The Gaussian mixture MCMC particle algorithm for dynamic cluster tracking

Avishy Carmi, François Septier, Simon J. Godsill

The Gaussian Mixture MCMC Particle Algorithm for Dynamic Cluster Tracking

Avishy Carmi[1], François Septier[2], Simon J. Godsill[1]

[1] *Signal Processing and Communications Laboratory*
Department of Engineering
University of Cambridge, U.K.
[2] *Signal Processing and Information Theory Group*
Institut TELECOM, Telecom Lille1, France

Abstract

We present a novel filtering algorithm for tracking multiple clusters of coordinated objects. Based on a Markov Chain Monte Carlo (MCMC) mechanism, the new algorithm propagates a discrete approximation of the underlying filtering density. A dynamic Gaussian mixture model is utilized for representing the time-varying clustering structure. This involves point process formulations of typical behavioral moves such as birth and death of clusters as well as merging and splitting. For handling complex, possibly large scale scenarios, the sampling efficiency of the basic MCMC scheme is enhanced via the use of a Metropolis within Gibbs particle refinement step. As the proposed methodology essentially involves random set representations, a new type of estimator, termed the probability hypothesis density surface (PHDS), is derived for computing point estimates. It is further proved that this estimator is optimal in the sense of the mean relative entropy. Finally, the algorithm's performance is assessed and demonstrated in both synthetic and realistic tracking scenarios.

Key words: Multiple cluster tracking; Feature tracking; Markov chain Monte Carlo filtering; Evolutionary MCMC.

1 Overview

Multi-Target tracking (MTT) poses major challenges for researchers in the fields of estimation and information fusion. The extensive studies that have been conducted over recent decades have yielded various tracking techniques which can be divided informally into two classes: non-statistical and statistical. Non-statistical methods typically rely on both image differencing techniques and heuristic smoothing algorithms for identifying targets' trajectories [23]. The non-statistical schemes are considered to be fast and viable and have been extensively deployed for tracking in various applications. Nevertheless, these methods are incapable of adequately handling complex tracking scenarios such as those studied here, in which there is significant statistical ambiguity and dynamical structure in the models used. Hence statistical inference approaches are now preferred in many cases.

Owing to the complex nature of MTT problems, statistical tracking methods usually involve smart implementation of tightly coupled data association and filtering schemes [1]. This in turn may result in computationally intensive algorithms such as the multiple hypothesis tracker (MHT) in [21]. Another major difficulty imposed by a typical MTT scenario is related to the mathematical modeling of complex interactions between entities. This consists mainly of birth and death of targets as well as coordinated behavioural patterns which arise in group motions.

The optimal filtering scheme involves the propagation of the joint probability density of target states conditioned on the data. Following the conventional approach, in which all states are concatenated to form an augmented vector, leads to a problematic statistical representation owing to the fact that the targets themselves are unlabeled and thus can switch positions within the resulting joint state vector. Furthermore, targets may appear and disappear thereby yielding inconsistencies in the joint state dimension. These problems can be circumvented by adopting one of the following approaches: 1) introducing some sort of labeling mechanism which identifies existing targets within the augmented vector [24], or 2)

* Part of this work was presented in the 15th IFAC Symposium on System Identification, (SYSID 2009), Saint-Malo, France, July 2009.

considering the joint state as a random finite set. The latter approach provides an elegant and natural way to make statistical inference in MTT scenarios. Nevertheless, its practical implementation as well as its mathematical subtleties need to be carefully considered [27].

Random sets can be thought of as a generalization of random vectors. The elements of a set may have arbitrary dimensions, and as opposed to vectors, the ordering of their elements is insignificant. These properties impose difficulties in constructing probability measures over the space of sets. This has led some researchers to develop new concepts based on belief mass functions such as the set derivative and set integral for embedding notions from measure theoretic probability within random set theory (e.g., Bayes rule). As part of this, point process statistics are commonly used for deriving probabilistic quantities [12]. The PHD filter presented in [17] is the first attempt to implement finite set statistics concepts for MTT. This algorithm uses a Poisson point process formulation to derive a semi closed-form recursion for propagating the first moment of the random set's intensity (i.e., the set's cardinality). A brief summary of the PHD algorithm can be found in [27].

