Contents lists available at ScienceDirect

Comptes Rendus Physique

www.sciencedirect.com

URSI-France 2018 Workshop: Geolocation and navigation / *Journées URSI-France 2018 : Géolocalisation et navigation*

A novel nonlinear least-squares approach to highly maneuvering target tracking

Une nouvelle méthode de moindres carrés non linéaires pour le pistage de cibles hyper-manœuvrantes

Marion Pilté [a,b,*], Silvère Bonnabel [a], Frédéric Livernet [c]

[a] *MinesParisTech, PSL University, 60, bd Saint-Michel, 75005 Paris, France*
[b] *Thales Land and Air Systems, voie Pierre-Gilles-de-Gennes, 91470 Limours, France*
[c] *Direction générale de l'armement (DGA), 83000 Toulon, France*

A R T I C L E   I N F O

Article history:
Available online 11 June 2019

Keywords:
Mono-target tracking
Radar
Nonlinear smoothing
Nonlinear state estimation
Probabilistic approach

Mots-clés :
Pistage mono-cible
Radar
Lissage non linéaire
Estimation d'état non linéaire
Approche probabiliste

A B S T R A C T

Trajectories of aerial and marine vehicles are typically made of a succession of smooth trajectories, linked by abrupt changes, i.e. maneuvers. Notably, modern highly maneuvering targets are capable of very brutal changes in the heading, with accelerations of up to 15 g. As a result, we model the target behavior using piecewise deterministic Markov models, driven by parameters that jump at unknown times. Over the past years, real-time (or incremental) optimization-based smoothing methods have become a popular alternative to nonlinear filters, such as the Extended Kálmán Filter (EKF), owing to the successive relinearizations that mitigate the linearization errors that inherently affect the EKF estimates. In the present paper, we propose to combine such methods for tracking the target during non-jumping phases with a probabilistic approach to detect jumps. Our algorithm is shown to compare favorably to the state-of-the-art Interacting Multiple Model (IMM) algorithm, especially in terms of target's velocity estimation, on a set of meaningful and challenging trajectories.

R É S U M É

Les trajectoires de véhicules aériens ou marins sont en général formées d'une succession de trajectoires lisses, séparées par des manœuvres brusques. En particulier, les cibles hyper-manœuvrantes modernes peuvent changer de cap de façon très abrupte, avec des accélérations pouvant aller jusqu'à 15g. Le comportement de la cible est donc modélisé par des modèles de Markov déterministes par morceaux, grâce à des paramètres à sauts. Depuis quelques années, les méthodes de lissage en temps réel sont devenues une alternative aux habituels algorithmes de filtrage, tels que le filtre de Kalman étendu (EKF). Dans cet article, de telles méthodes de lissage sont utilisées pour les phases sans sauts et sont combinées à une approche probabiliste pour déterminer les instants de sauts. L'algorithme ainsi proposé est comparé à un algorithme multi-modèles, l'IMM (*Interactirng*

* Corresponding author.
E-mail address: marion.pilte@thalesgroup.com (M. Pilté).

https://doi.org/10.1016/j.crhy.2019.05.019

Multiple Model), en particulier pour l'estimation de la vitesse de la cible, sur un ensemble de trajectoires représentatif et difficile.

1. Introduction

Considerable research has been devoted to the optimal estimation problem in the field of mono-target tracking. Applications span civilian airborne and marine surveillance and military tracking of highly maneuvering missiles. Although the formal equations of the optimal filter are easy to derive and have been known for decades, its implementation in real time is still a challenge. There are essentially two sources of difficulty. First, the use of models to accurately describe the target's motion, which are nonlinear, and then the use of multi-hypotheses regarding the unknown possible behavior of the target, which results in excessive combinatorics. Particle filters (PF), see, e.g., [1], have been a popular attempt over the past two decades to handle those two sources of difficulty.

Based on the idea that motion of manned and unmanned vehicles consists of a succession of smooth trajectories, with potentially abrupt changes from one type of trajectory (such as straight line) to the next (such as coordinated turn), S. Godsill and co-authors have advocated over the past decade the use of nonlinear piecewise deterministic Markov models (PDMM) to model the target's behavior, see, e.g., [2–4]. Between jumps, trajectories are modeled by ordinary differential equations driven by constant inputs. This kind of trajectories have long been a key model in tracking: see, for example, the constant velocity model in [5], the coordinated turn model [6], and our recent work [7]. Adapting PF techniques to the continuous-time setting of PDMM, S. Godsill and co-authors proposed the variable rate particle filter (VRPF). However, such filters are computationally demanding, as many particles are needed to fully cover the space of possible jumps and parameters.

In this paper, we consider PDMM that are akin to those considered in the VRPF literature. Instead of using a PF approach, we opt for a smoothing optimization-based approach. The use of such techniques for filtering and tracking have long been known, but only recent advances in computers have allowed them to be fully implementable. In robotics, and especially in the vast literature related to the problem of simultaneous localization and mapping (SLAM), optimization-based smoothing approaches, see [8–11], have virtually wholly replaced the once extremely popular PF-based approach [12]. Those methods currently enjoy much popularity because of the successive relinearizations they use until convergence, that mitigate the problem or linearization errors in nonlinear filtering.

In the present paper, inspired by, on the one hand, from the VRPD literature for target motion modeling, and, on the other one, by recent smoothing techniques from the robotics literature, we use smoothers to track the state of a PDMM driven by unknown constant inputs, and we use a probabilistic approach for jump detection. In the stationary phase, the state is very well tracked as our deterministic-based model provides smooth trajectories that are not fluctuating due to the assumption of process noise, and in turn the accuracy of the state estimates helps to rapidly detect jumps. It seems to us that our approach to estimate the target's state and possibly predict the motion of the target in the future is very similar to the way the eye of a human expert would proceed. The proposed estimator is shown to favorably compare to a state-of-the-art IMM for meaningful target motions, especially in terms of estimation of the velocity of the target.

2. Smoothing as an estimation procedure for target tracking

2.1. Classical smoothing approach

Consider a target that one must track. Assume a discrete time model and let the target's state at time i be denoted by $X_i \in \mathbb{R}^p$. The state typically consists of the position and velocity (and some additional quantities) of the target. Consider a nonlinear evolution model for the target of the form (1), with noisy measurements of the form (2).

$$X_i = f_i(X_{i-1}) + w_i, \tag{1}$$

$$y_i = h(X_i) + v_i \tag{2}$$

The goal of any filter, such as extended Kálmán filter (EKF) or interacting multiple models (IMM) filter, is to compute the distribution $p(X_n \mid y_{0:n})$ of the present state X_n conditionally on past and present measurements $y_{0:n} := \{y_1, \ldots, y_n\}$. In contrast, a smoother (sometimes referred to as Kálmán smoothing) computes the distribution $p(X_{0:n} \mid y_{0:n})$ of the entire past trajectory $X_{0:n} := \{X_0, \ldots, X_n\}$, conditionally on past measurements $y_{0:n} := \{y_1, \ldots, y_n\}$. Both a filter and a smoother allow us to find the best estimate of the state, that is, the most likely state X_n in the light of the information $y_{0:n}$ we have collected so far, which is referred to as the *maximum a posteriori* (MAP) estimate. The MAP estimate of the entire past trajectory $X_{0:n}$ is thus defined as $\operatorname{argmax}_{X_0,\ldots,X_n} P(X_{0:n}|y_{0:n})$, i.e.

