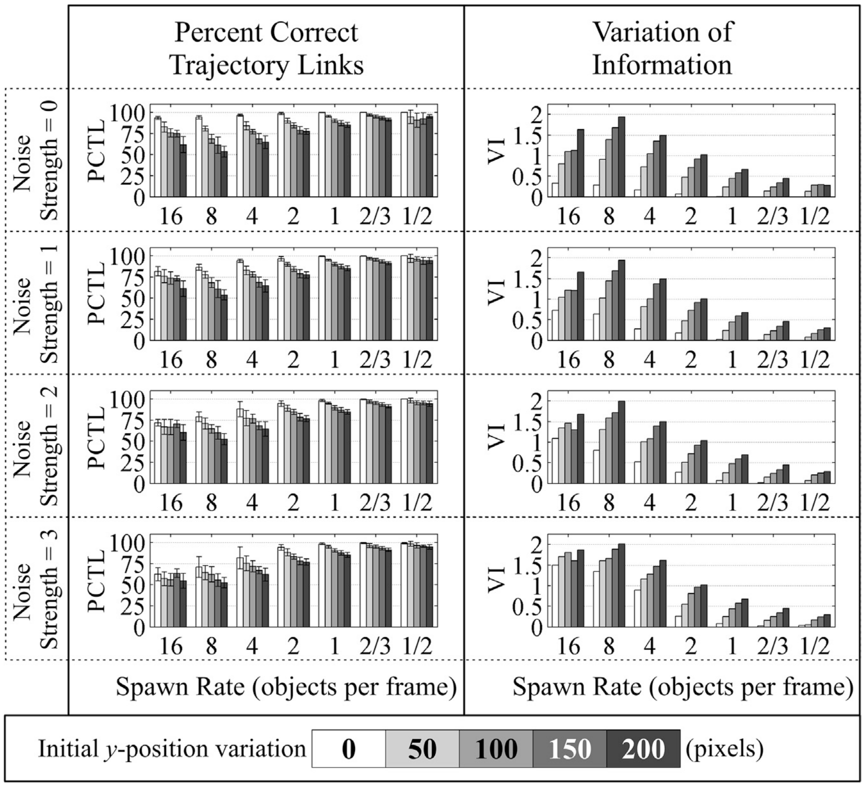

FIGURE 4 Results from simulations examining the initial *y*-position variation. While initial *y*-position variation was tested with values of 0, 50, 100, 150, and 200 (units of pixels), the remaining simulation parameters were held in the baseline configuration with $n_p = 30$, $x_0 = [1, 400]$, $v_{x0} = [0, 0]$, $v_{y0} = [0, 0]$, and $a_y = [9, 9]$. The two columns show the two performance metrics: percentage of correct trajectory links in the left column and variation of information in the right column. Four rows correspond to the four different noise strengths: zero, one, two, and three. The *x* axis for all plots is the spawn rate in units of objects/frame. Each spawn rate grouping contains five bars of different shades of gray, which represent the different initial *y*-position variations as shown in the legend at the bottom.

