Tracking Multiple Interacting Targets Using a Joint Probabilistic Data Association Filter

Arsène Fansi Tchango*†, Vincent Thomas†, Olivier Buffet†, Alain Dutech† and Fabien Flacher*
*Thales Services SAS Company, Vélizy-Villacoublay, France
Email: firstname.lastname@thalesgroup.com
†INRIA / Université de Lorraine, Nancy, France
Email: firstname.lastname@loria.fr

***Abstract*—In this paper, we describe and evaluate an original Monte Carlo JPDAF for tracking interacting autonomous targets in a cluttered environment. The originality of the proposed algorithm consists in reducing the complexity of the prediction step by selecting and separately updating groups of targets in interaction. The complexity of the correction step is addressed by Data Association and a gating procedure as found in literature. The main assumptions we make in this paper are (i) that the evolution of the state of each target only depends on the states of all the targets at the previous time step and (ii) that a generic simulator or a function modeling the targets' behaviors and their mutual interactions is available. We also build an approximate interaction graph between targets on the fly on the basis of simple information like their location, as it has been done in previous work. Experiments show that representing interactions this way can lead to good tracking efficiency with low computational cost.**

I. INTRODUCTION

In the field of state estimation, the problem of target tracking has been widely studied during the last decades. Classical methods have been developed in the case of a single target. The most popular one is the Kalman filter [1] in which it is assumed that both the dynamical model of the target and the observation model are linear and Gaussian. Studies have also been performed for application areas in which it is required to include non-linear and non-Gaussian elements in order to accurately model the underlying dynamics of the target. Several methods have been proposed to deal with such non-linear models, such as the Extended Kalman filter [2], the Unscented Kalman filter [3], and particle filters [4]. While variants of the Kalman filter aim to approximate the distribution on the system state as a Gaussian density probability, particle filters make no assumptions on the form of the probability densities [4], making particle filters more appropriate, in general, for non-linear and non-Gaussian problems.

When the number of considered targets is greater than one, the problem of tracking is referred to as Multiple Target Tracking (MTT). A naive approach for dealing with the MTT problem is to apply a single filtering method in which the considered state is the combination of the states of all the targets. However, this approach suffers from two main issues which contribute to making the MTT a challenging topic of research: (a) the complexity of the prediction step and (b) the data-association problem in the correction step. The computational complexity of the prediction step comes directly from the increase of dimension of the state space with the number of tracked targets [5]. The data association problem appears when the global observation function is defined by local target observation functions and when the local observations do not contain information about the target they are originated from. In this case, there is a combinatorial explosion in the space of possible multiple target trajectories due to the uncertainty in the association of the received observations with known targets at each time step. This is particularly significant in presence of missing reports (detection probability is less than one) and false reports (from clutter).

Several approaches have been proposed in the literature to deal with this MTT-related data-association problem. Among them, the most notable ones are the Multiple Hypothesis Tracker (MHT) [6] and the Joint Probabilistic Data Association Filter (JPDAF) [7]. The MHT, at each time step, considers all valid data-association hypotheses and maintains a branch for each of them. As more measurements are received, the likelihood of each branch is computed and branches with low probability are eliminated. The MHT is able to deal with a varying number of targets; however its downside resides in its high memory and processing requirements (which grow exponentially with the number of tracks). On the other hand, the JPDAF deals with a fixed number of targets and defines a way of combining all valid data-association hypotheses such that, at each time step, only one global update is needed during the correction step without requiring the memory of previous associations. Moreover, heuristics have been proposed to reduce the number of considered associations like the Gating procedure by considering proximity between target estimates and individual observations. Therefore, it has a lower computational complexity with respect to the MHT and is suitable for online implementation.

In their initial derivations [6] [7], the MHT and JPDAF approaches assume that the dynamical model of each target as well as the observation model are linear and Gaussian; thus the Kalman filter is used for target state estimation. Recently, solutions have been proposed to extend the JPDAF approach with particle filtering techniques for the general case of non-linear and non-Gaussian models [8] [9] [10] referring to the general term of Monte-Carlo JPDAF (MC-JPDAF).

Regarding the complexity of the prediction step, the above described approaches (MHT, JPDAF, MC-JPDAF) are meant to be applied when targets behave independently. In this case, the state of each target can be estimated by a specific distribution and the complexity of the prediction step is reduced by updating independently each distribution with the dynamics of the considered target. However, in some applications, like crowd surveillance for instance, the objective consists exactly

in monitoring interacting targets, so that the targets cannot be considered to behave independently anymore. Instead, their behaviors may be influenced by a set of rules characterizing their world. From a specific target point of view, these rules model the interactions with other targets present in the environment and can be of various kinds: for instance, social rules used to model social contexts but also be physical rules preventing two targets to be in the same exact location.

To consider interactions among targets, Khan et al. [11] assume that their interactions have a specific structure included in the transition model. They represent this application domain knowledge using potential fields via a Markov Random Field (MRF) given constraints between the states of the targets at the same timestep. This way of modeling constraints the authors to consider target interactions only at the correction step, which allow to eliminate particles that are not consistent with the application domain knowledge.

In this paper, we are also concerned with the problem of tracking a fixed number of interacting targets, but we make different assumptions about the dynamical model. While Khan et al. use a specific interaction term which limits the model they can use, we rather assume having a complex dynamical model that integrates interactions and where the state of each target depends only on the states of the other targets at the previous time. This constraint seems more in adequation with classical physical models as Craig Reynold's boids [12]. The approach proposed in this article consists then in reasoning on an interaction graph built on the fly to consider which targets are interacting among each other. This interaction graph is used to build subsets of interacting targets and to compute the prediction step by using K nearly independent filters, in order to approximate the joint targets states. The approach described in this paper is close to the one suggested by Khan et al. [11] in the sense that it reasons on interacting targets, but, in our approach, as the application domain knowledge is fully included within the dynamical model of the targets, the collaboration among filters occurs at the "prediction" step instead of the "correction" step.