$$X^*_{0:n} = \underset{X_{0:n}}{\operatorname{argmin}} - \log P(X_{0:n}|y_{0:n}) \tag{3}$$

Besides, under standard assumptions of independence of noises $(w_i, v_i)_{i\in\mathbb{N}}$, we get

$$P(X_{0:n}|y_{0:n}) = P(X_0)\prod_{i=1}^{n} P(X_i|X_{i-1})\prod_{k=1}^{n} P(y_k|X_k)$$

In this equation, $P(X_0)$ is a prior knowledge that we have on the initial state. Under the assumption of Gaussian noises $w_i \simeq \mathcal{N}(0, Q_i)$ and $v_i \simeq \mathcal{N}(0, N_i)$ to represent respectively model and measurement uncertainties, and the Gaussian prior $X_0 \sim \mathcal{N}(\bar{X}_0, P_0)$, we have from (1) that $P(X_i|X_{i-1}) = \tilde{C}\exp\left(||f_i(X_{i-1}) - X_i||^2_{Q_i}\right)$ and from (2), we have $P(y_k|X_k) = \bar{C}\exp\left(||h(X_{t_k}) - y_k||^2_{N_k}\right)$. Thus we end up with the following nonlinear least-squares problem

$$X^*_{0:n} = \underset{X_{0:n}}{\operatorname{argmin}} \left\{ ||X_0 - \bar{X}_0||^2_{P_0} + \sum_{i=1}^{n} ||f_i(X_{i-1}) - X_i||^2_{Q_i} + \sum_{k=1}^{n} ||h(X_k) - y_k||^2_{N_k} \right\} \tag{4}$$

where the norm is the Mahalanobis distance defined by $||e||_\Sigma = e^\intercal \Sigma^{-1} e$ for Σ a covariance matrix.

If the dynamical model f_i and the measurement function h are nonlinear, and a linearization point is not available, one must resort to non-linear optimization methods such as the Gauss–Newton or the Levenberg–Marquardt algorithm. The algorithm is based on successive linear approximations to (4), which iteratively improve the estimate $X_{0:n}$. Indeed, at each iteration, by denoting $\hat{X}_{0:n} = \{\hat{X}_0, \cdots, \hat{X}_n\}$ the current estimate, the problem may be linearized around $\hat{X}_{0:n}$ as follows. We let:

$$F_i = \left.\frac{\partial f_i(X)}{\partial X}\right|_{\hat{X}_{i-1}}, \quad H_k = \left.\frac{\partial h(X)}{\partial X}\right|_{\hat{X}_k}$$

Letting $a_i = \hat{X}_i - f_i(\hat{X}_{i-1})$, $c_k = y_k - h(\hat{X}_k)$, and $p_0 = \hat{X}_0 - \bar{X}_0$, the optimization problem can be approximated as

$$\delta X^* = \underset{\delta X}{\operatorname{argmin}} \left\{ ||p_0||^2_{P_0} + \sum_{i=1}^{n} ||F_i \delta X_{i-1} - \delta X_i - a_i||^2_{Q_i} + \sum_{k=1}^{n} ||H_k \delta X_k - c_k||^2_{N_k} \right\} \tag{5}$$

yielding at each iteration a linear least-squares problem to solve. Noting that we can re-write norms as follows:

$$||e||^2_\Sigma = e^\intercal \Sigma^{-1} e = (\Sigma^{-T/2} e)^\intercal (\Sigma^{-T/2} e) = ||\Sigma^{-T/2} e||^2$$

stacking the matrices F_i, H_k in a large matrix A and the vectors p_0, a_i, c_k in a large vector b, (5) may be re-written as

$$\delta X^* = \underset{\delta X}{\operatorname{argmin}} ||A\delta X - b||^2 \tag{6}$$

The solution to this linear least-squares problem is then notoriously obtained by equating the gradient of $||A\delta X - b||^2$ to 0, which yields

$$\delta X^* = (A^\intercal A)^{-1} A^\intercal b \tag{7}$$

A is a large but sparse matrix, and linear algebra methods can be used to compute efficiently this solution: as explained in, e.g., [10], the Cholesky decomposition or the QR matrix factorization allows us to efficiently compute $(A^\intercal A)^{-1}$. The obtained solution δX^* to (7) depends on a particular realization of random noises w_i, v_i, and varies due to fluctuations in the data y_i, which are stacked in vector b. Its variability over a large number of noise realizations is encoded in the covariance matrix Cov(δX^*, which can be shown to be equal to $(A^\intercal A)^{-1}$.

As already mentioned, the goal of a filter is to output the state that maximizes the posterior distribution $P(X_n|y_{0:n})$, whereas a smoother outputs the argmax of $P(X_{0:n}|y_{0:n})$. As time passes, n grows boundlessly and re-estimating the entire trajectory at each n may become intractable. Typically, the matrix A that appears in (6) at each iteration is of dimension $O(n^2)$, yielding a $O(n^3)$ complexity to evaluate (7). As a result, there have been various attempts to compute incrementally the MAP estimate for the smoothing problem. Notably, in robotics, the well-studied problem of SLAM has a structure that lends itself to such incremental methods, as proved in [10].

Another popular solution is to use a fixed-lag smoother, which aims to approximate $P(X_{n-k:n}|y_{0:n})$ for some fixed lag $k \in \mathbb{N}$. Such smoothers are obtained by marginalizing the old states $X_{0:n-k-1}$ out, see, e.g., [13], see also [14].

2.2. Restriction to a deterministic evolution model over a sliding window as a tuning strategy

The actual motion of objects such as aircrafts and marine vehicles typically consist of a succession of distinct maneuvers commanded by an operator. As a result, the trajectories of those objects look like a succession of smooth trajectories that are well described by continuous time ordinary differential equations (ODE). In Section 3.1, we will take into account the possibility of abrupt changes in the trajectory, but for now let us consider only the phase in between maneuvers where the trajectory is governed by deterministic equations. Prosaically, this means that the covariance matrix Q_i of process noise w_i in (1) is null. Thus, (4) becomes:

$$\begin{aligned} &\underset{X_{0:n}}{\text{minimize}} \quad \left\{ ||X_0 - \bar{X}_0||^2_{P_0} + \sum_{k=1}^{n} ||h(X_k) - y_k||^2_{N_k} \right\} \\ &\text{subject to} \quad X_i = f_i(X_{i-1}), \quad i = 1, \ldots, n \end{aligned} \tag{8}$$

Of course, such a model is too rigid in practice, as there are always fluctuations in the target behavior with respect to a model specified in advance. A boat or a plane may deviate slightly from its planned trajectory due to perturbations, or to slight motion adaptations from the pilot. This is why, in the target tracking literature, the covariance Q_i of noise w_i is always positive, and serves as a tuning parameter.