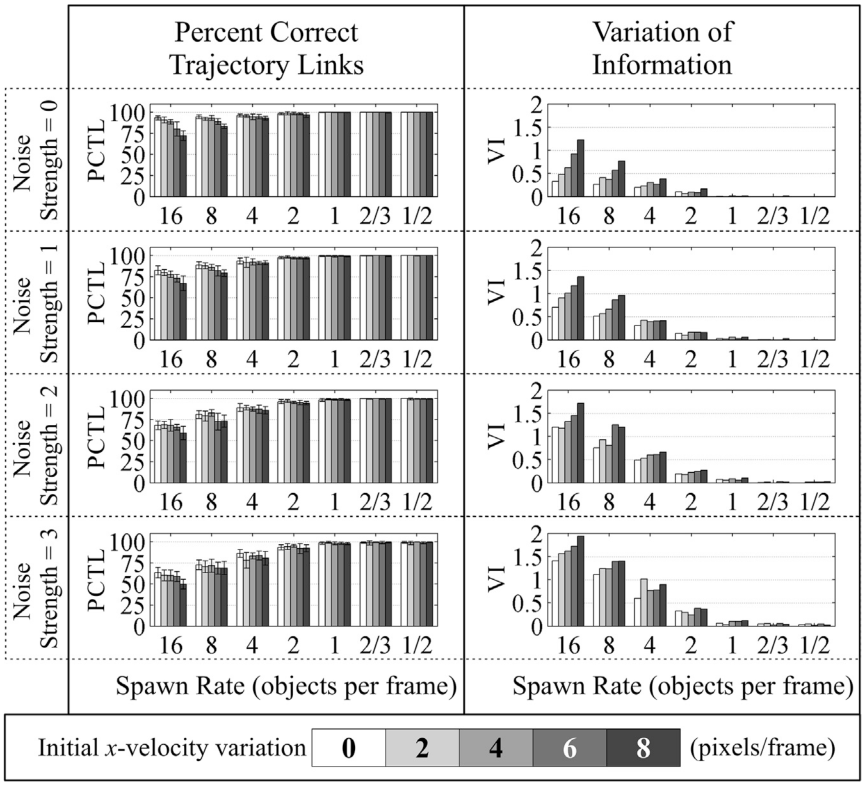

FIGURE 5 Results from simulations examining the initial x-velocity variation. While initial x-velocity variation was tested with values of 0, 2, 4, 6, and 8 (units of pixels/frame), the remaining simulation parameters were held in the baseline configuration with $n_p = 30$, $x_0 = [1, 400]$, $y_0 = [1, 1]$, $v_{y0} = [0, 0]$, and $a_y = [9, 9]$. The two columns show the two performance metrics: percentage of correct trajectory links in the left column and variation of information in the right column. Four rows correspond to the four different noise strengths: zero, one, two, and three. The x axis for all plots is the spawn rate in units of objects/frame. Each spawn rate grouping contains five bars of different shades of gray, which represent the different initial x velocity variations as shown in the legend at the bottom.

Fig. 2). The algorithm showed excellent performance with close to 100% correct links for all particle numbers tested. Results were weakly sensitive to noise, dropping to about 90% at noise level three for all n_p. Measured by VI, the performance degradation was expected but generally small (VI stayed below one for all noise and particle number values). However, for noise levels one and two, VI behaved nonmonotonically with particle numbers, increasing from 20 to 40, decreasing somewhat from 40 to 160, and rising again at 320.

In Fig. 3, we varied x_0 by drawing from intervals of different lengths [200, 200], [150, 250], [100, 300], [50, 350], and [1, 400]. Optimal performance occurs when x_0 is drawn from a wide distribution and the spawn rate is small (rate of one or lower). At rates between two and eight, wider sampling in x_0 leads to better performance; for instance, for a rate of four, the percentage of correct links varies from 70% to 95% as x_0 is sampled from intervals of length 0 to 400. VI results align with the percentage of link findings. For high spawn rates and point sources, the VI is close to three. However, without noise, performance significantly improves for rates of eight or less, with VI less than one, and almost zero for very low spawn rates. The trend remains consistent with the introduction of noise: performance improves the most as x_0 is drawn from longer intervals, approaching values with zero noise. Overall, we found weak sensitivity to noise across the parameter sets tested.

The initial y position intervals were taken from [0, 50, 100, 150, 200], as depicted in Fig. 4. Overall, the algorithm performed very well, with close to 100% correct identification and VIs less than one across a wide range of noise, spawn rates, and y_0 interval lengths. However, performance declined as the interval length increased, modulated by the spawn rate, and showed low sensitivity to noise when spawn rate and interval length were constant. At high noise (strength 3) and spawn rates (8 and 16), performance dropped to below 50% for trajectory link identification, especially with higher y position variation. Noise strength had limited impact on performance, except for specific cases like spawn rate two, where VI slightly increased from 0.1 to 0.25 at interval length zero for noise strength three. At lower spawn rates, the differences in VI for various noise values were negligible.

