Accelerated A-contrario Detection of Smooth Trajectories

Rémy Abergel, Lionel Moisan

ACCELERATED A-CONTRARIO DETECTION OF SMOOTH TRAJECTORIES

Rémy Abergel and Lionel Moisan

Université Paris Descartes, MAP5 (CNRS UMR 8145), France

ABSTRACT

The detection of smooth trajectories in a (noisy) point set sequence can be realized optimally with the ASTRE (A-contrario Smooth TRajectory Extraction) algorithm, but the quadratic time and memory complexity of this algorithm with respect to the number of frames is prohibitive for many practical applications. We here propose a variant that cuts the input sequence into overlapping temporal chunks that are processed in a sequential (but non-independent) way, which results in a linear complexity with respect to the number of frames. Surprisingly, the performances are not affected by this acceleration strategy, and are in general even slightly above those of the original ASTRE algorithm.

***Index Terms*—** trajectory analysis; point tracking; motion detection; a-contrario model.

1. INTRODUCTION

Many image processing tasks that concern video or image sequences are related to various forms of motion analysis like optical flow, object tracking, trajectory detection, motion compensation, etc. In the present work, we consider the fundamental problem of finding reliable trajectories in a point set sequence that has been previously extracted from an image sequence, without any attribute attached to each point.

This task, which has been considered several times in the literature [1–5], is the core of various applications including, e.g., particle velocimetry in fluid mechanics, dynamic analysis of fluorescent probes in biology, study of ant or termite behavior, pedestrian and car tracking, etc. For the case where smooth trajectories (more precisely, trajectories having a small maximum acceleration) are to be detected among a potentially high number of incoherent noise points, an algorithm with optimality guarantees, called ASTRE [6], has been recently built, but it turns out that it is nearly impossible to use it on long image sequences (say $K \geq 1000$ frames) because its time and memory complexity is quadratic with respect to K. We here propose to break this complexity limitation and describe a new algorithm for which the complexity is linear with respect to K, which substantially increases the possibilities of real-world applications. In Section 2, we describe the original ASTRE algorithm, and show that the introduction of a maximum speed threshold may bring an important speed-up, but does not break the $\mathscr{O}(K^2)$ complexity. This is why in Section 3 we present a new algorithm named CUTASTRE, which cuts the original image sequence into overlapping small temporal chunks and processes these chunks sequentially with the ASTRE algorithm, using an incremental strategy to detect long trajectories. This $\mathscr{O}(K)$ algorithm, though theoretically suboptimal in terms of detection performances, still offers (like ASTRE) a rigorous control of false detections in pure noise data. The new parameters (chunk size and overlapping ratio) are analyzed in Section 4, and appear to be rather easy to set. Moreover, numerical experiments on both synthetic and natural point set sequences reveal that the detection performances of CUTASTRE are very similar to those of ASTRE, which, considering the dramatic speed-up offered by CUTASTRE, opens very interesting perspectives.

2. THE FORMER ASTRE ALGORITHM

2.1. ASTRE methodology

The ASTRE algorithm is designed to perform trajectory detection over a sequence of K frames (with domain Ω) each containing N points, applying the a-contrario methodology [7]. The domain Ω can be continuous or discrete, but here we simply recall results in the continuous setting, when Ω is the square $[0,1] \times [0,1]$.

General a-contrario algorithms are based on two main ingredients: a naive model (called $\mathcal{H}_0$) describing what could be pure noise data, and a measurement function that characterizes the kind of structures looked for. In [6], the naive model $\mathcal{H}_0$ is a uniform draw of N points in each of the K frames, and the measurement function associated to a random trajectory $T = (X_{i_1}^{k_0}, X_{i_2}^{k_0+1}, \ldots, X_{i_\ell}^{k_0+\ell-1})$ with length ℓ is its acceleration

$$a(T) = \max_{p=3,\ldots,\ell} \left\| X_{i_p}^{k_0+p-1} - 2X_{i_{p-1}}^{k_0+p-2} + X_{i_{p-2}}^{k_0+p-3} \right\|,$$

where X_i^k is the i-th point of frame k. We shall denote the random trajectory T by $X_{i_1}^{k_0} \to X_{i_2}^{k_0+1} \to \cdots \to X_{i_\ell}^{k_0+\ell-1}$, and a link between two successive points X_i^k and X_j^{k+1} by $X_i^k \to X_j^{k+1}$. Similarly, the local discrete acceleration will be written $a(u,v,w) = a(u \to v \to w) = \|w - 2v + u\|$. The amount of surprise when observing an actual trajectory t with length ℓ and acceleration $\delta := a(t)$ can be estimated by

using a simple (but precise) upper bound of the probability of observing a trajectory with acceleration smaller than δ in $\mathcal{H}_0$,

$$\mathbb{P}_{\mathcal{H}_0}(a(T) \leq \delta) \leq \left(\pi\delta^2\right)^{\ell-2}. \quad (1)$$