II. PROBLEM STATEMENT

Let us consider a closed environment in which several targets evolve. The number K of targets in the environment does not change over time. Let $\mathbf{x}_t^k \in \mathbb{R}^{n_x}$ be the state of the k^{th} target at time t, where n_x represents the dimension of the state variable. The state of the overall system at time t, defined as $\mathbf{x}_t \triangleq \{\mathbf{x}_t^k\}_{k=1}^K$ is the collection of the states of the entities in the system. We consider the general case in which the trajectory followed by the k^{th} target may be influenced by other targets of the environment at each time step. But we assume that the evolution of the target state at the next time step depends solely on all targets states at the previous time step, that is,

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \prod_{k=1}^{K} p(\mathbf{x}_t^k|\mathbf{x}_{t-1}). \tag{1}$$

More precisely, we assume, for each target k, the availability of a generic function that, given any total number of targets, is able to compute (/simulate) the evolution of target k while accounting for the influence of other targets. Let $F_{t,K}^k : \mathbb{R}^{n_x K} \rightarrow \mathbb{R}^{n_x}$, the dedicated discrete-time dynamics of the k^{th} target. Its evolution may then may be expressed as

$$\mathbf{x}_{t+1}^k = F_{t,K}^k(\mathbf{x}_t) + \mathbf{w}_t^k, \tag{2}$$

where $\mathbf{w}_t^k$ is the process noise specific to target k.

The system is equipped with a sensor. It is assumed that each target can be detected with a probability P_D, and, if detected, a noisy observation of its state is provided. Furthermore, the sensor may provide false alarms, and the number of false alarms generated follows a Poisson distribution with a parameter $\lambda_{FT}V$ where λ_{FT} is the false alarm rate per unit time, per unit volume; and V is the volume of $\mathscr{R}$, the region of the environment where the targets may evolve.

Let $\mathbf{z}_t = (\mathbf{z}_t^1, \mathbf{z}_t^2, \cdots, \mathbf{z}_t^{M_t})$ be the observation received from the sensor at time t, where M_t represents the number of atomic observations. $\mathbf{z}_t$ includes both noisy measurements and false alarms. Let $\mathbf{z}_t^j \in \mathbb{R}^{n_z}$ be the j-th observation at time t for $j = 1, \cdots, M_t$, where n_z is the dimension of each observation data. It is assumed that each target generates a unique observation at each time step if it is detected. Also, an observation is originated from at most one target. Let $H : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_z}$ be the observation function for one target. Then, observations are generated according to:

$$\mathbf{z}_t^j = \begin{cases} H(\mathbf{x}_t^k) + \mathbf{v}_t^j & \text{if } \exists k \text{ s.t. the } j\text{-th observation is} \\ & \text{from the } k^{th} \text{ target,} \\ \mathbf{u}_t & \text{otherwise,} \end{cases} \tag{3}$$

where $\mathbf{v}_t^j$ is a zero-mean Gaussian noise with covariance matrix Q_{v_j}, $\mathbf{u}_t \sim Unif(\mathscr{R})$ is a uniform random process for false alarms.

The multi-target tracking problem consists in estimating the target states $\mathbf{x}_t^k$ for $k = 1, \cdots, K$, from the sequence of observations received so far.

III. MONTE CARLO JPDAF

In this section, we present the Monte Carlo JPDAF (MC-JPDAF) as it is applicable to the general case of non-linear and non-Gaussian models. As stated before, the MC-JPDAF uses common particle filter techniques to approximate the posterior density function $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ of the target states given the observations. However, previous works using JPDAF usually assume a total independency among targets. This is translated by modifying Equation 2 as follows:

$$\mathbf{x}_{t+1}^k = F_{t,1}^k(\mathbf{x}_t^k) + \mathbf{w}_t^k, \forall k = 1, \cdots, K, \tag{4}$$

where $\mathbf{w}_t^k$ is defined as previously.

In this section, we rely on Equation 4 to represent the dynamics of each target (interactions will be handled in Section IV). The main idea of the JPDAF algorithm is to recursively update the marginal filtering distribution for each target $p(\mathbf{x}_t^k|\mathbf{z}_{1:t})$ for $k = 1, \cdots, K$ instead of computing the joint filtering distribution $p(\mathbf{x}_t|\mathbf{z}_{1:t})$. The computation of these distributions is performed through the Bayesian filtering framework. In the MC-JPDAF paradigm, the particle filter is used to approximate the Bayesian estimation of the target state. Moreover, because of the uncertainty in the observation data regarding the target they are originated from, the computation

of these marginal distributions cannot be performed independently. Indeed, before computing each marginal distribution, a procedure is needed to resolve the data association problem, i.e., assign each target to its associated observation data. In the JPDAF framework, this procedure is referred to as Data Association and is the key feature of the JPDAF algorithm.

Thus the MC-JPDAF algorithm proceeds, at each time step, by (1) resolving the data association problem in the correction step, and then (2) using the particle filter techniques to recursively update the target marginal distribution. In what follows, we describe these two steps in more detail.

A. Particle Filter

Let us consider the problem of computing the posterior probability density function (pdf) $p(\mathbf{x}_t^k|\mathbf{z}_{1:t})$ for the k^{th} target. The Bayesian filtering framework [13] provides a recursive way of computing such a pdf assuming the availability of the prior belief $p(\mathbf{x}_0^k)$ regarding the target initial state at time step $t=0$. The Bayesian filter mainly proceeds through two steps:

the prediction step : the pdf at previous time step is modified according to the target's dynamics $p(\mathbf{x}_t^k|\mathbf{x}_{t-1}^k)$:

$$p(\mathbf{x}_t^k|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t^k|\mathbf{x}_{t-1}^k)p(\mathbf{x}_{t-1}^k|\mathbf{z}_{1:t-1})d\mathbf{x}_{t-1}^k; \quad (5)$$

the correction step : the pdf obtained at the prediction step is corrected using the received observation $\mathbf{z}_t$:

$$p(\mathbf{x}_t^k|\mathbf{z}_{1:t}) \propto p(\mathbf{z}_t|\mathbf{x}_t^k)p(\mathbf{x}_t^k|\mathbf{z}_{1:t-1}). \quad (6)$$