The initial x velocity (v_{x0}) was sampled from intervals with varying widths: [0, 0], [−1, 1], [−2, 2], [−3, 3], and [−4, 4], representing variations of 0, 2, 4, 6, and 8, respectively, as illustrated in Fig. 5. An increase in variation generally led to decreased algorithm performance for fixed noise and spawn rates. For instance, at noise level zero and spawn rate 16, the percentage of correct trajectory links decreased from 95% to 75%, and the VI increased from under 0.5 to close to 1.5 as

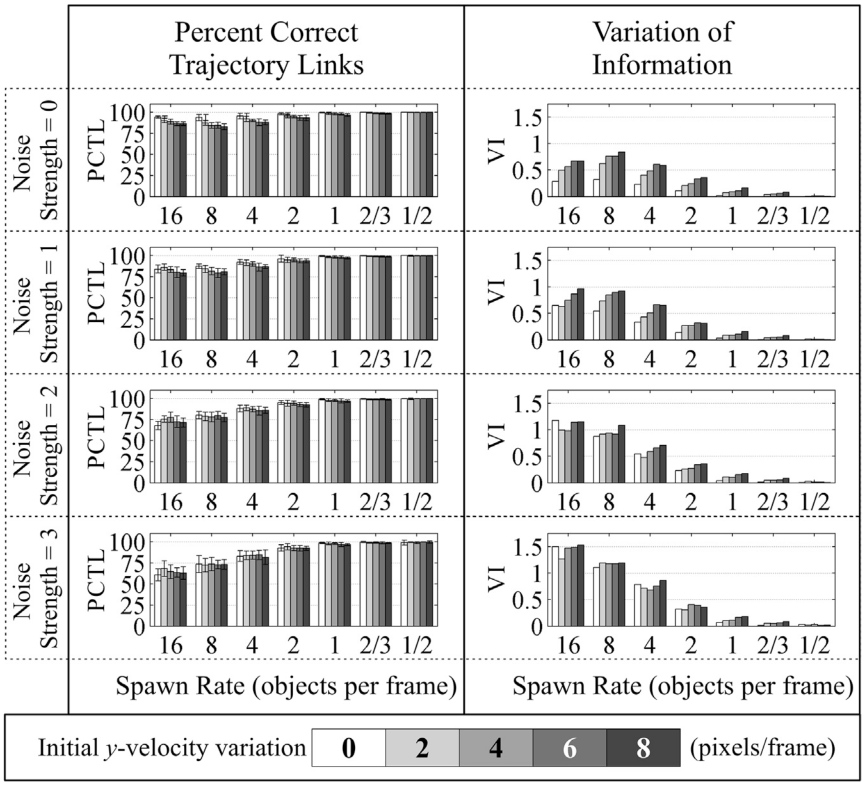

FIGURE 6 Results from simulations examining the initial y velocity. While initial y-velocity variation was tested with values of 0, 2, 4, 6, and 8 (units of pixels/frame), the remaining simulation parameters were held in the baseline configuration with $n_p = 30$, $x_0 = [1, 400]$, $y_0 = [1, 1]$, $v_{x0} = [0, 0]$, and $a_y = [9, 9]$. The two columns show the two performance metrics: percentage of correct trajectory links in the left column and variation of information in the right column. Four rows correspond to the four different noise strengths: zero, one, two, and three. The x axis for all plots is the spawn rate in units of objects/frame. Each spawn rate grouping contains five bars of different shades of gray, which represent the different initial y-velocity variations as shown in the legend at the bottom.

the interval widened. However, with lower spawn rates (e.g., two or less), the overall performance improved significantly, nearing 100% correct identification and VI close to zero. These results remained robust even with increased noise amplitude. The worst performance occurred at spawn rate one and noise three, but even then, over 95% of trajectory links were correctly identified, and VI remained below 0.25.

The simulations results presented in Fig. 6 were performed by selecting v_{y0} from the intervals [0, 0], [0, 2], [0, 4], [0, 6], and [0, 8], corresponding to variation lengths of 0, 2, 4, 6, and 8, respectively. At low noise strengths (zero and one), a slight decrease in algorithm performance was observed as initial y-velocity variation increased. However, at higher noise strengths (two and three), the performance difference between y-velocity variations disappeared. The percentage of correct trajectory links drops from the mid-90% range to the mid-80% range for spawn rates of 16, 8, and 4. VI measures did not show a discernible trend across spawn rates for noise strengths of two and three. In such cases, the alterations caused by high noise strengths overwhelmed any impact of initial y-velocity variations.