In recent years, sequential Monte Carlo (SMC) methods were applied for MTT. These methods, otherwise known as particle filters (PF), exploit numerical representation techniques for approximating the filtering probability density function (pdf) of inherently nonlinear non-Gaussian systems. Using these methods, the obtained estimates can be set arbitrarily close to the optimal solution (in the Bayesian sense) at the expense of computational complexity. An extensive survey and application of SMC methods is given in [4].

The MTT PF algorithms in the works [4, 9, 15, 26] are intended to work with a fixed number of targets. These PFs exploit point process formulations for properly assigning measurements to their originating targets. As part of this, smart procedures are used to eliminate nonprobable association hypotheses.

An extension of the PF technique to varying number of targets is introduced in [27], [19] and [16]. In [18, 27] a PF implementation of the PHD filter is derived. This algorithm maintains a representation of the filtering belief mass function using random set realizations (i.e., particles of varying dimensions). The samples are propagated and updated based on a Bayesian recursion consisting of set integrals. Both works of [19] and [16] develop a Markov Chain Monte Carlo (MCMC) PF scheme for tracking varying numbers of interacting objects. The MCMC approach does possess a reported advantage over conventional PF due to its efficient sampling mechanism. Nevertheless, in its traditional non-sequential form it is inadequate for sequential estimation. The techniques used by [19] and [16] amend the MCMC for sequential filtering (see also [2]). The work in [16] copes with inconsistencies in state dimension by utilizing the reversible jump MCMC method introduced in [14]. [19], on the other hand avoids the computation of the marginal filtering distribution as in [2] and operates on a fixed dimension state space through use of indicator variables for labeling of active target states (the two approaches being essentially equivalent, see [11]).

1.1 Cluster Tracking

In recent years there has been an increasing interest in tracking scenarios in which a very large number of coordinated objects evolve and interact. One could think of many fields in which such situation is frequently encountered: video surveillance, feature tracking in video sequences, biomedicine, neuroscience and meteorology to mention only a few. Considering the nature of the problem at hand an efficient approach would consist of tracking the clustering structure (i.e., the cluster formation) formed by object concentrations rather than individual entities. In some cases where the number of objects becomes excessively large it might be impractical to individually track them all. Such scenarios are frequently encountered when tracking features in a video sequence.

It should be noted that clusters can be thought of as extended objects that produce a large number of observations. This approach yields the Poisson likelihood formulations in [8]. In recent work [25] merging and splitting objects are modeled using point processes. This is another fundamental issue characterizing cluster behavior that is given full consideration in this work.

1.2 Proposed Method

The algorithm proposed in this paper is an extension of the MCMC filtering mechanism derived in [22]. While assuming that target locations resemble independent samples from a Gaussian mixture we parameterize each cluster by its mean and covariance. The evolution of cluster formation over time which is essentially affected by birth, death, merging and splitting moves, is described by means of a point process.

As opposed to the MCMC scheme derived in [16] which involves an excessive computation of the propagated pdf, our method incorporates an enhancement that renders the resulting filtering algorithm highly efficient and viable and allow its implementation with a relatively small number of particles. Thus, a Metropolis within Gibbs particle refinement step is introduced into the basic MCMC scheme for boosting its sample space exploration capabilities. In addition to that, we also examine the performance gain attained by using population and evolutionary MCMC steps. Our results concur with the theory as the chain mixing indeed improves when using the multiple chain approach. Nevertheless, in terms of tracking performance, the resulting MCMC schemes do not possess a significant advantage over the single chain MCMC.

Because the new MCMC filtering algorithm is intended to work with random sets rather than random vectors the commonly used approaches for computing point estimates (e.g., conditional mean and covariance) are strictly inadequate and cannot be used. This problem is alleviated in this work by adopting two techniques one of which consists of extending the notion of probability hypothesis density [17].