A particle filter (PF) [4] is an approximate Bayesian filter. At time t, a PF represents the pdf $p(\mathbf{x}_t^k|\mathbf{z}_{1:t})$ by a set of N_s weighted particles $\mathscr{S}_t^k = \{\mathbf{x}_t^{k,i}, w_t^{k,i}\}_{i=1}^{N_s}$ where $\mathbf{x}_t^{k,i}$ and $w_t^{k,i}$ are respectively the state and the weight of the i^{th} particle of the k^{th} target. The particle set $\mathscr{S}_t^k$ is typically constructed from the previous set $\mathscr{S}_{t-1}^k$ and the current observation $\mathbf{z}_t$ as follows:

1) **Prediction:** A sample $\mathbf{x}_{t|t-1}^{k,i}$ is generated from each sample $\mathbf{x}_{t-1}^{k,i}$ of the set $\mathscr{S}_{t-1}^k$ using a proposal density function $q(\mathbf{x}_t^k|\mathbf{x}_{t-1}^k, \mathbf{z}_t)$. Usually, $q(\mathbf{x}_t^k|\mathbf{x}_{t-1}^k, \mathbf{z}_t) = p(\mathbf{x}_t^k|\mathbf{x}_{t-1}^k)$.
2) **Weight Assignment:** Each propagated sample $\mathbf{x}_{t|t-1}^{k,i}$ is assigned an importance weight $w_t^{k,i}$ computed with

$$w_t^{k,i} = w_{t-1}^{k,i} \cdot \frac{p(\mathbf{z}_t|\mathbf{x}_{t|t-1}^{k,i}).p(\mathbf{x}_{t|t-1}^{k,i}|\mathbf{x}_{t-1}^{k,i})}{q(\mathbf{x}_{t|t-1}^{k,i}|\mathbf{x}_{t-1}^{k,i}, \mathbf{z}_t)}. \quad (7)$$

Once computed, the importance weights are normalized.
3) **Resampling:** This step consists in duplicating particles with high importance weights and suppressing those with low importance weights. At the end of the step, each resulting particle $\mathbf{x}_t^{k,j}$ is assigned an importance weight $w_t^{k,j} = 1/N_s$.

Furthermore, to encourage the state space exploration, the resampling phase may not be performed at every time step, but only when the effective size $\widehat{N}_{eff}$ of the filter goes below a given threshold N_T where

$$\widehat{N}_{eff} = \frac{1}{\sum_{i=1}^{N_s}(w_t^{k,i})^2}. \quad (8)$$

The one-time-step PF algorithm is presented in Algorithm 1.

Algorithm 1: Particle Filter

$[\{\mathbf{x}_t^{k,i}, w_t^{k,i}\}_{i=1}^{N_s}] = \text{PF } [\{\mathbf{x}_{t-1}^{k,i}, w_{t-1}^{k,i}\}_{i=1}^{N_s}, \mathbf{z}_t]$
for $i = 1 : N_s$ **do**
  sample $\mathbf{x}_{t|t-1}^{k,i}$ from $q(\mathbf{x}_t^k|\mathbf{x}_{t-1}^{k,i}, \mathbf{z}_t)$
  compute $w_t^{k,i}$ using Equation 7
normalize the importance weights: $w_t^{k,i} = \frac{w_t^{k,i}}{\sum_{i=1}^{N_s} w_t^{k,i}}$
compute $\widehat{N}_{eff}$ according to Equation 8
if $\widehat{N}_{eff} < N_T$ **then**
  $[\{\mathbf{x}_t^{k,j}, w_t^{k,j}\}_{j=1}^{N_s}] = \text{RESAMPLE } [\{\mathbf{x}_{t|t-1}^{k,i}, w_t^{k,i}\}_{i=1}^{N_s}]$
else
  for $i = 1, \cdots, N_s$ **do** assign $\mathbf{x}_t^{k,i} = \mathbf{x}_{t|t-1}^{k,i}$

B. Data Association Problem

According to Equation 7, the particle weight strongly depends on the likelihood $p(\mathbf{z}_t|\mathbf{x}_{t|t-1}^{k,i})$ of the received observation with respect to the state represented by the particle (We recall that $\mathbf{z}_t = (\mathbf{z}_t^1, \mathbf{z}_t^2, \cdots, \mathbf{z}_t^{M_t})$.). The question that arises is how to compute this likelihood function since an association has to be made between the observation data and the targets. The JPDAF deals with this issue as described below.

Let β_{jk}, $\forall j = 0, \cdots, M_t$, $\forall k = 1, \cdots, K$ be the probability of the k^{th} target being associated with the j^{th} observation data z_t^j at time t. The observation data z_t^0 is used to model the situations in which a given target has not been detected. Therefore, the likelihood of the observation $\mathbf{z}_t$ with respect to the k^{th} target is defined as

$$p(\mathbf{z}_t|\mathbf{x}_t^k) = \sum_{j=0}^{M_t} \beta_{jk}.p(\mathbf{z}_t^j|\mathbf{x}_t^k). \quad (9)$$

All that remains is to compute the β_{jk} probabilities.

Let an hypothesis θ be a set of pairs $(j,k) \in \{0, \cdots, M_t\} \times \{1, \cdots, K\}$. $(j,k) \in \theta$ means that the k^{th} target is associated with the j^{th} observation within hypothesis θ. θ is a feasible joint association event if it uniquely determines which observation is assigned to which target. Let Θ be the set of all feasible joint association events. Let us also consider Θ_{jk}, the set of all feasible joint association events in which observation k is assigned to target j, $\Theta_{jk} \subset \Theta$ such that $\Theta_{jk} = \{\theta : (j,k) \in \theta\}$. Then we have

$$\beta_{jk} = p(\Theta_{jk}|\mathbf{z}_{1:t}) = \sum_{\theta:(j,k)\in\theta} p(\theta|\mathbf{z}_{1:t}). \quad (10)$$

Given a feasible joint association event θ, one can easily compute N_{DT}, the number of observation data that are associated to a target and therefore $N_{FT} = M_t - N_{DT}$ the number of false alarms. Moreover, it is possible to determine the identity l_j of the target associated to the j^{th} observation data. We have $l_j = k$ if $(j,k) \in \theta$, and $l_j = 0$ otherwise.