We performed additional simulations with altered initial y acceleration: $a_y = [9, 9]$; [7, 11]; [5, 13]; [3, 15]; [1, 17], representing variation in values of 0, 4, 8, 12, and 16, respectively (see Fig. S4). For zero noise, a y-acceleration variation of zero led to notably better algorithm performance in both the percentage of correct trajectory links and VI. However, for y-acceleration variations greater than zero, there was no clear trend in performance. The differences in VI values were less noticeable at higher noise strengths, and in some cases, VI for y-acceleration variation of 0 was greater than the maximum variation of 16. When the noise strength is 3 and the spawn rate is 16, we see that VI for y-acceleration variation of 0 is greater than when the y-acceleration variation is at the maximum, 16. Overall, there was a slight improvement in performance with low y-acceleration variation, but the difference was often minimal and within the standard deviation of other values.

Experimental tests

We also carried out experimental tests to assess the accuracy of our method by releasing neutrally buoyant, micron-sized magnetic beads suspended in buffer in the magnetic field of a small bar magnet. In the low Reynolds number condition of the experiment, the drag force, which is proportional to the velocity, is equal to magnetic force. By computing the magnetic force in two independent ways, we were able to compare the particle-trajectory-based method to a second, orthogonal method. We found good agreement between the velocity-based method, which used the particle tracking algorithm to determine velocity, and the reference method.

Please see the supporting material for simulation studies designed to recapitulate experimental conditions and details on the experimental tests, as well as studies on algorithm run time.

DISCUSSION

We have presented an algorithm for determining trajectories of closely spaced, indistinguishable objects that are spawned at high rates and are moving rapidly in a directional force field. The algorithm combines a scoring function that considers expected motion due to the force field with a backtracking method inspired by measurement-assignment techniques. In order to test its performance, we carried out simulation-based validation and sensitivity studies in which we systematically varied the spawn rate, initial conditions for object position and velocity (x_0, y_0, v_{x0}, v_{y0}), object acceleration (a_y), and noise strength or the signal/noise ratio (σ). We also looked at how the performance scaled with n_p, the total number of particles tracked. These were complemented by additional simulations designed to recapitulate microsphere tracking experiments (described in the supporting material), and experimental validation was performed as well. Results were quantified in terms of 1) the percentage of correct links identified and 2) the VI score.

Starting with n_p, we found a weak dependence on particle number. This is not surprising since we expect entry of new objects to be compensated, to some extent, by the exit of others, roughly leaving the same number of particles to be tracked in each frame independent of the total number of particles.

The performance was generally robust as a function of the initial kinematical parameters of the particles (x_0, y_0, v_{x0}, v_{y0}, a_y), with successful link detection of 90% or higher over a wide range of values for these quantities, even for objects with large variations in velocities or accelerations (i.e., when, in simulations, we drew these values from intervals of varying lengths). Also, we found that the spawn rate and, to a lesser extent, the noise strength played a bigger role in limiting the algorithm.

As x_0 is drawn from wider intervals, the resulting trajectories tended to be laterally spaced apart, which helped with the identification task, as misidentifications would lead to an object in the next frame undergoing a large lateral displacement, an assignment that was heavily penalized by the scoring function. Conversely, a point source leads to tightly clustered paths (the location of the point source does not matter; data not shown), and this case is difficult to parse. This is because the lateral separation between links may only be a few pixels, making scoring-based discrimination less effective. This also explains why the performance measured in VI for the point source case is very sensitive to noise. Even for zero noise, trajectories are packed together tightly, leading to a challenging discrimination task. When noise is added, the chances of erroneous link assignments go up even more, making it that much harder to assign true trajectories. For y_0, at high spawn rates, increasing the length of the interval increases the probability that a new object A will spawn at time t_j very close to the position of object B at time t_i. If A is closer to B's location at t_{j-1} than B at t_j is to its previous value, then the algorithm will determine that A should be assigned to the trajectory of object B while assigning B at t_j to a new trajectory, leading to link (and trajectory) misidentifications.

