2007－CVIM－157 (30)
2007／1／12

Multiple-Tracking of Crowds Based on a Leader/Follower Strategy and the Dropping-Pebbles Method

J. Vergés-Llahí*, T. Wada†*, S. Nishio*,
N. Babaguchi‡*, N. Hagita*

* ATR Intelligent Robotics and Communication Laboratories, Kyoto 619-0288, Japan
Email: {jverges, nishio, hagita}@atr.jp
† Dept. of Computer & Comm., Wakayama University, Wakayama 640-8510, Japan
Email: babaguchi@comm.eng.osaka-u.ac.jp
‡ Grad. School of Engineering, Osaka University, Osaka 565-0871, Japan
Email: babaguchi@comm.eng.osaka-u.ac.jp

Abstract

Recent approaches to tracking a certain number of interacting targets have suggested the use of multiple-target tracking provided by particle filters which run in the joint configuration space. In practice, these algorithms unfortunately suffer from an increase in complexity in the number of tracked targets that can make them unusable in real cases, despite several approaches have been proposed to avoid this limitation. In this paper we suggest a novel way of sampling states from the joint particle filter which reduces the required number of particles. This is attained by collecting targets into groups based on a leader-follower strategy. States are asynchronously passed from leaders to followers by the new approach named dropping-pebbles method, which consists in attaching states to positions. This way, a distribution of the target states is created which works as a map suggests likely states to targets. This scheme has been successfully applied to synthetic data simulating the movement of people in crowds.

***Index Terms*—joint particle filters, multi-target tracking, leader/follower interaction.**

I. Introduction

Tracking human in video sequences is an important and basic tool in the analysis of human behaviour. There has been a considerable work in tracking people and other objects in recent years [1]. Usually those approaches deal with the problem of tracking isolated objects, which can be fairly reliably performed by some systems [2]. Nevertheless, our main interest is that of tracking people in crowds. This is a challenging problem deserving more research since a high number of people is present in the scene and their movements exhibit different degrees of occlusion and interaction, impeding an individual treatment.

Focusing our attention onto crowds, we have observed in video sequences that people usually tend to gather in groups that move as flocks. This suggests that their movements are somehow*related* each other and that there is a tendency in people belonging to the same group to move in a similar way, mimicking the movement of somebody before. Moreover, since the extraction of individuals is difficult when it comes to crowds, we plan to study the movement of smaller components instead of whole individuals. Therefore, one single person can give birth to several components that will show a similar behavior.

Recent approaches to tracking a certain number of interacting targets have suggested the use of multiple-target tracking provided by particle filters running in the *joint* configuration space [1]–[3]. In practice, these algorithms unfortunately suffer from an increase in complexity in the number of tracked targets that can make them unusable in real cases, despite several approaches have been proposed to avoid this limitation, such as clustering of particles to biding them to targets, probabilistic exclusion, or different types of MCMC[1] sampling.

Two are the contributions suggested in this work. First, a novel approach to deal with the problem of tracking

[1]Markov Chain Monte Carlo.

multiple targets which exhibit a degree of interaction. This is perceived as groups moving similarly to some previous targets to whom the group is following. In order to attain this goal, we adapt a conventional *joint* particle filter framework to incorporate a *leader/follower* strategy in the prediction of the states of followers.

Secondly, we introduce the *dropping-pebble* method, which will allow an *asynchronous* transmission of information between targets in the joint particles. Apart from the synchronism of available methods, where simultaneous states are shared to evaluate their interaction, in our case it is more suitable that followers may access to previous leader states corresponding to similar situations. Thus, a map of behaviors is thus generated by attaching those states to a grid which will be employed by the particle filter to sample new states for followers.

II. Independent PF vs. Joint PF

Recently, the use of *joint particle filters* (JPF) has been suggested in the case of multiple target tracking by a number authors [1]–[3], reporting examples of approaches that deal with this problem in areas such as that of data-association [3], targets interaction [2], or the cases where a joint model of targets is computed [1]. The use of joint particle filters is justified by the fact that both association, interaction, and multiple targets can be easily interpreted in terms of the joint configuration space, i.e., the state including the states of all targets.

The use of JPF might seem unnecessary since the use of *independent particle filters* (IPF) might suffice. When targets do not interact, we can run multiple IPF. In this case, in each of the particles, the state is simply the state of one target. Because each particle samples in a small state space, a good approximation of the posterior for the single target is obtained. However, this approach is susceptible to tracking failures when interactions do occur [2].

JPF suffers from a curse of dimensionality [3]. As the number of targets increases, the size of the joint state space increases exponentially. In spite of this *a priori* limitation, approaches based on JPF have been applied to tracking a small number of identical targets either by binding particles to specific targets or by using a mixture of particles filters along with a particle clustering step [2]. The work in [2] tries to overcome the exponential complexity using MCMC instead of SIR[2] since the former samples more efficiently high-dimensional state spaces than the latter.

In this work we focus on JPF and establish a novel way to ease their complexity in the task of tracking multiple targets in the case there exists interaction between *followers* and *leaders* targets. As it can be inferred from these categories, while leaders behave independently, follower states will be conditioned by their leaders. This way, more samples should be generated near the correct state space regions, lesser the number of particles needed.