Using the a-contrario methodology (see prop. 2 in [8]), one can design a *number of false alarms* by

$$\forall \ell,\ \forall t \in \mathbb{T}_\ell, \quad \mathrm{NFA}_t(\delta) = K(K-\ell+1)N^\ell \left(\pi\delta^2\right)^{\ell-2} \quad (2)$$

($\mathbb{T}_\ell$ denotes the set of trajectories of length ℓ) for the measurement $\delta = a(t)$, that is, a function that satisfies

$$\forall \varepsilon > 0, \quad \mathbb{E}_{\mathcal{H}_0}\left[\#\left\{T \;\middle|\; \mathrm{NFA}_T\left(a(T)\right) \leq \varepsilon\right\}\right] \leq \varepsilon. \quad (3)$$

Trajectories having a NFA smaller than ε are said to be ε-meaningful (or detected at level ε) and the so-called NFA-property (3) ensures that the average number of (false) detections made in $\mathcal{H}_0$ at level ε is less than ε.

The ASTRE algorithm has ε for unique parameter; it represents the maximal NFA value of a trajectory that the user wants to extract (usually one chooses $\varepsilon = 1$). In practice the extraction scheme is greedy: if m, the minimal NFA among all possible trajectories is less than ε, a trajectory having NFA equal to m is extracted, its points are removed from the sequence and the process is repeated until no trajectory with NFA less than ε can be found any more.

To compute the smallest NFA among all possible trajectories, a dynamic programming strategy is used. The problem boils down to compute, for any x, y, ℓ, the minimal acceleration $\mathscr{G}(x, y, \ell)$ of a trajectory $t \in \mathbb{T}_\ell$ ending with link $y \to x$ (see (9) in [6]). We refer the reader to [6] for a complete description of ASTRE, and in particular how to smoothly extend the method when the number of points is non-constant over the frame sequence, how to handle properly data quantization (when the domain Ω is discrete). Notice also that ASTRE can be extended to handle trajectory with missing points (that is, "with holes"), but this case will not be considered in the present work.

2.2. Improvement of the execution time

The main weakness of ASTRE is its quadratic time and memory complexity with respect to K, due to the extensive computation of $\mathscr{G}$. A very simple way to reduce the execution time of this algorithm is to introduce a threshold $\mathcal{S}_{\text{thre}}$ on the speed of the trajectories: as soon as the distance between two points x and y of two consecutive frames is higher than $\mathcal{S}_{\text{thre}}$, we consider that link $y \to x$ cannot exist. Hence we can avoid the computation of $\mathscr{G}(x, y, \ell)$ for those pairs of points.

A speed threshold is a physical parameter that can be easily adjusted in many applications. The use of this additional knowledge restricts the number of linking possibilities among the sequence, and reduces very significantly the execution time in general. However the complexity remains $\mathscr{O}(K^2)$ and ASTRE is still inapplicable to long data sequences, as can be seen in the ASTRE columns of Table 1.

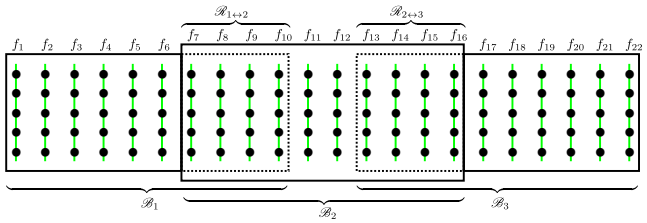

Fig. 1. Example of grouping for a twenty-two frames sequence (vertical segments) containing five points each (black disks). Here frames are grouped into three chunks of ten frames each, with four frames overlap.