When we looked at the effect of increasing the variation in v_{x0}, we expected improved performance since greater nonuniformity in v_{x0} can increase horizontal separation between objects. However, because we allow objects to have both positive and negative x velocity, the chance of multiple path intersections increases. The algorithm is capable of dealing with path intersections when the trajectories are spaced in time adequately (lower spawn rates), but as the objects are grouped closer and closer together (at high spawn rates), it can become quite challenging to discern between trajectories that intersect frequently within a small space. These trends were recapitulated for v_{y0}, where an increase in v_{y0} led to a slight decrease in performance. As the scoring function penalizes motion against the direction of the force, variations in the movement of objects in the direction of the force will see a smaller change in algorithm performance. As for a_y, the results were robust to nonuniformities in the acceleration of the objects.

When looking at how performance depends on noise and spawn rate, in general, we found a more prominent role for the spawn rate, although the signal/noise ratio affected results as expected. Indeed, increasing noise strength correlates with a decrease in performance. This is especially true for trajectories that are closely spaced since densely clustered objects can essentially swap positions and still have trajectories close to the true ones. This is evident in Fig. 3, which plots results from altering the initial x-position variation. In these simulations, the most densely clustered objects are represented by an initial x-position variation of 0 and a spawn rate of 16, with 16 objects appearing from the same exact point in every frame. As we look at the change in performance metrics from noise strength of zero to noise strength of one, we see a significant decrease in performance, with VI values increasing twofold for noise strength of one compared to noise strength of zero. As the objects increase in separation (the initial *x*-position variation is larger), the effect of noise on the algorithm performance is less profound. Furthermore, when the objects are separated both spatially and temporally, the noise strength has even less influence on the algorithm performance. Again, referring to Fig. 3, initial *x*-position variations of 300 or 400 and spawn rates of 1 or less lead to barely noticeable changes in VI values.

One of the fundamental limits of tracking algorithms is how closely the objects are located to one another. This is a function not only of the objects' motion in space but also of the rate at which they enter or exit the sensor's field of view, i.e., the rate at which they spawn. The role of the objects' spatial density was evident in our simulations when we studied how sensitive the algorithm was to variations in the initial positions, velocities, and accelerations. These three parameters were found to have the most significant effect on object density, with greater variation in their values leading to greater spatial separation. (The caveat is that greater variation in *x* velocity and *y* acceleration can also increase the frequency of trajectory intersections.)

We determined mean nearest-neighbor distances for objects in these trials, i.e., the mean distance between each object at time t and its nearest neighbor at that same time. Deciding between the nearest neighbor and the true object is the core challenge that the algorithm faces. These nearest-neighbor distances vary depending on the simulation parameters used, including both spatial parameters and spawn rates. At a spawn rate of 16, the mean nearest-neighbor distance is between 3 and 19 pixels, regardless of the spatial parameters. Reducing the spawn rate to 4, the mean nearest-neighbor distance grows to between 4 and 50 pixels. Then, at a spawn rate of 1, the mean nearest-neighbor distance spans the range of 40 to 140 for all spatial parameters. It is interesting to note that we see instances in the simulations where high spawn rate (≥ 8) and dense object clustering can often present nearest-neighbor distances of less than a single pixel and sometimes even with multiple object centroids within a single pixel. Parsing trajectory assignments for objects spaced so closely is incredibly challenging, particularly so without utilizing any special image processing methodology.

The spawn rate also played a significant role in the ability of the algorithm to discriminate trajectories. Indeed, in any of the Figs. 2, 3, 4, and 5, an examination of the same color bars of any subplot shows that the performance of the algorithm improves consistently as the object spawn rate is decreased. For object spawn rates of one or lower, the worst performance of the algorithm across all variables occurs for an initial *x*-position variation of zero and noise strength of three, in which 70% of trajectory links were correctly identified and a VI of 1.4 was achieved. This is perhaps one of the most difficult cases to parse accurately, and yet the algorithm is capable of detecting 70% of trajectory links correctly while maintaining a VI of 1.4. In fact, the only other case in which VI is greater than one for a spawn rate of one is when the initial *x*-position variation is zero and the noise strength is two. For all other parameter value combinations, VI is below one. Even for a spawn rate of two, the algorithm only has significant issues when the initial *x*-position variation is zero; for all other parameter combinations, VI at spawn rates of two does not significantly exceed one. In an experimental setting, this means that if the frequency of observation (frame rate) is high enough to reduce the spawn rate to some value around two, then the algorithm should have excellent performance regardless of object dynamics.