Let us temporarily assume that we are dealing with problem (8), though. To simplify the exposure, assume f_i, h_i are linear and let F_i, H_i denote the corresponding matrices. This means that $X_k = F_k \cdots F_1 X_0$, and thus $h(X_k) = H_k F_k \cdots F_1 X_0$. As a result, solving problem (8) is equivalent to minimizing $||X_0 - \bar{X}_0||^2_{P_0} + \sum_{k=1}^{n} ||H_k F_k \cdots F_1 X_0 - y_k||^2_{N_k}$ with respect to X_0. Let $\tilde{H}_0 = Id, \tilde{H}_1 = H_1 F_1, \cdots \tilde{H}_k = H_k F_k \cdots F_1$. We see, by applying the results of Section 2.1, that $\mathrm{Cov}(X_0^*) = (\sum_{i=0}^{n} \tilde{H}_i^{\mathsf{T}} \tilde{H}_i)^{-1}$, and as X_n^* is obtained deterministically from X_0^*, it has similar covariance. As a result, when there is no process noise, the confidence about the current state X_n^* obtained by solving problem (8) grows as $1/n$, where n is the number of measurements. However, as the model cannot be completely accurate due to the unpredictability of the target's behavior, new observations need to constantly impact the estimate for accurate tracking and it is not sensible to assume the confidence in the estimate to tend to 0 as $1/n$.

On the other hand, if process noise is considered, the covariance of X_n^* obtained by solving problem (4) is lower bounded by some matrix C^* that depends on the magnitude of the Q_i's. Matrix C^* is known as the Cramér–Rao bound. As a result, we see there are two different routes for the practitioner to tune its estimator. Either, one can attempt to solve (4) and tune process noise Q_i, leading to an asymptotic confidence C^* about the estimate. Or we can consider the estimation problem with no process noise $Q_i = 0$, leading to (8), but only on a sliding window of size $\bar{k}$, that is,

$$\begin{aligned} &\underset{X_{n-\bar{k}:n}}{\text{minimize}} \quad \left\{ \sum_{j=n-\bar{k}}^{n} ||h(X_j) - y_j||^2_{N_j} \right\} \\ &\text{subject to} \quad X_i = f_i(X_{i-1}), \quad i = n - \bar{k} + 1, \ldots, n \end{aligned} \tag{9}$$

Of course, those two routes are not strictly equivalent mathematically, but they may be viewed as alternative ways to tune the estimator. The second route that consists in solving (9) at each time step n is the one that we advocate in the present paper. In this case, the depth of the window $\bar{k}$ is the tuning parameter that appears as an alternative to process noise Q_i: one should bear in mind that the resulting uncertainty about the current state X_n is of magnitude $1/\bar{k}$, and this should be tuned in accordance with the fit between the evolution model $X_i = f_i(X_{i-1})$ and the actual motion of the target (in the extreme case where the motion of the target is exactly modeled by this deterministic approach, one may set $k = n$; on the other hand, if the fit of the model to actual motion is approximative, $\bar{k}$ should be kept at a moderate value).

In the literature devoted to system identification [15], the use of recursive least squares is pivotal. For real-time implementation, where one must track a parameter that varies (slowly) over time, it is common to use a "forgetting factor" that gives less weight to old observations. Our approach may be related to this practice, by setting a forgetting factor as 1 for the $\bar{k}$ latest observations and 0 for the preceding ones.

3. Smoothing applied to deterministic systems with random jumps

3.1. Considered systems and simplifying assumptions

As already explained, actual motions of objects such as aircrafts and marine vehicles typically consist of a succession of distinct maneuvers commanded by an operator. As a result, the trajectories of objects are in fact smooth and well described by continuous time ordinary differential equations (ODE) driven by constant inputs between change points. This was advocated in particular by S. Godsill with various co-authors who proposed the variable rate particle filter [2–4], a sequential Monte Carlo (SMC) method, well suited to piecewise deterministic models. Following [2], we consider the following piecewise deterministic Markov model:

$$\frac{\mathrm{d}}{\mathrm{d}t}x_t = f(x_t, u_{K(t)}) \tag{10}$$

where $x_t \in \mathbb{R}^p$ is the continuous time target's state, $K(t) \in \mathbb{N}$ is a stochastic point process that counts the number of random jumps up to time t, and $u_0, u_1, \cdots$ is a sequence of random inputs that drive the ODE (10). Moreover, at discrete time instants $t_0, t_1, \cdots$, we get (range and bearing) measurements of the form:

$$y_n = h(x_{t_n}) + v_n \tag{11}$$

The goal is to estimate the most likely state value, that is $\operatorname{argmax} p(x_t \mid y_{0:n})$ for $t_n \le t < t_{n+1}$. To simplify the estimation task, we will assume that jumps can only occur at pre-specified discrete times. This may look like a harmful approximation, but we will see in Section 4 that it is actually easy to modify the least-squares problem to mitigate its impact on the estimation. Furthermore, to keep the notation simple, we will assume that jump times coincide with observation times $t_1, t_2, \cdots$. We let r_k be the random variable indicating jump at time k ($r_k = 1$ if there is a jump). $K(t)$ is the number of jumps between times 0 and t. We also let $\tilde{K}_n$ be the number of jumps between 0 and t_n. We obviously have $K(t_n) = \tilde{K}_n$. Note that $\tilde{K}_n$ is a function of $r_{0:n}$. Finally, we let $\theta_n = (r_{0:n}, u_{0:\tilde{K}_n}, x_0)$ be the parameters that we seek to estimate. To recover x_t, we only need to integrate (10) based on the knowledge of θ_n for $t < t_{n+1}$.

3.2. *Corresponding smoothing problem*

For the problem described at the preceding paragraph, our goal is to find the most likely state x_t at present time. As just explained, for $t_n \le t < t_{n+1}$ this amounts to finding the parameter θ_n. First, note that we have

$$\log p(y_{0:n} \mid \theta_n) = \log p(y_{0:n} \mid r_{0:n}, u_{0:\tilde{K}_n}, x_0) = -\sum_{k=1}^{n} ||y_k - h(x_{t_k})||^2_{N_k} + Cste \tag{12}$$

where N_k is the covariance matrix of the Gaussian measurement noise v_k, and the x_{t_k} are obtained by integrating (10). Trying to estimate θ_n by maximizing the likelihood (12) is not suitable. Indeed, the optimal solution will jump at all times to stick to at most to the observations. Obviously, we need a prior on the average time between successive maneuvers. Letting $0 < p < 1$ be the probability of a jump at each time t_i, we have the following prior

$$p(\tilde{K}_n = j) = p(\{r_{0:n} \text{ contains } j \text{ ones and } n-j \text{ zeros}\}) = P(\text{bin}(n, p) = j) = \binom{n}{j} p^j (1-p)^{n-j} \tag{13}$$