1.3 Outline

This paper is organized as follows. Section 2 mathematically formulates the cluster tracking problem. The likelihood and time evolution models used by the filtering algorithm are presented in Section 3. Section 4 develops the Gaussian mixture MCMC particle filtering algorithm. Section 5 presents the results of a simulation study that has been conducted to assess the new algorithm's tracking performance. Finally, conclusions are offered in the last section.

2 Problem Statement

Assume that at time k there are l_k clusters, or targets at unknown locations. Each cluster may produce more than one observation yielding the measurement set realization $z_k = \{y_k(i)\}_{i=1}^{m_k}$, where typically $m_k >> l_k$. At this point we assume that the observation concentrations (clusters) can be adequately represented by a parametric statistical model.

Letting $Z_{1:k} = \{Z_1, \dots, Z_k\}$ and $z_{1:k} = \{z_1, \dots, z_k\}$ be the measurement history up to time k and its realization, respectively, the cluster tracking problem may be defined as follows. We are concerned with estimating the posterior distribution of the random set of unknown parameters, i.e. $p(\theta_k \mid z_{1:k})$, from which point estimates for θ_k and posterior confidence intervals can be extracted.

For reasons of convenience we consider an equivalent formulation of the posterior that is based on existence variables. Thus, following the approach adopted in [19] the random set θ_k is replaced by a fixed dimension vector coupled to a set of indicator variables $e_k = \{e_k^j\}_{j=1}^n$ showing the activity status of elements (i.e., $e_k^j = 1,\ j \in [1, n]$ indicates the existence of the jth element where n stands for the total number of elements). To avoid possible confusion, in what follows we maintain the same notation for the descriptive parameter set θ_k which is now of fixed dimension.

3 Bayesian Inference

Following the Bayesian filtering approach while assuming that the observations are conditionally independent given (θ_k, e_k) the density $p(\theta_k, e_k \mid z_{1:k})$ is obtained recursively using the conventional Bayesian recursion [8]. Thus, the filtering pdf is completely specified given some prior $p(\theta_0, e_0)$, a transition kernel $p(\theta_k, e_k \mid \theta_{k-1}, e_{k-1})$ and a likelihood pdf $p(z_k \mid \theta_k, e_k, m_k)$. These are derived next for the cluster tracking problem.

3.1 Likelihood Derivation

Recalling that a single observation $y_k(i)$ is conditionally independent given (θ_k, e_k) yields

$$p(z_k \mid \theta_k, e_k, m_k) \propto \prod_{i=1}^{m_k} p(y_k(i) \mid \theta_k, e_k) \tag{1}$$

In the above equation the pdf $p(y_k(i) \mid \theta_k, e_k)$ describes the statistical relation between a single observation and the cluster parameters. An explicit expression of (1) is derived in [8] assuming a spatial Poisson distribution for the expected number of observations. In this work we restrict ourselves to clusters in which the shape can be modeled via a Gaussian pdf. Following this only the first two moments, namely the mean and covariance, need to be specified for each cluster (under these restrictions, the cluster tracking problem is equivalent to that of tracking an evolving Gaussian mixture model with a variable number of components). Note however, that our approach does not rely on the Gaussian assumption and other parameterized density functions could equally be adopted in our framework. Thus,

$$\theta_k^j = \{\mu_k^j, \dot{\mu}_k^j, \Sigma_k^j, w_k^j, B_k^j\}, \quad \theta_k = \{\theta_k^j\}_{j=1}^n, \tag{2}$$

where μ_k^j, $\dot{\mu}_k^j$, Σ_k^j and w_k^j denote the jth cluster's mean, velocity, covariance and associated unnormalized mixture weight at time k, respectively. The additional set B_k^j consists of any other motion parameters affecting the cluster's behavior. At this stage we set $B_k^j = \{\rho_k^j\}$ the local turning radius of the jth cluster's mean at time k.

