Point tracking: an a-contrario approach

Maël Primet, Lionel Moisan

Preprint MAP5 no. 2012-06 (2012)

Point tracking: an a-contrario approach

Maël Primet · **Lionel Moisan**

Abstract In this work, we propose a new approach to recover trajectories from points previously detected in a sequence of images. In presence of spurious and missing detections, actual trajectories can be characterized by an a-contrario model, that estimates the probability of observing a similar trajectory in random data. This results in a single criterion combining trajectory characteristics (duration, number of points, smoothness) and data statistics (number of images and detected points), which can then be used to drive a dynamic programming algorithm able to extract sequentially the most meaningful trajectories. The performances obtained on synthetic and real-world data are studied in detail, and shown to compare favorably to the state-of-the-art ROADS algorithm.

Keywords point tracking · a-contrario detection · motion correspondence

1 Introduction

Object tracking plays an essential role in a large variety of Computer Vision tasks, among which, for example, particle image velocimetry (Gui and Merzkirch, 1996), monitoring cars (Koller et al, 1994), detecting and tracking cells in microscopy sequences (Smal et al, 2008; Sbalzarini and Koumoutsakos, 2005), recognizing human activities (Ali and Aggarwal, 2001), improving human-computer interfaces with head-tracking (Ashdown et al, 2005), generating special effects for movies (Pighin et al, 1999), or tracking particles in accelerators (Cornelissen et al, 2008). Numerous variations on this problem have been formulated, and several algorithms have been developed to try to solve them. A common strategy (see Yilmaz et al (2006) for a recent review) is to detect the objects in each image and associate them with a geometric representation, such as points (Veenman et al, 2003b; Serby et al, 2004), geometric shapes (Comaniciu et al, 2003), outlines (Yilmaz et al, 2004), skeletal models (Ali and Aggarwal, 2001), and with appearance features decribed, e.g., by templates (Fieguth and Terzopoulos, 1997), active appearance models (Edwards et al, 1998), or probability densities of object appearance (Zhu and Yuille, 1996; Paragios and Deriche, 2000). The detected objects are then tracked across frames using an algorithm that tightly depends on the object representation. According to Yilmaz et al (2006), tracking algorithms can be broadly classified in three categories:

1. Point tracking (Veenman et al, 2003b; Reid, 1979; Bar-Shalom et al, 1983; Streit and Luginbuhl, 1994; Shafique and Shah, 2003). Objects are represented as points and are generally tracked across frames by evolving their state (object position and motion);
2. Kernel tracking (Comaniciu et al, 2003; Tao et al, 2002; Black and Jepson, 1998; Avidan, 2001). Objects are represented by a combination of shape and appearance, for instance an ellipse with a color histogram. They are then tracked by computing the motion of the kernel in consecutive frames, often modeled with parametric transforms such as translations and rotations;
3. Silhouette tracking (Blake and Isard, 1998; Bertalmío et al, 2000; Ronfard, 1994; Kang et al, 2004; Huttenlocher et al, 1993; Sato and Aggarwal, 2004). Objects regions are estimated in each frame, and are usually tracked by either shape matching or contour evolution.

In this work, we restrict ourselves to the tracking of objects as points, without appearance information. We assume

This work was funded by the French National Research Agency (ANR) under contract ANR-09-PIRI-0030-03 (PAGDEG project).

M. Primet, L. Moisan
Université Paris Descartes, Sorbonne Paris Cité
45 rue des Saints-Pères 75006 Paris - France
Tel.: +33 183-945-848
Fax: +33 142-864-144
E-mail: {mael.primet,lionel.moisan}@parisdescartes.fr

that these points have already been detected in the sequence, but imperfectly, in the sense that we may have to deal with both spurious points and missing detections. Given such a sequence of frames containing the detected points in each image, the goal is to extract the trajectories as lists of points appearing in successive frames, possibly separated by holes (missing detections), while avoiding spurious points. As usual, we will assume that a given point can belong to only one trajectory, which means that point collisions are ruled by an occlusion principle.

1.1 Related work

The most well-known point tracking algorithm certainly is the Multiple Hypothesis Tracker (MHT) of Reid (1979), which, in theory, enumerates all possible trajectory combinations of the observed points, and selects the one having the maximal likelihood (a probabilistic motion model being given). This problem is indeed NP-hard, and leads to exponentially many trajectory combination lookups (if there are n frames and m points, and we assume there is no occlusion, noise points, or objects leaving or entering the scene, there are already $m!^{n-1}$ possible trajectory combinations), and thus approximate solutions and heuristics are needed to accelerate the search (often requiring thresholds to prune the search tree early). Moreover, this complex model implies parameter tuning to optimize the underlying motion model and the efficiency/coverage tradeoff of the heuristics.

To overcome the exponential growth of the state space, researchers have proposed a wealth of heuristics and approximations to the point tracking problem over the years. Such an approximation is the Joint Probabilistic Data Association Filter (JPDAF) proposed by Bar-Shalom, Fortmann, and Scheffe (1983), which relaxes the hypothesis that the points must be disjoint. Instead of assigning each track ending in frame $k-1$ to a particular object in frame k as in MHT (and thus having several possible hypotheses, resulting in an increase of complexity), the JPDAF algorithm assigns to every track ending in frame $k-1$ a weighted combination of all points of frame k, depending on a likelihood estimate with respect to a predicted position. However, this approach assumes that the number of objects tracked in the images is constant, and the relaxation of the disjointness hypothesis leads to trajectory mergings, which is often an undesired feature.

Sethi and Jain (1987) propose to solve the correspondence problem greedily. They initialize the trajectories using the nearest-neighbor criterion, and then improve the current solution by exchanging correspondences between frames in order to minimize the global cost. They also propose a modified algorithm that alternates between forward and backward passes through the sequence to help mitigate the problem of the nearest neighbor initialization. Their approach is much faster than MHT, but does not permit to take noise, occlusions, entries or exits into account.

Salari and Sethi (1990) address some shortcomings of the previous method, namely the fact that it assumes a constant number of points in the sequence, and that there can be no entries or exits of objects in the scene. It therefore allows for occlusion, entry and exit of points, as well as the presence of spurious points. Each trajectory is made of either points detected in a frame, or "phantom points", which correspond to added (interpolated) points when there are missing detections. Note that their approach involves two parameters (the maximum allowed speed and the maximum allowed acceleration) which may be difficult to set.

Rangarajan and Shah (1991) propose to solve the problem by using a proximal uniformity constraint which combines requirements on the maximum speed and the acceleration of objects. They propose to make the assigmnent choices in an order driven by a notion of minimal risk, still in a greedy way. They use an optical flow algorithm to initialize the point motion between the first two images, and deal with occlusion and missing detections by using linear interpolation. They do not allow for spurious points, or objects leaving or entering the scene.

Veenman, Reinders, and Backer (2003b) propose a greedy tracking algorithm called ROADS, which is capable of handling missing and spurious detections, as well as entries and exits of objects. Rather than optimizing a global cost, they consider a restricted temporal scope (usually two or three frames forward), and find the optimal assignments minimizing the cost on these frames. Since the restricted problem is still NP-hard, they have some heuristics that help them prune the search tree. They keep the assignment between the two first frames of the local scope, and iterate the process on the following frames. The assignment between the first two frames of the sequence is initialized using the nearest neighbor criterion, and the effect if this approximation is mitigated as in (Sethi and Jain, 1987) by a forward and a backward pass. The ROADS algorithm is an extension of the previous well-established GOA algorithm (Veenman et al, 2001), which itself compares favorably to three classical algorithms (Salari and Sethi (1990), Rangarajan and Shah (1991) and Chetverikov and Verestoy (1999)). The ROADS algorithm has been shown to outperform both GOA and the MHT algorithm of Reid (1979).

Another state-of-the-art algorithm, developped for fluorescence particle tracking, is described in (Sbalzarini and Koumoutsakos, 2005). However, it addresses a different (and easier) tracking problem, since it requires information on the intensity of the detections. Moreover, it uses a prior motion model (nearest neighbor) that is more adapted to small displacements than to the kind of smooth trajectories (low acceleration) that we will consider in the present paper.

Fleuret, Berclaz, Lengagne, and Fua (2008) propose to track multiple persons in multiple camera views using a probabilistic map of the individuals locations, coupled with a Dynamic Programming algorithm that tracks each person in isolation, rather than conjointly. They use both an appearance model and a motion model to describe the objects to track.

In essence, the approaches above try to restrict the search space either by constraining the optimization to local choices or by limiting the simultaneous number of objects tracked. A recent and promising approach introduced in Jiang, Fels, and Little (2007) tries to solve the original unconstrained problem by optimizing a global criterion using Linear Programming, where the correspondence decisions are not binary choices, but continuous values in $[0, 1]$, rendering the optimization problem convex, and thus efficiently computable. In practice, the estimated values are almost always equal to either 0 or 1, and it is easy to convert them into disjoint trajectories. The algorithm assumes that the number of objects in the images is constant, but has been later extended in Berclaz, Fleuret, and Fua (2009); Berclaz, Turetken, Fleuret, and Fua (2011) to accommodate a variable number of objects that can enter and leave the scene in prespecified locations. The authors also prove that when there exists a unique global optimum, it is necessarily a boolean optimum.

1.2 Trajectory estimation versus trajectory detection

These algorithms have some common limitations. First, when they take a varying number of objects into account, as well as spurious and missing detections, they face the difficulty of choosing an appropriate global cost for their optimization problem. Indeed, they need a penalization for spurious points, and most of the time they will simply introduce a fixed cost for them. This creates a subtle (and quite arbitrary) interplay between the cost of spurious points and the cost of detected trajectories, which is not easy to control and grasp. Second, these algorithms often have many parameters, which is fine in theory, but quickly becomes a hassle when one needs to set them for each practical use. Last, as they all try to solve an optimization problem, these algorithms suffer from a classical flaw: they always find something, since they try to find *the best explanation* of the data in terms of some structure, without trying first to *prove that the structure is present*. In particular, all these algorithms will, for some values of the parameters, find trajectories in random data made of pure random points (without motion coherence).

In the present work, we propose a new approach for trajectory detection, which can guarantee that no trajectory will be found in general in such random data. This work is based on the a-contrario framework (Desolneux et al, 2008), which permits to derive absolute detection thresholds. These thresholds are then used to drive an algorithm that is able to analyze trajectories globally in time while avoiding the three aforementionned limitations.

In Section 2, we first consider trajectories without holes, that is, the case where no data point is missing (but spurious points are expected). After recalling the basic principles of the a-contrario statistical framework, we derive an explicit criterion for trajectory detection and present an algorithm based on Dynamic Programming (Bellman, 1954). We also analyze some theoretical consequences of the a-contrario thresholds, in particular the link between the number of points, the number of images and the maximum allowed acceleration. Then Section 3 extends the theory and the algorithm of Section 2 to the more general case of trajectories that contain holes. In Section 4, the state-of-the-art ROADS algorithm (Veenman et al, 2003b) is considered and various experiments (following, for most of them, the methodology proposed in the original ROADS paper) are led to compare its performances with the a-contrario algorithm we propose. Aside from a very convenient reduction of the number of parameters (1 for the NFA algorithm, versus 4 in the ROADS experiments), the a-contrario algorithm significantly outperforms the ROADS algorithm in terms of precision, robustness, and sensitivity to parameters, both in the "no-hole" and in the "hole" versions. Experiments are also conducted on real data, namely a *snow* sequence (that we make publicly available online) for which the ground truth has been manually obtained. Again, the results are clearly in favor of the a-contrario algorithm. We finally conclude in Section 5, and comment on the strengths, limitations, and perspectives offered by the present work.

2 Trajectories without holes

In this part, we consider trajectories without holes, that is, we assume that there are no missing detections (but possibly spurious detections, and points leaving and entering the scene).

2.1 Principles of the a-contrario framework

The trajectory detection method that we propose relies on the a-contrario framework introduced by Desolneux, Moisan and Morel (see Desolneux et al (2008) for a recent presentation). The idea underlying its development (dubbed "Helmholtz Principle") is that the human visual system detects structures in an image as coincidences which could not appear by chance in a random setting. Conceived at first to detect structures issued from Gestal Theory (Wertheimer, 1922; Kanizsa, 1980), this methodology has been applied to a large variety of image processing tasks, aiming at detecting structures like alignments (Desolneux et al, 2000),

edges (Desolneux et al, 2001), stereo coherence (Moisan and Stival, 2004), spots (Grosjean and Moisan, 2009), image changes (Robin et al, 2010), etc. Note that it has also been used in the context of Motion Analysis to cluster motion-coherent regions in image sequences (Veit et al, 2007), or to distinguish moving regions from the scene background (Veit et al, 2006).

We here recall the formalization of the a-contrario framework as it was presented in Grosjean and Moisan (2009). The a-contrario methodology is based on two ingredients: a *naive model*, and one or several measurements defined on the structures of interest. The naive model describes typical situations where no structure should be detected. For instance, when trying to discover alignments of points in an image, a naive model could consist of uniform and independent draws of the point locations, where no interesting structure would usually appear (see Fig. 1).

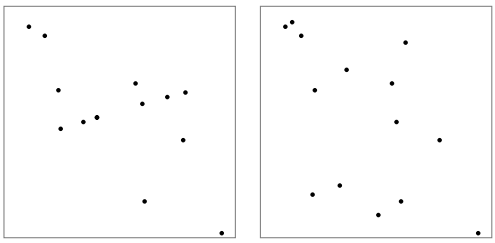


Fig. 1 Illustration of Hemholtz principle. Why can't we help seeing an alignment of dots on the left image? According to Helmholtz principle, we *a priori* assume that the dots should have been drawn uniformely and independently as in the right image, and we perceive a structure (here a group of aligned dots) because such an alignment is very unlikely to happen by chance. Alignments of three dots can be found in the right image, but they do not pop out, because they are likely to happen by chance considering the total number of points. The formalization of this principle is realized in a-contrario detection models.

To detect structures (e.g. alignments of points) in data using Helmholtz Principle, we need to define in what way an observation can be significant. If the measurement function is high when the structure is pregnant, we can relate the "amount of surprise" when observing the measurement x to the probability $\mathbb{P}(X \geqslant x)$, where X is the random variable corresponding to the distribution of x in the naive model. We will usually have several measurements $(x_i)_{i\in I}$ (in the example above, one for each possible alignment), and in the classical a-contrario framework the amount of surprise will be measured by a *number of false alarms*. More formally, we have the following

Definition 1 (Number of False Alarms) *Let $(X_i)_{1\leqslant i\leqslant N}$ be a set of random variables. A family of functions $\big(F_i(x)\big)_i$ is a* NFA *(number of false alarms) for the random variables $(X_i)_i$ if*

$$\forall \varepsilon > 0, \quad \mathbb{E}\big(\#\{i,\, F_i(X_i) \leqslant \varepsilon\}\big) \leqslant \varepsilon \tag{1}$$

(as usual, the notation "#S" stands for the cardinal of the set S).

A measurement x_i such that $F_i(x_i) \leqslant \varepsilon$ is said to be *detected at level ε*, or *ε-meaningful*. We say that a measurement is *meaningful* if it is 1-meaningful. This number of false alarms ensures that the average number of detections made in the naive model (that is, false detections) at level ε is less than ε.

The classical way to construct a NFA is given by the following proposition (see Grosjean and Moisan (2009)).

Proposition 1 (NFA construction) *Let $(X_i)_{1\leqslant i\leqslant N}$ be a set of random variables and $(w_i)_{1\leqslant i\leqslant N}$ a set of positive real numbers, then the function*

$$\mathrm{NFA}(i, x_i) = w_i \cdot \mathbb{P}(X_i \geqslant x_i) \tag{2}$$

is a NFA *as soon as $\displaystyle\sum_{i=1}^{N} \frac{1}{w_i} \leqslant 1$ and in particular if $w_i = N$ for all i.*

Remark 1 (NFA approximation) If $(F_i)_i$ is a NFA, then any family of functions $(G_i)_i$ verifying $F_i(t) \leqslant G_i(t)$ for all t is still a NFA. Hence, a function satisfying

$$\mathrm{NFA}(i, x_i) \geqslant w_i \cdot \mathbb{P}(X_i \geqslant x_i)$$

will define a NFA as soon as $\sum_{i=1}^{N} 1/w_i \leqslant 1$.

2.2 Trajectory detection

We are given a sequence of K images, each containing N points (to ease the notations we consider a constant number of points throughout the sequence but everything can be smoothly extended to the non-constant case as will be shown later), and whose support domain is taken to be the square $[0,1]\times[0,1]$ (again, the method can be adapted to arbitrary image sizes, as shown in Section 2.3). Following Helmholtz Principle, the naive model will here be a random uniform draw of N points in each of the K images (intuitively, we should not see trajectories appearing in the realizations of this model). The associated i.i.d. uniformely distributed random variables corresponding to the points of image k ($1 \leqslant k \leqslant K$) will be denoted by $X_1^k, \ldots, X_N^k$. We now define the structures of interest.

Definition 2 (Trajectories without holes) *A trajectory T of length ℓ starting at frame k_0 is a tuple $T = (k_0, i_1, \ldots, i_\ell)$, where $1 \leqslant i_p \leqslant N$ for all p and $1 \leqslant \ell \leqslant K - k_0 + 1$. We will denote by $\mathbb{T}$ the set of all trajectories. There is a natural equivalence between a trajectory $T \in \mathbb{T}$ and the tuple of variables $X_T = (X_{i_1}^{k_0}, \ldots, X_{i_\ell}^{k_0+\ell-1})$ that we shall therefore sometimes abusively call a (random) trajectory too.*

A realization t of the random variable X_T will be called a *realization of the trajectory* T. We have to keep in mind that we are working with the naive model (where points are randomly distributed), and thus, a realization of X_T *should not* look like what we would intuitively call a trajectory.

The second ingredient of the a-contrario model is the measurement function. For most applications, a natural choice is to define a "good trajectory" as a smooth one, and more precisely as a trajectory with a small acceleration. The proposed model could as well be easily adapted to a variety of other measurements; for instance, we could define a "good trajectory" as one that moves slowly, and take the measurement function to be the maximal speed of the object along the trajectory. Let us stick to the idea of good trajectories being those bearing a small acceleration. We still have many ways to define this, the two most obvious choices being to control the *maximal* acceleration, or to control the *average* acceleration of a trajectory realization $t = (\mathbf{y}_1, ..., \mathbf{y}_\ell)$. Again, the model can be adapted to both, but for practical reasons we will choose to control the maximal acceleration which involve simpler computations, thus leading to the measurement function

Definition 3 (Maximal acceleration) *The maximal acceleration of a tuple of points $t = (\mathbf{y}_1, ..., \mathbf{y}_\ell)$ is*

$$a_{\max}(t) = \max_{2 \leqslant i \leqslant \ell-1} \|\mathbf{y}_{i+1} - 2\mathbf{y}_i + \mathbf{y}_{i-1}\|. \tag{3}$$

We will now of course consider only trajectories having at least 3 points. Let T be a trajectory, Proposition 1 tells us that we can define a Number of False Alarms by an appropriate weighting of the probability $\mathbb{P}(a_{\max}(X_T) \leqslant \delta)$. In the naive model, this probability only depends on the length ℓ of the trajectory (and, of course, δ), and verifies

Proposition 2 (Probability bound) *If X_T is a random trajectory of length ℓ, and δ is a positive real number, then*

$$\mathbb{P}(a_{\max}(X_T) \leqslant \delta) \leqslant (\pi \cdot \delta^2)^{\ell-2}. \tag{4}$$

Proof — Assume that $T = (k_0, i_1, ..., i_\ell)$, and call $\bar{B}(x,r)$ the closed disc with center x and radius r. Writing $X'_p = X_{i_p}^{k_0+p-1}$, we get

$$\begin{aligned}
&\mathbb{P}(a_{\max}(\ X_T\ ) \leqslant \delta) \\
&\quad = \mathbb{P}\left(\bigcap_{p=3}^{\ell} \left\{ X'_p \in \bar{B}(2X'_{p-1} - X'_{p-2}, \delta) \right\} \right) \\
&\quad \leqslant \prod_{p=3}^{\ell} \mathbb{P}\left(X'_p \in \bar{B}(2X'_{p-1} - X'_{p-2}, \delta) \mid X'_{p-1}, X'_{p-2} \right) \\
&\quad \leqslant (\pi \cdot \delta^2)^{\ell-2}
\end{aligned}$$

because the area of $\bar{B}(x,\delta) \cap [0,1]^2$ is bounded from above by $\pi \cdot \delta^2$ for all x. □

By Remark 1, we know that we can use the upper bound (4) to construct a NFA, as in (2). There are many possibilities to define the weights $(w_T)_T$ subject to the constraint $\sum_T 1/w_T \leqslant 1$. This gives us a way to adjust the detection thresholds for each structure. We choose to group trajectories together according to their length, dividing the set of trajectories into K groups $\mathbb{T} = \mathbb{T}_1 \cup ... \cup \mathbb{T}_K$ (here, $\mathbb{T}_\ell$ denotes the set of trajectories of length ℓ), and weigh trajectories of a group uniformely by $w_T = K \cdot \#\mathbb{T}_\ell$ for any $T \in \mathbb{T}_\ell$. This choice is analyzed in detail in Section 2.6.3.