As discussed by Vermaak et al. [10], the posterior probability $p(\theta|\mathbf{z}_{1:t})$ of the joint association event θ can be expressed as

$$p(\theta|\mathbf{z}_{1:t}) \propto \lambda_{FT}^{N_{FT}} (1-P_D)^{K-N_{DT}} P_D^{N_{DT}} \prod_{j:l_j \neq 0} p^{l_j}(\mathbf{z}_t^j|\mathbf{z}_{1:t-1}). \quad (11)$$

with N_{FT} and N_{DT} depending on θ, $j : l_j \neq 0$ denoting the pairs in θ where the associated observation is not a false alarm, and λ_{FT} and P_D coming from the observation model.

The term $p^k(\mathbf{z}_t^j|\mathbf{z}_{1:t-1})$ represents the predictive likelihood of the j^{th} observation using the information from the k-th target, and is expressed as

$$p^k(\mathbf{z}_t^j|\mathbf{z}_{1:t-1}) = \int p(\mathbf{z}_t^j|\mathbf{x}_t^k) p(\mathbf{x}_t^k|\mathbf{z}_{1:t-1}) d\mathbf{x}_t^k. \quad (12)$$

In the particle filter paradigm, Equation 12 is approximated using the propagated samples or particles. That is

$$p^k(\mathbf{z}_t^j|\mathbf{z}_{1:t-1}) = \sum_{i=1}^{N} \alpha_t^{k,i} p(\mathbf{z}_t^j|\mathbf{x}_{t|t-1}^{k,i}), \quad (13)$$

where the predictive weights $\alpha_t^{k,i}$ are given by:

$$\alpha_t^{k,i} \propto w_{t-1}^{k,i} \frac{p(\mathbf{x}_{t|t-1}^{k,i}|\mathbf{x}_{t-1}^{k,i})}{q(\mathbf{x}_{t|t-1}^{k,i}|\mathbf{x}_{t-1}^{k,i}, \mathbf{z}_t)}, \qquad \sum_{i=1}^{N} \alpha_t^{k,i} = 1. \quad (14)$$

Also, we have

$$p(\mathbf{z}_t^j|\mathbf{x}_t^k) = \begin{cases} 1 & \text{if } j = 0, \\ \mathcal{N}_{0;Q_{v_j}}(\mathbf{z}_t^j - H(\mathbf{x}_t^k)) & \text{if not,} \end{cases} \quad (15)$$

where $\mathcal{N}_{0;Q_{v_j}}(.)$ is a function computing probability values from the zero-mean Gaussian pdf with covariance matrix Q_{v_j}.

Finally, using Equations 15, 13, 11 and 10, Equation 9 can be computed and be integrated to the PF algorithm.

C. Reducing the Number of Feasible Association Events

One limitation of the JPDAF algorithm is the necessity of enumerating all the feasible joint association events or hypotheses. In order to reduce the number of these hypotheses, Bar-Shalom *et al.* [14] introduced a technique named "Gating" allowing to associate to each target a validation region. Therefore, only observations falling within the target validation region are considered as possible candidates. This procedure allows to decrease the computational cost of the tracking system by reducing the number of considered association events. An illustration of the procedure is shown in Fig. 1. In this figure, the blue circles represent the different targets while the green squares are the observation data. The dashed ellipse around each target represents its validation region. As illustrated, only measurements Z_2 and Z_3 will be considered for target T_1 while no measurement has to be considered for the target T_4.

We are not presenting in this paper the detail of this procedure. However, readers are encouraged to refer to [15] for the application of this technique in the case of Monte-Carlo JPDAF. The one step MC-JPDAF is reported in Algorithm 2.

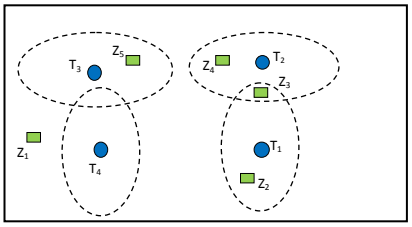

Fig. 1. Example of the Gating procedure: the blue circles represent targets and the green squares represent measurements. The validation region of each target is represented by the ellipse centered around it and only measurements falling in this region are considered as feasible for the target.

Algorithm 2: Monte Carlo JPDAF

$[\{\{\mathbf{x}_t^{k,i}, w_t^{k,i}\}_{i=1}^{N_s}\}_{k=1}^{K}]$ =MC-JPDAF$[\{\{\mathbf{x}_{t-1}^{k,i}, w_{t-1}^{k,i}\}_{i=1}^{N_s}\}_{k=1}^{K}, \mathbf{z}_t]$

for $k = 1, \cdots, K$ **do**
  for $i = 1, \cdots, N_s$ **do**
    sample $\mathbf{x}_{t|t-1}^{k,i}$ from $q(\mathbf{x}_t^k|\mathbf{x}_{t-1}^{k,i}, \mathbf{z}_t)$
  compute the measurement association possibilities (see section III-C)
compute the set Θ of feasible joint association events
for $k = 1, \cdots, K$ **do**
  for $j = 0, \cdots, M_t$ **do**
    compute β_{jk} using 10
  for $i = 1, \cdots, N_s$ **do**
    compute $w_t^{k,i}$ using Equations 9 and 7
  normalize the importance weights: $w_t^{k,i} = \frac{w_t^{k,i}}{\sum_{i=1}^{N_s} w_t^{k,i}}$
  compute the effective size $\widehat{N}_{eff}^k$ using Equation 8
  if $\widehat{N}_{eff}^k < N_T$ **then**
    $[\{\mathbf{x}_t^{k,j}, w_t^{k,j}\}_{j=1}^{N_s}]$ = RESAMPLE $[\{\mathbf{x}_{t|t-1}^{k,i}, w_t^{k,i}\}_{i=1}^{N_s}]$
  else
    for $i = 1, \cdots, N_s$ **do** assign $\mathbf{x}_t^{k,i} = \mathbf{x}_{t|t-1}^{k,i}$

IV. Tracking Interacting Targets

The previous section assumed that all the targets are independent. In this section, we consider the case where targets may interact with each other. We first present how Khan *et al.* deal with interactions in their work, then our contribution.

