Bioinformatics, 31(8), 2015, 1279–1285
doi: 10.1093/bioinformatics/btu793
Advance Access Publication Date: 4 December 2014
Original Paper

OXFORD

Bioimage informatics

Adaptive settings for the nearest-neighbor particle tracking algorithm

Javier Mazzaferri[1,*], Joannie Roy[1], Stephane Lefrancois[1,2] and Santiago Costantino[1,3]

[1]Centre de Recherche de l'Hôpital Maisonneuve-Rosemont, Montréal, Canada H1T 2M4, [2]Département de Médecine, Université de Montréal, Montréal, Canada H3T 3J7 and [3]Département d'Ophtalmologie et Institut de Génie Biomédical, Université de Montréal, Montréal, Canada H3T 1J4

*To whom correspondence should be addressed.

Associate Editor: Robert F. Murphy

Received on May 5, 2014; revised on October 29, 2014; accepted on November 25, 2014

Abstract

Background: The performance of the single particle tracking (SPT) nearest-neighbor algorithm is determined by parameters that need to be set according to the characteristics of the time series under study. Inhomogeneous systems, where these characteristics fluctuate spatially, are poorly tracked when parameters are set globally.
Results: We present a novel SPT approach that adapts the well-known nearest-neighbor tracking algorithm to the local density of particles to overcome the problems of inhomogeneity.
Conclusions: We demonstrate the performance improvement provided by the proposed method using numerical simulations and experimental data and compare its performance with state of the art SPT algorithms.
Availability and implementation: The algorithms proposed here, are released under the GNU General Public License and are freely available on the web at http://sourceforge.net/p/adaptivespt.
Contact: javier.mazzaferri@gmail.com
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Single particle tracking (SPT) has become an essential tool to characterize movement in live cell imaging experiments (Meijering *et al*., 2012). Typical applications span the dynamics of single molecules within the cell membrane (Weigel *et al*., 2011), cell migration (Suraneni *et al*., 2012) and intracellular trafficking (Akita *et al*., 2010; Mazzaferri *et al*., 2013). SPT methods extract the individual trajectories of objects registered in time-lapse images and provide information about both their individual and collective behavior. Although several studies track objects manually, the big leap for exploiting this technique has been brought by automatic multiple-tracking algorithms, because they can process large number of objects simultaneously (Kalaidzidis, 2009).

The vast majority of the implementations separate particle *detection*, i.e. the identification and location of particles in each image of the time series, from *tracking*. The latter assigns particle positions that result from *detection*, to individual trajectories. The simplest *tracking* approach, often named *nearest neighbors*, is based on minimizing the distance travelled globally by all particles. That is, object locations that are close enough in space and time are assigned to the same trajectory. However, the sole criterion of proximity is not enough under challenging experimental conditions. Tracking densely packed objects, and particles that merge, split or blink, requires additional information to uncover the real underlying tracks. In recent years, several methods have been proposed to address several of these shortcomings, which have been reviewed in detail (Kalaidzidis, 2009; Meijering *et al*., 2012). Typically, these methods employ sophisticated image processing and statistical tools, as well as a priori knowledge about the systems, to overcome clutter, noisy detection and eventual particle disappearance. The downside of methods that assume specific dynamical models is that they can bias the tracking results if they are used early on

systems where the dynamics is still unknown (Supplementary Note S5). In such cases a more secure procedure is to start the analysis with a conceptually simple tracking approach, such as the nearest-neighbor algorithm.

Regardless of how complex the method, its performance relies invariably on tuning accurately a set of parameters, which depend on the experimental details of the system (Chenouard *et al.*, 2014). If the features of the system fluctuate spatially it is unlikely that a unique set of parameters are optimal across the entire image. Systems such as, neutrophil migration assays, intracellular trafficking of vesicles, particles diffusing in media with heterogeneous viscosity, and tracking of receptors unevenly distributed on the cell membrane are examples where this problem is particularly important (Manley *et al.*, 2008; Valentine *et al.*, 2001). Typically, this is partially addressed by limiting the analysis to small regions of interest, where conditions are approximately homogeneous. However, if trajectories are long, this workaround cuts tracks and biases the description of the system's dynamics.

In this work, we propose a new SPT method for inhomogeneous systems that adapts automatically to the local density of particles, based on the popular *nearest-neighbor* tracking algorithm. One of the most cited and used nearest-neighbor algorithms was proposed by Crocker and Grier (C&G) 1996), and has a simple code and free open source implementations in IDL, Matlab, Phyton and ImageJ (Bricard *et al.*, 2013; Lund and Wustner, 2013; Manley *et al.*, 2008; Theves *et al.*, 2013). Despite its simplicity, a method based on this algorithm (Celler *et al.*, 2013) has shown to render a good performance in a recent study that makes a widespread and quantitative comparison of SPT techniques (Chenouard *et al.*, 2014).

Nearest-neighbor algorithms minimize the global displacement of the particles, and thus need to consider all possible combinations of particle positions from one frame to the following, which can be computationally too expensive. Therefore, it is necessary to introduce a parameter called maximum displacement (*mD*), which represents the largest distance that a particle can move from one frame to the next one, to discard unlikely bonds and reduce the computation time. If this parameter is not carefully adjusted according to particles' speed and their density, either the trajectories are interrupted or the computation time is prohibitive. Therefore, in systems where particle density is inhomogeneous, there is no global value of *mD* that can be optimal everywhere. The method we propose is especially well suited for this kind of problem. We perform a systematic analysis of the problem, based on numerical simulations, as well as a thorough comparison of the proposed method with the standard nearest-neighbor algorithm. We finally demonstrate the advantage of applying the adaptive method to the classic under-agarose neutrophils migration assay (Nelson *et al.*, 1975; Sackmann *et al.*, 2012).