Proposition 3 (Continuous NFA for trajectories without holes) *The family of functions $(\mathrm{NFA}_T)_{T \in \mathbb{T}}$ defined by*

$$\forall \ell, \forall T \in \mathbb{T}_\ell, \quad \mathrm{NFA}_T(\delta) = K(K - \ell + 1) N^\ell \cdot (\pi \cdot \delta^2)^{\ell-2} \tag{5}$$

is a Number of False Alarms for the measurement $a_{\max}$.

Proof — Since $\#\mathbb{T}_\ell = (K - \ell + 1)N^\ell$, we have

$$\sum_{T \in \mathbb{T}} \frac{1}{K(K-\ell+1)N^\ell} = \sum_{\ell=1}^{K} \sum_{T \in \mathbb{T}_\ell} \frac{1}{K \cdot \#\mathbb{T}_\ell} = 1,$$

and thus (5) defines a NFA thanks to Proposition 1. □

Let us quickly comment Proposition 3. We can rewrite (5) into $\mathrm{NFA}_T(\delta) = K(K-\ell+1)N^2 \cdot \alpha^{\ell-2}$ by using the relative density $\alpha = N\pi\delta^2$ (which corresponds to the average number of points falling in a disc with radius δ). We see that for a trajectory to be meaningful, we need to have $\alpha < 1$. In other terms, only trajectories with maximal acceleration $\delta < 1/\sqrt{N\pi}$ might be detected as meaningful. Such kinds of bounds will be analyzed more precisely in Section 2.6.

2.3 Data quantization

In many applications, point detection is realized on a discrete grid of integer pixel coordinates, so that it may happen that three successive points in the sequence have a null acceleration. This is a very strong contradiction to the naive model, since this event has probability zero. Thus, if a long trajectory has a subtrajectory with a null acceleration, an algorithm that detects the most meaningful trajectories first will cut the longer trajectory into chunks to isolate the (optimal) null-NFA subtrajectory.

To avoid this kind of behavior we need to handle data quantization carefuly. There are at least two ways to do this: assume that the data have been properly quantized on the integer grid of the image and define a discrete version of the NFA, or consider a measurement imprecision and always consider the "worst-case scenario" for the measurements when computing accelerations.

First, we assume that the data have been quantized on an integer grid, say a rectangle Ω of $\mathbb{Z}^2$ containing $\#\Omega$ pixels. We can define a discrete version of the NFA by replacing the

continuous acceleration area $\pi \cdot \delta^2$ by its discrete equivalent, the *discrete acceleration area* defined by

$$a^d(\delta) = \frac{\#S_\delta}{\#\Omega},$$

where $\#S_\delta$ is the number of pixels enclosed in the discrete disc $S_\delta = \mathbb{Z}^2 \cap \bar{B}(0, |\delta|)$ (see Fig. 2). In particular, when $\delta = 0$ we obtain a discrete area of $1/\#\Omega$ that no longer leads to a null NFA. This leads to

Definition 4 (Discrete maximal acceleration) *The discrete maximal acceleration of a tuple of points $t = (\mathbf{y}_1, ..., \mathbf{y}_\ell)$ is*

$$a^d_{\max}(t) = \max_{2 \leqslant i \leqslant \ell-1} a^d(\mathbf{y}_{i+1} - 2\mathbf{y}_i + \mathbf{y}_{i-1}). \tag{6}$$

Proposition 4 (Discrete NFA for trajectories without holes) *The family of functions $(\mathrm{NFA}^d_T)_{T \in \mathbb{T}}$ defined by*

$$\forall \ell, \forall T \in \mathbb{T}_\ell, \quad \mathrm{NFA}^d_T(a) = K(K-\ell+1)N^\ell \cdot a^{\ell-2} \tag{7}$$

is a Number of False Alarms for the measurement $a^d_{\max}$.

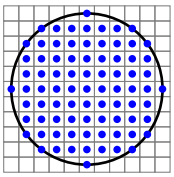


Fig. 2 Discrete discs. A continuous disc and its corresponding discretization composed of all pixels whose centers lie inside the continuous disc. A discrete measure of area is better suited to the analysis of quantized data which might result in observing degenerate null-area continuous disc (the corresponding discrete disc has a non-null area).

We now examine the "worst-case scenario" acceleration. We assume that we have an estimate $\eta > 0$ of the measurements imprecision (corresponding roughly to the radius of one pixel in the previous example). We keep the same NFA than in the continuous case (Equation 5), but we replace the measurement function by

$$a^w_{\max}(t) = \max_{2 \leqslant i \leqslant \ell-1} a^w(\mathbf{y}_i, \mathbf{y}_{i+1}, \mathbf{y}_{i+1}), \text{ where}$$

$$a^w(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \max_{dx,dy,dz \in \bar{B}(0,\eta)} \|(\mathbf{x}+dx) - 2(\mathbf{y}+dy) + (\mathbf{z}+dz)\|.$$

One easily shows that $a^w(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \|\mathbf{x} - 2\mathbf{y} + \mathbf{z}\| + 4\eta$, and therefore this amounts to keeping the same measurement function as in the continuous case and replacing the NFA (5) with

$$\mathrm{NFA}^w_T(\delta) = K(K-\ell+1)N^\ell \cdot (\pi \cdot (\delta + 4\eta)^2)^{\ell-2}. \tag{8}$$

We can see that a null-acceleration trajectory will be counted as an acceleration of 4η, thus incurring a penalty to all accelerations. This is why in practice we assume that the point positions are quantized in a grid (or force this quantization if needed), and use the discrete NFA.

2.4 Algorithm

In this section we consider the discrete NFA given in Equation (7). When a meaningful trajectory is present, any slight deviation from it (removing or adding a point, for instance) will usually also be meaningful, so that we expect to detect a large number of overlapping meaningful trajectories. Hence, we choose to detect the trajectories greedily, by iterating the following process:

1. compute the most meaningful trajectory, that is, the one having the smallest NFA;
2. remove its points from the data.

To compute the most meaningful trajectory in a sequence of points, that is, K images $I_1, ..., I_K$, each containing N points, we use a dynamic programming strategy. Indeed, we compute for each point $\mathbf{x}$ in image I_k the most meaningful trajectory ending in this point (note that in the following, we shall sometimes write $\mathbf{x}^k$ instead of $\mathbf{x}$ to recall that $\mathbf{x}$ belongs to I_k). Denoting by $\mathscr{G}(\mathbf{x}^k, \mathbf{y}^{k-1}, \ell)$ the minimal acceleration of a trajectory of length ℓ ending with points $\mathbf{y}^{k-1}$ and $\mathbf{x}^k$, we obtain a Bellman equation

$$\mathscr{G}(\mathbf{x}^k, \mathbf{y}^{k-1}, \ell) = \begin{cases} 0 & \text{if } \ell = 2, \\ \min_{\mathbf{z} \in I_{k-2}} \overline{\mathscr{G}}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \ell) & \text{otherwise,} \end{cases} \tag{9}$$

where

$$\overline{\mathscr{G}}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \ell) = \max\Big(a^d(\mathbf{x} - 2\mathbf{y} + \mathbf{z}) \, , \, \mathscr{G}(\mathbf{y}, \mathbf{z}, \ell - 1) \Big). \tag{10}$$

This recursive formulation translates to Algorithm 1.

Algorithm: `compute_`$\mathscr{G}$
input : $f_1, ..., f_K$ the sets of $\{n_k \leqslant N\}_{1 \leqslant k \leqslant K}$ points contained in each frame
output: $\mathscr{G}$

```
for 2 ⩽ k ⩽ K do
    for x in f_k do
        for y in f_{k-1} do
            G(x, y, 2) ← a^d(0)
            for 3 ⩽ ℓ ⩽ k do
                G(x, y, ℓ) ← +∞
                for z in f_{k-2} do
                    a ← max(a^d(x − 2y + z), G(y, z, ℓ − 1))
                    G(x, y, ℓ) ← min(a, G(x, y, ℓ))
                end
            end
        end
    end
end
return G
```

Algorithm 1: Dynamic programming computation of $\mathscr{G}$. We start by computing the values of $\mathscr{G}(x^k, y^{k-1}, \ell)$ for $k = 2$, then $k = 3$, ... each time reusing the results of the previous round according to the recursive formulation given by (9) and (10).

Now let us write

$$\mathrm{NFA}_\ell^d(a) = K(K-\ell+1)N^\ell \cdot a^{\ell-2}, \tag{11}$$

so that $\mathrm{NFA}_T^d(a) = \mathrm{NFA}_\ell^d(a)$ for any trajectory T with length ℓ. Since the function $a \mapsto \mathrm{NFA}_\ell^d(a)$ is monotone, the most meaningful trajectory with length ℓ is the one having the least maximal acceleration. Hence we can use Algorithm 1 to compute the smallest NFA of the point sequence (Algorithm 2). Moreover, if $\mathscr{B}(\mathbf{x}^k, \mathbf{y}^{k-1}, \ell)$ represents the most meaningful trajectory with length ℓ ending with points $\mathbf{y} \to \mathbf{x}$ (where $t \to \mathbf{x}$ denotes the concatenation of trajectory t and point $\mathbf{x}$), we can write

$$\mathscr{B}(\mathbf{x}^k, \mathbf{y}^{k-1}, \ell) = \begin{cases} \mathbf{y} \to \mathbf{x} & \text{if } \ell = 2, \\ \mathscr{B}(\mathbf{y}, \hat{\mathbf{z}}, \ell-1) \to \mathbf{x} & \text{otherwise,} \end{cases} \tag{12}$$

where $\hat{\mathbf{z}} \in \arg\min_{\mathbf{z}} \mathscr{G}(\mathbf{y}, \mathbf{z}, \ell-1)$ (strictly speaking, the most meaningful trajectory might not be unique, so we have to choose one most meaningul trajectory arbitrarily; to simplify the description we shall nevertheless use the term "the most meaningful trajectory" throughout the paper). Finally, for each $\mathbf{x}^k$, the most meaningful trajectory ending in $\mathbf{x}^k$ is $\mathscr{B}(\mathbf{x}^k, \hat{\mathbf{y}}^{k-1}, \hat{\ell})$ where

$$(\hat{\mathbf{y}}^{k-1}, \hat{\ell}) \in \arg\min_{(\mathbf{y}^{k-1}, \ell)} \mathrm{NFA}_\ell^d\left(\mathscr{G}(\mathbf{x}^k, \mathbf{y}^{k-1}, \ell)\right).$$

An algorithm similar to Algorithm 1 can hence be used to compute a trajectory having the smallest NFA (see Algorithm 2). In practice, we choose an arbitrary tuple $(\mathbf{x}, \mathbf{y}, \ell)$ such that there is an optimal trajectory of length ℓ ending on points $\mathbf{y} \to \mathbf{x}$, and we extract a trajectory by backtracking, each time selecting the predecessor that minimizes locally the maximal acceleration among all predecessors that lead to an optimal trajectory. Note that we could also extract the trajectory having the minimal average acceleration (for example) among all optimal trajectories, simply by rerunning a similar dynamic-programming algorithm restricted to optimal trajectories. In the end, by applying the process greedily (as mentioned above), we obtain an algorithm that extracts all meaningful (or ε-meaningful) trajectories from a sequence of points (Algorithm 3).

It is often possible to save computation time by extracting several trajectories at once without recomputing the function $\mathscr{G}$ each time, because removing points from the current data set cannot decrease any value of $\mathscr{G}$. This idea can be used, for instance, when the two most meaningful trajectories in the sequence do not share any point. The inner loop of Algorithm 4 does just that: if we have a way to define a set of disjoint minimal-NFA trajectories (for instance, greedily), we can extract them all at once (since they are of minimal NFA, and non-overlapping, we could extract them sequentially with interleaving $\mathscr{G}$ function recomputations, but this would not change the NFA of those trajectories, that would still be minimal).

Algorithm: `minimal_NFA`
input : $\mathscr{G}$
output: m the minimal NFA of a trajectory

```
m ← +∞
for 3 ⩽ k ⩽ K do
    for x in f_k do
        for y in f_{k-1} do
            for 3 ⩽ ℓ ⩽ k do
                m ← min(m, NFA_ℓ^d(𝒢(x, y, ℓ)))
            end
        end
    end
end
return m
```

Algorithm 2: Find the minimal NFA value among all trajectories.

Algorithm: `trajectory_detection`
input : ε the maximal allowed NFA
$f_1, \ldots, f_K$ the sets of points contained in each frame
output: $S = t_1, \ldots, t_m$ the extracted trajectories

```
S ← ∅
repeat
    compute_𝒢
    m ← minimal_NFA
    if m ⩽ ε then
        t ← a trajectory of NFA = m
        S ← S ∪ {t}, and remove all points in t from the
        corresponding frames
    end
until m > ε or there are no more points
return S
```

Algorithm 3: Greedy trajectories extraction using the NFA criterion.

Algorithm: `trajectory_detection_accelerated`
input : ε the maximal allowed NFA
$f_1, \ldots, f_K$ the sets of points contained in each frame
output: $S = t_1, \ldots, t_m$ the extracted trajectories

```
S ← ∅
repeat
    compute_𝒢
    m ← minimal_NFA
    stop ← false
    while m ⩽ ε and stop = false do
        U ← { x, ∃ a traj. with NFA = m ending in point x }
        V ← a set of disjoint trajectories of NFA = m ending
        on a point of U
        S ← S ∪ V
        remove all points from the trajectories in V
        stop ← true if not all points of U have been removed
        m ← minimal_NFA()
    end
until m > ε or there are no more points
return S
```

Algorithm 4: Greedy trajectories extraction, accelerated by extracting several trajectories at once.

Then if we have been able to extract a set of disjoint trajectories of minimal NFA that covers *every* pair of points where a trajectory of minimal NFA could end, we can continue doing the extraction for the next minimal NFA without recomputation. When this is no longer possible (because of some point removal) we need to recompute the $\mathscr{G}$ function to reactualize the NFAs.

We now examine the space and time complexity of algorithm 3 for an extraction round. The most expensive computation is that of function $\mathscr{G}$. The space (memory) required is $\mathscr{O}(N^2K^2)$, since we have to store a value for each triplet $(\mathbf{x}^k, \mathbf{y}^{k-1}, \ell)$ in each image frame k. Each value computation takes $\mathscr{O}(N)$ operations because we have to consider all the points in the previous image, leading to a $\mathscr{O}(N^3K^2)$ time complexity. The search for the minimal NFA and the extraction of the most meaningful trajectory have negligible time and space complexities. As the extraction must be repeated as long as there is any remaining meaningful trajectory, the global time complexity is $\mathscr{O}(sN^3K^2)$, where s denotes the number of extracted ε-meaningful trajectories. In practice, on a standard PC desktop, for $K = 50$ images, the number N of points per image can reach several hundreds (and about one thousand for $K = 20$).

2.5 Variable number of points

In real data, the number of points is hardly ever constant throughout the sequence, so that instead of having N points in each of the K images, we have $N_1, N_2, \dots N_K$ points on images $1, 2, \dots K$. Since the NFA is an upper bound on the average number of false alarms (Remark 1) we can simply take $N = \max_k N_k$ and keep the NFA unchanged. However, to obtain more accurate results, we can refine Proposition 4 with

Proposition 5 (Discrete NFA for trajectories without holes, general case) *The family of functions* $(\mathrm{NFA}_T^d)_{T\in\mathbb{T}}$ *defined, for any trajectory* T *with length* ℓ *starting at frame* k_0*, by*

$$\mathrm{NFA}_T^d(a) = K(K-\ell+1)\left(\prod_{k_0\leqslant k\leqslant k_0+\ell-1} N_k\right)\cdot a^{\ell-2}, \qquad (13)$$

is a Number of False Alarms for the measurement $a^d_{\max}$.

The proof is very similar to that of Proposition 3, except that now the set of trajectories $\mathbb{T}_\ell$ itself has to be decomposed with respect to the index of the starting frame.

2.6 Theoretical analysis

We now examine some theoretical consequences of Equation (5), which can have very practical consequences in the design of the data acquisition process. For simplicity reasons, we use the continuous NFA formulation, with a fixed number of points per image, but (7) or (13) would lead to similar conclusions.

2.6.1 Relation between the number of points and the maximal acceleration

Consider, as earlier, a sequence of K frames, each containing N points. We recall that the Number of False Alarms associated to a trajectory T with length $\ell \geqslant 3$ and maximal acceleration δ is (see Equation 5)

$$\mathrm{NFA}_T(\delta) = K(K-\ell+1)N^\ell \cdot (\pi\cdot\delta^2)^{\ell-2}.$$

Such trajectory is ε-meaningful as soon as

$$K(K-\ell+1)N^\ell \cdot (\pi\cdot\delta^2)^{\ell-2} \leqslant \varepsilon,$$

which can be rewritten $\delta \leqslant \delta_c$, where the upper bound is the critical acceleration

$$\delta_c = \frac{1}{\sqrt{N\pi}}\left(\frac{\varepsilon}{K(K-\ell+1)N^2}\right)^{\frac{1}{2\ell-4}}. \qquad (14)$$

Hence, as we already remarked at the end of Section 2.2, a necessary condition for trajectory detection is $\delta \leqslant \Delta$ with $\Delta = \frac{1}{\sqrt{N\pi}}$, which gives an order of magnitude of the typical accelerations that can be handled by the NFA approach (and, in some sense, by any approach since accelerations greater than Δ would allow detections in pure noise). Since the acceleration is inversely proportional to the squared frame rate (by doubling the frame rate, one divides accelerations by 4), this absolute bound can be useful in the design of the data acquisition process. Indeed, given the expected number of detected points in each frame (N), and the expected physical accelerations of objects (δ), one can compute the critical frame rate, under which no trajectory detection is possible. Note, however, that the upper bound Δ is not very accurate (see Table 1), thus using the exact value δ_c (Equation 14) is probably a better idea.

N	15	50	200	1000
Δ	146	80	40	18
δ_c ($l=5, K=20$)	23	8.3	2.6	0.7
δ_c ($l=10, K=20$)	74	35	15	5.4
δ_c ($l=10, K=50$)	64	30	13	4.7
δ_c ($l=30, K=50$)	117	61	29	12

Table 1 Acceleration bounds $\Delta = \frac{1}{\sqrt{N\pi}}$ and δ_c (Equation 14) in function of N, expressed in pixel.image^{-2} in a 1000×1000 image for some values of l and K ($\varepsilon = 1$).