A. MRF based Approach

Khan *et al.* [11] focus on the special case where the next state of a target does not directly depend on the previous other targets' states but on the other targets' states of the *same* time-step. This assumption leads them to propose to model interactions through a special structure of the transition function. In this approach, the probabilistic transition model can be decomposed into two products:

$$P(X_t|X_{t-1}) \propto \prod_i P(X_{it}|X_{i(t-1)}) \prod_{ij \in E} \psi(X_{it}, X_{jt}),$$

where $\prod_i P(X_{it}|X_{i(t-1)})$ considers that all targets behave independently and where the second product $\prod_{ij\in E}\psi(X_{it},X_{jt})$ corresponds to interaction potentials to represent the physical constraint that targets cannot overlap. Those potentials (and thus interactions) are computed using a Markov Random Fields(MRF).

An MRF is a graph (V,E) with undirected edges between nodes where nodes represent targets, and an edge between node k and node l symbolizes the fact that there is a local interaction between targets l and k. Interaction potentials are expressed by probabilities associated to each edge and modeled with a Gibbs distribution:

$$\psi(X_{it},X_{jt}) \propto exp(-g(X_{it},X_{jt})),$$

where g is a penalty function modeling the domain application, in this case the number of pixels of overlapping targets.

On this basis, it can be proved that the computation of the interaction can be done in the correction step and the update can be done as if targets were independent.

B. Interaction Graph based Approach

Through MRFs, Khan *et al.* assume that targets interact locally and model interactions as pairwise, reciprocal and instantaneous. We share the same locality assumption, but propose using interactions in a more frequently encountered form, where an agent reacts to its neighbors' previous states. This allows to directly use existing models like Craig Reynold's steering behaviors and numerous related models.

Our approach implies considering each target with its neighbors in the interaction graph. Formally, an interaction graph is a graph (V,E) where nodes represent targets, and an edge between node k and node l symbolizes a local interaction between targets k and l as presented in Fig. 2. In the worst case, the graph is complete and the evolution of each target depends on all the other targets, while, in the simplest case, there is no edge in the graph and the evolution of each target ignores the presence of other targets.

Let I_k be the set of indices of targets in the neighborhood of target k. Let $\mathbf{x}_{t-1}^{I_k} = \{\mathbf{x}_{t-1}^{j}\}_{j\in I_k}$, $\phi_{t-1}^{k} = \{\mathbf{x}_{t-1}^{k}, \mathbf{x}_{t-1}^{I_k}\}$, and $L = |I_k|+1$. Then, we have the following equality:

$$F_{t,K}^{k}(\mathbf{x}_{t-1}) = F_{t,L}^{k}(\phi_{t-1}^{k}). \tag{16}$$

Equation 16 stipulates that, in order to correctly determine the future behavior of a given target k, it is necessary to simulate only targets belonging to ϕ_{t-1}^{k} instead of simulating all the targets. In terms of probability, this translates to

$$p(\mathbf{x}_t^k|\mathbf{x}_{t-1}) = p(\mathbf{x}_t^k|\mathbf{x}_{t-1}^k, \mathbf{x}_{t-1}^{I_k}). \tag{17}$$

The algorithm we describe in this paper relies on the probability formulation of Equation 17.

C. Interaction Graph Building and Particle Updates

Because the targets are moving, the interaction graph evolves with time and must be built at each time step. A difficulty is that we have only access to distributions over targets' states and not the real hidden states. Our approach consists of using a "representative" for each target's distribution, which could be the particle with the largest weight, the mean of a targets' distribution... The interaction graph is built by adding edges between targets whose representatives are within a given distance from each other.

Once the interaction graph has been built, this graph is used for updating particles. For each target k, its corresponding distribution is updated by considering that the targets with which it interacts can be replaced by their associated representatives. This is a gross, but low-cost, approximation: the evolution of each particle involves only a single simulation containing this particle and the $|I_k|$ representatives of its neighbors (in the interaction graph) instead of $N_s^{|I_k|}$ simulations.

More formally, assuming that, at time $t-1$, the pdf of the k^{th} target is represented by the set $\{\mathbf{x}_{t-1}^{k,i}, w_{t-1}^{k,i}\}_{i=1}^{N_s}$, we compute, for each target k, its representative $\hat{\mathbf{x}}_{t-1}^{k}$. Then, based on the obtained representatives $\{\hat{\mathbf{x}}_{t-1}^{k}\}_{k=1}^{K}$, we build on the fly an interaction graph describing whether or not a given target is likely to locally interact with another target. The next step consists in making the particle of a given target k evolve. To update its associated distribution, we consider the set I_k of target k's neighbors, and its prediction through the system model is done according to the proposal distribution

$$q(\mathbf{x}_t^k|\mathbf{x}_{t-1}^k, \mathbf{z}_t) = p(\mathbf{x}_t^k|\mathbf{x}_{t-1}^k, \hat{\mathbf{x}}_{t-1}^{I_k}), \tag{18}$$

where $\hat{\mathbf{x}}_{t-1}^{I_k} = \{\hat{\mathbf{x}}_{t-1}^{j}\}_{j\in I_k}$.

In this work, we use the particle mean approach for computing the target representative. Thus, for the k^{th} target at time $t-1$, we have $\hat{\mathbf{x}}_{t-1}^{k} = \sum_{i=1}^{N_s} w_{t-1}^{k,i}\mathbf{x}_{t-1}^{k,i}$. The use of particle mean was also used by Khan *et al.* in [16], but for a different purpose: to compute the MRF factors. A drawback of Khan *et al.*'s method with respect to our approach is the fact that, by considering the interaction posteriorly (correction step), time may be wasted on particles that are not viable (at least from the domain application point of view). We embed this process within the MC-JPDAF leading to an algorithm whose one step description is depicted in Algorithm 3.