[2]Sequential Importance Resampling.

III. Bayesian Multi-Target Tracking

In what follows, we adopt a Bayesian approach to multi-target tracking. The Bayesian approach offers a systematic way to combine prior knowledge of target positions, modeling assumptions, and observations to the problems of tracking multiple targets [4]–[6]. A brief description of the Bayesian sequential estimation framework and its *Monte-Carlo* (MC) approximation, i.e., the *particle filter* (PF), is given in this section since its a core component of the algorithm described in Sect. IV.

We will describe the framework for a generic model parameterize by a state $\mathbf{x}_k$, where k denotes the discrete time index. For tracking, the distribution of interest is the posterior $p(\mathbf{x}_k|\mathbf{z}_{0:k})$, also known as the filtering distribution, where $\mathbf{z}_{0:k} = [\mathbf{z}_0, \ldots, \mathbf{z}_k]$ denotes all the observations up to the current time step. In the Bayesian sequential estimation framework, the filtering distribution can be computed according to the two step recursion

Prediction:

$$p(\mathbf{x}_k|\mathbf{z}_{0:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1}|\mathbf{z}_{0:k-1})\, d\mathbf{x}_{k-1} \tag{1}$$

Filtering:

$$p(\mathbf{x}_k|\mathbf{z}_{0:k}) = \frac{p(\mathbf{z}_k|\mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{z}_{0:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{0:k-1})} \tag{2}$$

The recursion requires the specification of a dynamic model describing the state evolution $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ and a model for the state likelihood in the light of the current measurement $p(\mathbf{z}_k|\mathbf{x}_k)$. Both dynamic and likelihood models rely on the Markov assumption [4]–[6].

The tracking recursion yields a closed-form expression in only a small number of cases, i.e., the Kalman Filter for both linear and Gaussian dynamic and likelihood models. For general non-linear and non-Gaussian models, the tracking recursion becomes intractable and approximations techniques are required, such as sequential Monte-Carlo methods, otherwise known as PF. For a complete review on such issue we suggest to refer to [4]–[6].

Accordingly to the MC approximation, at a certain time k, the posterior distribution $p(\mathbf{x}_{k-1}|\mathbf{z}_{0:k-1})$ can be approximated by a weighted set of samples $\{\mathbf{x}^i_{k-1}, \omega^i_{k-1}\}^{N_p}_{i=1}$ [6]. New samples are generated from a suitable designed proposal distribution, which may depend on the old state and the new measurement, i.e., $\mathbf{x}^i_k \sim \pi(\mathbf{x}_k|\mathbf{x}^i_{k-1}, \mathbf{z}_k), i = 1, \ldots, N_p$, where N_p is the number of particles. In order to maintain a consistent sample, the new importance weights

are set to

$$\omega_k^i \propto \omega_{k-1}^i \cdot \frac{p(\mathbf{z}_k|\mathbf{x}_k^i) \cdot p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{\pi(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{z}_k)} \tag{3}$$

where proportionality is up to a normalized constant.

The new set of particles $\{\mathbf{x}_k^i, \omega_{k1}^i\}_{i=1}^{N_p}$ is then approximately distributed according to $p(\mathbf{x}_k|\mathbf{z}_{0:k})$. Approximations to the desired *point* estimate can then be obtained by the MC technique, i.e., $p_{N_p}(\mathbf{x}_k|\mathbf{z}_{0:k}) \triangleq \sum_{i=1}^{N_p} \omega^i \delta_{\mathbf{x}_k^i}(\mathbf{x}_k) \approx p(\mathbf{x}_k|\mathbf{z}_{0:k})$.

A. Sequential Importance Resampling

Many are the choices when selecting an algorithm for sampling from an arbitrary distribution in MC-based Bayesian filtering. Just to mention the most important ones, we refer to the rejection sampling, *Metropolis-Hastings* (MH) algorithm, and *importance sampling* (IS) [4]–[6]. Since rejection sampling is typically inefficient and MH requires a substantial *burn-in* period and generates dependent samples, both techniques are unsuitable for online applications. Therefore, we will rely on importance sampling as a better option for these applications, especially its sequential extension, which is equivalent to the PF.

Algorithm 1 *Generic Joint Particle Filter*

INPUT: $\{\mathbf{x}_{k-1}^i, \omega_{k-1}^i\}_{i=1}^{N_p}, \mathbf{z}_k$

FOR $i = 1, \ldots, N_p$

  FOR $j = 1, \ldots, N_t$

    Draw $\tilde{\mathbf{x}}_{j,k}^i \sim \pi(\mathbf{x}_{j,k}|\mathbf{x}_{j,k-1}^i, \mathbf{z}_{j,k})$

  END FOR

  Compute ω_k^i with to Eq. (5)

END FOR

FOR $i = 1, \ldots, N_p$

  Normalize $\tilde{\omega}_k^i = \frac{\omega_k^i}{\sum_{j=1}^{N_p} \omega_k^j}$

END FOR

Compute $\hat{N}_{eff}$ using Eq. (8)