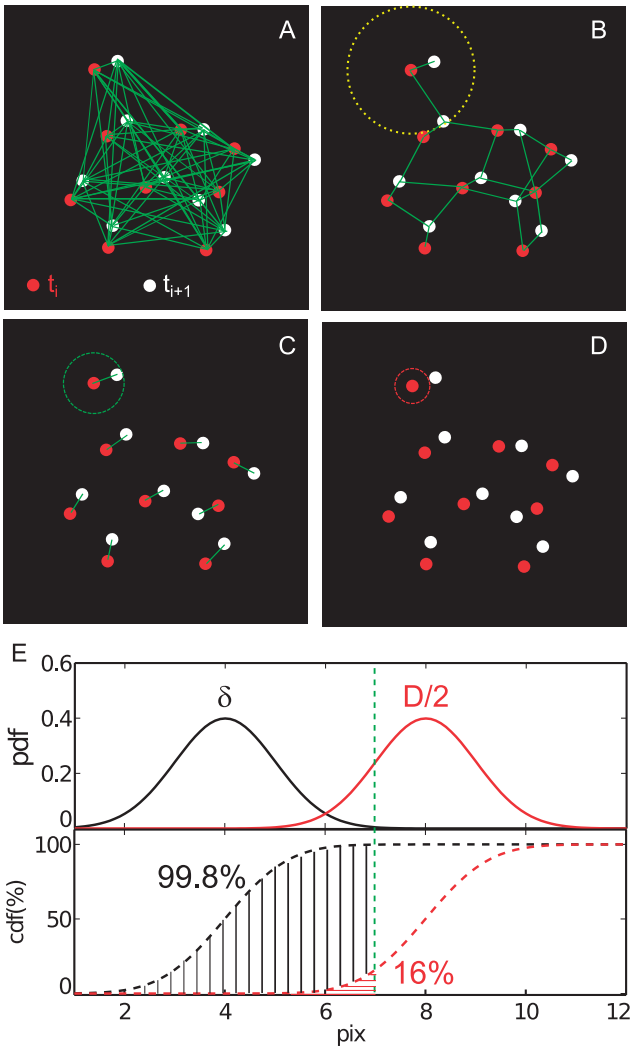

Fig. 1. Nearest-neighbor bond assignment: **(A)** The assignment of exclusive bonds between *P* locations at consecutive frames is computationally expensive because there are O(*P!*) possible combinations to consider. **(B)** Using the search radius *mD*, the algorithm only needs to consider bonds that are shorter than *mD*, thus significantly reducing the number of combinations to O(*P* ln(*P*)). **(C)** The value of *mD* can eventually be reduced to improve further the computation time. **(D)** If *mD* is smaller than the typical one-frame displacement of the particles, most bonds are discarded and the trajectories are interrupted. **(E)** Top: schematic probability density functions (pdfs) for δ and *D*/2. Bottom: cumulative density functions corresponding to the distributions above. Shaded regions below each distribution illustrates the trade off made when setting a value for *mD* (vertical dashed line). In this example, due to the overlap between distributions, for *mD*=7 pix, 99.8% of real bonds are detected (vertical lines shadow) at the cost of also detecting 16% of spurious bonds to neighboring particles (horizontal lines shadow)

2 Methods

2.1 Tracking algorithm and *mD*

Tracking algorithms identify which locations at frame t_i and t_{i+1} correspond to the same particle. In *nearest-neighbor* methods, this correspondence is established by creating unique bonds between particle locations based on proximity. These algorithms consider all possible assignments and choose the combination that yields the minimal total displacement for all particles. However, if each frame holds *P* particle positions, the network of bonds has O(*P*!) possible combinations, which makes the solution impractical even for small values of *P* (Fig. 1A).

As a workaround, nearest-neighbor methods limit the possible bonds to only those that are shorter than the preset parameter '*mD*' (Fig. 1B, dashed circle). In this way, the problem is reduced to solve a set of smaller sub-networks, where the number of combinations reduces to $O(P\ \ln(P))$, that has a significantly lower computational cost (Crocker and Grier, 1996).

However, *mD* has to be set wisely. It has to be small enough to discard false bonds, but large enough to detect all bonds that represent real particle displacements. It has been suggested that

the optimal magnitude of *mD* (Crocker and Grier, 1996) should satisfy:

$$\delta < mD < \frac{D}{2}, \quad (1)$$

where δ is the typical displacement of the particles during one frame (particle velocity times inter-frame interval), and D is the typical inter-particle distance. The first inequality ensures that real particle steps are detected, and the second inequality discards spurious bonds to nearby particles.

In practice, particle speeds (δ) are typically unknown a priori and it is customary to set *mD* as large as possible to minimize the risk of missing fast displacements. Although this strategy do not discard spurious bonds, while $\delta < D$ the algorithm will make the right connections. The real problem is that increasing *mD* boosts the number of candidate bond combinations, and computation time becomes too long. Figure 1A–D depicts the effect of different *mD* choices on the configuration of candidate bonds: while an overestimated *mD* yields an excessive number of bond combinations (Fig. 1A), when *mD* is too small trajectories are interrupted because real particle displacements are excluded (Fig. 1D). In summary, *mD* is typically chosen as large as possible while computation time remains practical.