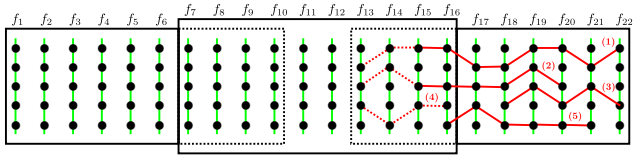

Fig. 2. Extraction of meaningful trajectories within chunk $\mathscr{B}_3$. Trajectories are being detected using ASTRE, taking $k_3 = 10$ instead of K for the NFA computation. Each time a trajectory is extracted, all its points are removed from the sequence. Once all trajectories are extracted, we set back to the sequence all points of overlapping frames (frames f_{13} to f_{16}). The corresponding links are removed (dotted links) excepting those linking a point of frame f_{15} with a point of frame f_{16}. Trajectory (4) being entirely included into $\mathscr{R}_{2\leftrightarrow 3}$, all its links are removed.

3. CUTASTRE

Our approach consists in grouping consecutive frames of the full sequence into overlapping chunks $\mathscr{B}_1, \ldots, \mathscr{B}_n$ (an example of grouping is proposed in Fig. 1). Trajectories will be detected within each chunk (starting from chunk $\mathscr{B}_n$) with an algorithm similar to ASTRE, which will be adapted to extend trajectories extracted from a chunk $\mathscr{B}_k$ when processing its predecessor $\mathscr{B}_{k-1}$. The function $\mathscr{G}$ will be computed only within a single chunk, allowing a drastic reduction of the time and memory complexity. In the following we will denote by k_i the number of frames contained in the chunk $\mathscr{B}_i$.

3.1. Description of the process

The first chunk to be processed is $\mathscr{B}_n$. For this very first chunk we simply apply the ASTRE algorithm on the corresponding subsequence of frames, and replace K by k_n in (2). When ASTRE terminates, we put back in the sequence all points that have been removed from the overlapping block of frames $\mathscr{R}_{n-1\leftrightarrow n}$. We remove the corresponding links except those linking points of the two last frames of $\mathscr{R}_{n-1\leftrightarrow n}$. Finally, when a trajectory is entirely included into $\mathscr{R}_{n-1\leftrightarrow n}$, all its links are removed (this is the case of trajectory (4) in Fig. 2). This ends the process for chunk $\mathscr{B}_n$.

Let us now describe the process for the other chunks $(\mathscr{B}_i)_{1\leq i<n}$. We say that a trajectory t is extendable to $\mathscr{B}_i$ if t has been extracted while processing chunk $\mathscr{B}_{i+1}$ (not $\mathscr{B}_i$)

and reaches the two last frames (that we denote $\mathscr{F}_0^i$ and $\mathscr{F}_1^i$) of $\mathscr{R}_{i\leftrightarrow i+1}$ (see Fig. 2, trajectories (1) and (2) are extendable to $\mathscr{B}_2$, as both reach frames $\mathscr{F}_0^2 = f_{15}$ and $\mathscr{F}_1^2 = f_{16}$). To process these chunks, we need to adapt the computation of $\mathscr{G}(x, y, \ell)$ and NFA $(\mathscr{G}(x, y, \ell))$ when x or y belongs to an extendable trajectory. Let us say we focus on chunk $\mathscr{B}_i$, four cases must be distinguished:

i. neither x nor y belongs to a trajectory extendable to $\mathscr{B}_i$,

ii. $x \in \mathscr{F}_1^i$ and x belongs to a trajectory t extendable to $\mathscr{B}_i$ that also contains y (necessarily $y \in \mathscr{F}_0^i$),

iii. $x \in \mathscr{F}_0^i$ and x belongs to a trajectory t extendable to $\mathscr{B}_i$,

iv. any other case.