IF $\hat{N}_{eff} < N_{thr}$ THEN

  $\{\mathbf{x}_k^i, \omega_j^i\}_{i=1}^{N_p} = \text{RESAMPLE}[\{\tilde{\mathbf{x}}_k^i, \tilde{\omega}_k^i\}_{i=1}^{N_p}]$

ELSE

  $\{\mathbf{x}_k^i, \omega_j^i\}_{i=1}^{N_p} = \{\tilde{\mathbf{x}}_k^i, \tilde{\omega}_k^i\}_{i=1}^{N_p}$

END IF

OUTPUT: $\{\mathbf{x}_k^i, \omega_j^i\}_{i=1}^{N_p}$

Fig. 1. Pseudo-code of a generic joint particle filter using the SIR algorithm.

The *sequential importance resampling* (SIR) algorithm is the sequential extension of the IS algorithm that will allows us to approximate posterior densities of the form $p(\mathbf{x}_k|\mathbf{z}_{0:k})$, as required for Bayesian filtering, since it is generally impossible to sample from it directly. Recursiveness of SIR comes from the form the proposal distribution in Eq. (3). The generic scheme of the SIR algorithm for a JPF can be seen in Fig. 1.

On how to choose a proposal distribution π, despite the possibility of choosing other more optimal expressions, a popular choice is the so-called prior distribution

$$\pi(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{z}_k) = p(\mathbf{x}_k|\mathbf{x}_{k-1}^i) \tag{4}$$

Sampling from the prior is often straightforward, for example in the case of an additive noise model, as described in the next section, since sampling from $p(\mathbf{x}_k|\mathbf{x}_{k-1}^i)$ amounts to sampling from the noise distribution. Furthermore, the update Eq. (3) assumes an even simpler form

$$\omega_k^i = \omega_{k-1}^i \cdot p(\mathbf{z}_k|\mathbf{x}_k^i) \tag{5}$$

The election of the prior as the proposal distribution is suboptimal because it does not take the measurement $\mathbf{z}_k$ into account. In cases where the variance of the system noise is significantly greater than the variance of the measurement noise, this could lead to poor estimations.

In this work, we assume a fixed number of N_t targets to be tracked. Each target is parameterized by a state $\mathbf{x}_{j,k}$, where $j = 1, \ldots, N_t$. The combined state is constructed as the concatenation of individual target states, i.e., $\mathbf{x}_k = [\mathbf{x}_{1,k}, \ldots, \mathbf{x}_{N_t,k}]$. The individual targets are assumed to evolve independently according to Markov dynamic models $p(\mathbf{x}_{j,k}|\mathbf{x}_{j,k-1})$. In Section IV-A, the case where targets are engaged in a leader/follower behavior is discussed.

This implies that the dynamics for the combined state factorizes over the individual targets as

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}) = \prod_{j=1}^{N_t} p(\mathbf{x}_{j,k}|\mathbf{x}_{j,k-1}) \tag{6}$$

Accordingly, the weight update Eq. (3) factorizes as

$$p(\mathbf{z}_k|\mathbf{x}_k^i) = \prod_{j=1}^{N_t} p(\mathbf{z}_{j,k}|\mathbf{x}_{j,k}) \tag{7}$$

Finally, it is necessary to resample particles to avoid *degeneracy* of the importance weights ω_k^i. This procedure essentially multiplies particles with high importance weights, and discards those with low importance weights. Our approach uses *importance resampling* [5] that computes an effective number of particles $\hat{N}_{eff}$ to establish how many of them are useful and those that need resampling

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N_p} \left(\tilde{\omega}_k^i\right)^2} \tag{8}$$

B. System Model

In this section, the model of the system used to simulate the behaviour of targets is described. The assumptions necessary for recursive Bayesian filtering are maintained, that is, $\mathbf{x}_{0:k}$ is Markovian and $\mathbf{z}_{0:k}$ is conditionally independent given $\mathbf{x}_{0:k}$, which means that for a dynamic system, noise vectors $\{\mathbf{v}_k\}$, $\{\mathbf{n}_k\}$, and the initial state $\mathbf{x}_0$ are both individually and mutually independent for all k. Therefore, we assume that the system model takes the linear Gaussian form

$$\begin{cases} \mathbf{x}_k &= \mathbf{F}_k \cdot \mathbf{x}_{k-1} + \mathbf{v}_k \\ \mathbf{z}_k &= \mathbf{H}_k \cdot \mathbf{x}_k + \mathbf{n}_k \end{cases} \tag{9}$$

where $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$, and $\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$ are white zero-mean Gaussian noise, and $\mathbf{F}_k$ and $\mathbf{H}_k$ are known linear functions represented by matrices. Specifically, matrix $\mathbf{H}_k$ is taken to be the identity, while $\mathbf{F}_k = diag[\mathbf{F}_{1,k}, \ldots, \mathbf{F}_{N_t,k}]$ is assumed to be a block diagonal matrix, where each block $\mathbf{F}_{j,k}$ corresponds to the j^{th} target and follows a *constant velocity* (CV) model below.