Additionally, both δ and D fluctuate from particle to particle and in time, and they are distributed as schematized in Figure 1E. If the δ and D distributions are well separated, a value of *mD* between them is enough, but when they overlap a trade-off is required (see Fig. 1E and F). When the overlap of these distributions is extensive, the system is ill-conditioned because the computationally affordable value of *mD* is not large enough for all δ, and some trajectories will necessarily be interrupted.

Specifically, the systems where D is spatially inhomogeneous are challenging for *tracking* because dense regions broaden and shift the $D/2$ distribution to the left, increasing the overlap and hampering the performance. Although this limitation originates from dense regions, the impact on performance is ubiquitous, because *mD* is chosen globally.

2.2 Locally adaptive SPT

Instead of using a global value of *mD* to determine the set of candidate bonds, we propose to build a function *mD(x,y,t)*, that adapts to local space and time fluctuations of the nearest-neighbor distance *D(x,y,t)*. In this way, we restrict the length of candidate bonds mostly within crowded regions, where it is more necessary, but allow particles in low density regions to make longer bonds.

In order to satisfy the needs of high-content studies, we propose an algorithm that computes *mD(x,y,t)* from previously detected particle locations $\{(x_i, y_i, t_i)\}$. In order to keep computation time practical, the algorithm chooses *mD(x,y,t)* to render a number of candidate bonds combinations (CBC) that the computer system can process in viable time.

We set $mD_i = mD(x_i,y_i,t_i)$ at each particle location to be proportional to the distance to the nearest-neighbor particle $D_i = D(x_i,y_i,t_i)$ (see Fig. 2A and B). However, fast fluctuations of D yield unnecessary oscillations of *mD*, that can be avoided by replacing D_i with an averaged version

$$D_\phi(x,y) = \left\langle \{D_i/(x_i,y_i) \in B(x,y,\phi) \wedge \forall t_i\} \right\rangle$$

where *B(x,y, ϕ)* is a circular neighborhood centered at (x,y) of diameter ϕ. The operator <.> indicates set average.

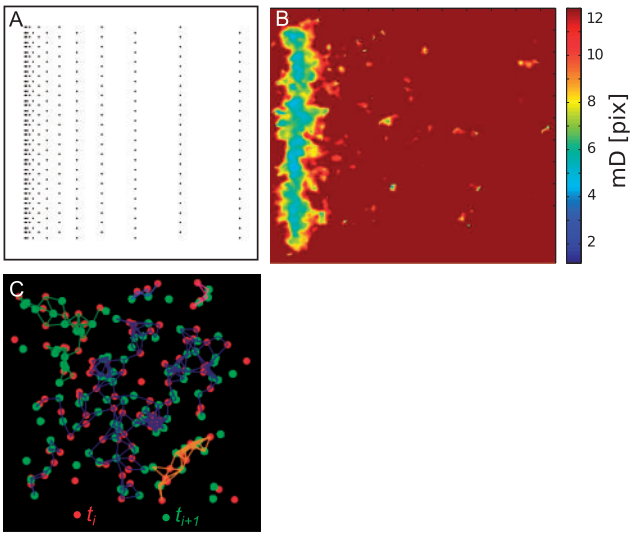

Fig. 2. (A) Unevenly distributed particles. **(B)** Color coded example of function *mD* for particles in **(A)**, $\alpha = 1$, and $mD_{max} = 12$ pixels. (C) Bond sub-networks for a specific *mD* function at two consecutive frames. Lines of the same color denote bonds in the same sub-network. Unconnected particle positions are due to either bonds longer than mD_i or trivial one-to-one bonds

The number *CBC* for a given *mD(x,y,t)* depends, not only on the number of bonds, but also on how they intertwine with one another in sub-networks (Fig. 2C). Because the tracking algorithm solves these sub-networks sequentially, what ultimately limits *CBC* is the size of the largest sub-network.

Because the quantity and structure of the sub-networks are hard to predict, we approached the selection of *mD(x,y,t)* using an heuristic and exhaustive method. We introduce a multiplicative factor α to scale D_ϕ at each frame, so that

$$mD(x,y,t) = \alpha(t) D_\phi(x,y)$$

Then, for each pair of consecutive frames we search iteratively the value of α for which the largest sub-network has the size of CBC. For an Intel Core i7-4770K processor running Matlab under linux OS, $CBC = 5 \times 10^4$ ensures that they can be processed in a few minutes, but this is a parameter that has to be set according to the available computing power.

A bisection algorithm is used to choose the value of α that renders a number of combinations as near as possible to predefined tolerance. For each candidate value of α, the whole set of candidate bonds is restricted to include only those shorter than *mD*, and all uniquely determined one-to-one bonds are discarded. Then, remaining bonds that are connected to each other by sharing a starting or ending position, are flagged as belonging to the same sub-network (Fig. 2C). We estimate the number of combinations for each sub-network by multiplying the number of candidate bonds for each particle in the sub-network as:

$$\prod_{i=1}^{P} \#(\{b_{ij}\})$$

where P is the number of particle positions that belong to the sub-network, $\{b_{ij}\}$ is the set of candidate bonds starting in position i and ending in j, and #(.) denotes set cardinality.