Let us assume as prior on the initial state $x_0 \sim \mathcal{N}(\bar{x}_0, P_0)$. Estimating the most likely state for the piecewise deterministic model of Section 3.1 boils down to the following optimization problem:

$$\begin{aligned}
\theta_n^* = (r_{0:n}, u_{0:\tilde{K}_n}, x_0)^* &= \underset{r_{0:n}, u_{0:\tilde{K}_n}, x_0}{\operatorname{argmin}} \; -\log p(r_{0:n}, u_{0:\tilde{K}_n}, x_0 \mid y_{0:n}) \\
&= \underset{r_{0:n}, u_{0:\tilde{K}_n}, x_0}{\operatorname{argmin}} \; [-\log p(y_{0:n} \mid r_{0:n}, u_{0:\tilde{K}_n}, x_0) - \log p(u_{0:\tilde{K}_n} \mid r_{0:n}) - \log p(r_{0:n}) - \log p(x_0)] \\
&= \underset{r_{0:n}, u_{0:\tilde{K}_n}, x_0}{\operatorname{argmin}} \; (\sum_{k=1}^{n} ||y_k - h(x_{t_k})||^2_{N_k} - \log[\binom{n}{\tilde{K}_n} p^{\tilde{K}_n}(1-p)^{n-\tilde{K}_n}] - ||x_0 - \bar{x}_0||^2_{P_0})
\end{aligned} \tag{14}$$

with the values of x_{t_k} obtained by integrating (10) with parameters $r_{0:n}, u_{0:\tilde{K}_n, x_0}$. The justification for removing $\log p(u_{0:\tilde{K}_n} \mid r_{0:n})$ from the optimization problem is that we assume a flat prior on the parameters $u_0, u_1, \cdots$, as we assume that each jump may correspond to a complete shift. Of course, alternative priors might be considered depending on the application.

4. Proposed algorithm

The optimization problem (14) is not tractable, owing to the combinatorics in the jump times. Unfortunately, this remains true even if the optimization is restricted to a sliding window along the lines of Section 2.2. Over a window of size $\bar{k}$, there are $2^{\bar{k}}$ possibilities for the discrete variable $r_{n-\bar{k}:n}$, each leading to a continuous optimization problem with respect to $u_{\tilde{K}_{n-\bar{k}}:\tilde{K}_n}$. To efficiently approximate the optimization problem, we propose the following tractable strategy.

Setting a horizon. We first choose a size $\bar{k}$ for a sliding window for the reasons explained in Section 2.2, corresponding to the forgetting horizon of the smoother in the absence of jumps: even if there are no jumps, this allows the state to deviate over time from the deterministic model (10) while continuing to be efficiently tracked.

Continuity assumption of x_t at jumps. In our model described in Section 3.1, we made the simplifying assumptions that jumps only occurred at discrete time instants $t_0, t_1, \cdots$, while they actually can occur at any time. We also said there was a way

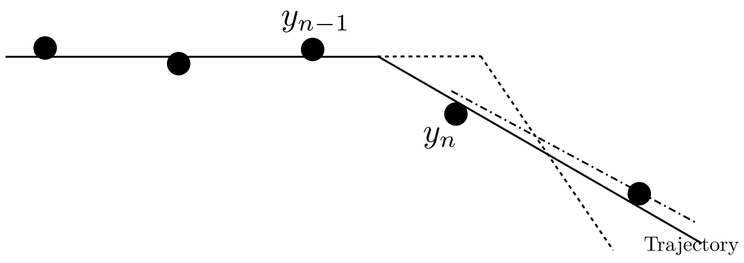


Fig. 1. Trajectory with velocity u jumping strictly between the observations at times t_{n-1} and t_n. Under the assumption that a jump may only occur at t_n and the trajectory x_t is continuous, we obtain the dotted line, which is a poor trajectory estimate. However, if we assume u jumps at time t_n but we relax the assumption that the trajectory x_t must be continuous and allow it to jump – see (15) –, we obtain a much better estimate (dashed line).

around the harmfulness of this approximation. Indeed, the "true" considered model (10) implies continuity in x_t, since $\dot{x}_t$ is bounded. If a jump actually occurs between t_{n-1} and t_n, for instance at time $(t_{n-1}+t_n)/2$, assuming it has occurred at time t_n and the trajectory is continuous may result in degraded accuracy, see Fig. 1 for a graphical illustration. However, by relaxing the continuity assumption and assuming small jumps in the state x_t may also occur at jumping times, may compensate for the assumption that jumps may only occur at pre-specified instants.

Assuming that a jump has occurred at time strictly *between* t_{l-1} and t_l, and given that no other jump has occurred until current time n, to find the most likely state x_{t_n} we relax the continuity assumption and solve the optimization problem

$$\underset{u,x_{t_l}}{\operatorname{argmin}}[||x_{t_l}-\bar{x}_{t_l}||^2_{P_{\text{jump}}}+\sum_{j=l}^{n}||y_j-h(x_{t_j})||^2_{N_j}] \tag{15}$$

where $\bar{x}_{t_l}$ is the value obtained at instant index l by integrating (10) until time t_l based on the former value of u (i.e. value obtained by continuity), and where P_{jump} is a covariance matrix that must be tuned as representative of the typical squared distance that x_t may achieve between successive observations.

Jump detection. Assume n denotes the current time step, and the last jump occurred at time $l > n-\bar{k}$. This means that $r_{l:n}$ contains one 1 followed by zeros. Using (12), we may solve the problem

$$\underset{u,x_{t_l}}{\operatorname{argmin}} -\log p(u,x_{t_l}\mid y_{l:n},r_{l:n}) = \underset{u,x_{t_l}}{\operatorname{argmin}}[||x_{t_l}-\bar{x}_{t_l}||^2_{P_{\text{jump}}}+\sum_{k=l}^{n}||y_k-h(x_{t_k})||^2_{N_k}] \tag{16}$$

where the x_{t_k} are obtained by integrating $\dot{x}_t = f(x_t,u)$ with initial condition x_{t_l}, and $\bar{x}_{t_l}$ corresponds to the estimate obtained by continuity with the model before the jump. As explained in Section 2.1, it is classically possible to assess a covariance matrix $\tilde{P}$ to the couple (u,x_{t_l}). As in the absence of jumps $x_{t_n}=\phi(u,x_{t_l})$ is a deterministic function of the parameters (u,x_{t_l}), with ϕ the flow of (10), covariance matrix of x_{t_n} is $P_n = D\phi\tilde{P}D\phi^\intercal$ where $D\phi$ denotes the differential of ϕ at the optimal values $(u^*,x^*_{t_l})$. This allows us to compute the associated Mahalanobis distance:

$$\Delta=\sqrt{(y_k-h(x^*_{t_n}))^\intercal(\mathrm{D}h^\intercal P_n \mathrm{D}h)^{-1}(y_k-h(x^*_{t_n}))} \tag{17}$$

where Dh denotes the differential of h at $x^*_{t_n}$. We may then apply the χ^2-test to determine if there is a jump, i.e. as soon as Δ goes above a certain value corresponding to a, say 95% quantile of χ^2, a potential jump is suspected.