The backtracking method was evaluated by comparing correct trajectory links of the scoring function to those of the complete algorithm. For the set of simulation parameters discussed previously, the number of correct trajectory links using the entire algorithm was compared to the number of correct trajectory links using the scoring function only, and the difference between the two values was calculated (see Fig. 7). A Gaussian fit with mean 8.16 and standard deviation 24.25 is also shown for comparison. In this figure, histogram bins less than zero represent

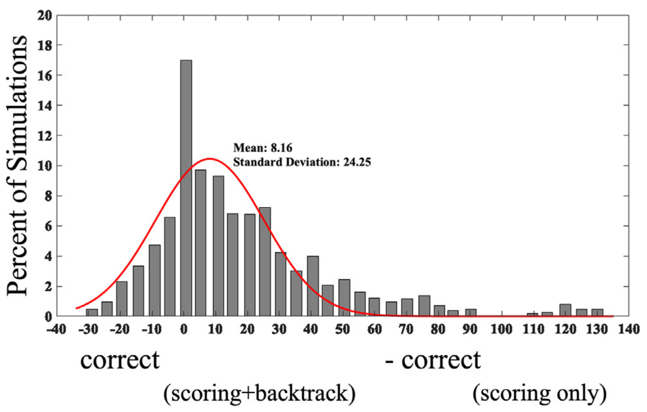

FIGURE 7 Examining the effect of the backtracking method on the scoring function output. Here, we show how the backtracking method may alter the scoring function output from the set of simulation trials shown previously. The *x* axis values correspond to results that were calculated by subtracting the number of correct assignments from the scoring function only from the number of correct trajectory assignments using both the scoring function and backtracking method. Thus, a positive number indicates trials that saw improvement with the backtracking method, while negative numbers correspond to trials in which the backtracking method caused additional incorrect assignments. These results were collected into bins of width 5, and so each bar shows the percentage of the total number of trials in that bin. The red curve is a Gaussian fit with fit parameters: mean = 8.6 and standard deviation = 24.25. The bin values are half-closed intervals, e.g., [0,5) for the bin labeled 0 or [5,10) for the bin labeled 5.

trials in which the backtracking method resulted in a decrease in the number of correct trajectory links, while positive bins count simulations where an improvement after backtracking was achieved. We see that while there are situations in which the backtracking method can reduce the effectiveness of the scoring-function-based trajectory assignment, most often, the backtracking method improves upon the initial trajectory assignments. We also note that for the bulk of the simulations, the backtracking method makes in the range of 0–25 trajectory assignment corrections after examining hundreds of inputs. This suggests that the scoring function is likely capable of performing rather well on its own, but an improvement on the scoring function is seen in the majority of cases after passing through the backtracking method.

Finally, we carried out experimental tests to assess the accuracy of the algorithm by comparing the forces imputed by the algorithm with those determined independently using an orthogonal comparator method. As discussed in the supporting material, we found good agreement between the algorithm-based and reference methods.

SUPPORTING MATERIAL

Supplemental information can be found online at https://doi.org/10.1016/j.bpr.2024.100148.

AUTHOR CONTRIBUTIONS

C.T. and S.G. programmed the algorithm and simulations; C.T. and S.G. carried out the experiments and related data analysis; C.T. and A.S. designed the algorithm and simulations; I.P. and A.S. designed and supervised the experiments and assisted with the data analysis; and C.T., S.G., I.P., and A.S. wrote the manuscript.