To always ensure track termination, we constrain *mD* so that it never exceeds the parameter mD_{max} that is determined by visual inspection of data.

At the end, we use *mD(x,y,t)* to reduce the set of candidate bonds with which the C&G algorithm builds the trajectories.

2.3 Performance assessment of tracking

In order to evaluate quantitatively the performance of tracking algorithms we compared their outcomes against a set of numerically simulated trajectories $T_k = \{(x_t^k, y_t^k, t), t = 1 \ldots F_k$, which we consider ground-truth (GT). For each test, a set $\{T_k, k = 1 \ldots N\}$ of N trajectories is produced, where each one is a sequence of particle locations along F_k time points. The initial coordinates for all trajectories are homogeneously distributed in a hexagonal lattice with inter-particle distance D (Fig. 3A). Each trajectory evolves with normally distributed step sizes (mean value $<\delta>$ and standard deviation σ_δ), during $F_{\max}$ frames or until it exceeds the image boundaries. Also, the direction of movement changes from frame to frame a certain amount θ that is uniformly distributed between 0 and 2π. Pseudo random number generation is done with the Matlab (R2013a) implementation of Mersenne Twister algorithm, using specific initialization seeds in order to make the simulations repeatible.

All the locations $\{(x, y, t/x, y, t) \in T_k \wedge T_k \in G\}$, without repetitions, are processed with the tracking algorithm under analysis, rendering the estimated trajectories $E = \{T_q, q = 1 \ldots M\}$. Because our study focuses only on the *tracking* step of SPT, we concentrate on evaluating the performance of making bonds between locations, and not in the *detection* of objects. To this end, we represent the trajectories in G and E as a sequence of bonds $T_k = \{b_{ij}^k\}$, where b_{ij} represents the bond between locations (x_i^k, y_i^k, i) and (x_j^k, y_j^k, j).

To characterize the tracking performance at the level of individual bonds we computed the number of them present in the G that are not in E (false negatives), and the number of bonds in E that are not present in the G (false positives). Additionally, to measure the ability to estimate whole trajectories, we computed the average tracked fraction

$$f = \left\langle \frac{|T_q|}{|T_k|} \right\rangle \tag{2}$$

where $T_k \in G$, T_q is the longest (in time) and continuous segment in E that matches T_k. Operator $|.|$ denotes duration in frames. Angle brackets indicate average over G. Because our analysis does not involve detection uncertainties, there is no ambiguity determining the matching locations between T_q and T_k.

3 Results and discussion

3.1 Tracking performance is determined by *mD*

The tracking performance of nearest-neighbors algorithms is strongly affected by the choice of *mD*. To explore this, we numerically simulated trajectories of particles with parameters $<\delta> = 4$ pix, $\sigma_\delta = 1$ pix, $N = 400$ particles, $F_{\max} = 100$ frames, and $D = 24$ pixels (Fig. 3A). We process them with the standard C&G algorithm (Crocker and Grier, 1996) implemented in Matlab (Mathworks, Inc.) using several values of *mD* (Fig. 3B and C). For *mD* above 7 pix, the performance of the tracking shows a plateau, but below this value the performance decreases: There are more false negatives and the average tracked fraction f gets smaller. This effect can be explained by comparing the value of *mD* with the particle step distribution (Fig. 3C, bottom panel). The value $mD = 7$ pix represents $<\delta> + 3\ \sigma_\delta$, that is >99% of particle possible steps, so most of the real bonds are detected and the performance is almost optimal. On the contrary, as *mD* gets smaller than $<\delta> + 3\ \sigma_\delta$, a greater fraction