Proposed strategy. We approximate the solution to the true optimization problem (14) by first restricting it to a sliding window of horizon $\bar{k}$. Then, we assume jumps are scarce (that is, the jump probability p is small) and we let $T_n \in \mathbb{N}$ denote the time index at which the last jump before the current (observation) time t_n occurs. At each jump time T_n, the window is re-initialized, since u jumps to an unknown arbitrary value. As a result, at time t_n, the current optimal parameter $u^*_{\bar{K}_n}$ is obtained as a solution to problem (16) with $l=\max(n-\bar{k},T_n)$. Assume that, according to the χ^2 test, a possible jump is detected at time index n, as (17) goes above some predefined quantile q. As this does not mean a jump has necessarily occurred, we initialize a second smoother based on jump at time n. The first smoother assumes "no jump" has occurred at n, it then solves:

$$\underset{u,x_{t_l}}{\operatorname{argmin}}\left(||x_{t_l}-\bar{x}_{t_l}||^2_{P_{\text{jump}}}+\sum_{j=l}^{n+k}||y_j-h(x_{t_j})||^2_{N_j}\right)$$

where the x_{t_j}'s are obtained by integrating $\dot{x}_t = f(x_t,u)$. Keeping in mind (14), the associated posterior log-likelihood writes:

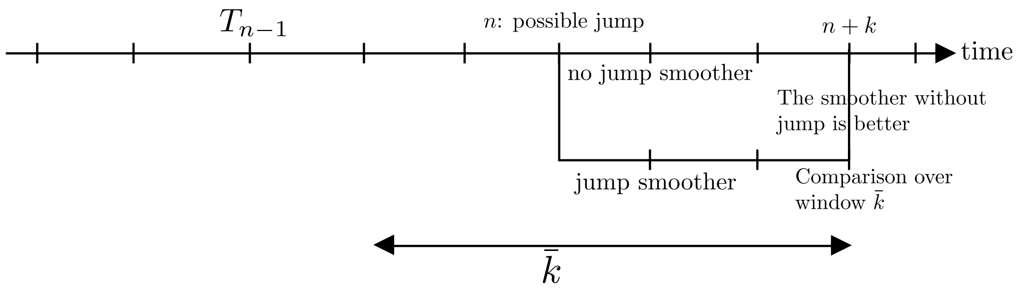


Fig. 2. Smoothing with jumps and underlying non-jumping smoother.

$$L_{\text{no jump}}(n+k) = \log p(y_{l:n+k} \mid u^*, x^*_{t_l}) + (n+k-l)\log(1-p) + \log p(x^*_{t_l}) \tag{18}$$

On the other hand, the candidate "jumping" smoother that assumes that one jump occurred at time index n solves:

$$\operatorname*{argmin}_{u,x_{t_n}} \Big(||x_{t_n} - \bar{x}_{t_n}||^2_{P_{\text{jump}}} + \sum_{j=n}^{n+k} ||y_j - h(x_{t_j})||^2_{N_j} \Big) \tag{19}$$

where the x_{t_j} are again obtained by integrating $\dot{x}_t = f(x_t, u)$. The associated posterior log likelihood writes:

$$L_{\text{jump}}(n+k) = \log p(y_{n:n+k} \mid u^*, x^*_{t_n}) + \log\big(kp(1-p)^{k-1}\big) + \log p(x^*_{t_n}) \tag{20}$$

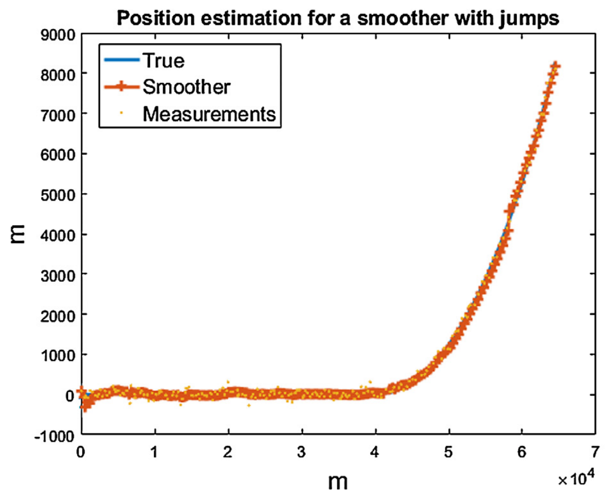
We clearly see the benefit of having a prior on jumps: as the size of the window for the jumping smoother is smaller, the residual of the least squares will be smaller, as it is easier to find a u^* that fits better a lesser number of data. But its likelihood will be penalized as soon as $kp < 1-p$, and in our strategy p must be assumed small. Thus, the binomial term acts as a regularization term that will favor the non-jumping smoother, and prevent the estimation from constantly jumping, which may end up in meaningless estimates. As p is assumed very small, the likelihood of two or more jumps is considered as negligible. After a possible jump detection at n, we let k increase until either $L_{\text{jump}}(n+k) > L_{\text{no jump}}(n+k)$ and then the jump at n is validated, leading to $T_n = T_{n+k} = n$, or until $k = \bar{k}$, in which case both smoothers coincide and the jumping smoother is suppressed. This strategy is illustrated by Fig. 2.

Note that there must be a minimum time elapsed after the candidate jump at n for comparing the smoothers. We will denote it by index k. Indeed, the problem (19) might be solved with 0 residual as long as $\dim(u) > k\dim(y)$. Jump penalization may not suffice to have $L_{\text{jump}}(n+k) > L_{\text{jump}}(n+k)$, and the jumping smoother is artificially favored.

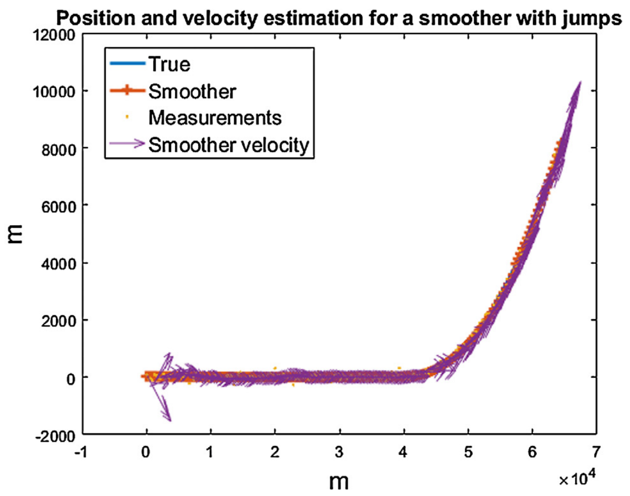
Algorithm. The pseudo code is displayed in Algorithm 1. During the phases where the jumping and the non-jumping smoothers are running in parallel, the user can choose whether to output the estimation of one or the other smoother.