Following the argument in [8] while recalling that the clusters are modeled via Gaussian pdfs, the likelihood (1) further assumes the following form

$$p(z_k \mid \theta_k, e_k, m_k) \propto \prod_{i=1}^{m_k} \left[\sum_{j=0}^{n} e_k^j \tilde{w}_k^j \mathcal{N}\left(y_k(i) \mid \mu_k^j, \Sigma_k^j\right) \right] \tag{3}$$

where $\tilde{w}_k^j = w_k^j / (\sum_{l=0}^n e_k^l w_k^l)$ is the jth normalized weight. The formulation (3) explicitly accounts for the clutter group (otherwise known as the bulk or null group [8]) for which the shape parameters are $w_k^0, \mu_k^0, \Sigma_k^0$

and $e_k^0 = 1$. An equivalent formulation of (3) designates the clutter probability as $\nu_0 := \tilde{w}_k^0 \mathcal{N}\left(y_k(i) \mid \mu_k^0, \Sigma_k^0\right)$, yielding

$$p(z_k \mid \theta_k, e_k, m_k) \propto \prod_{i=1}^{m_k} \left[\nu_0 + \sum_{j=1}^{n} e_k^j \tilde{w}_k^j \mathcal{N}\left(y_k(i) \mid \mu_k^j, \Sigma_k^j\right) \right] \quad (4)$$

3.2 Modeling Cluster Evolution

The cluster formation may exhibit a highly complex behavior resulting, amongst other things, from group interactions. This in turn may bring about shape deformations and may also affect the number of clusters involved in the formation (i.e., splitting and merging of clusters). At this stage, in order to maintain a generic modelling approach, the filtering algorithm assumes no such interactions which thereby yields the following independent cluster evolution model

$$p(\theta_k, e_k \mid \theta_{k-1}, e_{k-1}) = \prod_{j=1}^{n} p(\mu_k^j \mid \mu_{k-1}^j, \dot{\mu}_k^j) p(\dot{\mu}_k^j \mid \dot{\mu}_{k-1}^j, \rho_k^j) p(\Sigma_k^j \mid \Sigma_{k-1}^j) \times \prod_{j=1}^{n} p(w_k^j \mid w_{k-1}^j) p(e_k^j \mid e_{k-1}^j) p(\rho_k^j \mid \rho_{k-1}^j) \quad (5)$$

The time propagation models underlying the unique decomposition (5) are described next.

3.3 Time Propagation Models

Both the mean position and velocity μ_k^j and $\dot{\mu}_k^j$ are modeled assuming a 2-dimensional nonuniform circular motion. The kinematics of this type of motion which is well established in classical mechanics can be described in the random setting by means of a stochastic differential equation (SDE).

Let $\boldsymbol{u}_t(s)$ and $\boldsymbol{u}_n(s)$ be the tangential and normal unit vectors along the cluster's mean trajectory s, that is, $\boldsymbol{u}_t$ is in the direction of $\dot{\mu}$ whereas $\boldsymbol{u}_n$ is in the perpendicular direction along s. Because s is a parametric curve in t, similarly, one can express both $\boldsymbol{u}_t$ and $\boldsymbol{u}_n$ as functions of t. For the sake of notational simplicity we shall designate $\boldsymbol{u}_t(s(t))$ and $\boldsymbol{u}_n(s(t))$ by $\boldsymbol{u}_t(t)$ and $\boldsymbol{u}_n(t)$, respectively. Following this it can be easily shown that the jth cluster's mean velocity and position along s can be described via the following set of SDE's

$$\begin{bmatrix} d\dot{\mu}^j(t) \\ d\mu^j(t) \end{bmatrix} = \begin{bmatrix} a^j(t)\boldsymbol{u}_t(t) + \frac{\|\dot{\mu}^j(t)\|_2^2}{\rho_k^j} \boldsymbol{u}_n(t) \\ \dot{\mu}^j(t) \end{bmatrix} dt + \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix} dW(t) \quad (6)$$

where $a^j(t)$ and ρ_k^j denote the magnitude of the tangential acceleration and the turning radius, respectively. The SDE's (6) are driven by a Wiener process $W(t)$ with diffusion coefficients σ_1, σ_2. A rather simplified and explicit variant of (6) can be obtained by assuming that the unspecified quantity $a^j(t)\boldsymbol{u}_t(t)$ can be compensated by a proper tuning of the diffusion coefficient σ_1. Further recognizing that $\boldsymbol{u}_n(t) = [-\dot{\mu}_2^j(t), \dot{\mu}_1^j(t)]^T / \parallel \dot{\mu}^j(t) \parallel_2$ where $\dot{\mu}_l^i(t)$ denotes the lth element of $\dot{\mu}^j(t)$ (i.e., $\dot{\mu}^j(t) = [\dot{\mu}_1^j(t), \dot{\mu}_2^j(t)]^T$), the first equation in (6) can be rewritten as