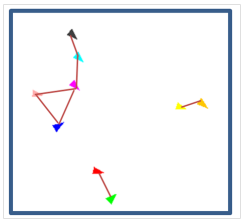

Fig. 2. An example of an Interaction Graph: The triangles represent targets being tracked. Only targets connected by an edge can locally interact. The behavior of a target is then affected by the behavior of other targets belonging to the same connected component in the graph. Targets from different connected components do not influence each other.

V. Experimental Results

We consider, as a basis of the experiments in this paper, the steering behavior model introduced by Reynolds in [12] for autonomous characters. The steering behavior model, in its genericity, describes a set of simple behaviors (represented as physical forces) that, combined together, allow an autonomous character to exhibit realistic high-level navigational behaviors such as obstacle avoidance, collision prevention, group formation, and so on, in games or in multi-agent computer simulations.

For the experiments, we used the 2-D java implementation proposed in [17] as our baseline simulator. Through this tool,

Algorithm 3: Interacting Targets MC-JPDAF

1 $[\{\{\mathbf{x}_t^{k,i}, w_t^{k,i}\}_{i=1}^{N_s}\}_{k=1}^{K}]$ = ITMC-JPDAF $[\{\{\mathbf{x}_{t-1}^{k,i}, w_{t-1}^{k,i}\}_{i=1}^{N_s}\}_{k=1}^{K}, \mathbf{z}_t]$
2 compute $\hat{\mathbf{x}}_{t-1}^{k}$, for $k = 1, \cdots, K$
3 build the Interaction Graph using $\{\hat{\mathbf{x}}_{t-1}^{k}\}_{k=1}^{K}$
4 **for** $k = 1, \cdots, K$ **do**
5 get the set I_k of targets from the Interaction Graph
6 **for** $i = 1, \cdots, N_s$ **do**
7 sample $\mathbf{x}_{t|t-1}^{k,i}$ from $q(\mathbf{x}_t^k | \mathbf{x}_{t-1}^{k,i}, \mathbf{z}_t)$ (Equation 18)
8 compute the measurement association possibilities (see Section III-C)
9 proceed as in MC-JPDAF (see Alg. 2), lines 6–17

we modeled target's dynamical behavior consisting in moving randomly within a closed arena while avoiding each other and collisions with obstacles. This is done by combining, within an agent (target), the following simple behaviors:

containment behavior: it helps an agent choose its path so as to avoid obstacles in its front as well as in its sides;
separation behavior: it makes agents keep a certain distance to each other;
wandering behavior: it allows agents to move randomly and without a specified goal through the scene.

The state of the k^{th} target in the 2-D plane comprises its location and velocity:

$$\mathbf{x}_t^k = [x_{k,t}, \dot{x}_{k,t}, y_{k,t}, \dot{y}_{k,t}].$$

The targets are physically represented as 1 cm side equilateral triangles. We also consider a polygonal arena inscribed in a rectangle of width 37.5cm and height 30cm (see Fig. 3). The system model (target behavior) is fully encoded within the simulator.

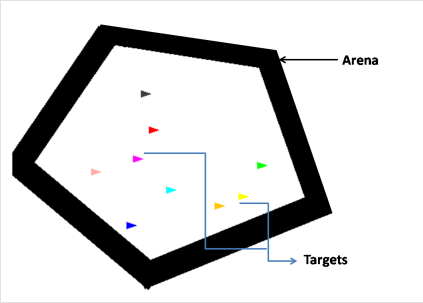

Fig. 3. The experimental scene: The considered arena is a pentagon. The colored triangles represent targets that should be tracked. The colors are just used for visual distinction and are **not** taken into account within the algorithm to facilitate data association.

The observation model function H is defined as:

$$H(\mathbf{x}_t^k) = B\mathbf{x}_t^k,$$

where B is the observation matrix defined as

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The zero-mean observation Gaussian noise is characterized by a diagonal covariance matrix with a standard deviation of 0.5 cm for both the x and y coordinates. Futhermore, the false alarm rate λ_{FT} is set to 0.8 while the detection probability $P_D = 0.95$. Finally, the radius for the gating procedure has been set to 4 cm. Observations are performed every time-step.

We consider scenarios involving 9 targets. For each scenario, each target is associated with $N_s = 500$ particles and the scenario is run for 500 time-steps. At the beginning of each experiment, each filter is associated to a specific target and knows its initial position.

Because we do not properly know which metrics the simulator used to determine how targets interact between each other, we consider sequentially several rules to build the interaction graph. Each rule is described by a specific tolerance in mm and according to this rule, 2 targets are considered to interact if their distance is lower than this tolerance. Thus, *Rule0* considers that targets are never in interaction and behave independently. On the contrary, *Rule200* considers that targets almost always interact since this distance (20cm) is close to the arena size. Intermediary rules have also been considered, i.e., *Rule10*, *Rule20* and *Rule50* to investigate the efficiency of the proposed approach.

For each rule, we run each scenario 20 times and we report the averages obtained. The algorithm was coded in the java language and the tests were performed on a 2.67 GHz Intel Core i5 CPU machine running under Windows.

The graphical results are reported in Figure 4. The plotted measures correspond to the average sum of the distance between targets and the mean of the particles filters. Two measures are considered: the one (a) where each distance is computed between each target and its initially associated filter and the one (b) where each distance is computed between each target and its closest filter. The latest measure (b) is a way to tolerate trackJumps in the efficiency measurement of the joint filter: since targets share the same behavior, once a target is lost, it would be not fair to expect the joint filter to re-associate this target to its originally associated filter.

It must be noted (see Figure 4) that *Rule200* and *Rule50* give similar results as confirmed by their standard deviation. The same result can be observed for *Rule0* and *Rule10*, meaning that considering an interacting distance of 10mm is too short to predict the correct behavior of interacting targets in our scenario. The standard deviations presented in Figure 5 show that *Rule10*, *Rule20* and *Rule50* exhibit significant differences. *Rule50* gives the best results in term of accuracy and corresponds to a rule where targets are almost always considered as being in interaction. Since *Rule0* considers that targets behave independently, the difference between results of *Rule200* and *Rule0* can be interpreted as the gain of efficiency of reasoning on interactions on the joint targets state. The *Rule20* is representative of our approach because targets are not considered to always interact but the filter has a better accuracy than filter with the independent targets behaviors assumption. Few obtained trajectories are presented in Figure 6 to give an insight of the conducted experiments.