Algorithm 1 Smoothing algorithm with jumps

```
Input: Initial prior $(\bar{x}_0, P_0)$; Observations $y_1, y_2, \cdots$,
 1: Set $P_{\text{jump}} = P_0$, $T_0 = 0$, $n = 0$
 2: Solve $(u^*, x^*_{t_l}) = \operatorname{argmin}_{u,x_{t_l}} [||x_{t_l} - \bar{x}_{t_l}||^2_{P_{\text{jump}}} + \sum_{j=l}^{n} ||y_j - h(x_{t_j})||^2_{N_j}]$ with $l = \max(0, n-\bar{k})$ and where $\bar{x}_{t_l}$ is either $\bar{x}_0$ or obtained by continuity through model (10) if $l > 0$.
 3: while $\Delta^2 < q$ with $\Delta$ defined by (17) and $q$ the user specified quantile for the $\chi^2$-test do
 4:   Set $T_n = T_{n-1}$
 5:   $n = n + 1$
 6:   if $\Delta^2 > q$, a candidate jump is detected at time index $n$ then
 7:     for $j = n : n + \bar{k}$ do
 8:       Compute estimations for a smoother with jump and with no jump
 9:     end for
10:     if for some $j$ we have $L_{\text{jump}}(j) > L_{\text{no jump}}(j)$ then
11:       Set $T_j = n$ and select the jumping smoother by selecting $(u^*, x^*_{t_l}) = \operatorname{argmin}_{u,x_{t_n}} [||x_{t_n} - \bar{x}_{t_n}||^2_{P_{\text{jump}}} + \sum_{j=n}^{n} ||y_j - h(x_{t_j})||^2_{N_j}]$ where $\bar{x}_{t_n}$ is obtained by continuity through model (10) with previous optimal $u^*$
12:     end if
13:     Set $n = j$
14:   end if
15: end while
    Output: $x_t$ is obtained for $t_l \le t < t_{n+1}$ with $l = \min(T_n, n-\bar{k})$ by integrating (10) with parameters $(u^*, x^*_{t_l})$.
```



(a)



(b)

Fig. 3. Example of position and velocity estimates using our approach. The real trajectory has two jumps. The smoothing algorithm detects the two jumps.

5. Application to a simple 2D target model

A simple and meaningful deterministic model for a target in 2D consists of a succession of straight lines and arcs of a circle (at constant speed). In other words, this is a piecewise constant speed and curvature assumption. This also coincides with the Frenet–Serret model that we introduced in [7], with noise turned off. The model is as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}\theta_t = \omega_t, \quad \frac{\mathrm{d}}{\mathrm{d}t}x_t = u_t \begin{pmatrix} \cos\theta_t \\ \sin\theta_t \end{pmatrix}, \quad \frac{\mathrm{d}}{\mathrm{d}t}\omega_t = 0, \quad \frac{\mathrm{d}}{\mathrm{d}t}u_t = 0 \tag{21}$$

where $x_t \in \mathbb{R}^2$ is the position of the target, $\theta_t \in \mathbb{S}^1$ denotes the orientation of the velocity vector, u_t its norm, and $\omega_t := \frac{\mathrm{d}}{\mathrm{d}t}\theta_t$ is related to the curvature of the trajectory. The observations are supposed to be of the form $y_n = x_{t_n} + v_n$, with v_n a Gaussian noise. Here again, we assume the parameters u_t, ω_t to be piecewise constant, and we assume that jumping times coincide with observation times.

5.1. *Solving the smoothing problem without jumps*

Let us first consider the problem without jumps, to explain how the corresponding optimization problem is solved. It is convenient to discretize the problem as exact discretization is possible. Consider indeed the discrete-time state (22). The state can be expressed as (23), and $y_k = h(X_k) + v_k$ where $h(X_k) = x_{t_k}$.

$$X_k = \begin{pmatrix} \theta_{t_k} & x_{t_k} & \omega_{t_k} & u_{t_k} \end{pmatrix}^{\mathsf{T}} \tag{22}$$

$$x_n = x_0 + u_0 \Delta t \sum_{k=0}^{n-1} \begin{pmatrix} \cos(\theta_0 + k\omega_0\Delta t)\frac{\sin(\omega_0\Delta t)}{\omega_0} - \sin(\theta_0 + k\omega_0\Delta t)\frac{1-\cos(\omega_0\Delta t)}{\omega_0} \\ \sin(\theta_0 + k\omega_0\Delta t)\frac{\sin(\omega_0\Delta t)}{\omega_0} + \cos(\theta_0 + k\omega_0\Delta t)\frac{1-\cos(\omega_0\Delta t)}{\omega_0} \end{pmatrix} \tag{23}$$

The corresponding least-squares problem is $||X_0 - \hat{X}_0||^2_{P_0} + \sum_{k=0}^{N} ||y_k - h(X_k)||^2_N$, and is amenable to a problem involving only X_0 through exact discretization (23). Then, we can solve the problem through successive linear approximations as explained in Section 2.1. Calculations are elementary, but all the developments take much space. Due to space limitations, they are not reproduced herein.

5.2. *Accounting for jumps*

For the 2D Frenet–Serret target model, with Cartesian observations, Algorithm 1 may be directly applied. In the present application, the piecewise constant (jumping) parameters are ω and u, the angular velocity and the norm of the velocity. An example is provided in Fig. 3. The trajectory presented has been created with a 2D Frenet–Serret target model, with random jumps for the angular velocity ω and the norm of the velocity u. The first jump, when the turn occurs, is shown in Fig. 4. The algorithm applied for the estimation is Algorithm 1, with the non-linear Frenet–Serret model (21).