The CV model is fitted to represent the movement of slowly manoeuvring targets in the xy plane. Target acceleration is modeled as zero-mean white Gaussian noise. The state of a single target comprises its position and velocity in the xy plane[3], i.e., $\mathbf{x}_k = [x_k, \dot{x}_k, y_k, \dot{y}_k]^T$. Assuming a uniform discretization with a sampling period of T seconds, matrix $\mathbf{F}$ in the state evolution equation in Eq. (9) is

$$\mathbf{F} = \begin{pmatrix} \mathbf{F}_{cv} & \mathbf{0}_{2\times 2} \\ \mathbf{0}_{2\times 2} & \mathbf{F}_{cv} \end{pmatrix}, \text{ where } \mathbf{F}_{cv} = \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix} \tag{10}$$

where $\mathbf{0}_{n\times m}$ denotes the $n \times m$ matrix of zeros. The state evolution noise $\mathbf{v}$ is assumed to be a zero-mean Gaussian with fixed and known covariance matrix $\mathbf{Q}$

$$\mathbf{Q} = \begin{bmatrix} \sigma_x^2 \mathbf{Q}_{cv} & \mathbf{0}_{2x2} \\ \mathbf{0}_{2x2} & \sigma_y^2 \mathbf{Q}_{cv} \end{bmatrix}, \mathbf{Q}_{cv} = \begin{bmatrix} \frac{T^3}{3} & \frac{T^2}{2} \\ \frac{T^2}{2} & T \end{bmatrix} \tag{11}$$

where σ is a parameter controlling the noise for the model. Under these assumptions individual target dynamics are linear, Gaussian, and non-singular.

IV. Asynchronous Multi-Target Tracking

Let us consider the problem of tracking multiple targets that shares a leader/follower relationship. In order to attain this goal, first we regard the problem of sampling states for those targets behaving as followers as well as introducing the method of *dropping pebbles* that assists the particle filter to hold *nonsimultaneous* state information. The final algorithm is a modification of the one described in Fig. 1 and has been summarized in Fig. 5.

[3]The image plane, in our tracking applications.

A. Leader/Follower Interaction

One contribution of this paper is the computation of the next state $\mathbf{x}_k^f$ of a follower not only from the previous state $\mathbf{x}_{k-1}^f$, as in Eq. (4), but also taking into account the state $\mathbf{x}^l$ of the leader being pursued by the follower. The idea is that the state $\mathbf{x}_k^f$ is conditioned by its leader, i.e., $\tilde{\mathbf{x}}_k^f \sim \pi(\mathbf{x}_k^f | \mathbf{x}_{k-1}^f, \mathbf{x}^l)$ as depicted in Fig. 2 (Up). The closer a follower is to a leader's previous position, the more intense this effect appears. The problem then is how to mix these two states in order to sample $\mathbf{x}_k^f$ and to what extent.

To answer the first question, we suggest to imagine a field of vectors around the position corresponding to the target as shown in Fig. 2 (Down). Those vectors represents the variation $\Delta\mathbf{x}$ that has to be added to a state $\mathbf{x}_{k-1}$ to get the next one, that is, $\mathbf{x}_k = \mathbf{x}_{k-1} + \Delta\mathbf{x}_k$. As explained in Sect. III-B, we can compute the next state of any target using Eq. (9). Accordingly, the increment[4] $\Delta\tilde{\mathbf{x}}$ as

$$\begin{aligned} \Delta\tilde{\mathbf{x}}_k &= \mathbf{x}_k - \mathbf{x}_{k-1} \\ &= \mathbf{F}_k \cdot \mathbf{x}_{k-1} + \mathbf{v}_{k-1} - \mathbf{x}_{k-1} \end{aligned} \tag{12}$$

This computation is independently performed for both the leader and the follower, i.e., $\Delta\mathbf{x}_k^l$ and $\Delta\mathbf{x}_k^f$, and then combined as follows

$$\Delta\mathbf{x}_k^f = \alpha \cdot \Delta\tilde{\mathbf{x}}_k^f + (1-\alpha) \cdot \Delta\tilde{\mathbf{x}}_k^l \tag{13}$$

where the variable α represents the extent of the effect of a leader onto its follower. This interaction takes into account the *credibility* of such information according to Eq. (16) in the Sect. IV-B. Then, the conditioned probabilities $p(\mathbf{x}_{k-1}^f | \mathbf{x}_{k-1}^l)$ and $p(\tilde{\mathbf{x}}_k^f | \mathbf{x}_{k-1}^f)$ combine as

$$\alpha = \frac{f_{cred} \cdot p(\mathbf{x}_{k-1}^f | \mathbf{x}_{k-1}^l)}{f_{cred} \cdot p(\mathbf{x}_{k-1}^f | \mathbf{x}_{k-1}^l) + p(\tilde{\mathbf{x}}_k^f | \mathbf{x}_{k-1}^f)} \tag{14}$$

This ratio weights the relative importance of $\tilde{\mathbf{x}}_k^f$ being a consequence of $\tilde{\mathbf{x}}_{k-1}^f$ with respect to $\mathbf{x}^f$ being a follower of $\mathbf{x}^l$. Fig. 2 (Down) pictures the interaction between leader and follower targets. The vector field on the leader decreases its action on a follower because of α as the states are more divergent, up to the extreme case where no interaction occurs (independent follower).