Each run was also evaluated according to the following criteria:

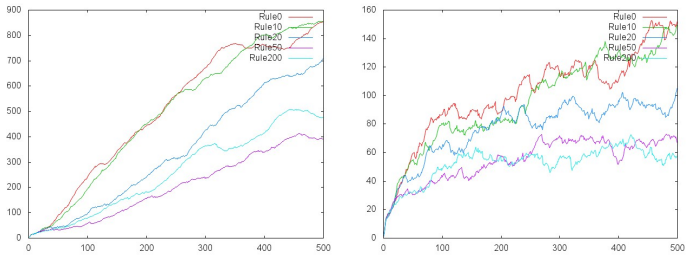

Fig. 4. Means of the sums of the distances (in mm) (a) between targets and their initially associated filters and (b) between targets and their closest filter. This sum is done at each time-step and is averaged over the experiments done for each considered rule.

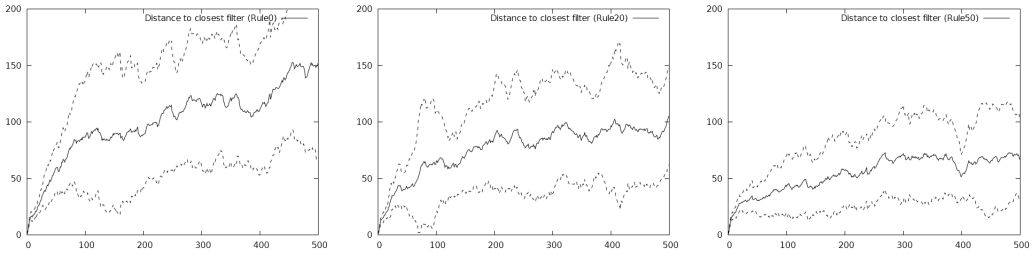

Fig. 5. Comparison of the sum of distances between each target and its closest filter (a) with *Rule0* (b) *Rule20* and (c) *Rule50*. Each figure presents the means of this sum and its standard deviation.

correct tracks : the averaged number of filters that are correctly representing their initial target. The correctness of a track is computed with respect to the corresponding target real data and an arbitrary 4mm drift, linked to the standard deviation of the observation function, is tolerated;

track jumps : The averaged number of filters referring to a target which is different to the initial one. The track jumping phenomenon generally happens when two targets pass very close to one another, so that the track in question shifts, in terms of correspondence, from one target to the other one. A trackJump is detected whenever the target is close with a 4mm drift to a filter meant to represent another target;

lost tracks : the averaged number of filters that do not correspond to any target in the scene. A lost track may occur principally because the detection probability that is lower than 1 and because the algorithm uses a gating procedure that considers an observation valid if and only if it falls within a validation region of at least one target.

run time : the total time to process all the 500 frames.

The numerical results are reported in Table I. We note that the average of runtime increases with the distance threshold used to build the interaction graph. When *Rule0*, *Rule10*, *Rule20* or *Rule30* are considered, the average computational time is close to 100 sec whereas there is a huge difference with *Rule200* where the computational time rises to almost 350 sec. The benefit of rules with low threshold distance is the consequence of considering less target representatives to compute the update step: the number of launched simulations is the same but simulations with low distance thresholds involve less targets than simulations with a high threshold. On the contrary, the number of lost tracks of the filter increases when this distance decreases. This means the filter is less and less accurate because it does not consider interactions between interacting targets. Filters *Rule20* and *Rule30* seem to be good compromises since their efficiency is close to the filter *Rule200* which considers that target always interact but their computational time is equivalent to the computational time of filter with independent targets assumption.

One shoud expect a direct comparison of our approach with respect to the MRF based one. Inspite the fact Khan et al [16] provide numerical results to refer to, they did not use MC-JPDAF in their work. Moreover, an explicit formulation of the penalty function $g(.,.)$ is not provided preventing us to encode the approach into MC-JPDAF. Further, While in our approach increasing the interaction distance leads to considering more target representatives in a particle evolution, in the MRF approach, it will consists in penalizing more and more distant particles. For these reasons, we do not linger to a quantitative comparison of both approaches.

VI. Conclusions

We examined the problem of tracking a fixed number of interacting autonomous targets in a cluttered environment under the assumptions that the behavior model of each target as well as the interaction model between targets is fully available (usually encoded within a simulator). We described a Monte Carlo JPDAF algorithm with nearly independent trackers in which local interactions between targets (as suggested by an interaction graph built on the fly) are taken into account when updating the track regarding a given target.

While this method allows to have significant results at lower cost with respect to a joint tracker approach (which suffers from exponential complexity), we show that attention has to be paid when designing criteria used to build the Interaction Graph. Interaction Graphs underestimating local interaction potentials (as implemented within the simulator) will yield poor tracking results, while Interaction Graphs overestimating these interaction potentials will result in an algorithm wasting time in simulating unnecessary situations in the sense that they would not improve the quality of the

	time=300			time=500			for 500 frames
Rule Distance (in mm)	Correct Tracks	Track Jumps	Lost Track	Correct Tracks	Track Jumps	Lost Track	Time (sec)
0	3.6 (±2.310)	3.8 (±1.749)	1.6 (±0.969)	1.85(±1.824)	4.9 (±1.700)	2.25(±1.299)	96.2(±5.249)
10	3.45(±1.745)	4.0 (±1.549)	1.55(±0.668)	2.0 (±1.224)	4.9 (±1.044)	2.1 (±0.830)	96.9(±2.624)
20	4.85(±2.080)	3.05(±1.856)	1.1 (±0.888)	3.3 (±1.615)	4.35(±1.388)	1.35(±0.909)	96.5(±3.734)
30	5.8 (±2.638)	2.45(±2.155)	0.75(±0.829)	5.1 (±2.447)	3.25(±2.299)	0.65(±0.653)	95.4(±3.023)
50	6.7 (±2.260)	1.45(±1.774)	0.85(±0.909)	5.4 (±2.034)	2.8 (±1.777)	0.8 (±0.678)	129.9(±7.258)
200	6.3 (±2.347)	2.1 (±1.997)	0.6 (±0.734)	5.15(±2.286)	3.3 (±1.977)	0.55(±0.804)	348.4(±15.376)

TABLE I. NUMERICAL RESULTS OF THE IMPLEMENTED ALGORITHM ON DEFINED SCENARIOS. EACH COLUMN CONTAINS MEAN VALUE AND STANDARD DEVIATION OVER ALL CONSIDERED EXPERIMENTS.