$$d\dot{\mu}^j(t) = \frac{\parallel \dot{\mu}^j(t) \parallel_2}{\rho_k^j} \begin{bmatrix} -\dot{\mu}_2^j(t) \\ \dot{\mu}_1^j(t) \end{bmatrix} + \sigma_1 dW(t) \quad (7)$$

In practice, the models (6) are numerically integrated over the time interval $[t_{k-1}, t_k)$ with the initial conditions

$$\dot{\mu}^j(t_{k-1}) = \dot{\mu}_{k-1}^j, \quad \mu^j(t_{k-1}) = \mu_{k-1}^j \quad (8)$$

for simulating random samples from the transition kernels $p(\dot{\mu}_k^j \mid \dot{\mu}_{k-1}^j, \rho_k^j)$ and $p(\mu_k^j \mid \mu_{k-1}^j, \dot{\mu}_k^j)$ in (5).

3.3.1 Covariance and Mixture Weights Propagation

Both the covariances $\Sigma_k^j \in \mathbb{R}^{2\times 2}$ and the mixture weights $w_k^j \in \mathbb{R}$ satisfy certain constraints that render their sampling procedure somewhat distinct compared to the previous motion parameters (see also [20]). Thus, any sampling scheme must maintain these parameters either positive or positive definite in the matrix case.

A natural approach for producing valid covariance samples assumes $p(\Sigma_k^j \mid \Sigma_{k-1}^j)$ obeys a Wishart distribution. New conditionally independent samples of Σ_k^j can then be drawn using

$$\Sigma_k^j \sim \mathcal{W}\left(\cdot \mid \frac{1}{n_1}\Sigma_{k-1}^j, n_1, n_2\right) \quad (9)$$

where $\mathcal{W}(\cdot \mid V, n_1, n_2)$ denotes the Wishart distribution with a scaling matrix V and parameters n_1 and n_2. Notice that following this approach the mean of the newly sampled matrices is exactly Σ_{k-1}^j.

The mixture weights are sampled in an analogous manner in which the Wishart distribution is replaced by a Gamma distribution. Thus, similarly to (9) we have

$$w_k^j \sim \Gamma\left(\cdot \mid \frac{1}{2}w_{k-1}^j, 2\right) \quad (10)$$

where $\Gamma(\cdot \mid a, n_1)$ denotes the Gamma distribution with shaping and scaling parameters a and n_1. Note that (10) implies that the normalized weights $\tilde{w}_k^j$, $j \in [1, n]$ obey a Dirichlet distribution (see also [20]).

3.3.2 *Turning Radius Propagation*

The cluster turning radius is assumed to change its direction according to a Markov chain. Let ν be the probability of keeping the current direction, then

$$p(\rho_k^j \mid \rho_{k-1}^j) = \begin{cases} \nu, & \text{if } \rho_k^j = \rho_{k-1}^j \\ 1-\nu, & \text{if } \rho_k^j = -\rho_{k-1}^j \end{cases} \quad (11)$$

Note that (11) does not necessarily imply that ρ_k^j assumes only two distinct values. In fact, the magnitude of ρ_k^j remains fixed over the entire time interval, i.e., $|\rho_k^j| = |\rho_0^j|, \; \forall k > 0$, whereas its initial value is drawn from $p(\rho_0^j)$. As a side remark, we note that other propagation models may be adopted for ρ_k^j depending on the scenario.