As a result, the final follower next state $\mathbf{x}_k^f$ can be computed from $\mathbf{x}_{k-1}^l$ and $\mathbf{x}_{k-1}^f$ as

$$\begin{aligned} \mathbf{x}_k^f &= \mathbf{x}_{k-1}^f + \Delta\mathbf{x}_k^f \\ &= \mathbf{x}_{k-1}^f + \alpha \cdot \Delta\tilde{\mathbf{x}}_k^f + (1-\alpha) \cdot \Delta\tilde{\mathbf{x}}_k^l \end{aligned} \tag{15}$$

In case $\alpha = 0$, $\mathbf{x}_k^f$ is computed only with the follower state, while if $\alpha = 1$, it is done with the leader.

[4]The tilde on $\Delta\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}$ means that they are intermediate estimations.

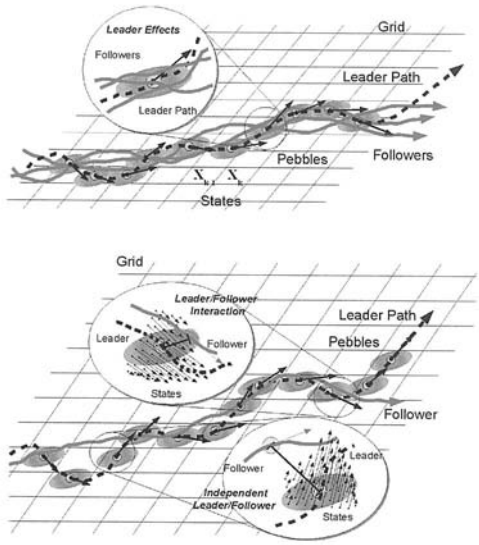

Fig. 2. (Up) Leader's effect regions. (Down) Leader/Follower State Combination.

B. Dropping Pebbles Method

To sample target states in the way explained in Sect. IV-A, where leaders affect followers, it is necessary to get for each time step the state of leaders in a nearby position. So far, interaction between targets has been performed using simultaneous state information. Nevertheless, in our case this information can be outdated (leader moved) or lost (leader left the scene). Consequently, we need an *asynchronous* kind of interaction between targets.

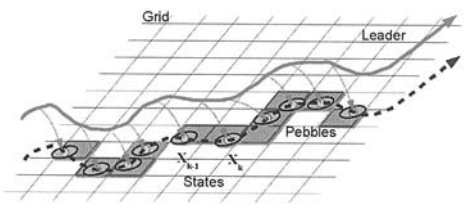

Fig. 3. Leader dropping pebbles in the grid.

Here, we propose a novel method to manage previous leader states in a way that can be posteriorly used by followers as described in Sect. IV-A. We call *pebbles* to the testimony of the pass of a given leader through a position in the xy space. In order to keep these pebbles associated to a certain place, the xy space is divided forming a grid (Fig. 3) where bins represent the area in the space where pebbles will exert their effect.

The main propose of a pebble is to maintain and asynchronously transfers information about a leader that was before in that position. It could be argued that a state holding some previous states might solve that difficulty. Nevertheless, one such approach has many side problems such as defining the temporal window and also the increase on the dimensionality of those states.

Specifically, our approach suggests the use of a grid $\mathcal{G}$ where to attach the states of leaders as they evolve in the image space. These states can be accessed and updated as they are used. A pebble $p_{\mathbf{x}}$ corresponds to a state $\mathbf{x}$ that a leader deposit in the grid $\mathcal{G}$. The usefulness of a pebble $p_{\mathbf{x}}$ is defined by a measure of credibility f_{cred} as

$$f_{cred} = exp(-k_{cred}\frac{\Delta t_{drop}}{N_{votes}}) \tag{16}$$

where f_{cred} depends on the *age* of the pebble, i.e., the time Δt_{drop} elapsed since the pebbles was last accessed (dropped or read), and its *usage* N_{votes}, that is, the number of times the information has been read. Credibility f_{cred} captures the certainty of the information carried by the pebble as a mean to predict the behaviour of a follower from its leader state. All this information is included in each pebble as $p_{\mathbf{x}} = [\mathbf{x}, \Delta t_{drop}, N_{votes}, f_{cred}]$.

Three actions can be performed on a pebble, namely

Drop At each step k, leaders will drop their states $\mathbf{x}_j^l$ as pebbles $p_{\mathbf{x}_j^l}$ in the bins in grid $\mathcal{G}_{k-1}$, as in Fig. 3. We note the dropping of pebbles as $\mathcal{G}_k = \mathcal{G}_{k-1} \bigoplus_{j=1}^{N_l} p_{\mathbf{x}_j^l}$, where N_l is the number of leaders. If the bin already contains a pebble belonging to the same leader, the state $\mathbf{x}_j^l$ is updated as $\mathbf{x}_{j,new}^l = \frac{1}{N_{votes}+1}\left(N_{votes} \cdot \mathbf{x}_{j,old}^l + \mathbf{x}_j^l\right)$. Then, usage N_{votes} is increased by one and Δt_{drop} reset to zero.