2.6.2 Influence of the trajectory length

A nice property of the a-contrario approach is that it permits us to relate different parameters by observing the way they are linked in the NFA formula. Table 1 shows that the trajectory length has a significant impact on the critical acceleration δ_c (whereas the number of frames, K, has a much smaller impact). Thus, it could be interesting to study more precisely how the NFA balances the trajectory length and the acceleration, that is, how the critical acceleration grows as the trajectory lentgh increases. Since $1 \leqslant K-\ell+1 \leqslant K$, we can write $\log(K(K-\ell+1)) = 2\beta(\ell)\log K$ for some function $\beta(\ell)$ (taking values in $[1/2,1]$), so that from (14) we get

$$\log \delta_c = \log \Delta - \frac{\log N + \beta(\ell)\log K - \log\sqrt{\varepsilon}}{\ell-2}.$$

Hence, $\log\delta_c$ grows approximately like $1/\ell$, and attains for $\ell = K$ a value close to (and below) $\log\Delta$. This is illustrated in Fig. 3.

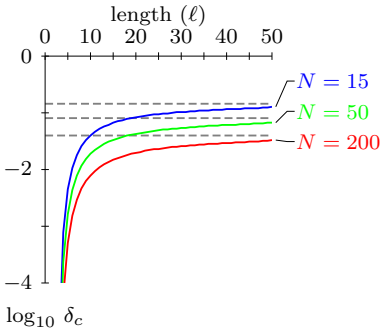


Fig. 3 Asymptotic and non-asymptotic critical accelerations. The critical acceleration δ_c (here, in base-10 log scale) is an increasing function of ℓ that approaches its upper bound (dotted line) $\Delta = \frac{1}{\sqrt{N\pi}}$ when $\ell = K$. Here $K = 50$ and the three curves correspond to $N = 15$, $N = 50$, and $N = 200$ respectively.

Last, we show the monotony of the critical acceleration with respect to the trajectory length (that is, the longer the trajectory, the looser the constraint on the acceleration).

Proposition 6 *If $\varepsilon \leqslant 1$, then the critical acceleration δ_c given by (14) increases with respect to ℓ.*

Proof — Rather than using (14), we go back to (5) and write, for $\ell \in \{1,\dots,K\}$,

$$F(\ell,\delta) = N^2K(K-\ell+1)\cdot(N\pi\delta^2)^{\ell-2}$$

(thus, $F(\ell,\delta)$ is the NFA associated to a trajectory with size ℓ and maximal acceleration δ). Now, if ℓ is such that $F(\ell,\delta)$ is smaller than 1 (since we assumed $\varepsilon \leqslant 1$), then $N\pi\delta^2 < 1$ so that both $\ell \mapsto (N\pi\delta^2)^{\ell-2}$ and $\ell \mapsto N^2K(K-\ell+1)$ are decreasing with respect to ℓ, and so is $\ell \mapsto F(\ell,\delta)$. Hence, if $\ell_1 > \ell_2$, we have, for $\delta = \delta_c(\ell_2)$,

$$F(\ell_1,\delta_c(\ell_2)) < F(\ell_2,\delta_c(\ell_2)) = \varepsilon,$$

which proves that $\delta_c(\ell_1) > \delta_c(\ell_2)$ since $\delta \mapsto F(\ell_1,\delta)$ is increasing. □

2.6.3 Asymptotic bounds and the importance of the combinatorial factor

Now we would like to assess the importance of the combinatorial factor in the definition of the NFA. As was discussed above before Proposition 3, there are several ways to define the weights of the structures. In (5), we chose to weigh the trajectories uniformely with respect to their length (that is, such that the expected number of false alarms is equally shared among all possible trajectory lengths), that is

$$\mathrm{NFA}_T(\delta) = K(K-\ell+1)N^\ell \cdot \mathbb{P}(a_{\max}(X_T) \leqslant \delta). \tag{15}$$

Another more classical choice would be to set a uniform weight $w_T = \#\mathbb{T}$ for *all* trajectories, thus obtaining

$$\mathrm{NFA}'_T(\delta) = \left(\sum_{m=1}^{K}(K-m+1)N^m\right)\cdot\mathbb{P}(a_{\max}(X_T)\leqslant\delta). \tag{16}$$

Suppose that we observe a trajectory t with length ℓ and maximal acceleration $\delta = a_{\max}(t)$, what difference will each NFA definition make? This trajectory is detected if $\mathrm{NFA}_T(\delta)$ (or $\mathrm{NFA}'_T(\delta)$) is below a certain threshold ε, hence it is interesting to estimate the ratio

$$\begin{aligned}\frac{\mathrm{NFA}'_T(\delta)}{\mathrm{NFA}_T(\delta)} &= \frac{1}{K(K-\ell+1)N^\ell}\sum_{m=1}^{K}(K-m+1)N^m \\ &\geqslant \frac{1}{K^2N^\ell}\sum_{m=\ell}^{K}N^m \\ &\geqslant \frac{N^{K-\ell}}{K^2}.\end{aligned}$$

This lower bound shows that when ℓ is small, the detection penalty incurred when using NFA′ is very large and it will thus be more difficult to detect small trajectories with this criterion.

In the following, we compare more precisely NFA and NFA′ and compute asymptotic estimates when the number of frames (K) becomes large. Let us deal first with the function $NFA_T(\delta)$. We consider a trajectory T spanning $\#T = \mu K$ images of the sequence for a fixed $\mu \in (0,1]$, with a maximal acceleration δ, among N points per frame. We write

$\alpha = N\pi\delta^2$, and propose to determine, when K gets very large, if the trajectory is meaningful. We have

$$\begin{aligned}\mathrm{NFA}_T(\delta) &= K\cdot(K-\mu K+1)\cdot N^{\mu K}\cdot(\pi\delta^2)^{\mu K-2}\\ &\underset{K\to\infty}{\sim} \pi^{-2}\delta^{-4}K^2(1-\mu)\alpha^{\mu K},\end{aligned}$$

so that

$$\log\mathrm{NFA}_T(\delta) \underset{K\to\infty}{\sim} \mu K\log\alpha$$

and

$$\lim_{K\to+\infty}\mathrm{NFA}_T(\delta)\leqslant 1 \iff \alpha<1 \qquad (17)$$

This means that for any $\mu\in(0,1)$ and any maximal acceleration $\delta<\Delta$ (that is, such that $\alpha<1$), for K large enough, any trajectory with maximal acceleration at most δ that spans μK frames among K will be meaningful. In practice, trajectories that lead to values of α near 1 are difficult to detect because they need to be observed on a large number of images (large K) to be meaningful. This phenomenon is illustrated in Fig. 4. Another non-asymptotic relationship between the acceleration δ, the number of frames K and the number of points per frame N is illustrated in Fig. 5.

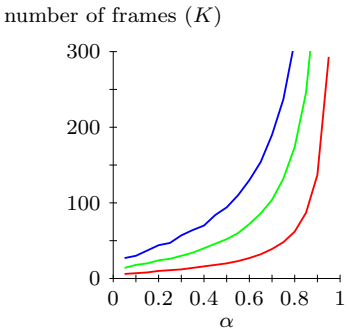


Fig. 4 Non-asymptotic counterpart of (17). Theses three curves represent, as a function of $\alpha = N\pi\delta^2$, the minimum number of frames (K) required for a trajectory with length $\ell = \lfloor \mu K\rfloor$ and maximal acceleration δ to be 1-meaningful according to the NFA criterion (15). Upper (blue) curve: $\mu = 0.3$; middle (green) curve: $\mu = 0.5$; lower (red) curve: $\mu = 1$. The number of points per frame is $N = 100$.

Now we study the asymptotic behavior of $\mathrm{NFA}'_T(\delta)$ (see Equation 16). First, we notice that

$$\sum_{m=1}^{K}(K-m+1)N^m = N^K\sum_{m=1}^{K} mN^{-m-1} \underset{K\to\infty}{\sim} \frac{N^K}{(N-1)^2}$$

and thus

$$\mathrm{NFA}'_T(\delta) \underset{K\to\infty}{\sim} \frac{N^{K+2-\mu K}\alpha^{\mu K-2}}{(N-1)^2}.$$

Therefore,

$$\log\mathrm{NFA}'_T(\delta) \underset{K\to\infty}{\sim} K\Big((1-\mu)\log N+\mu\log\alpha\Big)$$

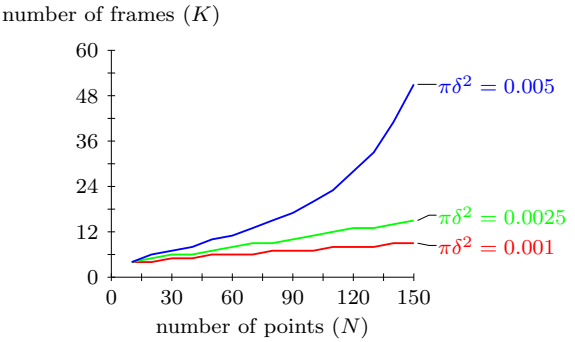


Fig. 5 Minimal number of frames required for detectability. Theses three curves represent, as a function of the number of points N, the minimum number of frames (K) required for a trajectory with length $\ell = K$ and various values of the maximal acceleration δ to be 1-meaningful according to the NFA criterion (15).

and

$$\lim_{K\to+\infty}\mathrm{NFA}'_T(\delta)<1 \iff \log\alpha<\frac{\mu-1}{\mu}\log N. \qquad (18)$$

Since $\frac{\mu-1}{\mu}\to-\infty$ as $\mu\to 0^+$, Equation (18) shows that it is indeed asymptotically much harder to detect small trajectories with NFA′ than with NFA. This fact is illustrated by Fig. 6, on which we can see that even when μ is slightly smaller than 1, NFA permits to detect more trajectories than NFA′, both asymptotically and non-asymptotically.

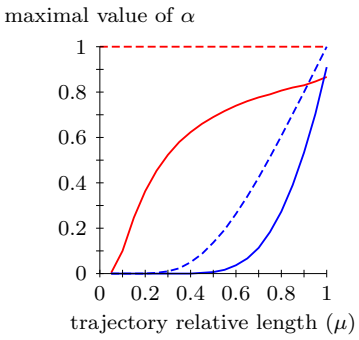


Fig. 6 Comparison between the NFA and NFA′ models (15, 16). Each curve represents, as a function of μ, the maximal value of $\alpha = N\pi\delta^2$ allowed for a trajectory with length $\ell = \lfloor\mu K\rfloor$ and maximal acceleration δ to be 1-meaningful in a sequence of K images, among $N = 100$ points per frame. The red upper curves are obtained with the criterion NFA, whereas the blue lower curves are obtained with the criterion NFA′. The full curves correspond to $K = 100$, and the dashed curves correspond to the asymptotical estimates obtained when $K\to+\infty$, that is, $\log\alpha = 0$ for NFA and $\log\alpha = \log N\cdot(\mu-1)/\mu$ for NFA′. These curves clearly demonstrate that not only NFA is better suited to the detection of small trajectories than NFA′ (it allows for trajectories having a much larger maximal acceleration), but it is also more efficient even for relatively large values of μ (NFA′ being slightly better only for almost complete trajectories). Asymptotically, NFA is always the best choice.

3 Trajectories with holes

In many practical situations, because of occlusions or acquisition noise, some trajectory points will not be detected in one or more frames. In this section, we generalize the previous framework to trajectory detection in the case of missing points.

3.1 Number of false alarms

The naive model remains unchanged (keeping the notations of the previous section): we are given K images $I_1,...,I_K$, each image I_k containing N points $X_1^k,...,X_N^k$, and we assume by Helmholtz principle that all random points X_i^k are independent and uniformly distributed in $[0,1]^2$.

Definition 5 (Trajectories with holes) *A* trajectory *of size s is a sequence of pairs $T=\{(i_1,\tau_1),...,(i_s,\tau_s)\}$, such that $\tau_1<...<\tau_s$. We denote by $\mathbb{T}$ the set of all trajectories, and by $\mathbb{T}_s$ the set of trajectories of size s. We bijectively associate to the trajectory T the tuple of random (i.i.d., uniformely distributed) variables $X_T=(X_{i_1}^{\tau_1},...,X_{i_s}^{\tau_s})$, that we also abusively call a (random) trajectory.*

Since the definition above is more general than the special case of trajectories without holes (Definition 2), we chose to keep the same word (trajectory) and the same notations ($\mathbb{T}$, $\mathbb{T}_s$) as in Section 2. This should not lead to ambiguities, since we will only consider trajectory with holes in this section.

As in Section 2, we would like to build an a-contrario detection model to detect trajectories (here, with holes). We consider three parameters of interest for the computation of the NFA of a trajectory $X_T=(X_{i_1}^{\tau_1},...,X_{i_s}^{\tau_s})$: the trajectory length, its size and its number of runs. The *length* is the total number of frames that the trajectory spans ($\tau_s-\tau_1+1$), the *size* is the number of (detected) points it contains (s), and a *run* is a maximal set of consecutive points. Note that if we call *hole* a maximal set of consecutive missing points, then the number of runs equals the number of holes plus one.

We first need to generalize the notion of maximal acceleration $a_{\max}$ (Definition 3) to the case of trajectories with holes. A natural way to do this consists in interpolating the missing points of the trajectory and compute its maximal acceleration. Since we would like to keep using an algorithm based on dynamic programming, we use the most local choice, that is, a simple constant speed interpolation. This leads to the following

Definition 6 (Maximal acceleration with holes) *The maximal acceleration of the realization $t=(\mathbf{y}_1^{\tau_1},...,\mathbf{y}_s^{\tau_s})$ of a trajectory T is*

$$a^h_{\max}(t)=\max_{2\leqslant i\leqslant s-1}\|a^h(\mathbf{y}_{i-1}^{\tau_{i-1}},\mathbf{y}_i^{\tau_i},\mathbf{y}_{i+1}^{\tau_{i+1}})\|, \tag{19}$$

with, for all points $\mathbf{x}^i,\mathbf{y}^j,\mathbf{z}^k$ $(i<j<k)$,

$$a^h(\mathbf{x}^i,\mathbf{y}^j,\mathbf{z}^k)=\frac{\mathbf{z}-\mathbf{y}}{k-j}-\frac{\mathbf{y}-\mathbf{x}}{j-i}. \tag{20}$$

We now compute, as in Proposition 2, a probability bound for the maximal acceleration of a random trajectory with holes.

Proposition 7 (Simple probability bound) *If a random trajectory X_T with size s has holes of size $h_1,...,h_{p-1}$, then for any $\delta>0$ one has*

$$\mathbb{P}(a^h_{\max}(X_T)\leqslant\delta)\leqslant(\pi\cdot\delta^2)^{s-2}\cdot\prod_{1\leqslant i\leqslant p-1}(h_i+1)^2. \tag{21}$$

Proof — We assume that $T=\{(i_1,\tau_1),...,(i_s,\tau_s)\}$, and write $X'_q=X_{i_q}^{\tau_q}$ and

$$M_q=X'_{q-1}+\frac{\tau_q-\tau_{q-1}}{\tau_{q-1}-\tau_{q-2}}(X'_{q-1}-X'_{q-2}),$$

so that

$$\begin{aligned}
&\mathbb{P}(a_{\max}(\ X_T\ )\leqslant\delta)\\
&\quad\leqslant\ \mathbb{P}\left(\bigcap_{q=3}^{s}\left\{X'_q\in\bar{B}(M_q,(\tau_q-\tau_{q-1})\delta)\right\}\right)\\
&\quad\leqslant\ \prod_{q=3}^{s}\mathbb{P}\left(X'_q\in\bar{B}(M_q,(\tau_q-\tau_{q-1})\delta)\ \mid\ X'_{q-1},X'_{q-2}\right)\\
&\quad\leqslant\ (\pi\cdot\delta^2)^{s-2}\prod_{q=3}^{s}(\tau_q-\tau_{q-1})^2\\
&\quad\leqslant\ (\pi\cdot\delta^2)^{s-2}\prod_{i=1}^{p-1}(h_i+1)^2.
\end{aligned}$$

□

For efficiency reasons, we want to design an algorithm that can share computations, that is, we want to be able to reuse the computations made on subtrajectories and extend them to obtain the results for bigger trajectories. To do this efficiently, we shall not consider, for a given trajectory, the individual sizes h_i of its holes, but simply its length ℓ, its size s and its number of runs p. This is why we derive from (21) the following

Proposition 8 (Practical probability bound) *If a random trajectory X_T has length ℓ, size s and number of runs p, then for any $\delta>0$ one has*

$$\mathbb{P}(a^h_{\max}(X_T)\leqslant\delta)\leqslant(\pi\cdot\delta^2)^{s-2}\cdot\left(\frac{\ell-s}{p-1}+1\right)^{2p-2} \tag{22}$$

with the convention that the right-hand parenthesis equals 1 ($(\frac{0}{0}+1)^0$) when $p=1$.

Proof — We consider the maximum value of the right-hand term of (21) over all possible hole sizes $h_1,...,h_{p-1}$ that are feasible for parameters ℓ, s and p. Relaxing the constraint that the h_i have to be integers, we face the optimization problem

$$\max_{(h_i)} \prod_{i=1}^{p-1} (h_i+1)^2 \quad ; \quad \sum_i h_i = \ell - s, \text{ and } \forall i,\ h_i \geqslant 0,$$

which, denoting $\xi_i = h_i + 1$, has the same solutions as the problem $\max_{\xi \in C} E(\xi)$, with $E(\xi) = \sum_{i=1}^{p-1} \log(\xi_i)$ and

$$C = \left\{ \xi \in [1,+\infty)^{p-1},\ \sum_{i=1}^{p-1} \xi_i = \ell - s + p - 1 \right\}.$$

Now if $\xi \in C$ has not identical coordinates, we can choose two different values, say $1 \leqslant \xi_{i_1} < \xi_{i_2}$, and replace them both by $(\xi_{i_1} + \xi_{i_2})/2 \geqslant 1$ to form a new $\xi' \in C$. Then,

$$E(\xi') - E(\xi) = 2\log((\xi_{i_1} + \xi_{i_2})/2) - \log(\xi_{i_1}) - \log(\xi_{i_2}),$$

and this quantity is positive by strict concavity of the log function. Thus, the unique solution of $\max_{\xi \in C} E(\xi)$ satisfies $\xi_i = (\ell - s)/(p-1) + 1$ for all i, and the maximum value of $\prod_i (h_i+1)^2$ over feasible hole sizes $h_1,...,h_{p-1}$ is bounded from above by

$$\left(\frac{\ell - s}{p-1} + 1 \right)^{2p-2}.$$

Using this bound in (21) yields the announced result. □

Now we need to define the combinatorial term w_T that premultiplies the NFA. Recalling the discussion of the end of Section 2.2, we choose to group the trajectories by their length and size, and to use uniform weights in each category. The number of trajectories of length ℓ and size s that can fit in K frames is bounded from above by $(K-\ell+1)\binom{\ell}{s} N^s$, and since we cluster the trajectories by their lengths and sizes, we have to count the number of such clusters. Indeed, it is bounded from above by $K\ell$, since there are less than K ways to choose the length, and knowing that the length is ℓ there are less than ℓ ways to choose the size. Combining these remarks with Proposition 8 establishes the following

Proposition 9 (NFA **for trajectories with holes**) *The family of functions* $(\mathrm{NFA}_T)_{T \in \mathbb{T}}$ *defined for any trajectory* T *of length* ℓ*, size* s *and number of runs* p *by*

$$\mathrm{NFA}_T(\delta) = K\ell(K-\ell+1)\binom{\ell}{s} N^s (\pi\delta^2)^{s-2} \left(\frac{\ell - s}{p-1} + 1 \right)^{2p-2} \tag{23}$$

is a Number of False Alarms for the measurement $a^h_{\max}$.