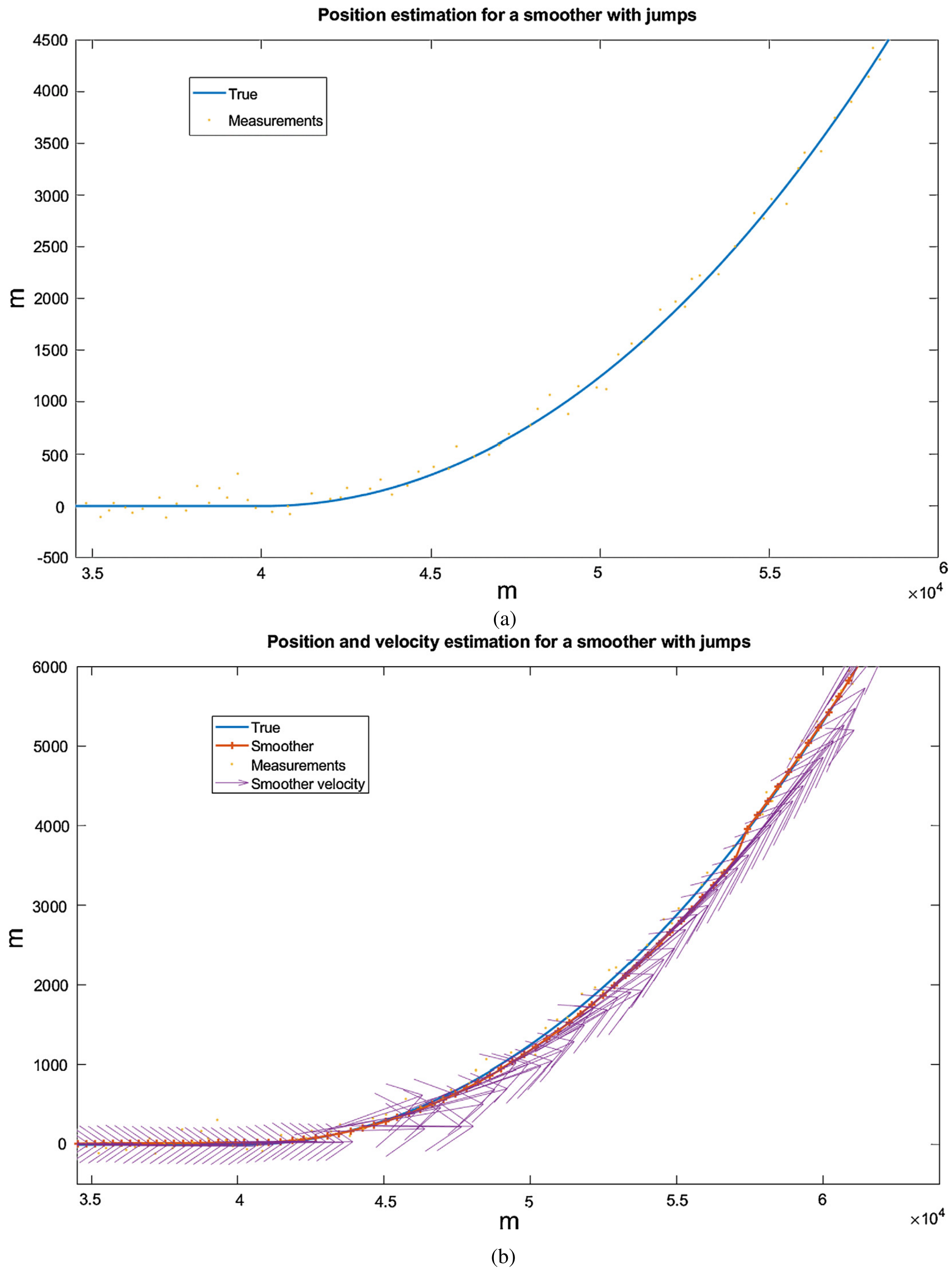


Fig. 4. Zoom on the first jump, between a straight line and a turn. The estimation of the velocity adapts smoothly to this turn.

6. Comparison with state-of-the-art IMM

In this section, we compare our smoothing Algorithm 1 with the IMM algorithm. The IMM was introduced in [16] for systems with Markovian switching. Owing to its computational efficiency, its versatility, and its accuracy in terms of tracking performances, it has become prevalent in the field, where it is considered as the (industrial) state-of-the-art filter for mono-target tracking. Moreover, good performances on the type of problems we consider in Section 3.1 may be anticipated, as the IMM filter is inherently based upon randomly switching models.

Table 1
RMSE for the smoothing algorithm and the IMM on trajectories simulated with 100 Monte Carlo simulations of random jumps of heading, angular velocity, and norm of velocity.

Parameter	Smothing: smoothed	Smoothing: real time	IMM
Position (m)	40	60	43
Norm of velocity (m/s)	15	32	75
Orientation (rad)	0.12	0.24	0.26

Table 2
RMSE for the smoothing algorithm and the IMM for the trajectory of Fig. 5.

Parameter	Smothing: smoothed	Smoothing: real time	IMM
Position (m)	33	49	46
Norm of velocity (m/s)	13	35	81
Orientation (rad)	0.18	0.45	0.67

6.1. Results

We compare an IMM with three models (constant velocity, constant turn, and constant acceleration) with our smoothing Algorithm 1. The smoothing algorithm is tuned as follows: the sliding window horizon is taken to be the entire trajectories, and we tune the number of observations we need to wait after a candidate jump to accept or reject as $k = 5$. The jump probability is tuned very low: the average jump probability by unit step p is tuned such that the average number of jumps over the whole trajectory is 0.02, whereas the actual number of jumps in the trajectory is around 2. The transition probability matrix for the IMM is $\begin{pmatrix} 0.4 & 0.3 & 0.3 \\ 0.3 & 0.4 & 0.3 \\ 0.3 & 0.3 & 0.4 \end{pmatrix}$.

6.1.1. Monte Carlo simulations results

We provide RMSE values computed for a set of 100 randomly generated trajectories with random jumps in heading θ, norm of velocity u, and angular velocity ω. The results are displayed in Table 1. Smoothing "real-time" is an implementation of Algorithm 1, whereas smoothing "smoothed" returns the trajectory estimated by the smoother at the end of the experiment. The difference is that if a candidate jump at time n is actually validated at time $n+k$, the real-time smoother estimates the state with the "no-jump" smoother between n and $n+k$ (while expecting the jump to be validated) whereas the smoothed one provides the estimates of the jumping smoother during transition. As a result, our smoothed estimates may be viewed as nearly optimal.

For the comparison to be fair, we must compare the real-time smoother to the IMM. We see that our smoothing-based estimator outperforms the IMM in terms of velocity estimation (norm and orientation of velocity vector). However, for the position precision, we observe that the estimations provided in real time of the smoothing algorithm are less precise than the ones of the IMM. Indeed, the IMM has a quite large process noise, so the position tends to stick more to the observations, whereas the smoothing position estimation can deviate a little. The IMM is tuned so that the convergence after the jumps is fairly fast. This implies high-process noise, so it introduces less accuracy (for the velocity especially) during constant motions. Moreover, the transition probabilities of the IMM have also to be tuned. Note that velocity estimation may be considered as more important that position, since modern radars measure very accurately the position. However, velocity is not measured and an accurate estimate may be pivotal for, e.g., beam repointing during active tracking.

6.1.2. Results for a challenging trajectory

We also run the filters on a trajectory generated as follows: first, a constant velocity linear motion, then an abrupt 90-degree turn that occurs between two observations, followed by another straight-line motion, and a slow turn, containing several observations. The results are displayed in Fig. 5. There again, we see that our smoothing approach is better suited to this type of trajectories than an IMM, especially after a jump in the trajectory. The error values averaged over the entire trajectory are collected in Table 2. Note that this kind of challenging trajectory is used by the French Department of Defense (DGA) as a key trajectory to benchmark tracking algorithms.

6.2. Discussion about IMM

During model changes, the sampling of the observations (which might be over one second, for rotating antenna radars) is a source of erroneous estimates. Indeed, during these transition phases, there exist an infinite number of likely solutions, when the exact time of the transition is not known. During these transition phases, recursive Kálmán filters estimate a unique solution, which has a high probability to be erroneous. To overcome this problem, and avoid the divergence of the Kálmán filters during transitions, multiple models filters (IMM) are commonly used. A well-known caveat to practitioners of IMM is that at least one of the models must be tuned with a high-process noise (and usually a constant acceleration