Read Whenever it is required, giving a position in the grid or a follower state $\mathbf{x}^f$, the corresponding pebble $p_{\mathbf{x}^l}$ so as to $\left(\mathbf{x}^f \rightsquigarrow \mathbf{x}^l\right)$ is recovered[5]. Consulting a pebble updates N_{votes} and t_{drop} as before.

Delete As the usage of a pebble $p_{\mathbf{x}}$ decays or its age increases, its credibility will decrease, in accordance to Eq. (16). Whenever f_{cred} is too low, the pebble will be erased from the grid $\mathcal{G}$.

[5] $(\mathbf{x}^f \rightsquigarrow \mathbf{x}^l)$ means that $\mathbf{x}^f$ follows $\mathbf{x}^l$.

C. Lost Followers

The correspondence between a follower $\mathbf{x}^f$ and its leader $\mathbf{x}^l$, i.e., $(\mathbf{x}^f \rightsquigarrow \mathbf{x}^l)$, can be lost in two different cases, as depicted in Fig. 4. Whenever the follower differs too much from its leader, this is considered as a *lost* follower which needs to find a new leader. If none, the target becomes a new leader itself (upper diagram in Fig. 4). However, it may also happen the disparity differs just temporally and the follower resumes to the leader after a short period of dissent (lower diagram in Fig. 4).

We compute the probability p_{lost} that a given target is really lost as the combination of the two former notions, expressed as a probability dependent on the time Δt_{lost} since the follower is apparently lost and the probability $(1 - p(\mathbf{x}^f \rightsquigarrow \mathbf{x}^l))$ that target $\mathbf{x}^f$ is actually not following its leader $\mathbf{x}^l$, that is

$$p_{lost} = \left(1 - exp\left(-\frac{\Delta t_{lost}}{2k_{lost}^2}\right)\right) \cdot \left(1 - p(\mathbf{x}^f \rightsquigarrow \mathbf{x}^l)\right) \quad (17)$$

where Δt_{lost} is the time elapsed since the follower is being detached from its leader.

The probability $p(\mathbf{x}^f \rightsquigarrow \mathbf{x}^l)$ that a certain follower $\mathbf{x}^f$ is after a leader $\mathbf{x}^f$ is computed as

$$p(\mathbf{x}^f \rightsquigarrow \mathbf{x}^l) = \begin{cases} exp\left(-\frac{d^2}{2\sigma^2}\right) & \text{if } d \leq 3\sigma \\ 0 & \text{otherwise} \end{cases} \quad (18)$$

where $d = \|\mathbf{x}^f - \mathbf{x}^l\|$ and $\sigma = R/2$ are the difference between follower and leader states and the variance measuring the spread of the distribution, respectively. R is the radius where the leader exert influence onto its followers (see Fig. 4) and relates to the size of bins in the grid $\mathcal{G}$. Distribution $p(\mathbf{x}^f \rightsquigarrow \mathbf{x}^l)$ is truncated to avoid computations farther than 3σ, since this is the 99.73% of the area under the Gaussian in Eq. (18).

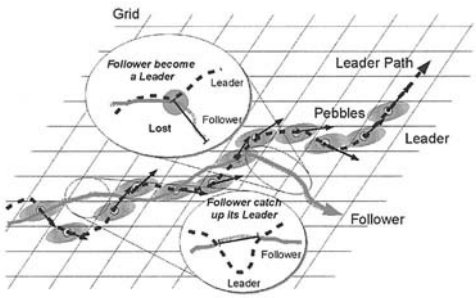

Fig. 4. Leader dropping pebbles in the grid.

Algorithm 2 *Joint Pebbles Particle Filter*

INPUT: $\{\mathbf{x}_{k-1}^i, \omega_{k-1}^i\}_{i=1}^{N_p}, \mathbf{z}_k, \mathcal{G}_{k-1}$

FOR $i = 1, \ldots, N_p$
  FOR $j = 1, \ldots, N_t$
    Get Pebble $\mathbf{x}^l$ from $\mathcal{G}_{k-1}$
    $\tilde{\mathbf{x}}_{j,k}^i \sim \pi_j(\mathbf{x}_{j,k} | \mathbf{x}_{j,k-1}^i, \mathbf{x}^l)$
  END FOR
  Compute ω_k^i with Eq.(5)
END FOR
FOR $i = 1, \ldots, N_p$
  Normalize $\tilde{\omega}_k^i = \frac{\omega_k^i}{\sum_{j=1}^{N_p} \omega_k^j}$
END FOR
FOR $i = 1, \ldots, N_p$
  Compute $p(\mathbf{x}^f \rightsquigarrow \mathbf{x}^l)$
  Compute p_{lost}
  Update $\mathcal{G}_{k-1}$
END FOR
Compute $\hat{N}_{eff}$ using Eq.(8)
IF $\hat{N}_{eff} < N_{thr}$ THEN
  $\{\mathbf{x}_k^i, \omega_j^i\}_{i=1}^{N_p} = \text{RESAMPLE}[\{\tilde{\mathbf{x}}_k^i, \tilde{\omega}_k^i\}_{i=1}^{N_p}]$
ELSE
  $\{\mathbf{x}_k^i, \omega_j^i\}_{i=1}^{N_p} = \{\tilde{\mathbf{x}}_k^i, \tilde{\omega}_k^i\}_{i=1}^{N_p}$
END IF

OUTPUT: $\{\mathbf{x}_k^i, \omega_j^i\}_{i=1}^{N_p}, \mathcal{G}_k$

Fig. 5. Pseudo-code of asynchronous joint particle filter.