This new function NFA_T is a kind of generalization of (5). Indeed, for a trajectory T without hole (that is, such that $p = 1$, and consequently $\ell = s$), we have

$$\mathrm{NFA}_T(\delta) = \ell \cdot K(K-\ell+1) N^\ell (\pi\delta^2)^{\ell-2}$$

which is, up the a new factor ℓ, the value given in (5). This new factor simply comes from the fact that we do not know *a priori* that the number of runs of the trajectory is one.

3.2 Algorithm

In the practical implementation that we describe below, we use a discrete version of the acceleration, obtained by replacing the norm involved in (19) by a discrete area measure, exactly as we did in Section 2 (see Definition 4).

We want to compute, for each point $\mathbf{x}$ of each image I_i (that we denote by $\mathbf{x}^i$), the most meaningful trajectory that ends in $\mathbf{x}^i$ (or, to be more precise, one of such most meaningful trajectories) . This information can be extracted from the function $\mathscr{G}(\mathbf{x}^i, \mathbf{y}^j, \ell, s, p)$, which represents the least maximal acceleration of a trajectory of length ℓ, size s, and having p consecutive runs (that is, $p-1$ holes), ending with the point $\mathbf{y}^j$ in frame $j < i$ followed by the point $\mathbf{x}^i$ in frame i.

We say that a tuple (i, j, ℓ, s, p) is *undefined* if there is no trajectory ending with its two last points in frame i and j, with length ℓ, size s and having p runs of consecutive points. For instance, if $\ell < s$ or $s < 2$, the tuple is undefined. We define for $i > j$

$$\mathscr{G}(\mathbf{x}^i, \mathbf{y}^j, \ell, s, p) = \begin{cases} +\infty \text{ if } (i,j,\ell,s,p) \text{ is undefined,} \\ 0 \text{ if } s = 2, \\ \min_{u \geqslant 1, \mathbf{z} \in I_{j-u}} \bar{\mathscr{G}}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \ell, s, p) \text{ else,} \end{cases} \tag{24}$$

with the convention, for $i > j > k$, that

$$\bar{\mathscr{G}}(\mathbf{x}^i, \mathbf{y}^j, \mathbf{z}^k, \ell, s, p) = \max\left(a^h(\mathbf{x}, \mathbf{y}, \mathbf{z}),\ \mathscr{G}(\mathbf{y}, \mathbf{z}, \ell - (i-j), s-1, p - \mathbf{1}_{i \neq j+1}) \right) \tag{25}$$

and as usual $\mathbf{1}_{a \neq b} = 1$ if $a \neq b$ and 0 otherwise. Notice that as in Section 2.4, the superscript k in $\mathbf{x}^k$ simply reminds us that the point $\mathbf{x}$ belongs to image I_k, so we sometimes omit it and simply write $\mathbf{x}$.

We deduce from (25) a dynamic programming algorithm to compute $\mathscr{G}$, similar to the one we presented in Section 2 for the trajectories without holes. We can then backtrack to find the most meaningful trajectory ending in each point $\mathbf{x}^i$ by defining, for $i > j$, the recursive function

$$\mathscr{B}(\mathbf{x}^i, \mathbf{y}^j, \ell, s, p) = \begin{cases} \text{undefined if } (i,j,\ell,s,p) \text{ is undefined,} \\ \mathbf{y} \to \mathbf{x} \text{ if } s = 2, \\ \mathscr{B}(\mathbf{y}, \hat{\mathbf{z}}^{j-\hat{u}}, \ell - (i-j), s-1, p - \mathbf{1}_{i \neq j+1}) \to \mathbf{x} \text{ else,} \end{cases}$$

where $\hat{\mathbf{z}}^{j-\hat{u}}$ realizes the minimum in the last line of (24). Finally, for each $\mathbf{x}^i$, the most meaningful trajectory with holes ending in $\mathbf{x}^i$ is $\mathscr{B}(\mathbf{x}^i, \hat{\mathbf{y}}^{\hat{j}}, \hat{\ell}, \hat{s}, \hat{p})$, where

$$(\hat{j}, \hat{\mathbf{y}}^{\hat{j}}, \hat{\ell}, \hat{s}, \hat{p}) = \arg \min_{j<i, \mathbf{y}^j, \ell, s, p} \mathrm{NFA}^d_{\ell,s,p}(\mathscr{G}(\mathbf{x}^i, \mathbf{y}^j, \ell, s, p)).$$

We can analyze the spatial and temporal complexities of the algorithm. As in the case of trajectories without holes, the most costly operation is the computation of the $\mathscr{G}$ function. Its complexity is $\mathscr{O}(N^2K^5)$ in space and $\mathscr{O}(N^3K^6)$ in time. Since the extraction must be repeated until there are no more meaningful trajectories, the global time complexity is $\mathscr{O}(sN^3K^6)$, where s is the number of extracted ε-meaningful trajectories. In practice, on a standard PC desktop, for $K = 30$ images, the number N of points per image can go up to about one hundred. For long sequences containing much more than 30 images, the algorithm cannot be used directly (due to the N^6 term in the time complexity), but one could probably obtain good results by cutting the image sequence into small parts and applying the algorithm on each part (raising the issue of long trajectories spanning several parts).

3.3 Variable number of points and rectangular images

As for the NFA of trajectories without holes (Section 2), we can adapt the NFA given in Proposition 9 to the case of a variable number of points per image. Let us write N_k the number of points present in image k. The simplest strategy consists in applying directly the definition of Proposition 9 with $N = \max_k N_K$. If N_k has strong variations, a more sensitive detection can be obtained by replacing in $\mathrm{NFA}_T(\delta)$ the term N^s by

$$\max_{k_0=i_1<i_2<\ldots<i_s=k_0+\ell-1} N_{i_1} \cdot \ldots \cdot N_{i_s},$$

where T is a trajectory starting in image k_0, with length ℓ and size s. This term is easily computed once the sequence $(N_k)_{1\leqslant k\leqslant K}$ has been sorted.

As in the case of trajectories without holes, the NFA can also be adapted to rectangular images (see Section 2.3).

3.4 Theoretical results

We now analyze the asymptotic behavior of the NFA on some particular cases. They are all composed of one trajectory spanning the K images, and $N-1$ additional spurious points in each frame. The trajectory has a maximal acceleration of δ.

3.4.1 Long trajectory with a single hole

We first study the case where the trajectory is composed of two parts separated by a unique hole of length $h = \varepsilon K$. We thus have $\ell = K, s = (1-\varepsilon)K, p = 2$, and we write $\alpha = N\pi\delta^2$ as in Section 2.6.3.

We study under which conditions, when K gets large, the trajectory is meaningful, and if it is more meaningful than its first (or equivalently last) part. First we derive an asymptotic expansion of

$$\begin{aligned}\mathrm{NFA}_T(\delta) &= K \cdot (K-\ell+1) \cdot \ell \cdot \binom{\ell}{s} \cdot \frac{\alpha^s}{(\pi\delta^2)^2} \cdot (\varepsilon K)^{2(p-1)} \\ &= K^2 \binom{K}{(1-\varepsilon)K} \frac{\alpha^{(1-\varepsilon)K}}{(\pi\delta^2)^2} (\varepsilon K)^2.\end{aligned}$$

From Stirling's Formula, one easily derives the expansion

$$\log \binom{K}{\eta K} = -Kh(\eta) - \frac{1}{2}\log K + \underset{K\to+\infty}{\mathscr{O}}(1),$$

where $\eta \in (0,1)$ is fixed, and

$$h(\eta) = \eta \log(\eta) + (1-\eta)\log(1-\eta).$$

Hence, we have

$$\log \mathrm{NFA}_T(\delta) = K\Big((1-\varepsilon)\log\alpha - h(\varepsilon)\Big) + \frac{7}{2}\log K + \underset{K\to+\infty}{\mathscr{O}}(1),$$

which proves the asymptotic equivalence

$$\lim_{K\to+\infty} \mathrm{NFA}_T(\delta) \leqslant 1 \quad \Longleftrightarrow \quad \log(\alpha) < \frac{h(\varepsilon)}{1-\varepsilon}. \tag{26}$$

This asymptotic condition on α is illustrated in Fig. 7. We can notice that the asymptotic limit on $\log\alpha$ given by the righ-hand term of (26) is quite accurate (and almost linear) as long as the relative hole size ε is not too large. If the hole size is half the trajectory length ($\varepsilon = 1/2$), then the asymptotic condition is $\alpha < \frac{1}{4}$.

Now we would like to investigate the condition under which the complete trajectory is more meaningful than its starting or ending parts. Writing $\gamma = (1-\varepsilon)/2$ so that each small trajectory has a size γK, this condition writes

$$\mathrm{NFA}_{T_{1+2}}(\delta) / \mathrm{NFA}_{T_1}(\delta) \leqslant 1,$$

that is

$$\frac{K}{K} \cdot \frac{1}{(1-\gamma)K+1} \cdot \frac{K}{\gamma K} \cdot \frac{\binom{K}{2\gamma K}}{\binom{\gamma K}{\gamma K}} \cdot \frac{\alpha^{2\gamma K}}{\alpha^{\gamma K}} \cdot \frac{((1-2\gamma)K+1)^2}{1} \leqslant 1$$

or equivalently

$$-Kh(2\gamma) + \gamma K \log\alpha + \log K + \underset{K\to+\infty}{\mathscr{O}}(1) \leqslant 0.$$

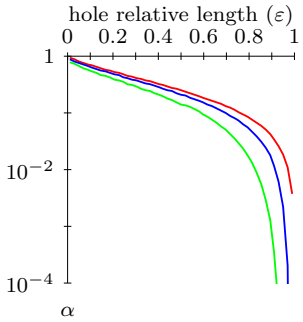


Fig. 7 Influence of the hole size. Plot, in function of ε, of the maximal value of α (in log scale) for a trajectory with a single hole of size $h = \lfloor \varepsilon K \rfloor$ to be meaningful. The total number of points in each frame is $N = 100$, and the length of the sequence is $K = 100$ for the green lower curve and $K = 400$ for the blue middle curve. The red upper curve is the asymptotical limit $h(\varepsilon)/(1-\varepsilon)$ of $\log \alpha$ corresponding to the case $K = +\infty$. We can see that when the trajectory hole becomes fairly important, the maximal allowed acceleration for a trajectory to be meaningful plummets.

Since $h(2\gamma)/\gamma = 2h(\varepsilon)/(1-\varepsilon)$, we thus have the property that the whole trajectory is asymptotically more meaningful than its parts when $K \to +\infty$ if and only if

$$\log(\alpha) < 2\frac{h(\varepsilon)}{1-\varepsilon}. \tag{27}$$

Note that this inequality constraint on α is stronger (the quantities are negative) than the one obtained in (26), hence the case when a trajectory is 1-meaningful but less meaningful than its parts can be encountered.

3.4.2 Dotted trajectories

If the trajectory is made of a succession of single points and one-frame holes, what is the condition as $K \to +\infty$ to have a meaningful trajectory? We now have $\ell = K$, $s = (K+1)/2$ (K being odd), $p = (K-1)/2$, so that from (23) we can derive the asymptotic expansion

$$\log \mathrm{NFA}_T(\delta) = \frac{K}{2}\log(16\alpha) + \frac{3}{2}\log K + \underset{K\to+\infty}{\mathscr{O}}(1). \tag{28}$$

Hence, a dotted trajectory is asymptotically meaningful when $K \to +\infty$ as soon as

$$\alpha < \frac{1}{16}. \tag{29}$$

3.4.3 Dashed trajectories

In the more general setting of a dashed trajectory made of p runs of u consecutive points separated by holes spanning v frames, we have $\ell = K = p(u+v) - v$ and $s = pu$, so that if u and v are fixed,

$$\log \mathrm{NFA}_T(\delta) = 2\log p + \log \binom{p(u+v)-v}{pu} + pu\log\alpha + 2p\log(1+v) + \underset{p\to+\infty}{\mathscr{O}}(1).$$

Now we have

$$\frac{pu}{p(u+v)-v} = \eta + \underset{p\to+\infty}{\mathscr{O}}\left(\frac{1}{p}\right) \quad \text{with} \quad \eta = \frac{u}{u+v},$$

so that

$$\log \binom{p(u+v)-v}{pu} = -p(u+v)h(\eta) - \frac{1}{2}\log p + \underset{p\to+\infty}{\mathscr{O}}(1)$$

and

$$\log \mathrm{NFA}_T(\delta) = \frac{3}{2}\log p + pu\left(\log\alpha + \frac{2}{u}\log(1+v) - \frac{h(\eta)}{\eta}\right) + \underset{p\to+\infty}{\mathscr{O}}(1).$$

Hence, a dashed trajectory is asymptotically meaningful when p tends to infinity if and only if

$$\log\alpha \leqslant \frac{h(\eta)}{\eta} - \frac{2}{u}\log(1+v), \tag{30}$$

where $\eta = \frac{u}{u+v}$ is the asymptotic density of known points. This formula yields an interesting relation between the density of known points and the allowed maximum acceleration, as illustrated in Fig. 8. When $u = v = 1$, we have $\eta = 1/2$, $h(\eta) = -\log 2$ and the right-hand term of (30) is $-\log 16$, in accordance with (29).

4 The ROADS algorithm, and comparison

4.1 The ROADS tracking algorithm

As we mentioned in Introduction, the state-of-the-art ROADS algorithm (Veenman et al, 2003b) is an extension of GOA that compares favorably to most classical tracking algorithms like MHT (Reid, 1979), Salari and Sethi (1990), Rangarajan and Shah (1991), Chetverikov and Verestoy (1999), and GOA itself (Veenman et al, 2001). It can handle points entering and leaving the scene, as well as missing and spurious points. It requires the setup of several parameters, which are listed in Table 2.

The criterion measuring the local smoothness of a trajectory on the consecutive points $(\mathbf{x},\mathbf{y},\mathbf{z})$ is

$$\phi(\mathbf{x},\mathbf{y},\mathbf{z}) = w\left[1 - \frac{v(\mathbf{x},\mathbf{y})\cdot v(\mathbf{y},\mathbf{z})}{\|v(\mathbf{x},\mathbf{y})\|\cdot\|v(\mathbf{y},\mathbf{z})\|}\right] + (1-w)\left[1 - 2\frac{\sqrt{\|v(\mathbf{x},\mathbf{y})\|\cdot\|v(\mathbf{y},\mathbf{z})\|}}{\|v(\mathbf{x},\mathbf{y})\| + \|v(\mathbf{y},\mathbf{z})\|}\right]$$

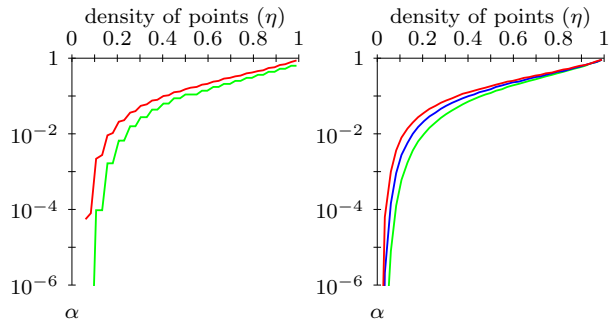


Fig. 8 Required precision for "dashed" trajectories. Left: The green lower curve shows the maximal value of α allowed (in log scale) for a trajectory to be meaningful when it is made of repetitions of runs of length u and holes of length v, with $u+v=20$ and $u=\lfloor\eta(u+v)\rfloor$. The length of the sequence is $K=100$, and the total number of points in each frame is $N=100$. The upper red curve is the asymptotical limit $h(\eta)/\eta-2\log(1+v)/u$ corresponding to $p=+\infty$. (note that the staircasing effect is due to the definition of u). As long as the density of known points (η) is large enough, the critical value of $\log\alpha$ is quite well approximated by the asymptotic bound and the relation to η is almost linear. When the density η becomes too small, the maximal allowed acceleration for the trajectory to be meaningful quickly plummets. Right: The asymptotic condition on α given by (30), for a variety of values of $u+v=10$ (lower green curve), 20 (middle blue curve) and 40 (upper red curve). The curves are close, hence showing that the minimal required precision for dashed trajectories to be meaningful mostly depend on the density $\eta=u/(u+v)$ and not much on the period $u+v$ of the runs.

w	smoothness model parameter
$d_{\max}$	maximal allowed speed
$\phi_{\max}$	maximal allowed smoothness criterion
s	scope width parameter
$a_{\max}$	max. # of missing consecutive points on a track
$p_{\min}$	min. # of present consecutive points on a track
F_g^γ, F_l^γ	optimization cut-off constants
$h_{\max}$	max. # of hypotheses made when optimizing

Table 2 Parameters used in the ROADS algorithm.

where $v(\mathbf{x},\mathbf{y})=\mathbf{y}-\mathbf{x}$. As we can see, this criterion combines (with a weight parameter w) an angular variation (first term) and a speed variation (second term). Assume that M_k objects are tracked until the k^{th} frame, and N_{k+1} points are observed in frame $k+1$. The trajectories already constructed can either link to one of the observed points, or to a missing "slave measurement", meaning the corresponding object in frame $k+1$ is missing. Additionally, a point of frame $k+1$ can be tagged as spurious. All these possibilities are called *individual assignments*.

Each individual assignment a has a cost $c(a)$. The cost of linking a trajectory to a point in frame $k+1$ is the smoothness criterion as defined above (if one of the past measurements is missing, we estimate its position through linear interpolation). The cost of considering a point in frame $k+1$ as spurious and the cost of linking a trajectory to a slave (missing) measurement are both equal to the value of the parameter $\phi_{\max}$ (eg. a missing point has the cost of the worst possible trajectory continuation). The algorithm restricts its possible assignments using its cut-off values, for instance, two points in consecutive frames can be linked only if they are at most $d_{\max}$ pixels apart.

The local cost of all the individual assignments between two consecutive frames is obtained by averaging their costs. Let $A^k=\{a_1,\ldots,a_p\}$ be the set of individual assignments between frame k and $k+1$, that is, such that every trajectory tracked in frame k appears in exactly one of the assignments, and every measurement in frame $k+1$ appears in exactly one of the assignments, then the local cost is

$$C^k(M_k,A^k)=\frac{1}{M_k}\sum_{i=1}^{p}c(a_i).$$

The optimization of this cost for a fixed k is a minimum-weight perfect matching problem, and can be solved efficiently using for instance the Hungarian algorithm (Munkres, 1957). Finally, the *global motion model* averages costs over the whole sequence, leading to the minimization of

$$C(A)=\sum_{k=2}^{K-1}C^k(M_k,A^k)$$

where $A=(A^2,\ldots,A^{K-1})$ is a multi-assignment. Other optimization objectives are: as many points as possible should be included in a trajectory, and there should be as few trajectories as possible. In its generality, the global motion model optimization is a NP-hard problem, thus intractable in practice. One of the approximation made by the ROADS algorithm is to sequentially optimize the global model on a restrained time window (typically using $s=2$ or $s=3$), hence computing

$$A_{\min}^{k:s}=\operatorname*{argmin}_{A^{k:s}}C^{k:s}(A^{k:s})$$

where

$$C^{k:s}(A^{k:s})=\sum_{p=1}^{s}C^{k+p-1}(M^{k+p-1},A^{k:s}[p])$$

and $A^{k:s}=(A^k,\ldots,A^{k+s-1})$ is a multi-assignment. The approximation to the global solution is then

$$A=(A_{\min}^{2:s}[1],\ldots,A_{\min}^{K-1:s}[1]).$$

This approach results in an initialization problem at the beginning of the sequence: the assignment between the first two frames is considered given. To mitigate this strong requirement, the ROADS algorithm uses a "minimal-motion" criterion $c(\mathbf{x},\mathbf{y})=\|\mathbf{y}-\mathbf{x}\|$ to initialize the assignment between the first two frames of the sequence, and then a successive up- and down-processing to reduce the imprecision of the initial assignments. We refer the reader to Veenman

et al (2003b) for a detailed explanation, including how tracks can be ended and started.