3.3.3 *Birth and Death Moves*

The existence indicators, denoted here by either e_k^j or $\bar{e}_k^j$, $j = 1, \ldots, n$, are assumed to evolve according to a Markov chain. Denote γ_l the probability of staying in state $l \in \{0, 1\}$, then

$$p(\bar{e}_k^j \mid e_{k-1}^j = l) = \begin{cases} \gamma_l, & \text{if } \bar{e}_k^j = l \\ 1-\gamma_l, & \text{otherwise} \end{cases} \quad (12)$$

In what follows, the existence indicators $\bar{e}_k^j$ undergo an additional propagation stage for simulating complex clustering behaviour. If the next stage is omitted then $e_k^j = \bar{e}_k^j, \; \forall j$.

3.4 *Complex Interactions: Merging and Splitting Clusters*

As previously mentioned, two additional types of moves, merging and splitting, are considered for adequate representation of typical clustering behaviour. The transition kernels for these moves follow the point process formulation (12) with the only difference being possibly a state dependent probability γ. The idea here is that the probability of merging is related to the clusters' spatial location. This allows smooth and reasonable transitions, essentially discouraging 'artificial' jumps to some physically unlikely clustering structure.

The detailed methodology for performing either splitting or merging proceeds as follows. Firstly, two distinct clusters j_1,j_2 are picked uniformly at random between 1 and n (i.e., there are exactly $\frac{n!}{2(n-2)!}$ possible combinations (j_1, j_2)). The chosen clusters are then either splitted or merged with some probability. The transition kernel corresponding to these interactions is given by

$$p(e_k^{j_1}, e_k^{j_2} \mid \bar{e}_k^{j_1}, \bar{e}_k^{j_2}, \theta_k^{j_1}, \theta_k^{j_2}, j_1, j_2) = \begin{cases} \gamma_{j_1,j_2}^{merge}, & \text{If } e_k^{j_1} + e_k^{j_2} = 1 \text{ and } \bar{e}_k^{j_1} + \bar{e}_k^{j_2} = 2 \\ \gamma^{split}, & \text{If } e_k^{j_1} + e_k^{j_2} = 2 \text{ and } \bar{e}_k^{j_1} + \bar{e}_k^{j_2} = 1 \\ 1 - \gamma_{j_1,j_2}^{merge} - \gamma^{split}, & \text{Otherwise} \end{cases} \quad (13)$$

where $\gamma^{split} \in (0, 1)$ and

$$\gamma_{j_1,j_2}^{merge} = \begin{cases} \gamma^m, & \text{If } \parallel \mu_k^{j_1} - \mu_k^{j_2} \parallel_2 \leq d_{\min} \\ 0, & \text{Otherwise} \end{cases} \quad (14)$$

for some $\gamma^m \in (0, 1)$ and some minimal merging distance $d_{\min} > 0$. It should be noted that the probability of having either one of these interactions $\gamma_{j_1,j_2}^{merge} + \gamma^{split}$ should be chosen carefully bearing in mind that excessive application of these operations may be a departure from typical realistic settings which in turn might deteriorate the tracking performance.

The parameters $\theta_k^{j_1}$, $\theta_k^{j_2}$ of either splitting or merging clusters should be updated properly. This consists of finding a single cluster representation $\theta_k^+ = \{\mu_k^+, \dot{\mu}_k^+, \Sigma_k^+ \, w_k^+, \rho_k^+\}$ which forms the outcome of the pair $\theta_k^{j_1}$, $\theta_k^{j_2}$. One way to accomplish this is by matching the first and second moments of the Gaussian, that is

$$\int g^+(x)\mathcal{N}\left(x \mid \mu_k^+, \Sigma_k^+\right) dx = \xi \int g^{j_1}(x)\mathcal{N}\left(x \mid \mu_k^{j_1}, \Sigma_k^{j_1}\right) dx + (1-\xi) \int g^{j_2}(x)\mathcal{N}\left(x \mid \mu_k^{j_2}, \Sigma_k^{j_2}\right) dx \quad (15)$$

where $\xi \in (0, 1)$ is a weighting parameter, and $g^a(x)$, $a \in \{j_1, j_2\}$, may be either x or $(x - \mu_k^a)(x - \mu_k^a)^T$ corresponding to the first two statistical moments. When merging clusters, we set the weighting parameter as $\xi = w_k^{j_1}/(w_k^{j_1} + w_k^{j_2})$ and solve for both μ_k^+ and Σ_k^+. Thus,

$$\mu_k^+ = \xi\mu_k^{j_1} + (1-\xi)\mu_k^{j_2} \quad (16a)$$

$$\Sigma_k^+ = \xi\Sigma_k^{j_1} + (1-\xi)\Sigma_k^{j_2} + \xi(1-\xi)\left[\mu_k^{j_1}(\mu_k^{j_1})^T + \mu_k^{j_2}(\mu_k^{j_2})^T - 2\mu_k^{j_2}(\mu_k^{j_1})^T\right] \quad (16b)$$