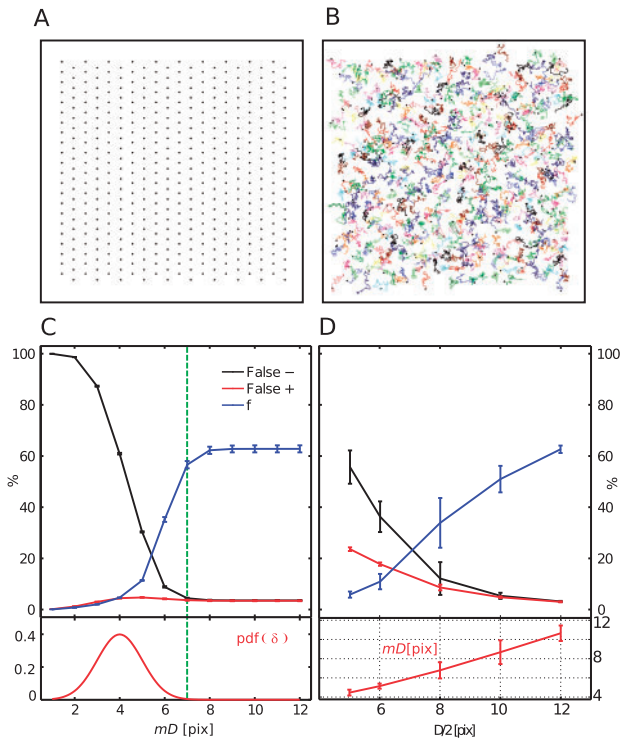

Fig. 3. Performance of nearest-neighbors algorithm versus *mD* and inter-particle distance. All trajectories sets *G* were numerically generated with $n = 400$, $F_{\max} = 100$ frames, $<\delta> = 4$ pixels, $\sigma_\delta = 1$ pixel and several values of inter-particle distance *D*. **(A)** Distribution of particles at $t = 0$ for $D = 24$ pixels. **(B)** Snapshot trajectories *E* obtained with C&G algorithm using $mD = 12$ pixels at frame 40 for the same time series of (A). **(C)** Descriptors of tracking performance on trajectories generated with $D = 24$ pixels versus *mD* used by C&G algorithm (top panel), and pdf of δ (bottom panel). The green vertical line indicates percentile 99.7% of δ. Error bars indicate the SD for time series generated with seeds 2, 4–8, 10 and 12–14. **(D)** Descriptors of tracking performance versus *D*/2 (top panel). Each set of trajectories was tracked with *mD* computed with the automatic algorithm described in Section 2.2 by setting $D_\phi(x,y)$ constant and choosing the minimal α among all frames (bottom panel). Error bars indicate the SD for time series generated with seeds 1–10

of the particles get out of the search radius and the performance decreases (see also Fig. 1E).

In summary, the performance gets better by increasing *mD*, but this also requires longer computation time because a larger number of linking candidates fall inside the search radius. Additionally, computation time grows when particles get near one another. Therefore the choice of *mD* is also related to the interparticle distance (Equation 1). To study this, we generated time series, similar to those above, but in this case we varied the interparticle distance *D*. We performed particle tracking on each time series, computing *mD* with the algorithm described in Section 2.2 by setting $D_\phi(x,y)$ constant and choosing the minimal α among all frames (Fig. 3D, bottom panel). In the top panel of Figure 3D, we plotted the performance descriptors versus the inter-particle half distance. The inter-particle distance has a strong impact, not only on the false negatives percentage, but also on the false positive fraction. In fact, as the inter-particle distance gets smaller, *mD* has to be reduced in order to keep computation time low, what yields an increase of false negatives. Additionally, as particles get too close to each other, the probability of making bonds to nearby trajectories is larger, resulting in more false positives.

In summary, when the particles are densely distributed, two effects limit the tracking performance: higher false negatives (and lower tracked fraction *f*) produced by smaller values of *mD*, and higher false positives due to more frequent bonds to nearby particles. The latter is an intrinsic problem of time series where the inter-particle half distance is similar to the typical particle step. Algorithms that use more sophisticated criteria for assigning bonds can improve the performance in these cases (Jaqaman *et al.*, 2008; Serge *et al.*, 2008).

3.2 Tracking performance is low for inhomogeneous distributions of particles

The effects mentioned above are also observed when only part of the image has densely packed particles. In such cases, the regions of high density limit the maximum value of *mD*, and this has an impact in the performance of the tracking everywhere. To study such systems, we simulated time series in which the particle density varies along the horizontal coordinate x (see Fig. 4B–D). We generated such movies by modifying the initial position of trajectories with a gamma transformation applied to x coordinates of all particle positions (Equation 3, Fig. 4A):

$$x' = \left(\frac{(x - x_{\min})}{(x_{\max} - x_{\min})}\right)^{\gamma} (x_{\max} - x_{\min}) + x_{\min}, \qquad (3)$$

where x is the horizontal coordinate of the particles in an homogeneous distribution (Fig. 3A), $x_{\min}$ and $x_{\max}$ are the minimum and maximum values of x, and x' are the transformed x coordinates of particles. We generated time series with parameters similar to those studied in Section 3.1 for several values of γ.

We tracked each one of them with the standard C&G algorithm and plot the performance descriptors versus γ in Figure 4E (top, solid lines and Supplementary Video S1). For each movie we computed the global value of *mD* (Fig. 4E, bottom) with our automatic algorithm by setting $D_{\phi}(x,y)$ constant and choosing the minimal α among all frames.

For the homogeneous case ($\gamma = 1$) both false negatives and false positives are ~3.5% while the mean tracked fraction f is 63%. However, the performance progressively worsens as γ increases because the region on the left of the image becomes crowded with particles, which force the automatic algorithm to choose smaller values of *mD* (Fig. 4E, bottom).

3.3 Adaptive algorithm improves tracking performance

The decrease of performance observed on inhomogeneous systems (Fig. 4E, top panel) can be mostly explained by the small values of *mD* (Fig. 4E, bottom panel) imposed by the regions of high density of particles. The effect is similar to the one observed in dense systems (Fig. 3D) but the tracking in the inhomogeneous system can be improved because the particles located on the right are sparse enough to afford larger *mD*. We propose using the spatially adaptive approach for optimizing the value of *mD* locally, according to the local density of particles. To test this idea, the same set of time series processed with the standard algorithm (Fig. 4E, solid lines), was processed with the adaptive algorithm described in Section 2.2 (Fig. 4E, dashed lines, and Supplementary Video S2). The parameter ϕ, was set equal to 10 pixels (see Supplementary Note S4).

Although the overall performance is reduced as γ increases, it is clearly better than for the standard algorithm. The difference between the two approaches shows mostly in the false negatives fraction and the tracked length, although there is also a slight improvement on the fraction of false positives. This is consistent with the fact that the performance improvement for the adaptive algorithm is mostly due to better matching between *mD* and the local particle density.