The core of the ROADS algorithm (see Algorithm 5) is the computation of the local scope optimization. In order to compute $A^{k:s}$ at each frame k, the algorithm recursively enumerates all potential assignments between successive frames in the scope. Of course, this set of assignments is too large to be exhaustively enumerated, and therefore the algorithm uses a branch-and-bound approximation strategy. It first makes an "optimal cost bound" guess C_b by initializing the global solution on the time scope with the local solutions of the minimum-weight perfect matching between each two consecutive frames in the scope. This cost bound is then gradually lowered.

At each recursion step (that is, each frame in the scope), the bound on the current optimal matching cost is lowered by using a cost-bound constraint called $\gamma_{\max}$. It is derived from the cost $C^k_{\min} = C^k(M_k, A^k_{\min})$ of the best possible assignment $A^k_{\min}$ between frames k and $k+1$ (which can be obtained by the Hungarian algorithm) and the current global cost bound C_b by

$$\gamma_{\max} = \min(F_l^{\gamma} C^k_{\min}, F_g^{\gamma} C_b / s'),$$

where $F_l^{\gamma} \geqslant 1$ is the local cost factor, $F_g^{\gamma} \geqslant 1$ is the global cost factor and s' is the length of the remaining scope.

The intuition behind this bound is that the cost of the assignment at frame k corresponding to the optimal solution on the time scope cannot be too far from the cost $C^k_{\min}$ of the optimal (local) assignment between the frames k and $k+1$, and that the cost of the assignment on the time scope is more or less uniformly distributed between all pairs of frames, and thus should not be too far from C_b/s'. To enumerate the successive best assignment between frames, ROADS uses Murty's algorithm (Murty, 1968), which takes a cost matrix D^k and a set of previous assignments Y and returns the next best assignment not in Y. The set of all assignements between frames k and $k+1$ is denoted by U^k.

The $costMatrix(A^{k-1}, k)$ function returns a matrix containing the cost of each possible assignment between a trajectory of A^{k-1} and a point of the kth image.

Algorithm: $ROAD(A^{k-1}, k, C_b, A^{k:s}_{sol})$

input : A^{k-1} the previous assignment,
k the current frame number,
C_b the current cost bound,
$A^{k:s}_{sol}$ the current best assignment
output: $A^{k:s}_{sol}$ the new best assignment

$D^k \leftarrow costMatrix(A^{k-1}, k)$
if $s = 1$ **then**
  $A^k_{\min} = minCostAssignment(D^k)$
  if $C^k(M^k, A^k_{\min}) < C_b$ **then**
    $A^{k:s}_{sol} \leftarrow (A^k_{\min})$
  end
else
  $Y \leftarrow \emptyset$
  repeat
    $A \leftarrow nextBestAssignment(Y, D^k)$
    $Y \leftarrow Y \cup \{A\}$
    $C_0 \leftarrow C^k(M^k, A)$
    $T_{sol} \leftarrow A^{k:s}_{sol}[2..s]$
    $R \leftarrow ROAD(A, k+1, s-1, C_b - C_0, T_{sol})$
    $A^{k:s} = (A) :: R$
    if $C^{k:s}(M^k, A^{k:s}) < C_b$ **then**
      $C_b \leftarrow C^{k:s}(M^k, A^{k:s})$
      $A^{k:s}_{sol} \leftarrow A^{k:s}$
    end
    $\gamma_{\max} = \min(F_l^{\gamma} C^k_{\min}, F_g^{\gamma} C_b / s)$
  until $Y = U^k$ *or* $C_0 \geqslant C_b$ *or* $C_0 \geqslant \gamma_{\max}$
end
`return` $A^{k:s}_{sol}$

Algorithm 5: Core of the ROADS algorithm

4.2 Experiments

In the following, we propose to compare the NFA algorithm with ROADS to evaluate its strengths and weaknesses against a state-of-the-art solution. We start with experiments on synthetic data, similar to those used by the authors of ROADS in their presentation papers (Veenman et al, 2003a,b). Let us first briefly present the way they generate trajectories using the Point-Set Motion Generation (PSMG) algorithm (experiments having different parameters will be signaled):

- the initial position of each trajectory is chosen uniformly at random in the first image;
- the initial velocity magnitude is chosen using a normal random variable $v_0 \sim \mathcal{N}(\mu = 5, \sigma = 0.5)$ and its angle β_0 is chosen using a uniform distribution in $[0, 2\pi]$;
- the velocity magnitude and angle are updated in each frame using $\begin{cases} v_{k+1} \sim \mathcal{N}(\mu = v_k, \sigma = 0.2) \\ \beta_{k+1} \sim \mathcal{N}(\mu = \beta_k, \sigma = 0.2). \end{cases}$

The image domain is divided in 100×100 pixels, and the length of the sequence is set to 20 frames (see Fig. 9 for an illustration of the trajectories generated). Most of the experiments are realized with 20 trajectories (like in the ROADS paper).

Since the ROADS authors were comparing their algorithm with an algorithm that did not allow trajectories entering or leaving the scene, they required that all trajectories fit completely inside the frames and span the whole sequence, and we will usually do the same (if a trajectory does not fit inside the frame, we regenerate it). They also impose that in the experiments where points are missing, all points are still detected in the first and last two frames. To have experiments coherent with theirs we generally impose the same constraints, but in some experiments (with a great number of trajectories in the images) constraining trajectories to stay inside the frame seemed unnatural since it forced trajectories

to have a beating and swirling motion inside the frames. We chose to keep a constant number of trajectories, but to allow them to leave the image, and when this happens, to generate a new random trajectory that starts on the border of the frames (we force all trajectories to have at least three points in the frame).

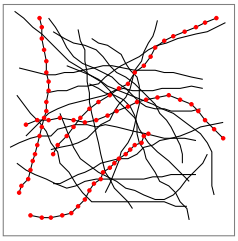


Fig. 9 Point-Set Motion Generation (PSMG) algorithm. Here we display a sample of 20 trajectories spanning 20 frames generated with the PSMG algorithm, which will be used to produce synthetic data to assess the performance of the ROADS and NFA algorithms. The trajectories have an homogeneous and smooth motion. The points of some trajectories have been highlighted to show the speed.

Additionaly, we impose the following constraints:

- we quantize the trajectory coordinates to the nearest integers, thus *implicitely defining the scale of the measurements* to the size of one pixel;
- to avoid ambiguities when comparing the detection results to the ground truth, trajectories cannot share points (otherwise we regenerate one of the trajectories) and when adding noise points we avoid covering already existing points.
- when we remove points we target solely trajectory points (we do not remove noise points, so we can make experiments with a varying number of trajectory points removed, while keeping a constant number of noise points). We choose a certain uniform probability α of removing a point.

Performance estimates gathered in the experiments below are averages of a measure over 400 runs of the algorithmson random data. However, in some experiment results, the measure that we compute might be undefined (for instance when no trajectory was detected). In this case, we only take experiment results that have a defined measure into account, and measurements might thus consist of averages of less than 400 repetitions.

A well-known interest of a-contrario methods are their small number of parameters, which simplifies their use and their study. More accurately, the NFA algorithm has exactly one explicit parameter, the maximal NFA value of a trajectory we can extract. The effect of this threshold is simple: it drives the selection of a subsequence of the successively extracted trajectories. In other words, if $\varepsilon < \eta$, $\mathbb{T}(\varepsilon) \subseteq \mathbb{T}(\eta)$, where $\mathbb{T}(x)$ is the set of trajectories extracted by the algorithm for a maximal value of the NFA equal to x. This implies that changing the threshold will not dramatically change the results, contrary to methods like ROADS that use their parameters *in* the computations. In practice, as usually done in a-contrario methods (see Desolneux et al (2008)) and unless otherwise specified, we set this threshold ε to 1.

In contrast, ROADS has many parameters, which can be tuned to set ROADS in different "modes" that may be better suited to certain types of data. Since these parameters might (at least in theory) be learned on data, we felt it was fair to try several sets of parameters and show the best results that could be achieved in the comparisons.

Here is the way we proceeded: we tested six "modes" for the ROADS algorithm on a small batch of data (40 repetitions) for each experiment. We then selected the three modes that would compare the best with the NFA algorithm on the various experiments. Some of the modes will have strengths and weaknesses compared to others, but they mostly have the same global behaviour. In practice, the strengths and weaknesses of the NFA method when compared to ROADS do not dramatically change when including several modes, rather than just the most general parameters for the ROADS algorithm (mode 1 below). However, we include the results of the three selected modes for the sake of completeness.

To be fair with the ROADS algorithm, which relies on knowing the maximal speed and maximal smoothness criterion of the trajectories in the data, we compute these values and give them to the algorithm. More precisely, for each experiment, and each parameter (eg. number of noise points added), we compute the maximal speed $d_{\max}$ and maximal smoothness criterion $\phi_{\max}$ before crippling (eg. removing points) across the batch of 400 repetitions (rather than on a per-file basis), and we feed them to the algorithm when processing those 400 repetitions.

The first mode is the general ROADS algorithm with the minimal number of present points set to $p_{\min} = 1$ and the maximal number of interpolated points equal to $a_{\max} = +\infty$. The second mode is $p_{\min} = 1$ and $a_{\max} = 0$, that is, we disallow interpolation. The third mode is $p_{\min} = 3$ and $a_{\max} = 0$, which means that we disallow interpolation and we expect to see at least 3 consecutive points on each trajectory segment. For the three other modes, we set $p_{\min} = 3$, $a_{\max} = 3$, but rather than choosing the maximal speed and maximal smoothness criterion as given by their maximal value on the batch of 400 repetitions, we select in turn: $d^4_{\max} = 0.8 \cdot d_{\max}$, $\phi^4_{\max} = 0.8 \cdot \phi_{\max}$ for mode 4, $d^5_{\max} = 0.5 \cdot d_{\max}$, $\phi^5_{\max} = 0.5 \cdot \phi_{\max}$ for mode 5 and $d^6_{\max} = 0.5 \cdot d_{\max}$, $\phi^6_{\max} = 0.8 \cdot \phi_{\max}$ for mode 6.

The other default ROADS parameters given in the implementation that was sent to us by its authors were kept unchanged ($w = 0.1$, $F_\ell = F_g = 1.05$, $s = 2$). We tried to make some experiments with $s = 3$, but this would generally

not change the results (and sometimes even have a negative impact) and be much more computationally intensive. The maximal number of hypotheses $h_{\max}$ that ROADS can explore when trying to find the best assignment in the time scope has been kept equal to 300 as in the given implementation.

After running all the ROADS modes on small batches of repetitions, we selected modes 1 (original ROADS), 3 and 4 as giving the best results. They will now be called modes A, B and C (see Table 3). Note that these modes are not real algorithms, since their parameters depend on true data values ($d_{\max}$ and $\phi_{\max}$) that are not estimated but computed from an oracle. In that sense, the methodology we use to compare the NFA and ROADS algorithms minimizes the issue of parameter selection that is recurrent with ROADS (but this issue will be discussed later, in particular in Section 4.3.4). Note also that ROADS results could probably be slightly improved by trying a larger number of parameters, however our goal is not to make a study of ROADS, but rather to give an idea of the state of the art performances, to make it possible for the reader to appreciate the NFA results.

mode	$p_{\min}$	$a_{\max}$	$d_{\max}$	$\phi_{\max}$
A	1	$+\infty$	$1 \cdot d_{\max}$	$1 \cdot \phi_{\max}$
B	3	0	$1 \cdot d_{\max}$	$1 \cdot \phi_{\max}$
C	3	3	$0.8 \cdot d_{\max}$	$0.8 \cdot \phi_{\max}$

Table 3 Parameters defining the three (best) modes of the ROADS algorithm in the experiments used for comparison with the NFA algorithm. Note that these three modes are based on an oracle that observes the values of $\phi_{\max}$ and $d_{\max}$ on the (supposedly unknown) true trajectories.

4.2.1 Comparison criteria

In the literature, tracking algorithms are generally compared using two sets of criteria: the qualitative description of the situations that the algorithm can handle (missing points, entry of points, etc), and the quantitative criteria given by the number of real structures found in the sequence (eg. the number of real trajectories, of real links between points, etc.), as well as the *precision* and *recall* of the algorithm for these different structures, defined by

$$\text{precision} = \frac{\#\text{ of correct structures found}}{\#\text{ of structures found}}, \tag{31}$$

$$\text{and recall} = \frac{\#\text{ of correct structures found}}{\#\text{ of actual structures}}. \tag{32}$$

The precision allows to measure the number of false positives (more precisely, $1-$precision is the proportion of false positives amoung found structures), while the recall is linked to the number of false negatives ($1-$recall represents the proportion of false negatives among actual structures). It is important to realize that the analysis of an algorithm must be done by considering simultaneously the precision and recall (or equivalent variables), since varying a parameter or a threshold of an algorithm generally does not improve both quantities but sets a different trade-off between the two, resulting in a better recall and worse precision or vice-versa.

In some experiments, the presence of noise points limits the interest of the number of real (whole) trajectories found as a significant criterion, although it is widely used in the litterature. Indeed, a well-placed noise point can sometime better fit the trajectory than its "real" counterpart, thus giving a realistic and usable trajectory as output, yet one that will not be counted as a real trajectory. We therefore chose to generally use the *number of correct links* as a significant structure for the precision and recall criteria. A link is simply two points that appear consecutively on a trajectory (possibly separated by a hole). Thus, if a noise point better fits a trajectory than its "real" counterpart, we will only "miss" two correct links (that include the real point), and "create" two false links (that include the noise point). However, when using the number of correct links, we do not account for trajectory over-segmentation, under-segmentation, or mixing. More precisely, if we split a trajectory in half, or if we join two distinct trajectories, we will barely notice it from the point of view of the number of correct links criterion, but we would have noticed it using the number of correct trajectories criterion. The same is true for "mixed" trajectories: if two trajectories cross at a point in time, we might start by following trajectory A, and then either choose to continue with trajectory A or to "hop" on trajectory B. In the latter case, the number of correct links criterion will barely be affected, but the number of correct trajectories criterion would. This particular problem of crossing trajectories appears however to be difficult to solve properly, and would certainly requires *a priori* knowledge. We believe that once the trajectories have been detected, even if they have been mixed, a simple post-processing task might be sufficient to split crossing trajectories in part at the crossing points, and reconstruct the real trajectories using an *a priori* (having the trajectories bounce if we are following billard balls, or having them cross if we are looking at fishes in an aquarium).

For the qualitative criteria, ROADS is able to account for missing and spurious points, as well as points leaving and entering the scene. The NFA algorithms come in two flavors, one that allows for missing points, and the other that does not. The latter is used for computational reasons (it is much faster) in some of the following experiments. Both NFA algorithms allow spurious points, as well as points leaving and entering the scene.

4.3 Trajectories without holes

4.3.1 Variable number of spurious points experiment

First, we investigate the influence of noise (spurious points) on trajectory detection. We generate sequences spanning 20 frames and having 20 trajectories, and we add 0 to 320 noise points uniformly at random to each image. Since we do not remove points, we can use the version of the NFA algorithm that does not take holes into account (Section 2). Then we run the NFA ($\varepsilon = 1$) and the ROADS (modes A,B,C) algorithms, and compare the results by computing the average recall and precision over 400 repetitions. For the precision estimate, the averaging is limited to the repetitions that lead to at least one detection, since the precision is not well defined when no structure is found.

As we explained earlier, the precision and recall are computed for two different criteria: the number of correct links and the number of correct trajectories. For the criterion based on the number of correct links (Fig. 10), the NFA algorithm performs much better in terms of precision: the precision remains very high (above 80%) for the NFA algorithm, but drops very fast for ROADS (under 20% when the number of spurious points exceeds 140). This illustrates a classical property of a-contrario detection models: the robustness to noise. As concerns the recall, the NFA algorithm performs better than all versions of ROADS up to 200 spurious points, and is slightly under the C mode of ROADS beyond this level of noise. Considering the number of false detections made by ROADS at these levels of noise (the precision is under 10%), the global comparison remains in favor of the NFA algorithm. Table 4 clearly illustrates this: if ROADS manages to find a lot of correct links, it is solely because it makes a huge number of detections when the number of noise points increases, whereas the NFA algorithm correctly finds 20 trajectories in low noise and makes fewer detections when the noise level increases.

When we look at the number of correct trajectories found (Fig. 11), we see that ROADS is very good when there are no noise points. The NFA algorithm is a bit less efficient (both in terms of precision and recall) when the number of spurious points is under 40, but for higher levels of noise it is much more robust than ROADS, whose performances collapse very quickly (both in terms of precision and recall). A comparison with Fig. 10 is interesting here, because it shows that the "number of correct trajectories" is a very specific criterion that is not very relevant for complex or noisy data (the performances drop very quickly, much before the "number of correct links" is significantly affected). As we remarked before, there are plenty of reasons why a detected trajectory could be counted as undetected while it is very near an actual trajectory (a missing endpoint, a noise point fitting better the trajectory smoothness, trajectory crossings, etc.). Also, it is clear that applications based on data corrupted by a medium or high level of noise are more interested in a high rate of local point tracks (links) than in the perfect reconstruction of a very small proportion of the complete actual trajectories. This is why we shall not use the "number of correct trajectories" criterion any more in the following, but focus instead on the broader "number of correct links" criterion.

# spurious points	0	40	120	200	280	320
NFA	20.1	20.2	18.9	15.5	7.1	6.1
ROADS(A)	20.0	70.0	151.5	227.1	300.9	335.7

Table 4 Average number of detected trajectories depending on the level of noise. We compare the average number of trajectories detected by the NFA and ROADS algorithms on data made of 20 real trajectories spanning the entire sequence plus a varying number of spurious points (from 0 to 320) in each frame. We see that NFA is very conservative in its detections (it only detects the trajectories that it considers to be non-ambiguous), and this results in a high precision (see Fig. 10). On the other hand, ROADS makes many false detections (it should not find more than 20 trajectories per sequence).

4.3.2 Variable density experiment

We now test how the algorithms behave when we increase the number of points. In this experiment, we do not consider spurious or missing points, so there is no noise and the difficulty of the trajectory detection problem only comes from the ambiguities produced by the large number of mixed trajectory points. We generate sequences of 20 frames, containing 10 to 140 points moving according to the PSMG model, where we allow trajectories to leave the image frame (when a trajectory leaves the image frame, we generate a new trajectory starting at a random position in the image frame, in order to keep a constant number of points throughout the sequence). Then we compute the precision and recall for the correct links criterion (Fig. 12) for the ROADS and NFA (without holes) algorithms.

When using the standard threshold ($\varepsilon = 1$) in the NFA algorithm, we obtain results which are similar (slightly better) than ROADS in terms of precision but significantly worse in terms of recall. However, knowing that there is no noise in these data, it makes sense to try to set the NFA threshold to $+\infty$ (that is, no threshold), and in this case the results obtained by the NFA algorithms are similar to the best modes of ROADS. This is an unexpected good surprise for the NFA algorithm, which detects trajectories in a greedy way (by iterating a best-trajectory-detection/trajectory-removal process) without considering at all the global inter-frame assignment problem like ROADS. In the absence of noise points, one could have expected this assignment step to bring a significant edge to ROADS.