V. Simulation and Experiments

This section is devoted to consider the synthetic generation of data required to study the performance of the particle filter described in previous sections in this work and to the series of experiments that has been carried out to support our statements.

A. Data Simulation

In order to study the behavior of the suggested *asynchronous joint particle filter* (AJPT), a certain amount of data has been synthetically generated resembling the movement of people in crowds. Despite our goal is using this algorithms with real data from images depicting their movement, we are using synthetically generated data as a starter. The great benefit of using such kind of data in the preliminary stages of our research is that it can be easily obtained, manipulated, and modulated to study different aspects of the algorithm.

The kind of data generated consists of sets of trajectories closely related to the apparent movement of people in crowded situation. The parameters of those trajectories

have been changed as needed, varying the initial and final points as well as the shape of the curves. Besides, since we are interested in the dynamic of groups, the number of trajectories that can be observed forming a group as well as the number of such groups have also been varied. For space reasons we only show Fig. 6 and Fig. 7 as examples of such trajectories.

Since the time required by the particle filter to conclude the computations is proportional to the number of targets being followed, we have limited to 10 the number of them. Therefore, the data sets generated consists in different gatherings of targets up to that number. Accordingly, up to 22 sets where targets collected in 1, 2, 3, 4, and 5 groups have been obtained. Fig. 6 exhibits one group of trajectories including 3, 5, and 7 targets.

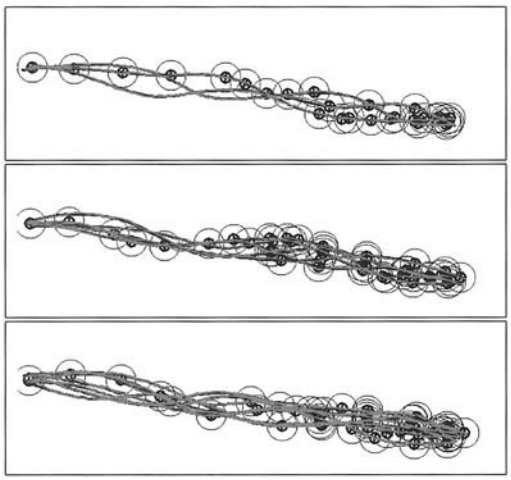

Fig. 6. Example of AJPF using trajectories in the same direction with an increasing number of targets (3, 5, and 7).

B. Experiments

Experiments carried out consist in the application to the sets of synthetical generated measures of two sorts of algorithms, i.e., the generic joint particle filter (JPF) and the asynchronous particle filter (AJPF). The goal of these experiments is, first, to establish the validity of our algorithm and then comparing it to the usual JPF. We want to show how it works and that our approach is better.

To attain this goal, we have conducted a set of computations where the precision, the required time, and the least number of particles needed to reach them have been

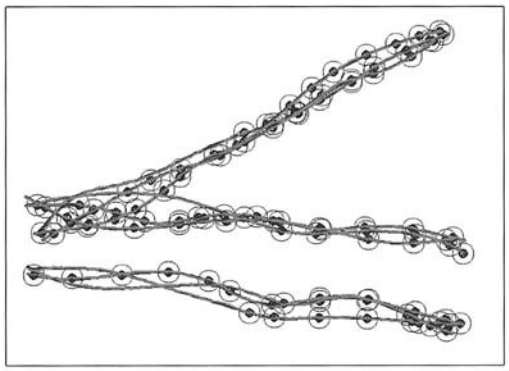

Fig. 7. Example of AJPF using trajectories in several directions with 9 targets.

studied. The last feature aims to decide which of the two algorithms requires fewer particles to obtain reasonable results. Because of the degeneracy inherent in SIR algorithms, if the number of particles is diminished too much, it can happen that at some point the particles in the filter diverge and lose the targets.

Whenever the filter does not converge to any result, the computations are redone increasing the number of particles. This way, an estimation of the number of particles is obtained. As explained before, the main drawback of JPF is its computational complexity, therefore, the lesser number of particles required the better.

C. Results

First, let us consider the performance of our algorithm. In Fig. 6 we want to illustrate the results obtained using one single group of 3, 5, and 7 elements going in the same direction depending on the distance where the leader effect is applied. The circles represent pebbles dropped by leaders. The thicker circle corresponds to the area inside one σ in Eq. (18) and the thinner exterior one to a distance of 3σ. The curves correspond to the computed trajectories which overlay the real measured trajectories in black.