The same equations are used when splitting clusters. However, in this case one should properly determine both ξ and either $\theta_k^{j_1}$ or $\theta_k^{j_2}$ for finding the missing parameters of the couple $\theta_k^{j_1}$, $\theta_k^{j_2}$. In this work splitting is carried out using $\xi \sim U(0, 1)$, and $\mu_k^{j_1} \sim \mathcal{N}(\cdot \mid \mu_k^{j_2}, \sigma I_{2\times 2})$,

$\Sigma_k^{j_1} \sim \mathcal{W}(\cdot \mid \frac{1}{n_1}\Sigma_k^{j_2}, n_1, n_2)$ where the parameters σ, n_1 and n_2 pertain to spatial uncertainty.

The remaining parameters which are indirectly involved in determining the spatial locations of the clusters, $\dot{\mu}_k^+$ and ρ_k^+, are either averaged or inherited based on the values of the corresponding parameters of the parent clusters.

3.5 *Dynamic Bayesian Network Representation*

The underlying models governing the time evolution of the latent and observed variables, (θ_k, e_k) and z_k respectively, can be schematically illustrated using a dynamic Bayesian network (DBN). Thus, the state transitions are specified by the kernel (5) while the observations are related to the (hidden) states via the likelihood (3). The DBN corresponding to this process is depicted in Fig. 1 where it is assumed that no interactions between clusters exist (i.e., this DBN lacks the splitting and merging moves). In this figure, indicator transitions $e_{k-1}^j \to e_k^j \to e_{k+1}^j$ and state transitions $\theta_{k-1}^j \to \theta_k^j \to \theta_{k+1}^j$ are shown in the upper and lower channels, respectively. Both these channels contribute in producing the observation sequence z_{k-1}, z_k, z_{k+1} shown in the middle of this figure.

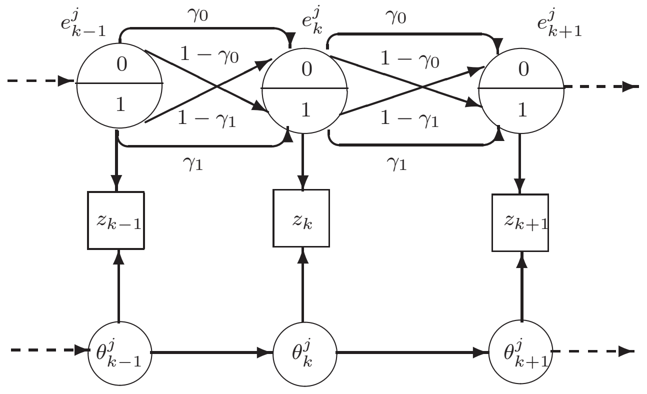


Fig. 1. DBN representation of the jth cluster time evolution (assuming no interactions between clusters). Showing the state transition and emission probabilities (arrows), and the latent and observed variables (circles and rectangles, respectively).

4 MCMC Particle Algorithm

In practice the filtering pdf $p(\theta_k, e_k \mid z_{1:k})$ cannot be obtained analytically and approximations should be made instead. In this section we introduce a sequential MCMC particle filtering algorithm for approximating $p(\theta_k, e_k \mid z_{1:k})$. Two variants of the algorithm are introduced in this section. Firstly, the plain MCMC particle filter is derived consisting of a secondary Gibbs refinement stage that is aimed at increasing the efficiency of the overall sampling procedure. The second, and more innovative, variant uses the plain MCMC scheme in a multiple-chain setting in which genetic operators produce improved proposals. Compared to the single-chain MCMC approach, this scheme increases the efficiency of the filtering algorithm mainly due to its ability to better explore the sample space in a reasonable time.