In Figure 4F we plot the performance descriptors for both tracking methods within different spatial regions for inhomogeneous time series with $\gamma = 6$. Essentially, the trajectories studied in Figure 4E, are now analyzed within five intervals of x: $R_1 = [0, 20)$, $R_2 = [20, 40)$, $R_3 = [40, 60)$, $R_4 = [60, 80)$ and $R_5 = [80, 100]$, for values of x expressed in percentage of $x_{\max}$ (Fig. 4D). Although in R_1, the performance improvement is modest because particles are densely packed in R_5 the improvement is dramatic. Due to the ability of adapting the value of *mD* to the local density, the tracking performance in the sparse regions on the right of the image are not limited by the high density of particles that happens on the left. In this way, the performances are greatly optimized.

In order to test the adaptive algorithm in typical challenging conditions, and to compare its performance with state of the art methods, we used the tools developed for the SPT challenge (Chenouard *et al.*, 2014). The results, described in Supplementary Note S1, show that the adaptive method, using a simple detection algorithm (Supplementary Note S2), presents a performance that is fairly comparable with state of the art SPT methods. However, these tests are not designed to evaluate performance under inhomogeneous conditions, which is the main advantage of the proposed method. Additionally, we compared the adaptive algorithm with u-track tracking which has open source code (Jaqaman *et al.*, 2008). This analysis, described in Supplementary Note S6 shows that u-track performs very similarly to the adaptive algorithm for systems similar to those in Figure 4E, although it shows a small improvement for very inhomogeneous conditions. However, when particles undergo abruptly long displacements, the adaptive algorithm renders better performance than u-track (Supplementary Notes S5 and S6).

3.4 Adaptive algorithm improves biophysical measurements

Inaccurate particle tracking may produce biased biophysical measures like particle speed and mean square displacement (MSD) (Chenouard *et al.*, 2014; Sackmann *et al.*, 2012; Weimann *et al.*, 2013). According to the performance analysis in Sections 3.2 and 3.3, it is expected that the accuracy of these biophysical measures may also be hampered in inhomogeneous systems.

We analyzed the distribution of one-frame displacements δ (instantaneous speed) and MSD curves, since they are biophysical descriptors typically derived from SPT analyzes. We compared the results of analyzing the GT and trajectories estimated with both C&G and the Adaptive algorithm. Because the improvement provided by the adaptive method is mostly within regions where particle density is not extremely high (see Fig. 4F), we compared only trajectories placed in the rightmost 80% of the movie (Fig. 5A). In Figure 5B, we compare the distributions of δ using boxplots. Although the median of δ distributions on both GT and Adaptive trajectories match the parameter $<\delta>$ of generated trajectories, the values estimated with C&G are clearly biased towards zero. Similarly, the MSD curve generated from C&G tracking reveals that this method underestimates particle displacement (Fig. 5C) in inhomogeneous movies. Also, since most trajectories are interrupted by missing bonds, the obtained trajectories are short, making it impossible to compute the MSD at large time delays (Δt). The linear fits of the MSD curves yield diffusion coefficients of: $D_{GT} = (1.69 \pm 0.002)$, $D_{C\&G} = (0.65 \pm 0.1)$, $D_{adap} = (1.59 \pm 0.04)$, all expressed in pix^2/frame, that also show the inaccuracy of the