In case i, K is replaced by k_i in (2), and $\mathscr{G}(x, y, \ell)$ and NFA$(\mathscr{G}(x,\ y, \ell))$ are computed exactly as in [6]. In cases ii and iii, K is replaced by $k_i + k_{i+1}$. Let us consider the sub-trajectory t_0 of t that is obtained by removing from t all points that do not belong to chunk $\mathscr{B}_{i+1}$, and let a_0 denotes the acceleration of the (sub-)trajectory t_0 and ℓ_0 its length. In case ii, t_0 starts with link $y \to x$, so we replace $\mathscr{G}(x, y, \ell)$ by $\max(a_0, \mathscr{G}(x, y, \ell))$ and replace ℓ by $\ell + \ell_0 - 2$ in the NFA formula when computing NFA $(\mathscr{G}(x, y, \ell))$. In case iii, t_0 starts with a link $x \to w$ (x belongs to frame $\mathscr{F}_0^i$ and w to frame $\mathscr{F}_1^i$) so we replace $\mathscr{G}(x, y, \ell)$ by $\max(a_0, a(y, x, w), \mathscr{G}(x, y, \ell))$ and replace ℓ by $\ell + \ell_0 - 1$ in the NFA formula when computing NFA $(\mathscr{G}(x, y, \ell))$. Last, in case iv we simply set $\mathscr{G}(x, y, \ell) = +\infty$ and NFA$(\mathscr{G}(x, y, \ell)) = +\infty$ in order to avoid the detection of a trajectory ending with link $y \to x$ (or equivalently, we just do not compute $\mathscr{G}(x, y, \ell)$ for such pairs of points). Moreover, all the NFA values considered in the four cases are multiplied by the number of chunks n.

The strategy concerning trajectories extraction is exactly the same that in [6]; when all ε-meaningful trajectories are extracted, we set back again to the sequence all points belonging to $\mathscr{R}_{i-1\leftrightarrow i}$ and unless $\mathscr{B}_i$ is chunk $\mathscr{B}_1$, we repeat the link suppression process (see Fig. 3).

We would like to emphasize the ability of this algorithm to not necessarily redraw removed links when a trajectory extension is done (look carefully at trajectory (4)), hence avoiding edge effects that would fatally occur if the overlapping areas contained only two frames. These areas can be seen as decision areas and removed links as hypotheses that can be validated or not when trajectories are being extended.

3.2. Preservation of the NFA property

A natural question arises: do we still control the number of false detections by (3) like the ASTRE algorithm? The answer is yes, and it simply comes from the fact that the number of new trajectories extracted in a given chunk is controlled by ε/n, thanks to the multiplication of the individual NFA by the number of chunks n in the four cases considered above.

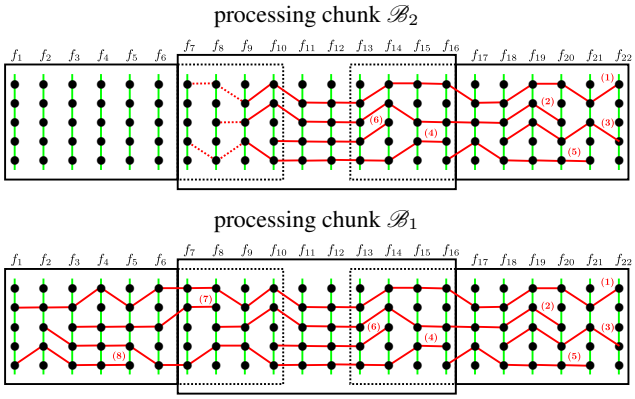

Fig. 3. **Top**: extraction of meaningful trajectories within chunk $\mathscr{B}_2$ with several links suppression within $\mathscr{R}_{1\leftrightarrow 2}$. Trajectories (1) and (2) have been extended, trajectory (4) is detected again but is now longer. **Bottom**: extraction of meaningful trajectories within chunk $\mathscr{B}_1$, trajectories (1), (2) and (4) have been extended, no link suppression must be done, as trajectories will not be extended anymore.

4. EXPERIMENTS

We first compare the ASTRE and CUTASTRE algorithms on synthetic sequences produced by the Point-Set Motion Generation (PSMG) algorithm described in [9] (see also [6], 4.2):

- The initial position of a trajectory is chosen uniformly on the (continuous) image domain Ω;
- The initial velocity magnitude is $\nu_0 \sim \alpha|Z|$, where $Z \sim \mathscr{N}(\mu = 5, \sigma = 0.5)$ is a Gaussian random variable and α a scale factor whose setting will be detailed later. The initial velocity angle β_o is uniformly chosen on $[0, 2\pi]$.
- The velocity magnitude and angle are updated in each frame using $\begin{cases} \nu_{k+1} = |Z|, \quad Z \sim \mathscr{N}(\mu = \nu_k, \alpha \cdot \sigma_\nu) \\ \beta_{k+1} \sim \mathscr{N}(\mu = \beta_k, \sigma_\beta). \end{cases}$
- The generation ends when the trajectory reaches the last time index or when it goes outside Ω. Once the trajectory is generated, its points are quantized on a discrete grid.