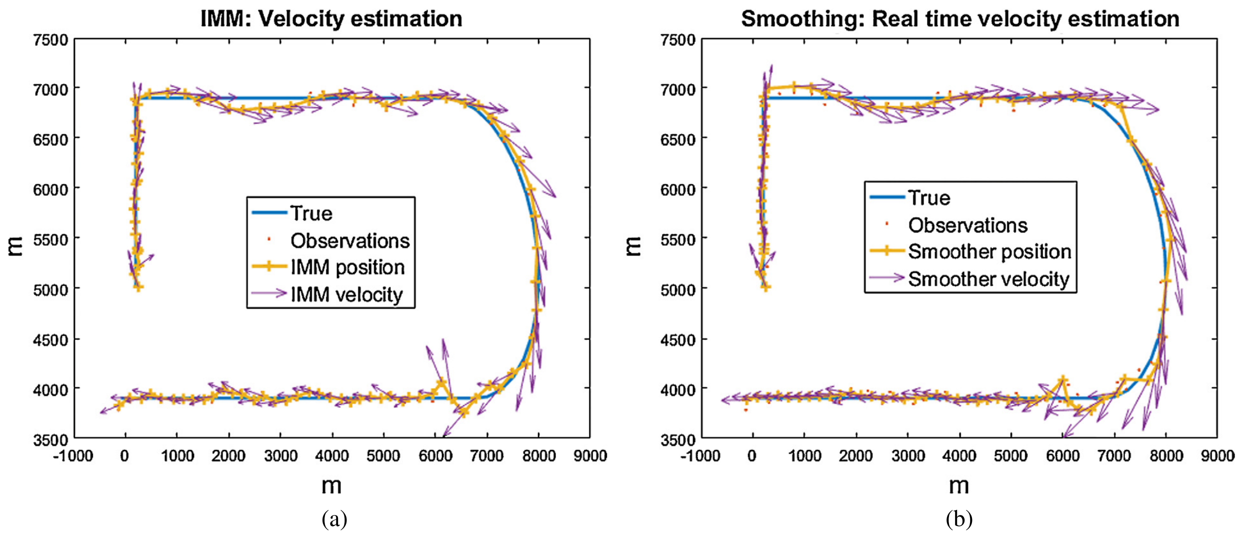


Fig. 5. Velocity vector along the trajectory for the IMM (Fig. 5a), for the smoothing algorithm, with the results output in real time (Fig. 5b), and for the smoothing algorithm with the final result (Fig. 5b). The results are more precise for the smoothing algorithm, even when considering the real time output, and not the final smoothed estimation. The smoothing algorithm is able to manage the jumps, and to be accurate during straight lines, whereas the IMM algorithm, while managing the jumps has troubles during straight lines, because of the high process noise.

model), to ensure that at least one of the filters in the IMM never diverges. Engineers sometimes compare it to a "garbage collector" model, which saves the day when none of the other models is properly fitting the measurements. The problem is that this model generates uncertainty on the estimate and usually delays the convergence of the filter after a transition during phases with constant kinematic model.

The jumping smoother with the Frenet–Serret model allows one to optimize these transition phases. The conception of the algorithm is designed to be optimal during constant input phases, and to handle jumps gracefully. We see indeed the smoothing algorithm with jumps works very well in the presence of very abrupt jumps.

6.3. *Discussion about tuning*

Another advantage of the smoothing algorithm proposed in this chapter for practitioners is that there are very few parameters to tune, a feature in sharp contrast with other methods that involve process noise covariance matrices. The three scalar parameters to tune are the number of observations we expect after a possible jump to accept the jump or to reject it, called k in Algorithm 1, the size of the sliding window $\bar{k}$ and the probability p of a jump, which is used in (13) to compare the smoother that has just jumped with the one that has not jumped. Moreover, our experience is that the filter is robust to small variations in the parameters. In contrast, the IMM has to be tuned carefully, and must match the amplitude of the jumps. Indeed, if the process noise is too low, then the filter cannot accommodate the jumps, whereas if it is too high, the accuracy elsewhere is degraded.

7. Conclusion

In this paper, a novel estimation method has been applied to the mono-target tracking problem, using nonlinear least-squares and a target dynamic model based on piecewise deterministic Markov models. As no process noise is assumed during the non-jumping phases, the estimation is quite smooth and accurate in the absence of jumps (stationary phases), and allows us to accurately detect jumps in the trajectory (transition phases). Some types of targets, such as modern missiles, may exhibit very abrupt changes in heading between two observations (while changes in the linear velocity/acceleration are necessarily much more limited). Our estimation algorithm should prove especially useful for this type of targets, as illustrated by the displayed comparisons to the IMM.

This smoothing algorithm has only been derived for a 2D target model, in a vector space. Perspectives include extension to 3D, using the general Lie group setting introduced in [7].

References

[1] A. Doucet, N. De Freitas, K. Murphy, S. Russell, Rao-blackwellised particle filtering for dynamic Bayesian networks, in: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., 2000, pp. 176–183.

[2] S. Godsill, J. Vermaak, Variable rate particle filters for tracking applications, in: Proc. IEEE/SP 13th Workshop on Statistical Signal Processing, 2005, IEEE, 2005, pp. 1280–1285.

[3] P. Bunch, S. Godsill, Dynamical models for tracking with the variable rate particle filter, in: Proc. 15th International Conference on Information Fusion (FUSION), IEEE, 2012, pp. 1769–1775.
[4] P. Bunch, S. Godsill, Particle smoothing algorithms for variable rate models, IEEE Trans. Signal Process. 61 (7) (2013) 1663–1675.
[5] Y. Bar-Shalom, X. Li, T. Kirubarajan, Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software, Wiley, 2004, https://books.google.fr/books?id=xz9nQ4wdXG4C.
[6] M.R. Morelande, N.J. Gordon, Target tracking through a coordinated turn, in: Proc. ICASSP '05, IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005, vol. 4, 2005, pp. 21–24.
[7] M. Pilté, S. Bonnabel, F. Barbaresco, An innovative nonlinear filter for radar kinematic estimation of maneuvering targets in 2d, in: Proc. 18th International Radar Symposium (IRS), IEEE, 2017, pp. 1–10.
[8] S. Thrun, M. Montemerlo, The graph slam algorithm with applications to large-scale mapping of urban structures, Int. J. Robot. Res. 25 (5–6) (2006) 403–429.
[9] F. Dellaert, M. Kaess, Square root sam: simultaneous localization and mapping via square root information smoothing, Int. J. Robot. Res. 25 (12) (2006) 1181–1203.
[10] M. Kaess, A. Ranganathan, F. Dellaert, iSAM: incremental smoothing and mapping, IEEE Trans. Robot. 24 (6) (2008) 1365–1378, https://doi.org/10.1109/TRO.2008.2006706.
[11] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J.J. Leonard, F. Dellaert, iSAM2: incremental smoothing and mapping using the Bayes tree, Int. J. Robot. Res. 31 (2) (2012) 216–235, https://doi.org/10.1177/0278364911430419.
[12] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al., FastSLAM: a factored solution to the simultaneous localization and mapping problem, in: AAAI/IAAI, 2002, pp. 593–598.
[13] T.-C. Dong-Si, A.I. Mourikis, Motion tracking with fixed-lag smoothing: algorithm and consistency analysis, in: Proc. IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2011, pp. 5655–5662.
[14] G. Sibley, L. Matthies, G. Sukhatme, Sliding window filter with application to planetary landing, J. Field Robot. 27 (5) (2010) 587–608.
[15] L. Ljung, System Identification: Theory for the User, Prentice-Hall, 1987.
[16] H.A. Blom, Y. Bar-Shalom, The interacting multiple model algorithm for systems with Markovian switching coefficients, IEEE Trans. Autom. Control 33 (8) (1988) 780–783.