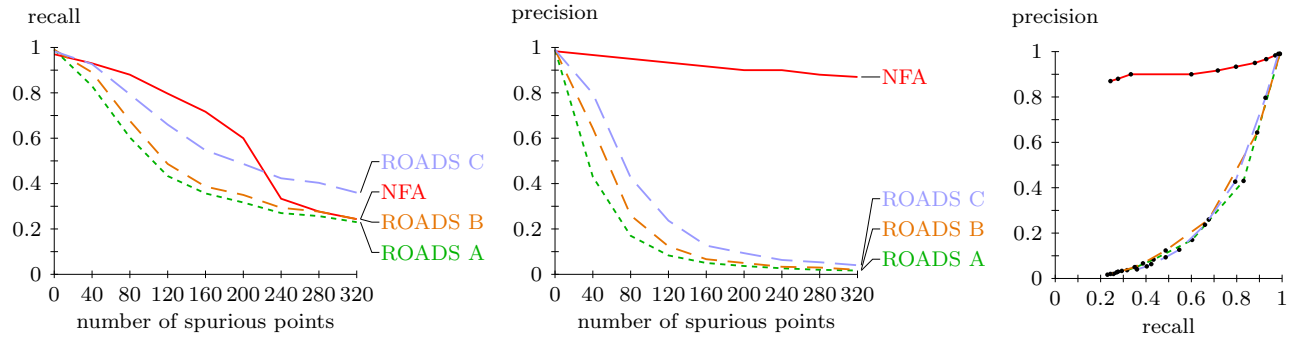


Fig. 10 Influence of spurious points (# correct links criterion). On synthetic data containing 20 real trajectories spanning the entire sequence (20 frames) plus a varying number of spurious points (from 0 to 320), we compute the average recall (left) and precision (middle) obtained with the NFA and ROADS algorithms over 400 realizations, as a function of the level of noise (number of spurious points), or together (right). The most striking result here is that the precision of the NFA algorithm is almost constant, no matter the number of spurious points. This means that the NFA algorithm makes very few false detections (which is how we designed it), while keeping a recall rate that is above the one of ROADS as long as the number of spurious points is under 200 (which is more surprinsing). On the contrary, the poor precision of the ROADS algorithm in medium or high noise conditions makes its recall values quite unsignificant : if ROADS finds a large number of correct links, it is moslty because it proposes a high number of links, most of which are false detections (see also Table 4).

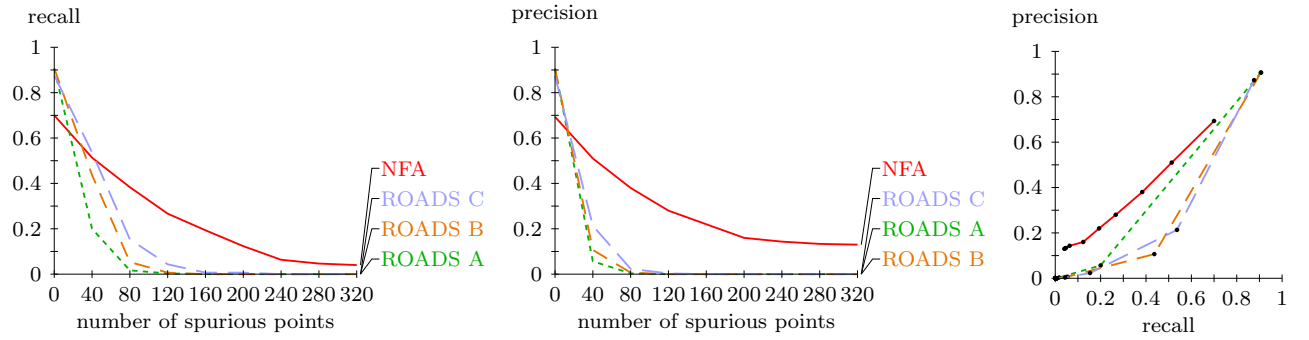


Fig. 11 Influence of spurious points (# correct trajectories criterion). The results of the experiments conducted in Fig. 10 are now analyzed with a different criterion (the number of correct trajectories, instead of the number of correct links) for the definition of precision and recall. We can see that the NFA algorithm behaves much better than all ROADS modes as soon as there is a reasonable level of noise, while ROAD gives slightly better results when the noise level is very low. Note that the "number of correct trajectories" is a very specific criterion that is quite sensitive to local ambiguities (and to the selection of trajectory endpoints), and for that reason it is not very relevant when dealing with complex and/or noisy data.

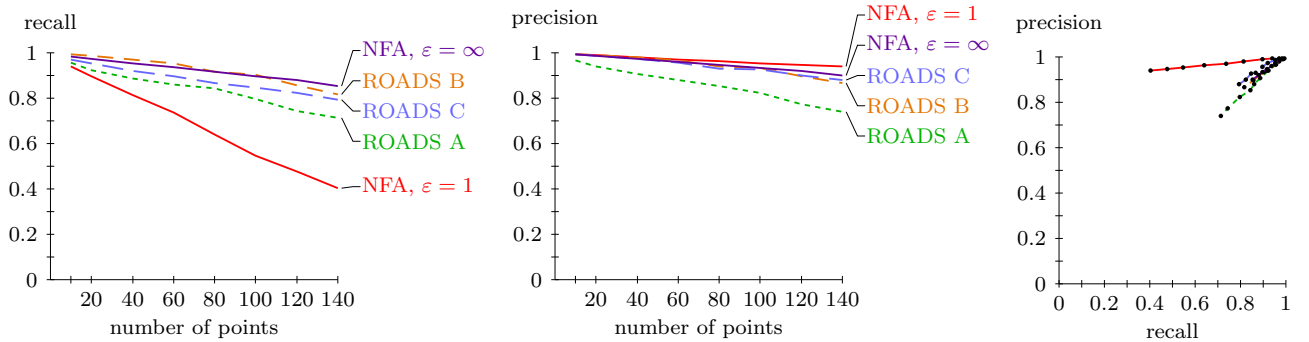


Fig. 12 Influence of the number of trajectories. The average recall (left) and precision (middle) are computed for the number of correct links criterion on 400 repetitions of synthetic data made of a given number of random trajectories (varying from 10 to 140) in a sequence of 20 frames. The analyzed algorithms are the three modes of ROADS (A, B, C), and two variants of the proposed NFA algorithm: the standard variant (threshold $\varepsilon = 1$ on the expected number of false alarms), and the no-threshold variant ($\varepsilon = +\infty$). As we can see, the precision of both NFA variants is very high (like for ROADS B and C), but the recall of the standard NFA algorithm is significantly worse than the one of ROADS. In this setting where no noise points are present, these missing detections can be avoided by removing the thresholding process in the NFA algorithm: for this $\varepsilon = +\infty$ variant, both recall and precision are as good as the best modes of ROADS.

4.3.3 Sensitivity to data smoothness

The trajectories generated in the previous experiments are somehow smooth and quasi-linear (see Fig. 9). In order to see if the algorithms can cope with trajectories that do not completely fit the model, we try to detect trajectories having a potentially high acceleration. For that purpose, we consider different values of σ, the standard deviation of the acceleration magnitude used in the PSMG synthesis procedure (see the very begining of Section 4.2). The effect of this parameter on the synthetized trajectories is illustrated in Fig. 13.

We first reproduce the last experiment (Fig. 12), in which there are no noise points and the number of synthetized trajectories ranges from 10 to 140, and evaluate the effect of the σ parameter both for the unthresholded NFA algorithm (Fig. 14) and the ROADS B algorithm (Fig. 15). Whereas the performance of the NFA algorithm barely depends on σ (and remains high), ROADS exihibits a high sensitivity to this parameter, and its performance quickly collapses as σ increases. The same conclusion arises from the analysis of data generated with 10 noise points per frame plus 20 synthetic trajectories (see Fig. 16).

Thus, the sensitivity to data smoothness is a major difference between the NFA and ROADS algorithms. The poor results obtained by ROADS for $\sigma = 1$ (see Fig. 15) could probably be improved by a specific choice of the ROADS parameters (specially adapted to $\sigma = 1$), but this kind of optimization will not be efficient on most real-world data, since one can expect to observe a high variability of accelerations on such data. Conversely, the robustness of the NFA algorithm to the σ parameter is an indication that it can probably handle well real-world data containing various levels of acceleration.

4.3.4 Parameter tuning

One major interest of most a-contrario methods is that they yield "parameterless" detection algorithms, or, more precisely, algorithms for which there exist natural values of the parameters that work well in all situations. Both NFA algorithms we propose here (the no-hole and hole versions) have only one parameter: the threshold ε used to decide whether a trajectory should be detected or not. Since ε corresponds to an upper bound on the expected number of false alarms in pure noise data, its default value is classically set to 1 (see Desolneux et al (2008)). In Fig. 17, we examine the sensitivity of the NFA no-hole algorithm with respect to the choice of ε. We use the same experimental setting as in Fig. 10 (20 frames containing 20 real trajectories plus several spurious points), and examine how recall and precision are affected by different choices of ε. The results clearly show that the default value $\varepsilon = 1$ ($\log_{10}\varepsilon = 0$) is nearly optimal, in the sense that it is small enough to guarantee a strong precision control, and large enough to offer good recall performances. It is nonetheless interesting to notice that slightly better performances (same precision and better recall) can be obtained with greater values of ε (typically $\log_{10}\varepsilon = 2$ or 3).

In Fig. 18, the average precision/recall curve obtained with the NFA algorithm for different values of the threshold $\log_{10}\varepsilon$ (in the case of 160 spurious points) is displayed on the left. On the right, we report the average performances of ROADS on the same data points, considered for several values of the two main parameters of this algorithm, namely the maximal speed and the maximal smoothness. The maximum speed parameter varies from the actual value in the $[-50\%, +50\%]$ range (from one curve to another) and the maximum smoothness varies from the actual value in the $[-95\%, 50\%]$ range (inside each curve). Note that the best performances of ROADS (-25% speed, -90% smoothness) are obtained inside these ranges.

As we can see, not only the performances of ROADS are way under those of NFA on these data, but also the parameter tuning is much more difficult and crucial (we have to explore carefully a bidimensionnal domain, while $\varepsilon = 1$ is almost optimal for the NFA algorithm).

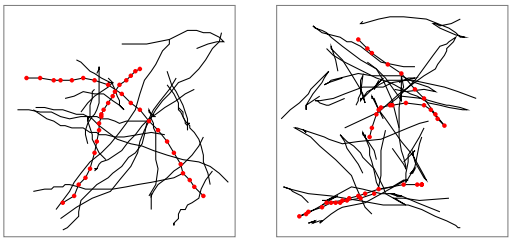


Fig. 13 Changing the acceleration variance. A sample of 20 trajectories generated using the PSMG algorithm, when the standard deviation of the acceleration magnitude (σ) is 1 (left) and 4 (right). The points of two trajectories have been highlighted to show the speed. We study the sensitivity of the algorithms to data variability by analyzing their performances when we increase σ.

4.3.5 NFA as a criterion for trajection selection

Contrary to ROADS, which is by nature an algorithm (relying in particular on some heuritics), the NFA we propose here is first and foremost a criterion to compare trajectories. The greedy algorithm we described, based on the iteration of a "best trajectory (minimal NFA) detection / trajectory removal" process, is only one possibility to use the NFA criteria (5) and (23), and it is possible to design other algorithms based on these criteria. In particular, given a trajectory detection algorithm, it is always possible to use the NFA criterion as a post-processing step that simply keeps from the output of the considered algorithm the trajectories having a NFA under a certain threshold ε.

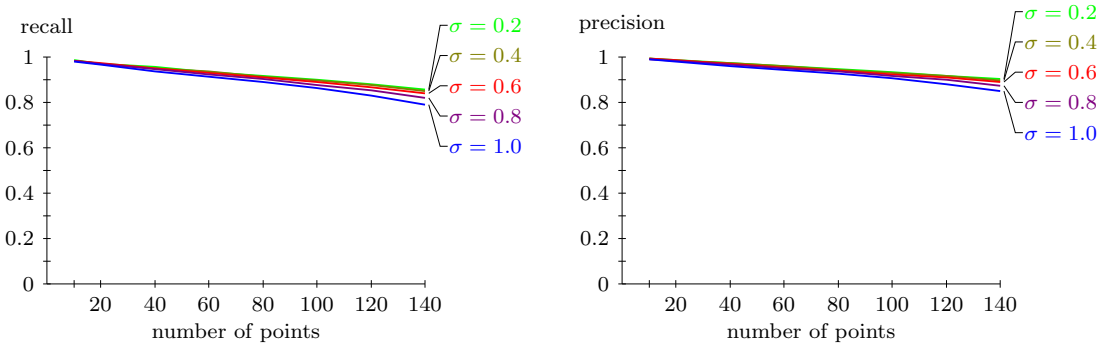


Fig. 14 Influence of the data smoothness (unthresholded NFA algorithm). We use the same experimental setting as Fig. 12, and examine the sensitivity of the unthresholded ($\varepsilon = \infty$) NFA algorithm to the smoothness of the analyzed synthetic data. More precisely, we consider several values of σ, the standard deviation of the acceleration magnitude (a parameter of the synthesis algorithm, PSMG), and estimate the precision and recall (correct links criterion) as functions of the number of synthetic trajectories. We can see that the NFA algorithm is extremly robust to σ, since both the precision and recall performance curves remain unchanged when σ varies.

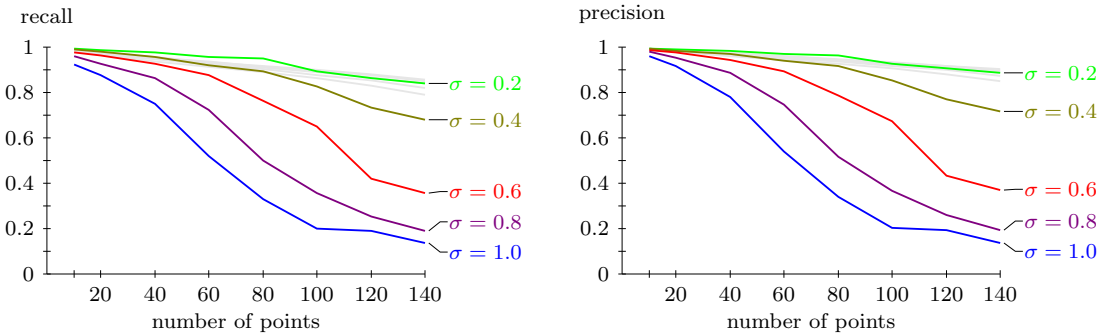


Fig. 15 Influence of the data smoothness (ROADS B algorithm). We analyze the same data as in Fig. 14, now with the ROADS algorithm, mode B (the best mode for these data, see Fig. 12). Contrary to what happens for the NFA algorithm, the ROADS method exhibits a severe sensitivity to σ, since both recall and precision performances, which were at the same level as the NFA algorithm for $\sigma = 0.2$ (grey shadow curves), are strongly affected when σ increases. As we shall see in Section 4.5, the sensitivity/robustness to data variability has strong consequences when real-world data are analyzed. Note incidentally the strong similarity between the recall and the precision curves, which comes from the fact that in this set of experiments, the number of detected links is most of the time equal to the number of actual links (see Equation (31) and (32)), probably because there are no spurious points.

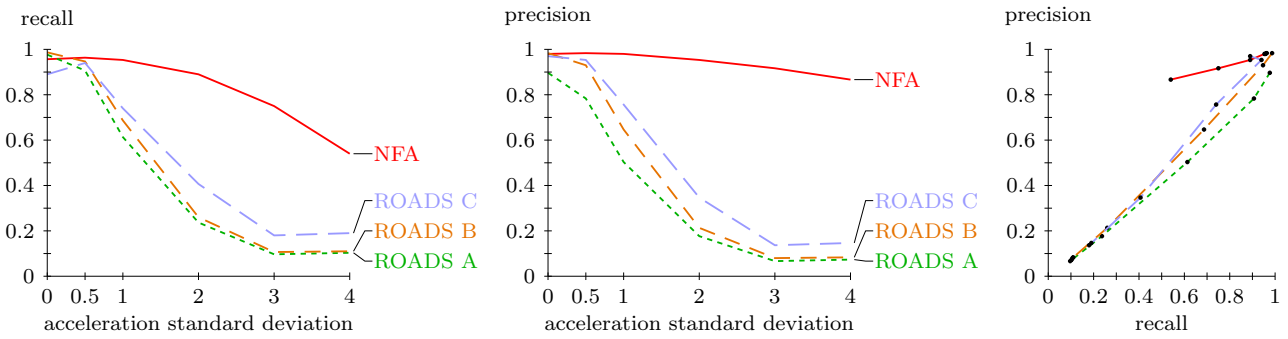


Fig. 16 Sensitiviy to data smoothness. The average recall and precision of the ROADS algorithm (modes A, B, C) and the standard NFA algorithm ($\varepsilon = 1$) are compared in synthetic data made of 10 noise points per frame plus 20 random trajectories spanning 20 frames, in function of the standard deviation of the acceleration magnitude (σ). These results corroborate the ones obtained in Fig. 14 and 15: the performances of the NFA algorithm are not too much affected by the increase of σ (except for the recall when the variance becomes large, probably because the problem of recovering the true trajectories becomes objectively difficult), whereas the performances of all ROADS algorithms collapse, both in terms of precision and recall.

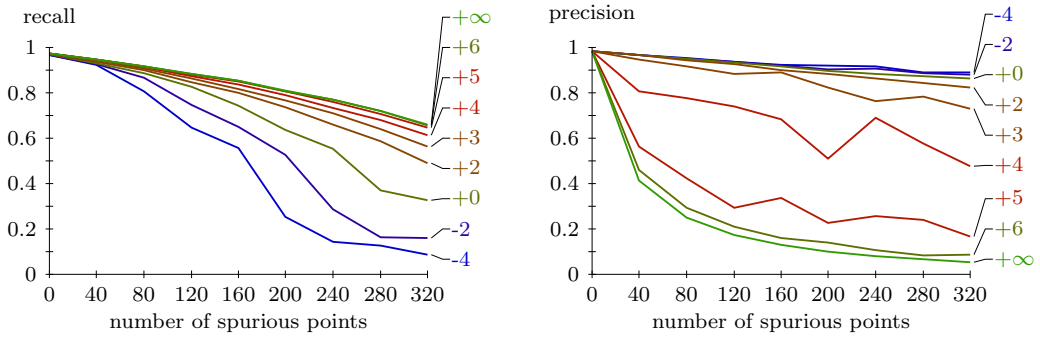


Fig. 17 Influence of the NFA threshold (ε). We consider the same experiment as in Fig. 10 (that is, 20 real trajectories spanning 20 frames, with a given number of spurious points in each frame), and examine the influence of the threshold ε arising in the NFA algorithm. Recall and precision curves are plotted in function of the number of spurious points, for different values of $\log_{10}\varepsilon$ (ranging from -4 to $+\infty$). We can see that the good precision control predicted by the theory for $\varepsilon \leqslant 1$ ($\log_{10}\varepsilon \leqslant 0$) is well achieved, since the first significant precision losses occur around $\log_{10}\varepsilon = 3$. Hence, the default value $\log_{10}\varepsilon = 0$ is a good compromise in this experiment, even if slightly better recalls (without significant precision losses) can be achieved by using greater values like $\log_{10}\varepsilon = 2$.