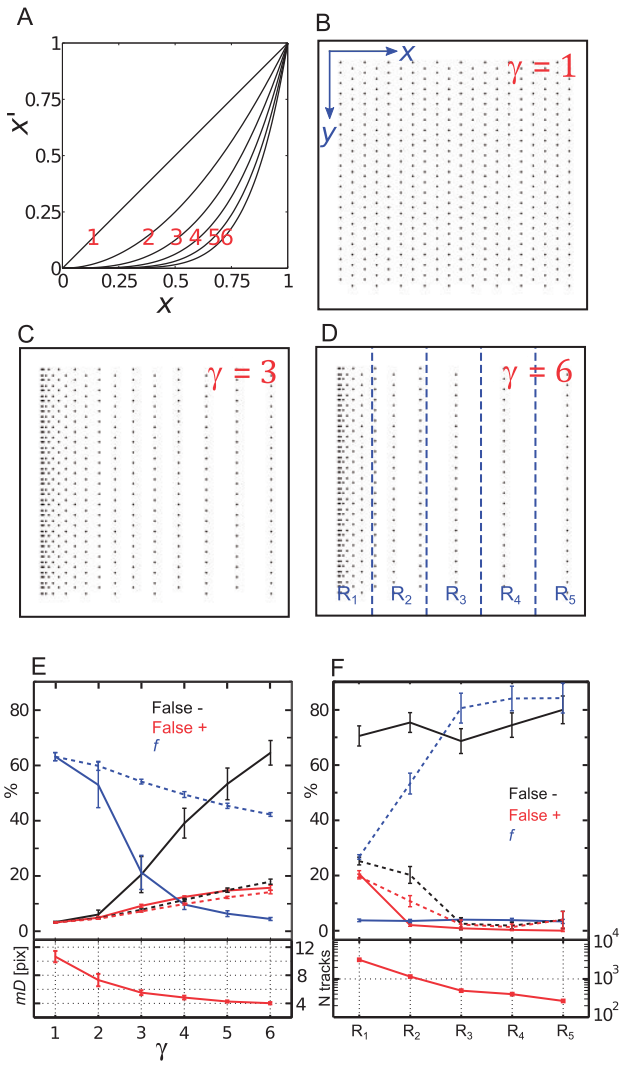

Fig. 4. Tracking performance of spatially inhomogeneous time series. All trajectories sets G were numerically generated with $n=400$, $F_{max}=100$ frames, $<\delta> = 4$ pixels, $\sigma_\delta=1$ pixel and $D=24$ pixels. The initial positions of particles were redistributed with Equation (3) for several values of γ. **(A)** Gamma curves of Equation (3) for $\gamma=1$–6. **(B–D)**: Initial particle locations for $\gamma=1$, 3 and 6, respectively. **(E)** Performance descriptors for the standard C&G (full lines) and Adaptive (dashed lines) algorithms as a function of γ (top), and values of mD used for the C&G algorithm computed with the automatic algorithm by setting $D_\phi(x,y)$ constant and choosing the minimal α among all frames (bottom). Error bars indicate the SD for time series generated with seeds 1–10. **(F)** Top: local tracking performance at regions depicted in (D). Bottom: number of tracks analyzed in each region

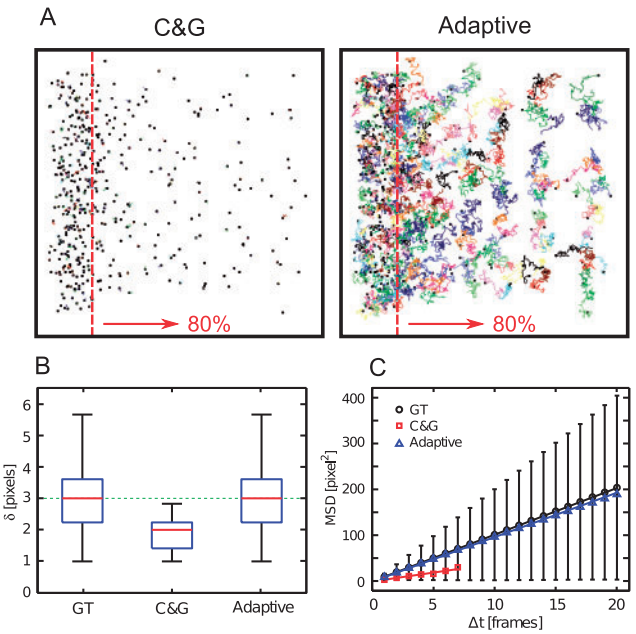

Fig. 5. Performance of biophysical measures. Ten trajectories sets G were numerically generated (seeds = 1–10) with $n=400$, $F_{max}=100$ frames, $<\delta>=3$ pixels, $\sigma_\delta=1$ pixel, $D=17$ pixels and $\gamma=5$. **(A)** Snapshots of trajectories estimated with C&G and Adaptive algorithms at frame 50 of the same time series. **(B)** Boxplots of δ for particles placed in the rightmost 80% of the movies, for GT, and trajectories estimated with C&G and Adaptive algorithm. The horizontal dotted line indicates the value of $<\delta>$ used to generate the trajectories. **(C)** MSD curves with corresponding linear fits for the same trajectories analyzed in **(B)**. Error bars for MSD of GT trajectories indicate the SD of squared displacements

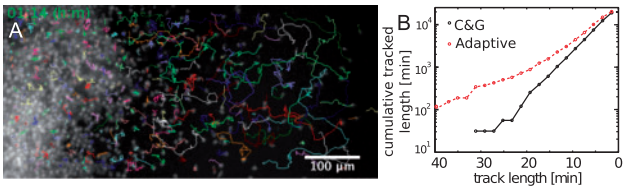

Fig. 6. Adaptive algorithm applied to neutrophils migrating in an under-agarose assay imaged at 4 frames per minute: **(A)** Sample frame with superimposed trajectories. See Supplementary Video S3. **(B)** Cumulative tracked length (in minutes) of the trajectories of neutrophils versus length of individual trajectories (in minutes): C&G (continuous line), Adaptive (dashed line)

global C&G in this case. All these results confirm the problems that arise from using C&G algorithm in inhomogeneous movies with global parameters, and that the method presented here can solve them.

3.5 Adaptive algorithm for Neutrophils migration assay

We used the adaptive algorithm to track the movement of neutrophils in the classical under-agarose chemotaxis assay (Nelson *et al.*, 1975), using the detection algorithm described in Supplementary Note S3. In this assay, neutrophils migrate by squeezing between a coverslip and a layer of agarose where a diffusing peptide produces a graded concentration of a chemotactic cue. The neutrophils, tagged with a florescent dye, enter the system from the left (Fig. 6A) and migrate towards the right, where the peptide concentration increases. The density of cells is remarkably inhomogeneous and they become difficult to track as explained in previous sections. Using the adaptive algorithm, long trajectories are detected despite the high concentration of cells on the left (see Fig. 6A, and Supplementary Video S3). To quantify the improvement of performance obtained with the proposed method, we also tracked the cells with the standard C&G algorithm. We computed the cumulative tracked length (in time) starting from the longer trajectories, and we plotted it versus the trajectory length (in minutes) for both SPT methods (see Fig. 6B). These results show that the trajectories obtained with the adaptive algorithm are remarkably longer than those obtained with C&G, rendering less biased and more accurate description of the neutrophil migration dynamic system. We finally compared the

algorithm with u-track (Supplementary Note S6). The u-track renders shorter trajectories possibly because it fails to detect unusually long displacements. For tracking objects in dense environments, u-track restricts the search radius using a priori knowledge of particle dynamics. However, this feature comes at the price of missing unexpectedly long displacements while the adaptive algorithm is more flexible for these types of dynamics.