In all our experiments we set $\sigma_\beta = 0.2$, $\sigma_\nu = 0.2$ or 0.5, and the frame domain Ω is quantized in 1000×1000 pixels. The setting of the other parameters (length K of the sequence, length ℓ of the trajectories) will be signaled in each experiment. The scale factor α is equal to $(\frac{\#\Omega}{100\times 100})^{1/2}$ (that is, $\alpha = 10$ in our experiments), which allows to change the domain quantization while maintaining the acceleration and speed characteristics of the trajectories. Last, when a trajectory does not cover the whole sequence (that is $\ell < K$), its starting frame index is chosen uniformly among $\{1, \ldots, K - \ell + 1\}$.

The detection performances are evaluated using the F_1-score criterion defined by

$$F_1\text{-score} = 2 \cdot \frac{\text{recall} \cdot \text{precision}}{\text{recall} + \text{precision}}, \tag{4}$$

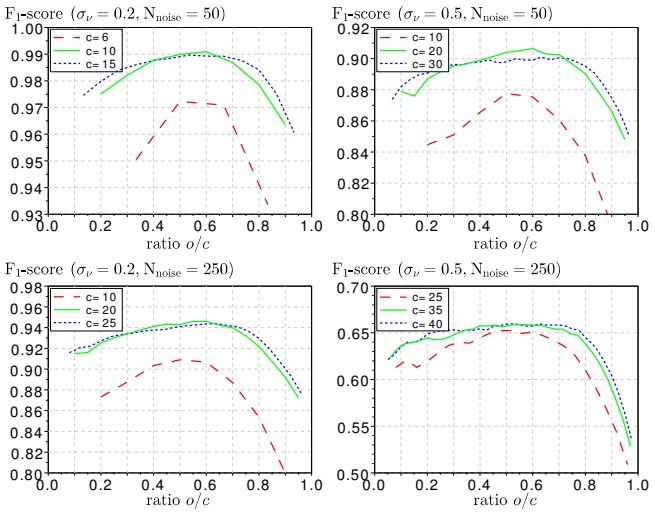

Fig. 4. Setting chunk size (c) and overlap size (o). We compute (over 50 realizations) the average F_1-score obtained with CUTASTRE on synthetic sequences ($\sigma_\nu = 0.2$ or 0.5) of $K = 90$ frames, each containing 20 trajectories with length $\ell \geq 45$ and $N_{noise} \in \{50, 250\}$ spurious points (uniformly drawn) per frame. Several chunk sizes (c) are tested for all possible overlap sizes ($2 \leq o \leq c-1$). We display the F_1-score as a function of the ratio o/c. As could be expected, the optimal chunk size c_{opt} increases with σ_ν and N_{noise} (the algorithm needs bigger chunks to catch the temporal coherence of the motion), but surprisingly enough it remains quite small compared to K. The performance is stable according to the choice of c as soon as c is not chosen too small. Conversely, once c is set, taking $o = \frac{c}{2}$ seems to be the optimal (or at least a reasonable) choice for the overlap size.

where $1 -$ precision measures the proportion of false positive links among found links, and $1 -$ recall measures the proportion of false negative links among actual links, that is,

$$\text{precision} = \frac{\#\text{of correct links found}}{\#\text{of links found}}, \quad \text{recall} = \frac{\#\text{of correct links found}}{\#\text{of actual links}}.$$

Unless explicitly signaled, we systematically take $\varepsilon = 1$ for both algorithms.

4.1. Setting the chunk-size and overlap-size parameters

The ASTRE algorithm has the NFA threshold ε as unique parameter, which is remarkably easy to set (ε is a simple bound on the average number of detections that would be done in pure noise data, usually one chooses $\varepsilon = 1$). The CUTASTRE algorithm introduces two new parameters, which are the chunk and overlap sizes. Fortunately, the setting of these parameters appears to be quite simple, according to the experiments performed on synthetic and real-life data (see Fig. 4 and 6-left). Indeed, a standard setup like $c = 30$ (or more) and $o = c/2$ seems to lead to near-optimal performances in most situations.