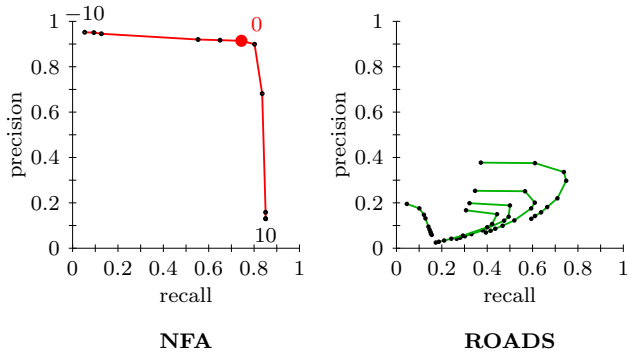


Fig. 18 Performance and parameter tuning. We consider a particular case of Fig. 17, that is, synthetic data made of sequences of 20 frames containing 20 real sequences and 160 spurious points on each frame. The average performances in terms of precision/recall is then evaluated for the NFA algorithm (left) and the ROADS algorithm (right), with varying values of the algorithm parameters. For the NFA algorithm, the only parameter is the treshold ε (or $\log_{10}\varepsilon$, as displayed on the figure), and we can see that the default value $\log_{10}\varepsilon = 0$ is very near to be optimal, as was remarked earlier in the comment of Fig. 17. For ROADS, not only the performances are lower (especially in terms of precision), but they are also quite sensitive to the choice of the maximum speed and maximal smoothness parameters.

We tested this possibility with the ROADS algorithm, and reported in Fig. 19 the results obtained on the synthetic data used in the previous section (parameter tuning). It appears that the mixed ROADS+NFA algorithm we obtain this way performs much better than ROADS alone in terms of precision (because the NFA filtering permits to eliminate most false detections), but the performances in terms of recall do not attain the ones of the NFA algorithm alone. Hence, the "NFA filtering" strategy is efficient but does not provide a particularly interesting new algorithm when applied to ROADS. It is not impossible, however, that such a strategy could be successful, in particular in situations where only special kinds of trajectories appear and a good detection algorithm (in terms of recall) exists. In that kind of situation, one could expect NFA filtering to increase the precision up to a high level, without damaging to much the recall performances. Note that such a strategy guarantees, thanks to the properties of the NFA criterion (1), the control of the number of false detections in random data.

4.4 Trajectories with holes

We now examine the performances of the second NFA algorithm (Section 3), which is able to handle trajectories with holes. We compare it to ROADS using the same kind of conditions as in Fig. 10 (20 real trajectories, 20 frame, several spurious points added in each frame), except that we now consider uncomplete trajectories (20% of the points of the true trajectories are removed before spurious points are added). The conclusions made in the no-hole case remain unchanged (see Fig. 20) : the ROADS algorithm detects true trajectory links as well as the NFA algorithm, but at the price of many false detections, whereas the NFA algorithm makes almost no false detection (the precision remains above 0.9, even for 70 spurious points per frame).

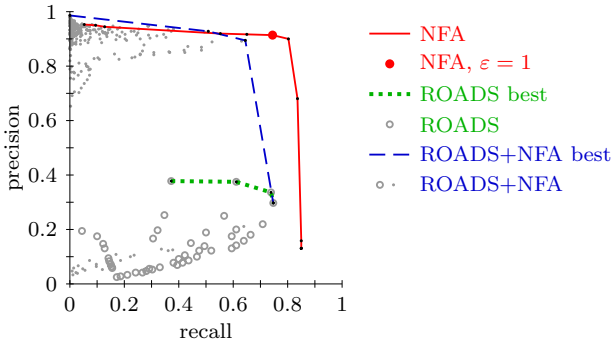


Fig. 19 ROADS output filtered by the NFA algorithm. We consider the same synthetic data as in Fig. 18, but now add to the comparison of NFA and ROADS algorithms a combination of them that consists in detecting trajectories with ROADS and keeping only those having a NFA under a certain threshold ε. Since each algorithm depends on parameters (1 for NFA, 2 for ROADS, 3 for ROADS+NFA), we explore systematically all parameter values and compute the upper performance enveloppe (curves named *best*). As we can observe, the major drawback of ROADS (which is its high rate of false detections) can be corrected by NFA fitering, which results in a dramatic increase of precision (up to the level of the NFA algorithm alone). However, this correction does not permit to attain the same level of recall (around 0.75 for NFA, versus 0.6 for ROADS+NFA in the high precision zone). Note also that the mixed ROADS+NFA algorithm would be much more complicated to use than NFA alone, due to the 3 parameters that have to be set.

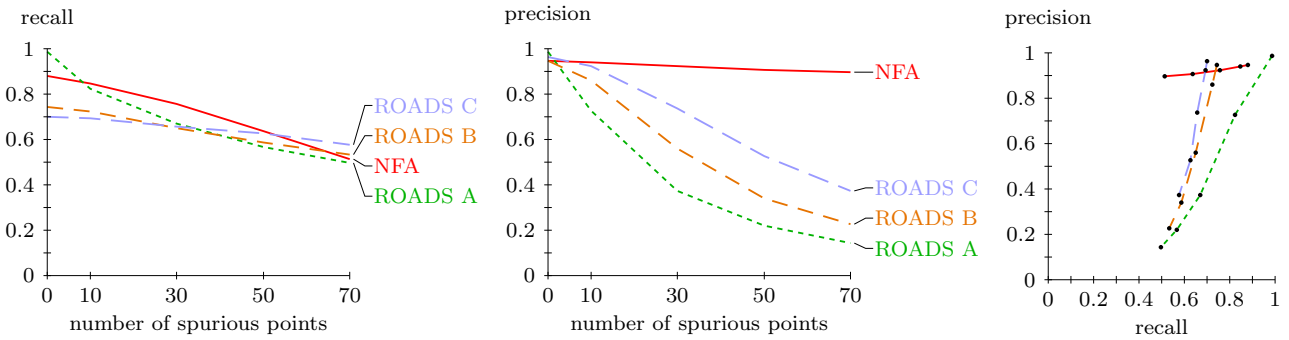


Fig. 20 Influence of spurious points for trajectories with holes. We generate 20 trajectories spanning the whole sequence (20 frames), and remove randomly 20% percent of the points, before we add a varying number of spurious points (from 0 to 70). On these synthetic data (with 400 repetitions), we estimate the recall (left) and the precision (middle) of the ROADS and NFA algorithms for the "number of correct links" criterion. The obtained results are very similar to those of Fig. 10: the recall values are roughly the same for all algorithms, but only the NFA algorithm manages to maintain a high precision (above 0.9) as the number of spurious points increases, while all ROADS variants make lots of false detections.

4.5 Trajectories of real-world images

4.5.1 The snow sequence

In this part, we evaluate the relative performances of NFA and ROADS algorithms on a real-world sequence named *snow*. To produce this sequence, we filmed falling snowflakes in front of a dark metal door with a high-speed (210 frames per second) camera, and then subsampled the high speed sequence at 30 fps by taking 1/7 of the original frames. This way, we obtained a classical 30 fps sequence of 40 images of 480×360 pixels, on which we ran a simple point extraction process that we describe below. The high-speed version was processed in the same way and used in order to build a hand-made ground truth for trajectories.

We purposefully used a very simple extraction process to produce data as objectively as possible, without trying to adapt the detection algorithm in a way that would affect (and ease) the tracking part. The snowflakes (but also some stains on the metal door background) were detected in the following way: we smoothed the images using a simple Gaussian kernel, and we computed the mean background image on a few frames of the subsampled (30 fps) sequence. We then thresholded the image differences, processed the result with a morphological closing, and extracted the connected components. For each connected component, we kept the centroid position, rounded to the nearest point on the integer grid, as a trajectory data point.

In the resulting point sequence, many objects were detected as several close points in the sequence (in particular the stains on the background and some big snowflakes). This

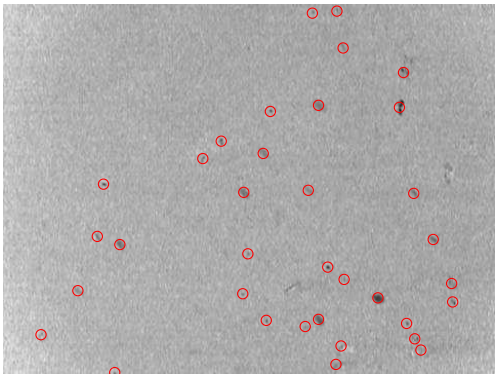


Fig. 21 An image of the *snow* sequence (inverted grayscale), with overlayed detections.

made it sometimes hard to define the ground truth trajectories. To alleviate this difficulty, we kept only isolated points by removing all points in the sequence that were below a certain distance of another point (we chose the smallest radius that would resolve almost all ambiguities, in practice 12 pixels). An example of detections on one frame of the sequence is displayed in Fig. 21. We finally extracted the ground truth trajectories by hand.

The resulting point sequence is interesting because it presents a mix of difficulties: there are widely varying trajectory types (points in the background which practically do not move, very slow snowflakes with curvy trajectories, very fast snowflakes with almost linear trajectories). There are missing points (missing detections or detections removed because of the non-isolated point removal process), and a few noise points (but the relatively high detection threshold gave more missing points and fewer noise points).

Finally, we subsampled the high-speed point sequence by keeping only 1/7 of the frames, and subsampled accordingly the ground truth trajectories. The resulting trajectories containing less than 3 points were eliminated from the ground truth reference, but the corresponding points were kept in the data (thus becoming noise points). The final result of this process (30 fps *snow* point sequence and associated ground truth) is available on the website

`http://www.mi.parisdescartes.fr/~moisan/astre/`

The first row of Fig. 25 gives an idea of the ground truth trajectories extracted from the *snow* sequence.

4.5.2 Parameter tuning

There are several parameters to set for ROADS (see Table 2), and different settings lead to varying results. Namely, we can set the size s of the time scope (we chose 2, giving the best results), the minimal number p_{min} of consecutive present points for a trajectory to be considered (we chose 1, 3, 5 or 7), the maximal length of interpolation a_{max} before we loose the trajectory (we chose 0, 4, 8 or $+\infty$), the maximal smoothness criterion $\phi_{\max}$ and the maximal speed $d_{\max}$. The way to choose the best parameters is not obvious, but it appears in Fig. 22 that the most important parameter is the maximum allowed speed $d_{\max}$. The choices $p_{\min} = 1$, $a_{\max} = 0$ and $\phi_{\max} = 0.4$ are among the best possible for the *snow* point sequence, and would probably achieve reasonable performances on similar sequences too. As concerns the choice of $d_{\max}$, the ground truth value (160) is much too large, and much better results are obtained with $d_{\max} = 15$. This fact, which comes from the unability of ROADS to deal with a variety of trajectory speeds at the same time, is analyzed more precisely later. Note that the ground truth value of $\phi_{\max}$ is 0.58.

On the *snow* sequence, extracting trajectories using the two NFA algorithms (the one with holes and the one without holes) would return trajectories having a value of $\log_{10}(\mathrm{NFA})$ varying from -40 to $+10$, and the "optimal" precision/recall values would be obtained (for both algorithms) by thresholding this value with $\log_{10}\varepsilon = +5$ (see Fig. 23). Even without access to the ground truth, finding this value is relatively easy, since one simply has to look for values slightly above the (nearly optimal) default value $\log_{10}\varepsilon = 0$. This strategy works well in all synthetic experiments we considered earlier, and also in the present case of the *snow* sequence. In view of the false detection control offered when $\log_{10}\varepsilon = 0$, such a strategy is probably efficient on most (if not all) point sequences.

Thus, as we mentioned before, one great interest of the NFA algorithm is that the parameter tuning step is *much* more easier than in other algorithms like ROADS, for which it can be a real burden, especially when dealing with complex data (with unknown ground truth) on which the effect of a parameter change can be very difficult to evaluate. This relative parameter sensitivity is illustrated in Fig. 24.

4.5.3 Comparison of ROADS and NFA algorithms

To compare the results obtained by the ROADS and NFA algorithms on the *snow* sequence, we use for each algorithm two different settings: the *default* setting and the *best* setting.

For ROADS, the *default* setting corresponds to $a_{\max} = +\infty$, $p_{\min} = 1$, $d_{\max} = 130$, and $\phi_{\max} = 0.58$. Note that $\phi_{\max} = 0.58$ corresponds to the oracle value, that is, the (theoretically unknown) maximum value of ϕ on the ground truth trajectories. For $d_{\max}$, we chose the value $d_{\max} = 130$ to allow ROADS to detect all the trajectories in the main bulk of trajectories (choosing $d_{\max}$ as the real maximal speed (160) would give worse results). The *best* setting for ROADS was chosen after a careful (and a bit cumbersome) parameter analysis (see Fig. 22), which leads to $a_{\max} = 0$, $p_{\min} = 1$, $\phi_{\max} = 0.6$, and $d_{\max} = 20$.

Concerning the NFA algorithms, the *default* and *best* settings simply correspond to $\log_{10}\varepsilon = 0$ and $\log_{10}\varepsilon = +5$ re-

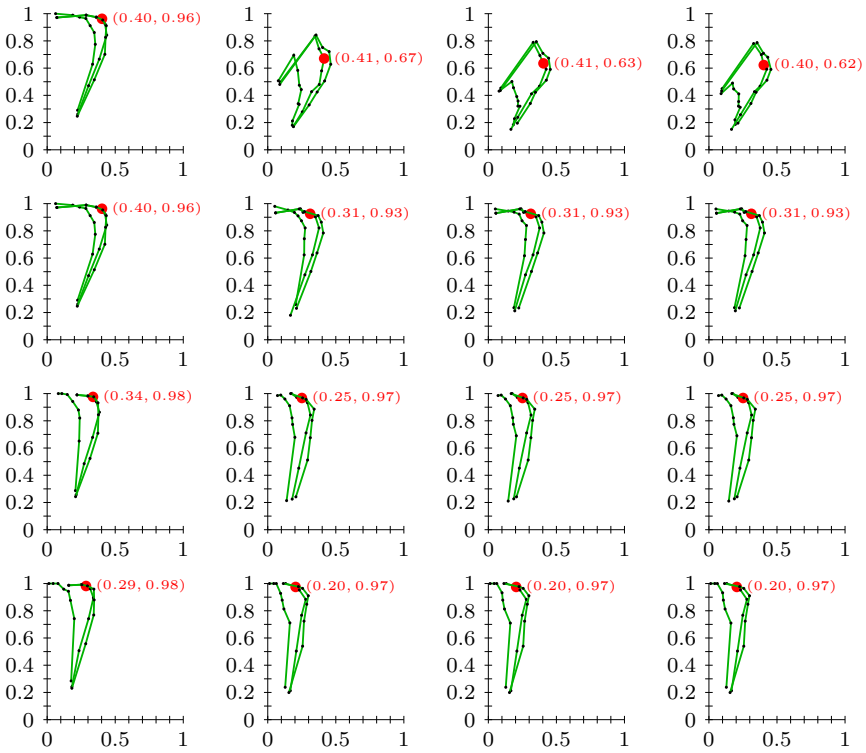


Fig. 22 ROADS parameter tuning on the snow sequence. We vary all ROADS parameters on the *snow* point sequence, and show the associated performances in the (recall,precision) plane using the available ground truth for that sequence. Each column has a distinct $a_{\max} = 0, 4, 8, +\infty$, and each row has a distinct $p_{\min} = 1, 3, 5, 7$. Each curve corresponds to a different maximal smoothness criterion value $\phi_{\max} = 0.2, 0.4, 0.6$ and each point of a given curve corresponds to a different maximal speed criterion $d_{\max} = 2, 5, 10, 15, 20, 25, 30, 40, 80$. The big red dot corresponds to the parameters $\phi_{\max} = 0.4$ and $d_{\max} = 15$, which seem to achieve a good precision/recall compromise for all values of $a_{\max}$ and $p_{\min}$. The numbers indicate the corresponding recall and precision.

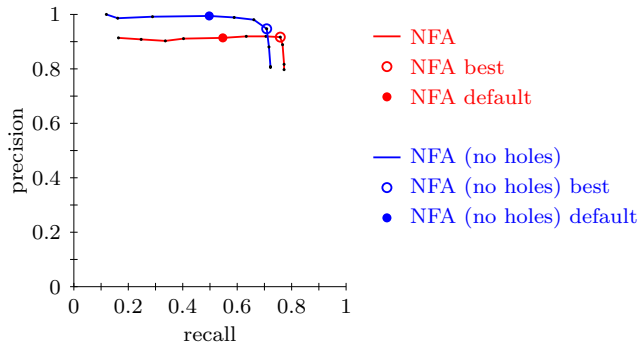


Fig. 23 NFA parameter tuning on the snow sequence. The performances of the two NFA algorithms (with holes and without holes) on the *snow* sequence are represented in the (recall,precision) plane in function of the threshold parameter $\log_{10} \varepsilon$. While the precision remains merely constant, a good recall is obtained with the default value (0) of $\log_{10} \varepsilon$, but the results can be improved by choosing a slightly greater value ($\log_{10} \varepsilon = +5$), which corresponds for the two algorithms to the "NFA best" point.

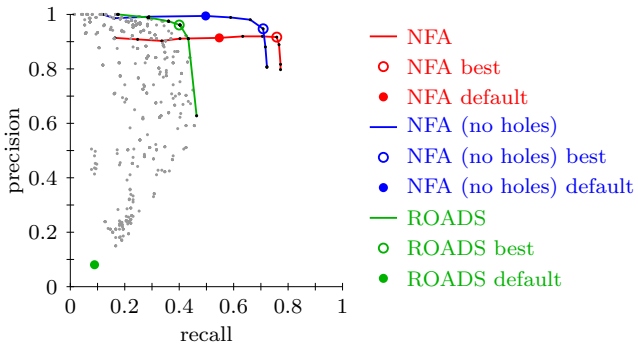


Fig. 24 Comparison of NFA and ROADS algorithms on the snow sequence. All results obtained on the *snow* sequence with the NFA (with holes) and ROADS algorithms are represented in the (recall,precision) plane, with a point for each set of parameters (thus, the optimal performance of each algorithm is the curve obtained as the upper-right enveloppe of its points). We can see not only that ROADS is much more sensitive to the parameter choice than the NFA algorithms, but also that its overall performance in terms of recall is significantly worse, even with an optimal choice of its parameters.

spectively (see Fig. 23). As for ROADS, the *best* setting was chosen a bit arbitrarily, as one of the best recall values avoiding a significant loss of precision. The strong L-shaped aspect of the precision-recall curves made that choice quite easy (in the sense that other possible choices would not differ much).

The results obtained with ROADS and NFA algorithms on the *snow* sequence (for both *default* and *best* settings) are shown in Fig. 25, and the precision-recall performances reported in Table 5. In the *best* configuration, all algorithms attain similar performances in terms of precision, but at the price of a high number of false detections (poor recall) for the ROADS algorithm. Moreover, the *default* parameter setting of ROADS gives very poor results, while the *default* parameter setting of NFA ($\varepsilon = 1$) achieves a better performance than the *best* ROAD settings. These results are analyzed in greater details in Fig. 26, where it appears that the main limiting factor of the ROADS algorithm seems to be its unability to handle simultaneously (that is, with a same set of parameters) trajectories with various lengths and speeds.

mode	**algorithm**	**recall** (fg, bg)	**precision**
default	ROADS	0.09 (0.19, 0.00)	0.08
	NFA no holes	0.50 (0.44, 0.55)	0.99
	NFA holes	0.55 (0.44, 0.64)	0.91
best	ROADS	**0.40** (0.13, 0.65)	**0.96**
	NFA no holes	**0.71** (0.74, 0.68)	**0.95**
	NFA holes	**0.76** (0.78, 0.74)	**0.92**

Table 5 Performances of ROADS and NFA **algorithms on the snow sequence.** The ROADS and NFA algorithms are run on the *snow* sequence with their *default* and *best* settings, and their performances are analyzed in terms of recall and precision. In order to permit a more accurate analysis, separate recall scores are also computed by considering separately fast foreground (fg) objects (snowflakes) and the almost static background (bg) objects (staints on the background door). As we can see, the comparison is clearly in favor of the NFA algorithm. With the *best* settings, the obtained precision is roughly the same, but the ROADS algorithm is unable to achieve an interesting detection rate on the foreground objects (the snowflakes), which results in a poor overall recall.