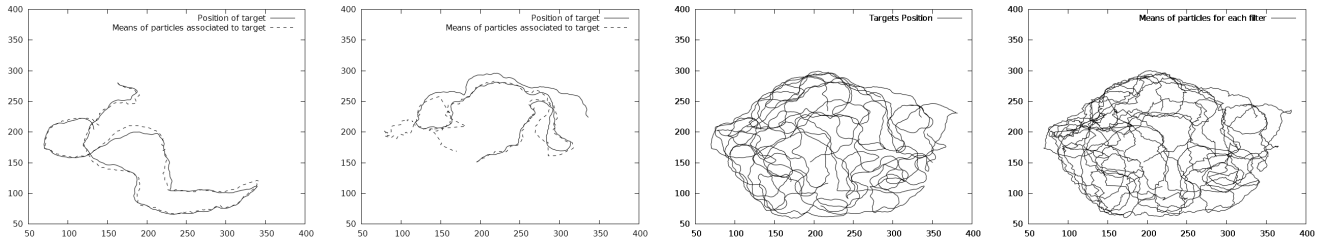

Fig. 6. Trajectories of targets and their associated filters for a *Rule20* experiment. (a), (b) presents trajectories of two randomly chosen targets, (c) and (d) present all the trajectories for all targets (c) and all means of filter (d). These two latest shapes illustrate that targets are indeed evolving in a small space and can frequently interact

tracking process. Nevertheless, this article has shown that, for this specific problem, it is possible to find a compromise between the computation time and the efficiency of the filter. Work should be undertaken to pursue in this direction.

This approach has currently several limitations. First, it is based on the means of the particles of the filters to build the interaction graph, but the mean is a very simple information and may not be always relevant —when the dynamics have strong non-linearities, i.e., when a target has to choose between two paths— nor available —when the states of the targets cannot easily be quantified—. We would like to investigate in the future how to build better representatives of the distributions estimating each targets. Based on the shape of the distributions, the objective would be to represent a distribution with the help of several representatives in order to be more accurate but without having to suffer too much from the curse of dimensionality. The second limitation corresponds to the way the interaction graph is generated, i.e., based on a simple heuristic and on elementary information. More work has to be done on how to build this interaction graph when using a multi-agent simulation to represent the dynamics of a real phenomenon like crowd behavior. A lot of work has been undertaken to represent groups of interacting agents in multi-agent systems, and one promising perspective would be to investigate how it is possible to take advantage of a specific interaction formalism in order to build on the fly an interaction graph consistent with the considered simulations.

REFERENCES

[1] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Trans. of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[2] A. Jazwinski, *Stochastic Processes and Filtering Theory*. Academic Press, 1970.

[3] S. J. Julier and J. K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *Proc. of AeroSense: The 11th Int. Symp. on Aerospace/Defence Sensing, Simulation and Controls*, 1997, pp. 182–193.

[4] M. S. Arulampalam, S. Maskell, and N. Gordon, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, 2002.

[5] J. Liu, C. M., and R. J.E., "Multi-target Tracking in distributed sensor networks," *Signal Processing Magazine, IEEE*, vol. 24, no. 3, pp. 36–46, 2007.

[6] D. B. Reid, "An algorithm for tracking multiple targets," *IEEE Trans. Automat. Contr.*, vol. 24, pp. 843–854, 1979.

[7] T. Fortmann, Y. Bar-Shalom, and M. Scheffé, "Sonar tracking of multiple targets using joint probabilistic data association," *IEEE J. Oceanic Eng.*, vol. 8, July 1983.

[8] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers, "People tracking with a mobile robot using sample-based joint probabilistic data association filters," *International Journal of Robotics Research (IJRR)*, vol. 22, no. 2, p. 99–116, 2003.

[9] O. Frank, J. Nieto, J. Guivant, and S. Scheding, "Multiple target tracking using sequential Monte Carlo methods and statistical data association," in *Proc. of the IEEE / RSJ International Conference on Intelligent Robots and Systems*, 2003.

[10] J. Vermaak, S. J. Godsill, and P. P. Érez, "Monte Carlo filtering for multi-target tracking and data association," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 41, pp. 309–332, 2005.

[11] Z. Khan, T. Balch, and F. Dellaert, "MCMC-based particle filtering for tracking a variable number of interacting targets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, 2005.

[12] C. Reynolds, "Steering behaviors for autonomous characters," in *Game Developers Conference*, 1999, pp. 763–782.

[13] J. Diard, P. Bessière, and E. Mazer, "A survey of probabilistic models, using the Bayesian programming methodology as a unifying framework," in *Proc. of the 2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS)*, 2003.

[14] Y. Bar-Shalom and T. E. Fortmann, *Tracking and data association*, ser. Mathematics in Science and Engineering. San Diego, CA, USA: Academic Press Professional, Inc., 1987, vol. 179.

[15] A. G. Daronkolaei, S. Shiry, and M. B. Menhaj, "Multiple target tracking for mobile robots using the JPDAF algorithm," in *Proc. of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2007, pp. 137–145.

[16] Z. Khan, T. Balch, and F. Dellaert, "Efficient particle filter-based tracking of multiple interacting targets using an MRF-based motion model," in *IEEE Proc. IROS*. IEEE, 2003, pp. 254–259.

[17] S. Christian and F. Thomas, "The Steering Behaviour project," 12 2007. [Online]. Available: http://www.steeringbehaviors.de/