It can be seen in Fig. 6 how pebbles are dropped along the trajectory. When trajectories start separating each other and some follower is lost for too long, it becomes a leader itself and new pebbles are dropped. That is why there are two sets of pebbles as targets approaches to the rightmost end of curves. In Fig 7 the behavior of pebbles in a more complex set of trajectories is depicted. In general, it tends to be one single leader per group until trajectories diverge too much each other. These are examples on how the *dropping pebbles* method works.

N_t	1	2	3	4	5
Error(p)	1.36	1.71	2.32	3.10	3.86
Time(s)	0.92	1.98	3.13	4.14	5.30
TPS(s)	0.7e-4	1.4e-4	2.2e-4	2.9e-4	3.8e-4
N_p	100	100	100	100	100
N_t	**6**	**7**	**8**	**9**	**10**
Error(p)	5.58	4.61	4.87	3.10	5.18
Time(s)	6.31	38.53	55.50	120.81	418.13
TPS(s)	4.5e-4	5.5e-4	5.6e-4	6.6e-4	7.4e-4
N_p	100	500	700	1300	3700

TABLE I. Results of the asynchronous JPF.

Finally, we will discuss the relative performance of the AJPF with respect to the generic JPF. Table. I and Table. II summarize the results corresponding to the RMS error between the computed trajectory and the real one (*Error*, in pixels), the time elapsed during the filtering process (*Time*, in sec.), the time divided by the number of particles and the number of times steps (*TPS*, in sec.), and the number of particles required to obtain those results (N_p).

The set of trajectories in these computations are like those in Fig. 6, where targets follow similar directions. The number of targets N_t goes from 1 to 10, and the maximum number of particles is 4000, due to memory and time limitations. When this number is reached without convergence, the corresponding place in the tables is left empty, meaning that more particles would be needed.

From the results included in Table. I and Table. II some facts can be drawn. First, differences in the precision of the two PF are small, being AJPF slightly better. This seems logic since both of them use the same kind of dynamics to predict the movements of targets. Secondly, the use of the *dropping pebbles* method does not suppose an extra burden since TPS is fairly similar for both PF. In fact, AJPF seems faster because of a smaller number of resampling steps.

Nevertheless, the important fact is that while there is one point from where it is impossible to reach convergence from the usual JPF, the AJPF is possible to obtain results under that limit. Moreover, in cases where generic JPF also converges, AJPF usually requires fewer particles. This is pretty obvious as the number of targets increases.

N_t	1	2	3	4	5
Error(p)	1.59	1.77	2.40	3.61	4.13
Time(s)	2.66	4.25	5.25	11.78	17.78
TPS(s)	1.9e-4	3.0e-4	3.7e-4	4.2e-4	4.2e-4
N_p	100	100	100	200	300
N_t	**6**	**7**	**8**	**9**	**10**
Error(p)	4.51	4.95	–	–	–
Time(s)	18.56	19.07	–	–	–
TPS(s)	4.4e-4	4.5e-4	–	–	–
N_p	300	300	–	–	–

TABLE II. Results of the generic JPF.

VI. Conclusions and Future Work

The main aim in the present work was the introduction of two novel approaches to the problem of multiple target tracking using joint particle filters in relation to the task of extracting trajectories of people in crowds. First, in order to better adapt the dynamic of targets to the real movement of people in such situations, we suggest the use of a *leader/follower* model of sampling target states. This fact has lead us to the *dropping pebbles* method as a way of performing *asynchronous* transmission of target states as an alternative to the usual approaches where only simultaneous states within the particle can be shared.

Our goal in this paper consisted in showing the feasibility of this approach as well as the computational advantages with respect to a usual joint particle filter. To show this we have carried out a series of experiments with synthetical data. Our approach is able to attain similar results with less number of particles than a generic JPF without any further increase in the computational burden.

The next work will include the extension of these ideas to other algorithms apart from SIR, such as Metropolis-Hastings, taking into account a variable number of targets, and also the inclusion of data from real images of crowds.

Acknowledgments

This research was supported by the Ministry of Internal Affairs and Communications of Japan.

References

[1] T. Zhao and R. Nevatia, "Tracking multiple humans in crowded environment," in *Proc. Computer Vision and Pattern Recognition (CVPR'2004)*, vol. 2, July 2004, pp. 406–413.

[2] Z.Khan, T. Balch, and F. Dellaert, "Mcmc-based particle filtering for tracking a variable number of interacting targets," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 11, pp. 1805–1819, 2005.

[3] J. Vermaak, S. Godsill, and P. Perez, "Monte carlo filtering for multi-target tracking and data association," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 41, no. 1, pp. 309–332, 2005.

[4] C. Andrieu, N. Freitas, A. Doucet, and M. Jordan, "An introduction to mcmc for machine learning," *Machine Learning*, vol. 50, pp. 5–43, 2003.

[5] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filter for online nonlinear/non-gaussian bayesian tracking," *IEEE Trans. on Singal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[6] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and Computing*, vol. 10, pp. 197–208, 2000.