ACKNOWLEDGMENTS

We gratefully acknowledge the assistance of the following individuals: German Uritsky, Kara Fagerstrom, Josh Maher, Dunja Skoko, C.L. Chou, Eric Fischer, Robert Mohr, and Roberto Fabian. A.S. would like to acknowledge funding from Vitreous State Laboratory and The Catholic University of America.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

1. Thomann, D., J. Dorn, …, G. Danuser. 2003. Automatic fluorescent tag localization II: Improvement in super-resolution by relative tracking. *J. Microsc.* 211:230–248. https://doi.org/10.1046/j.1365-2818.2003.01223.x.
2. Crocker, J. C., and D. G. Grier. 1996. Methods of digital video microscopy for colloidal studies. *J. Colloid Interface Sci.* 179:298–310. https://doi.org/10.1006/jcis.1996.0217.
3. Meijering, E., O. Dzyubachyk, and I. Smal. 2012. Methods for cell and particle tracking. *Methods Enzymol.* 504:183–200. https://doi.org/10.1016/B978-0-12-391857-4.00009-4.
4. Chenouard, N., I. Smal, …, E. Meijering. 2014. Objective comparison of particle tracking methods. *Nat. Methods.* 11:281–289. https://doi.org/10.1038/nmeth.2808.
5. Anderson, C. M., G. N. Georgiou, …, R. J. Cherry. 1992. Tracking of cell surface receptors by fluorescence digital imaging microscopy using a charge-coupled device camera. Low-density lipoprotein and influenza virus receptor mobility at 4 degrees C. *J. Cell Sci.* 101:415–425.
6. Bar-Shalom, Y., and E. Tse. 1975. Tracking in a cluttered environment with probabilistic data association. *Automatica.* 11:451–460. https://doi.org/10.1016/0005-1098(75)90021-7.
7. Kalman, R. E. 1960. A new approach to linear filtering and prediction problems. *J Basic Eng.* 82:35–45. https://doi.org/10.1115/1.3662552.
8. Cerveri, P., A. Pedotti, and G. Ferrigno. 2003. Robust recovery of human motion from video using Kalman filters and virtual humans. *Hum. Mov. Sci.* 22:377–404. https://doi.org/10.1016/S0167-9457(03)00004-6.
9. Barniv, Y. 1985. Dynamic programming solution for detecting dim moving targets. *IEEE Trans. Aero. Electron. Syst.* 21:144–156. https://doi.org/10.1109/TAES.1985.310548.
10. Fortmann, T., Y. Bar-Shalom, and M. Scheffe. 1983. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE J. Ocean. Eng.* 8:173–184. https://doi.org/10.1109/JOE.1983.1145560.
11. Hashiro, M., T. Kawase, and I. Sasase. 2002. Maneuver target tracking with an acceleration estimator using target past positions. *Electron. Commun. Jpn. Part I.* 85:29–37. https://doi.org/10.1002/ecja.10026.
12. Logothetis, A., V. Krishnamurthy, and J. Holst. 2002. A Bayesian EM algorithm for optimal tracking of a maneuvering target in clutter. *Signal Process.* 82:473–490. https://doi.org/10.1016/S0165-1684(01)00198-0.
13. Blanding, W. R., P. K. Willett, and Y. Bar-Shalom. 2007. Offline and real-time methods for ML-PDA track validation. *IEEE Trans. Signal Process.* 55:1994–2006. https://doi.org/10.1109/TSP.2007.893212.
14. Chen, B., and J. K. Tugnait. 2001. Tracking of multiple maneuvering targets in clutter using IMM/JPDA filtering and fixed-lag smoothing. *Automatica.* 37:239–249. https://doi.org/10.1016/S0005-1098(00)00158-8.
15. Hong, L., N. Cui, …, D. Wicker. 1998. An interacting multipattern data association (IMPDA) tracking algorithm. *Signal Process.* 71:55–77. https://doi.org/10.1016/S0165-1684(98)00134-0.
16. Reid, D. 1979. An Algorithm for Tracking Multiple Targets. *IEEE Trans. Automat. Control.* 24:843–854. https://doi.org/10.1109/TAC.1979.1102177.
17. Cox, I., and S. L. Hingorani. 1996. An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Trans. Pattern Anal. Mach. Intell.* 18:138–150. https://doi.org/10.1109/34.481539.
18. Noyes, S. P., and D. P. Atherton. 2004. Control of false track rate using multiple hypothesis confirmation, target tracking. *In* Algorithms and Applications IEEE, pp. 115–120. https://doi.org/10.1049/ic:20040062.