4 Conclusion

The field of particle tracking has evolved significantly during the last two decades, allowing to perform in very challenging conditions such as dense systems, blinking particles and low-quality images. Several methods assume dynamic models of particle movement to face some of these challenges, but these assumptions may in certain cases bias results if the behavior of the system is not fully characterized a priori.

In this work, we propose a novel approach for optimizing the performance of the popular nearest-neighbor SPT algorithm by C&G in systems with inhomogeneous particle density. This method adapts the parameter *mD* to the local density of the particles both in space and time, boosting the tracking performance dramatically. Furthermore, even in systems that are not manifestly inhomogeneous, the standard nearest-neighbor algorithm is unstable due to slight variations of density that make the computation time to increase abruptly. Our improvement prevents these situations, providing the robustness needed for high content studies. Finally, this approach can be applied to more complex methods (Jaqaman *et al.*, 2008) and potentially other algorithms that depend on a parameter equivalent to *mD* (Chenouard *et al.*, 2013).

We demonstrate quantitatively the advantages of the adaptive method by comparing it with the standard C&G algorithm and other state of the art methods using numerical simulations and experimental data.

Funding

This work was funded by grants from the Fondation de l'Hôpital Maisonneuve Rosemont (La néphrologie et son Impact) to S.L., and the National Sciences and Engineering Research Council (NSERC) to S.C. S.C. and S.L. are recipients of salary awards from Fonds de la Recherche en Santé du Quebec (FRSQ). J.M. is a recipient of a postdoctoral fellowship from the Fondation de l'Hôpital Maisonneuve-Rosemont (La néphrologie et son Impact). J.R. was supported by funds from the NSERC-CREATE Training Program in Neuro-Engineering and an FRQS doctoral scholarship.

Conflict of Interest: none declared.

References

Akita,H. *et al.* (2010) Particle tracking of intracellular trafficking of octaarginine-modified liposomes: a comparative study with adenovirus. *Mol Ther*, **18**, 955–964.

Bricard,A. *et al.* (2013) Emergence of macroscopic directed motion in populations of motile colloids. *Nature*, **503**, 95–98.

Celler,K. *et al.* (2013) Single particle tracking of dynamically localizing TatA complexes in Streptomyces coelicolor. *Biochem Biophys Res Commun*, **438**, 38–42.

Chenouard,N. *et al.* (2013) Multiple hypothesis tracking for cluttered biological image sequences. *IEEE Trans Pattern Anal Mach Intell*, **35**, 2736–3750.

Chenouard,N. *et al.* (2014) Objective comparison of particle tracking methods. *Nat Meth*, **11**, 281–289.

Crocker,J.C. and Grier,D.G. (1996) Methods of digital video microscopy for colloidal studies. *J Colloid Interf Sci*, **179**, 298–310.

Jaqaman,K. *et al.* (2008) Robust single-particle tracking in live-cell time-lapse sequences. *Nat Methods*, **5**, 695–702.

Kalaidzidis,Y. (2009) Multiple objects tracking in fluorescence microscopy. *J Math Biol*, **58**, 57–80.

Lund,F.W. and Wustner,D. (2013) A comparison of single particle tracking and temporal image correlation spectroscopy for quantitative analysis of endosome motility. *J Microsc*, **252**, 169–188.

Manley,S. *et al.* (2008) High-density mapping of single-molecule trajectories with photoactivated localization microscopy. *Nat Methods*, **5**, 155–157.

Mazzaferri,J. *et al.* (2013) Analysis of AQP4 trafficking vesicle dynamics using a high-content approach. *Biophys J*, **105**, 328–337.

Meijering,E. *et al.* (2012) Methods for cell and particle tracking. *Method Enzymol*, **504**, 183–200.

Nelson,R.D. *et al.* (1975) Chemotaxis under agarose: a new and simple method for measuring chemotaxis and spontaneous migration of human polymorphonuclear leukocytes and monocytes. *J Immunol*, **115**, 1650–1656.

Sackmann,E.K. *et al.* (2012) Microfluidic kit-on-a-lid: a versatile platform for neutrophil chemotaxis assays. *Blood*, **120**, e45–e53.

Serge,A. *et al.* (2008) Dynamic multiple-target tracing to probe spatiotemporal cartography of cell membranes. *Nat Methods*, **5**, 687–694.

Suraneni,P. *et al.* (2012) The Arp2/3 complex is required for lamellipodia extension and directional fibroblast cell migration. *J Cell Biol*, **197**, 239–251.

Theves,M. *et al.* (2013) A bacterial swimmer with two alternating speeds of propagation. *Biophys J*, **105**, 1915–1924.

Valentine,M.T. *et al.* (2001) Investigating the microenvironments of inhomogeneous soft materials with multiple particle tracking. *Phys Rev E Stat Nonlin Soft Matter Phys*, **64**, 061506.

Weigel,A.V. *et al.* (2011) Ergodic and nonergodic processes coexist in the plasma membrane as observed by single-molecule tracking. *Proc Natl Acad Sci U S A*, **108**, 6438–6443.

Weimann,L. *et al.* (2013) A quantitative comparison of single-dye tracking analysis tools using Monte Carlo simulations, *Plos One*, **8**, e64287.