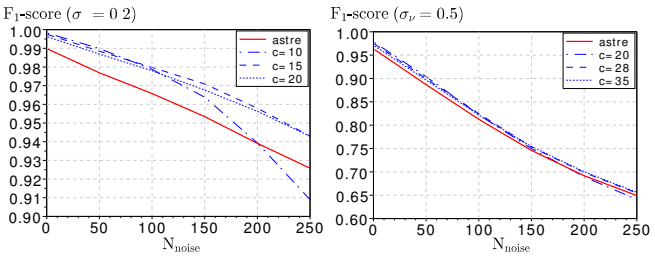

Fig. 5. Comparison of ASTRE and CUTASTRE F_1-scores (same data sequences as in Fig. 4). CUTASTRE is tested for several (near optimal) chunk sizes and the overlap size is set to $c/2$ (integer part). We observe that better performances can be reached with CUTASTRE especially for data sequences with $\sigma_\nu = 0.2$ (smooth trajectories). When working with $\sigma_\nu = 0.5$ (trajectories with high accelerations) CUTASTRE remains slightly better but performances are very close.

4.2. Performances ASTRE versus CUTASTRE

We evaluated both algorithms on synthetic data sequences (with different characteristics, see Fig. 5) but also on a real one (see Fig. 6). It turns out from our experiments that ASTRE and CUTASTRE lead to similar performances when dealing with highly accelerated trajectories ($\sigma_\nu = 0.5$) and a high level of noise. Conversely, CUTASTRE achieved better detection on the smooth synthetic data set ($\sigma_\nu = 0.2$) and the snow sequence (when $\varepsilon = 1$). This result is surprising as the globality in time of ASTRE should intuitively lead to better performances. It would be interesting to further study the mechanisms of this unexpected behavior.

4.3. Time and space complexity

The introduction of a speed threshold discussed in section 2.2 can be applied to both algorithms; it decreases the execution time, but does not change the time and memory complexities, which are respectively $\mathcal{O}(N^3K^2)$ and $\mathcal{O}(N^2K^2)$ for ASTRE, and respectively $\mathcal{O}(N^3K)$ and $\mathcal{O}(N^2K)$ for CUTASTRE. Examples of practical execution time are given in Table 1.

4.4. Tuning the threshold parameter ε

In general, the *numbers of false alarms* of a-contrario algorithms are built using a probability upper bound (like (1)) that is not necessarily sharp. Furthermore, in the case of ASTRE, trajectories are greedily extracted, thus the number of trajectories extracted at level ε in any data sequence is always less than the number of ε-meaningful trajectories. As a consequence, ε is in practice a pessimistic estimation of the number of detections that really occur in pure noise data, and the user can usually obtain better detection results by increasing the NFA-threshold parameter ε, as illustrated in Fig. 6 and 7.

	K	no speed threshold		$\mathcal{S}_{\text{thre}} = 150$	
		ASTRE	CUTASTRE	ASTRE	CUTASTRE
$N_{\text{noise}} = 10$	200	30	1.4	1.2	0.09
	500	270	3.6	11	0.26
	1000	2160	7.6	80	0.51
	3000	-	24.8	1230	1.64
	5000	-	39.7	-	2.76
$N_{\text{noise}} = 50$	200	718	27	18	0.91
	500	10^4	88	226	2.88
	1000	-	158	1686	5.13
	3000	-	444	-	15.20
	5000	-	743	-	24.87

Table 1. **Comparison of typical execution times** on synthetic sequences ($\sigma_\nu = 0.2$) with various values of the number of frames K. Each sequence contains $K/10$ trajectories with length $\ell \in [100, 200]$ and N_{noise} spurious points per frame. We compare the execution time (in seconds) of ASTRE and CUTASTRE algorithms, with and without speed threshold (we took $\mathcal{S}_{\text{thre}} = 150$, which was three times the typical maximal speed that we could observe in the data). This experiment shows that the use of a speed threshold (even pessimistic) reduces significantly the execution time (for both algorithms), but does not break the $\mathcal{O}(K^2)$ complexity of ASTRE, which is prohibitive for long data sequences. With CUTASTRE, the execution time increases linearly with the number of frames.