We end this section with a little discussion on computation times. On the *snow* sequence, the ROADS algorithms runs in about 10 seconds for the *default* parameter values, and in 0.1 second for the *best* parameter values (this does not take into account, of course, the time needed to find these best parameters values). In comparison, the NFA "no-hole" algorithm runs in 1.1 second on the same data, while the NFA "hole" algorithm takes 35 minutes. The speed ratio is large, but it should be noted that in several applications such a computation time is not a problem, because the production of the point sequence may take much more time (consider a biological experiment relying on cell tracking for example). Moreover, there is an intermediate way of speeding up the algorithm while keeping the interesting property of allowing holes (which may be crucial for some data). It consists in using the NFA with holes criterion (Equation 23) while limiting the exploration of trajectories to those that do not have holes longer than h (this requires only a very simple modification of the algorithm). In this modified algorithm, the integer parameter h has no influence on the detection thresholds, but greatly decreases the computation time by limiting the search to the most common trajectories. It may be very useful to find a tradeoff in situations where computation time matters more than full detection performances, or in a quick-analysis stage used before running the exact ($h = \infty$) NFA "hole" algorithm. On Table 6, we reported the influence of the h parameter on the computation time and detection performances (precision/recall) for the snow sequence ($\log_{10} \varepsilon = 5$).

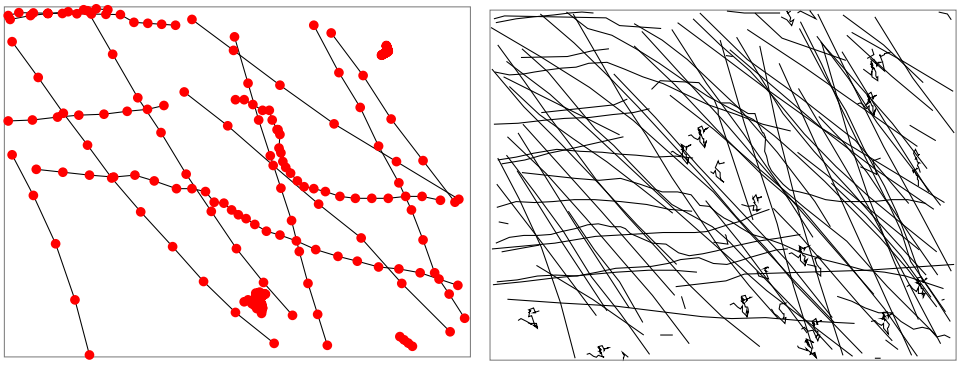
h	0	1	2	5	∞
time	1.1 s	1 min	4 min	12 min	35 min
recall	0.71	0.75	0.76	0.76	0.76
precision	0.95	0.94	0.93	0.92	0.92

Table 6 Limiting the size of holes in the NFA algorithm. This table reports the computation time and the precision/recall performances obtained after running on the *snow sequence* the NFA "hole" algorithm ($\log_{10} \varepsilon = 5$) restricted to trajectories having no hole longer than h (for $h = 0$, the "no-hole" NFA algorithm is used). As h decreases, the computation time drops quickly, while maintaining a high level of precision and recall on that sequence.

5 Conclusion

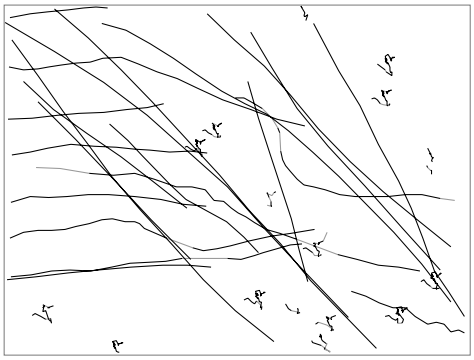
We presented two point-tracking algorithms based on the a-contrario framework, which are able to detect trajectories in point sequences without additional information. The first algorithm (NFA "no-hole") is restricted to complete trajectories (that is, without holes), while the second (NFA "hole") can recover trajectories with missing points. Both algorithm are very robust to noise, in the sense that they are designed to avoid halucinating trajectories in noise data. Another strength of these algorithms, which comes from the a-contrario approach, is that they do not require to set parameters: even the only threshold that can be set (ε, which balances between precision and recall) may be left to its default value ($\varepsilon = 1$).

When compared to the state-of-the-art ROADS algorithm, these two algorithms perform very well, both on simulated and real-world data. In particular, they show a very high level of precision (that is, a very low rate of false alarms) while maintaining a good level of recall (actual trajectories detected). The absence of required parameter setting and the robustness to trajectory variability (speed, length, acceler-

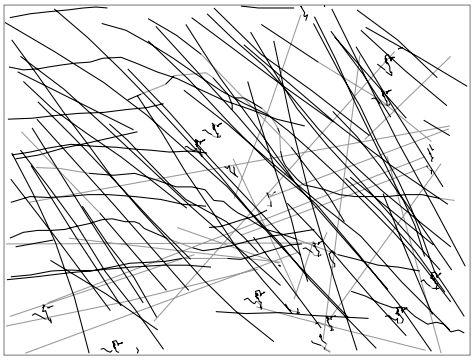


(1a) Sample trajectories from ground truth

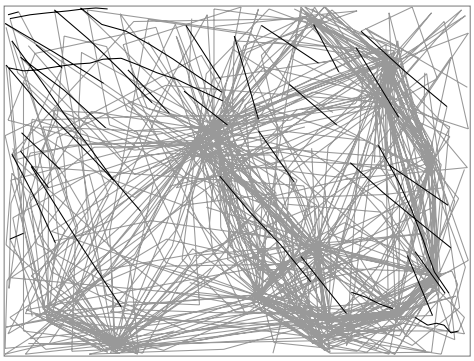
(1b) Ground truth



(2a) NFA default



(2b) NFA best



(3a) ROADS default

(3b) ROADS best

Fig. 25 Trajectories found by ROADS and NFA **algorithms on the snow sequence.** First row: trajectories from ground truth (right) and a sample of ground truth trajectories where successive points are represented as dots to give an idea of the objects speeds. Second row: trajectories found by the NFA algorithm, for the default parameter value ($\varepsilon = 1$, left) and for the optimal parameter value ($\log_{10} \varepsilon = 5$, right). The well-detected trajectory links are drawn in black, and the wrong ones in gray. Third row: trajectories found by the ROADS algorithm, for the default parameter values (left) and the optimal parameter values (right). As we can observe, the ROADS algorithm makes many false detections with the default parameter values, and very few detections with the optimal parameter values. On the contrary, the NFA algorithm with the default parameter value finds a large part of the actual trajectories and yields almost no false detection (see Fig. 23). When using the best parameter value, almost all trajectory links are found, and only a few spurious detections occur.

ation) appears to be a very useful feature when real-world point sequences have to be processed.

Compared to several classical tracking algorithm, the two NFA algorithms we proposed have the advantage that they are based on the exact minimization (and thresholding) of a simple criterion, without any heuristic or approximation. This criterion not only permits to rate the quality of each trajectory, but might also be used in a very simple way to remove false detections from to the output of another algorithm.

The "no-hole" algorithm is rather fast on standard data, but the principal limitation of the "hole" algorithm is its high computational and memory costs. However, these costs may be mitigated by using the easily parallelizable nature of the

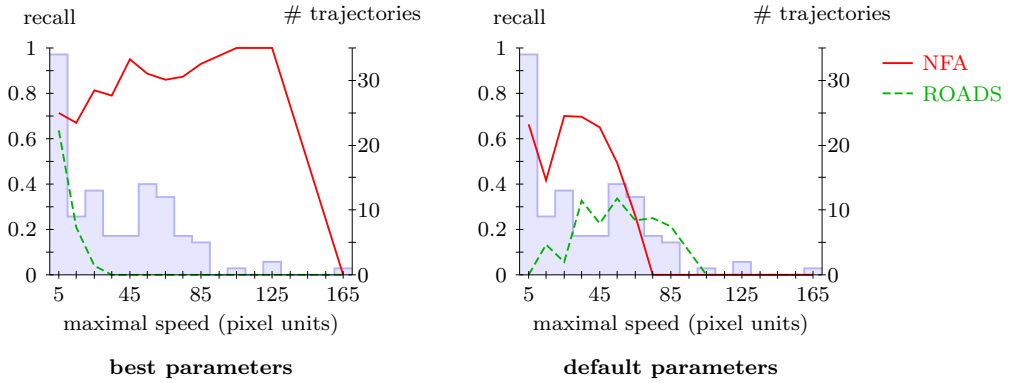


Fig. 26 Detailed analysis of ROAD and NFA recall performances. The recall performances of ROADS and NFA "hole" algorithms on the *snow* sequence (as given in Table 5) are analyzed in function of the maximal trajectory speed, for both *best* (left) and *default* parameter settings. The possible values of the maximal trajectory speed are divided into bins (horizontal axis), and the blue histogramms indicate the number of corresponding actual trajectories. Then, the recall of each algorithm is analyzed inside each bin, in red for NFA and in dashed green for ROADS. As we can see, the ROADS algorithm does not manage to handle simultaneously trajectories with various speeds: the detection is focused either on trajectories with middle-range speeds (*default* setting), or on very slow trajectories (*best* setting). The NFA algorithm, which combines the trajectory smoothness and length into a single NFA criterion (hence avoiding a speed threshold), does not suffer from this dilemma, as it clearly appears on the left (*best* setting) graph.

algorithm, or by reducing the size of the trajectory search space with additional constraints (for example, a bound on the maximum number of consecutive missing points).

We chose in this paper to use a trajectory smoothness criterion based on the maximum acceleration, but similar approaches could be developped as well for other local criteria (speed, angular acceleration, etc.) and for other global costs (a sum cost instead of a max cost for example). The presentation was made in a two-dimensional setting, but the generalization to higher dimensions (3D points, or more) is straightforward. Note also that the proposed framework could probably be extended to the case when points come with features (intensity, shape, etc.), in the same spirit as Noury et al (2010) extended the framework of Moisan and Stival (2004) for Fundamental Matrix Estimation.

Acknowledgments

We are very grateful to C.J. Veenman for providing us with a C implementation of the ROADS algorithm.

Available resources

The *snow* point sequence, as well as open-source codes for the proposed algorithms are freely available on the ASTRE (A-contrario Smooth TRajectory Extraction) website

http://www.mi.parisdescartes.fr/~moisan/astre/

It contains a simple Python implementation, an optimized one in C language working under Linux Operating System, and a detailed installation and usage manual.

References

Ali A, Aggarwal JK (2001) Segmentation and recognition of continuous human activity. In: *IEEE Workshop on Detection and Recognition of Events in Video*, pp 28–35

Ashdown M, Oka K, Sato Y (2005) Combining head tracking and mouse input for a GUI on multiple monitors. In: *Proceedings of the 2005 Conf. on Human Factors in Computing Systems (CHI)*, pp 1188–1191

Avidan S (2001) Support vector tracking. In: *Proceedings of the Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp 184–191

Bar-Shalom Y, Fortmann T, Scheffe M (1983) Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering* 8(3):173–184

Bellman R (1954) The theory of dynamic programming. *Bulletin of the American Mathematical Society* 60:503–516

Berclaz J, Fleuret F, Fua P (2009) Multiple Object Tracking using Flow Linear Programming. In: *12th IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance (Winter-PETS)*, pp 1–8

Berclaz J, Turetken E, Fleuret F, Fua P (2011) Multiple object tracking using k-shortest paths optimization. *IEEE Trans on Pattern Analysis and Machine Intelligence* 33(9):1806–1819

Bertalmío M, Sapiro G, Randall G (2000) Morphing active contours. *IEEE Trans on Pattern Analysis and Machine Intelligence* 22(7):733–737

Black MJ, Jepson AD (1998) EigenTracking: Robust Matching and Tracking of Articulated Objects Using a

View-Based Representation. *Int Journal of Computer Vision* 26(1):63–84

Blake A, Isard M (1998) *Active Contours: The Application of Techniques from Graphics,Vision,Control Theory and Statistics to Visual Tracking of Shapes in Motion*, 1st edn. Springer-Verlag New York, Inc., Secaucus, NJ, USA

Chetverikov D, Verestoy J (1999) Feature point tracking for incomplete trajectories. *Computing* 62:321–338

Comaniciu D, Ramesh V, Meer P (2003) Kernel-based object tracking. *IEEE Trans on Pattern Analysis and Machine Intelligence* 25(5):564–575

Cornelissen T, Elsing M, Gavrilenko I, Liebig W, Moyse E, Salzburger A (2008) The new atlas track reconstruction (newt). *Journal of Physics: Conference Series* 119(3)

Desolneux A, Moisan L, Morel JM (2000) Meaningful alignments. *Int Journal of Computer Vision* 40(1):7–23

Desolneux A, Moisan L, Morel JM (2001) Edge detection by Helmholtz principle. *Journal of Mathematical Imaging and Vision* 14(3):271–284

Desolneux A, Moisan L, Morel JM (2008) *From Gestalt Theory to Image Analysis: A Probabilistic Approach* (Interdisciplinary Applied Mathematics). Springer

Edwards GJ, Taylor CJ, Cootes TF (1998) Interpreting face images using active appearance models. In: *Procedings of the Int. Conference on Face & Gesture Recognition (FG)*, pp 300–305

Fieguth PW, Terzopoulos D (1997) Color-Based Tracking of Heads and Other Mobile Objects at Video Frame Rates. In: *Proceedings of the Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp 21–27

Fleuret F, Berclaz J, Lengagne R, Fua P (2008) Multi-Camera People Tracking with a Probabilistic Occupancy Map. *IEEE Trans on Pattern Analysis and Machine Intelligence* 30(2):267–282

Grosjean B, Moisan L (2009) A-contrario detectability of spots in textured backgrounds. *Journal of Mathematical Imaging and Vision* 33(3):313–337

Gui L, Merzkirch W (1996) A method of tracking ensembles of particle images. *Experiments in Fluids* 21(6):465–468

Huttenlocher D, Noh J, Rucklidge W (1993) Tracking non-rigid objects in complex scenes. In: *Proceedings of the Int. Conf. on Computer Vision (ICCV)*, pp 93–101

Jiang H, Fels S, Little JJ (2007) A linear programming approach for multiple object tracking. In: *Proceedings of the Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp 1–8

Kang J, Cohen I, Medioni GG (2004) Object reacquisition using invariant appearance model. In: *Proceedings of the Int. Conf. on Pattern Recognition*, vol 4, pp 759–762

Kanizsa G (1980) *Grammatica del vedere*. Il Mulino, Bologna

Koller D, Weber J, Malik J (1994) Robust multiple car tracking with occlusion reasoning. In: Eklundh JO (ed) *Proceedings of the European Conf. on Computer Vision (ECCV)*, pp 189–196

Moisan L, Stival B (2004) A probabilitic criterion to detect rigid point matches between two images and estimate the fundamental matrix. *Int Journal of Computer Vision* 57(3):201–218

Munkres J (1957) Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics* 5(1):32–38

Murty KG (1968) An algorithm for ranking all the assignments in order of increasing cost. *Operations Research* 16(3):682–687

Noury N, Sur F, Berger MO (2010) Determining point correspondences between two views under geometric constraint and photometric consistency, INRIA report RR-7246

Paragios N, Deriche R (2000) Geodesic Active Contours and Level Sets for the Detection and Tracking of Moving Objects. *IEEE Trans on Pattern Analysis and Machine Intelligence* 22(3):266–280

Pighin FH, Szeliski R, Salesin D (1999) Resynthesizing facial animation through 3d model-based tracking. In: *Proceedings of the Int. Conf. on Computer Vision (ICCV)*, pp 143–150

Rangarajan K, Shah M (1991) Establishing motion correspondence. In: *Proceedings of the Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp 103–108

Reid D (1979) An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control* 24(6):843–854

Robin A, Moisan L, Hégarat-Mascle SL (2010) An A-Contrario Approach for Subpixel Change Detection in Satellite Imagery. *IEEE Trans on Pattern Analysis and Machine Intelligence* 32(11):1977–1993

Ronfard R (1994) Region-based strategies for active contour models. *Int Journal of Computer Vision* 13(2):229–251

Salari V, Sethi I (1990) Feature point correspondence in the presence of occlusion. *IEEE Trans on Pattern Analysis and Machine Intelligence* 12(1):87–91

Sato K, Aggarwal JK (2004) Temporal spatio-velocity transform and its application to tracking and interaction. *Computer Vision and Image Understanding* 96(2):100–128

Sbalzarini I, Koumoutsakos P (2005) Feature point tracking and trajectory analysis for video imaging in cell biology. *Journal of Structural Biology* 151(2):182–95

Serby D, Koller-Meier E, Gool LJV (2004) Probabilistic object tracking using multiple features. In: *Proceedings of the Int. Conf. on Pattern Recognition*, vol 2, pp 184–187

Sethi IK, Jain R (1987) Finding trajectories of feature points in a monocular image sequence. *IEEE Trans on Pattern Analysis and Machine Intelligence* 9(1):56–73

Shafique K, Shah M (2003) A non-iterative greedy algorithm for multi-frame point correspondence. In: *Proceedings of the Int. Conf. on Computer Vision (ICCV)*, pp 110–

115

Smal I, Niessen WJ, Meijering EHW (2008) A new detection scheme for multiple object tracking in fluorescence microscopy by joint probabilistic data association filtering. In: *Proceedings of the Int. Symp. on Biomedical Imaging: From Nano to Macro*, pp 264–267

Streit RL, Luginbuhl TE (1994) Maximum likelihood method for probabilistic multihypothesis tracking. In: Drummond OE (ed) *Signal and Data Processing of Small Targets 1994*, vol 2235, pp 394–405

Tao H, Sawhney HS, Kumar R (2002) Object Tracking with Bayesian Estimation of Dynamic Layer Representations. *IEEE Trans on Pattern Analysis and Machine Intelligence* 24(1):75–89

Veenman CJ, Reinders MJT, Backer E (2001) Resolving motion correspondence for densely moving points. *IEEE Trans on Pattern Analysis and Machine Intelligence* 23(1):54–72

Veenman CJ, Reinders MJT, Backer E (2003a) Establishing motion correspondence using extended temporal scope. *Artificial Intelligence* 145(1-2):227–243

Veenman CJ, Reinders MJT, Backer E (2003b) Motion tracking as a constrained optimization problem. *Pattern Recognition* 36(9):2049–2067

Veit T, Cao F, Bouthemy P (2006) An a contrario decision framework for region-based motion detection. *Int Journal of Computer Vision* 68(2):163–178

Veit T, Cao F, Bouthemy P (2007) Space-time a contrario clustering for detecting coherent motion. In: *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp 33–39

Wertheimer M (1922) Untersuchungen zur lehre von der gestalt. *Psychologische Forschung* 1(1):47–58

Yilmaz A, Li X, Shah M (2004) Contour-Based Object Tracking with Occlusion Handling in Video Acquired Using Mobile Cameras. *IEEE Trans on Pattern Analysis and Machine Intelligence* 26(11):1531–1536

Yilmaz A, Javed O, Shah M (2006) Object tracking: A survey. *ACM Comput Surv* 38(4)

Zhu SC, Yuille AL (1996) Region Competition: Unifying Snakes, Region Growing, and Bayes/MDL for Multiband Image Segmentation. *IEEE Trans on Pattern Analysis and Machine Intelligence* 18(9):884–900