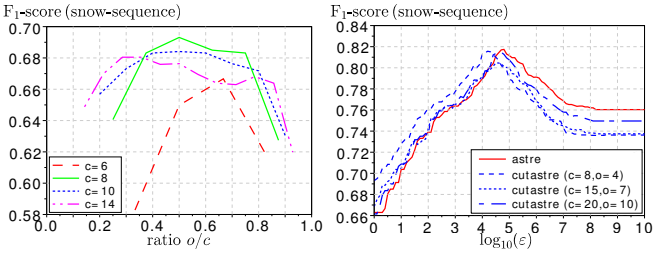

Fig. 6. **Performances evaluation on a real sequence.** We evaluated the algorithms on the *snow sequence* described in [6] (available online at http://www.mi.parisdescartes.fr/∼moisan/astre/). On the left, we reproduce the parameter exploration of Fig. 4. We find $c_{\text{opt}} = 8$ and the performance is stable for $c \geq c_{\text{opt}}$. Also, the setting $o = c/2$ remains a good choice (often the best) once the parameter c is set. On the right, we display the evolution of the F_1-score with respect to $\log_{10}(\varepsilon)$. We can see, as in Fig. 5, that the two algorithms achieve similar performances (actually slightly better for CUTASTRE when the parameters c and o are optimally set).

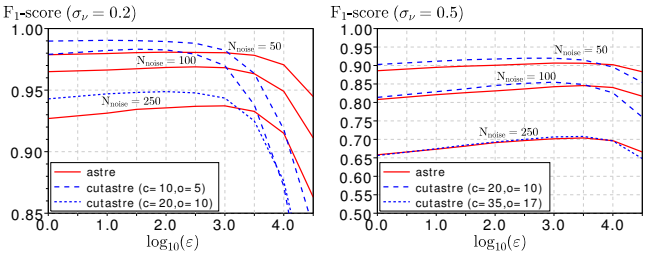

Fig. 7. **Influence of the threshold parameter** ε (same data sequences as in Fig. 4-5). We display the (average) F_1-score as a function of ε. Algorithm CUTASTRE is used with a near-optimal setting of parameters c and o (which revealed to be robust to ε changes). For both algorithms, the F_1-score increases with ε up to a global maximum, then it falls down. We observe as in Fig. 5 that the performances of CUTASTRE are similar to those of ASTRE (and even slightly better for low accelerations and low noise levels).

5. CONCLUSION AND PERSPECTIVES

We proposed a new algorithm that manages to break the quadratic $\mathcal{O}(K^2)$ time and memory complexity of ASTRE, while showing similar (or slightly higher) detection performances and a good ease-of-use since the two algorithmic parameters involved in the time-cut strategy of CUTASTRE are easy to set. This fast variant could be extended to handle missing points (trajectories "with holes"), as this functionality is already available with ASTRE but is still too computationally expensive for many applications.

REFERENCES

[1] C.J. Veenman, M.J. T. Reinders, and E. Backer, "Resolving motion correspondence for densely moving points," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 1, pp. 54–72, 2001.

[2] K. Shafique and M. Shah, "A non-iterative greedy algorithm for multi-frame point correspondence," in *Int. Conf. Comp. Vision*, 2003, pp. 110–115.

[3] Z. Khan, T. Balch, and F. Dellaert, "MCMC-based particle filtering for tracking a variable number of interacting targets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 11, pp. 1805–1819, 2005.

[4] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua, "Multiple object tracking using k-shortest paths optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 9, pp. 1806–1819, 2011.

[5] R.T. Collins, "Multitarget data association with higher-order motion models," in *IEEE Conf. on Comp. Vision and Pattern Recognition (CVPR)*, 2012, pp. 1744–1751.

[6] M. Primet and L. Moisan, "Point tracking: an a-contrario approach," *preprint MAP5*, 2012.

[7] A. Desolneux, L. Moisan, and J.-M. Morel, *From Gestalt Theory to Image Analysis. A Probabilistic Approach*, Springer-Verlag, collection Interdisciplinary Applied Mathematics, 2008.

[8] B. Grosjean and L. Moisan, "A-contrario detectability of spots in textured backgrounds," *Journal of Mathematical Imaging and Vision*, vol. 33:3, pp. 313–337, 2009.

[9] J. Verestóy and D. Chetverikov, "Experimental comparative evaluation of feature point tracking algorithms," in *Performance Characterization in Computer Vision*, pp. 167–178. Springer, 2000.