4.1 *Basic Sampling Scheme*

The following sequential scheme is partially based on the inference algorithm presented in [19] (see also [2]). Suppose that at time $k-1$ there are N samples $\{\theta_{k-1}^{(i)}, e_{k-1}^{(i)}\}_{i=1}^N$ drawn approximately from the filtering density $p(\theta_{k-1}, e_{k-1} \mid z_{1:k-1})$ (i.e., the previous time target distribution). A new set of samples $\{\theta_k^{(i)}, e_k^{(i)}\}_{i=1}^N$ representing $p(\theta_k, e_k \mid z_{1:k})$ can be then simulated using the following Metropolis Hastings (MH) scheme.

4.1.1 *Metropolis Hastings Step*

The MH algorithm generates samples from an aperiodic and irreducible Markov chain with a predetermined (possibly unnormalized) stationary distribution. This is a constructive method which specifies the Markov transition kernel by means of acceptance probabilities based on the preceding time outcome. As part of this, a proposal density is used for drawing new samples. In our case, setting the stationary density as the joint filtering pdf $p(\theta_k, e_k, \theta_{k-1}, e_{k-1} \mid z_{1:k})$ (of which the marginal is the desired filtering pdf), a new set of samples from this distribution can be obtained after the MH burn-in period. This procedure is described next.

First, we simulate a sample from the joint propagated pdf $p(\theta_k, e_k, \theta_{k-1}, e_{k-1} \mid z_{1:k-1})$ by drawing

$$(\theta_k', e_k') \sim p(\theta_k, e_k \mid \theta_{k-1}', e_{k-1}') \tag{17}$$

where $(\theta_{k-1}', e_{k-1}')$ is uniformly drawn from the empirical approximation of $p(\theta_{k-1}, e_{k-1} \mid z_{1:k-1})$ given by

$$\hat{p}(\theta_{k-1}, e_{k-1} \mid z_{1:k-1}) = N^{-1} \sum_{i=1}^{N} \delta(\theta_{k-1}^{(i)} - \theta_{k-1}) \delta(e_{k-1}^{(i)} - e_{k-1}) \tag{18}$$

where $\delta(\cdot)$ denotes the Dirac delta measure. This sample is then accepted or rejected using the following Metropolis rule.

Let $(\theta_k^{(i)}, e_k^{(i)}, \theta_{k-1}^{(i)}, e_{k-1}^{(i)})$ be a sample from the realized chain of which the stationary distribution is the joint filtering pdf. Then the MH algorithm accepts the new candidate $(\theta_k', e_k', \theta_{k-1}', e_{k-1}')$ as the next realization from the chain with probability

$$\alpha = \min\left\{1, \frac{p(z_k \mid \theta'_k, e'_k, m_k)}{p(z_k \mid \theta_k^{(i)}, e_k^{(i)}, m_k)}\right\} \tag{19}$$

That is,

$$(\theta_k^{(i+1)}, e_k^{(i+1)}, \theta_{k-1}^{(i+1)}, e_{k-1}^{(i+1)}) = \\ \begin{cases} (\theta'_k, e'_k, \theta'_{k-1}, e'_{k-1}), & \text{If } u \le \alpha \\ (\theta_k^{(i)}, e_k^{(i)}, \theta_{k-1}^{(i)}, e_{k-1}^{(i)}), & \text{otherwise} \end{cases} \tag{20}$$

where the uniform random variable $u \sim U[0,1]$. The converged output of this scheme simulates the joint density $p(\theta_k, e_k, \theta_{k-1}, e_{k-1} \mid z_{1:k})$ of which the marginal is the desired filtering pdf.

It has already been noted that the above sampling scheme may be inefficient in exploring the sample space as the underlying proposal density of a well behaved system (i.e., of which the process noise is of low intensity) introduces relatively small moves. This drawback is alleviated here by using a secondary Gibbs refinement stage.

4.1.2 Gibbs Refinement

In this work the accepted cluster means undergo a refinement procedure for improving the algorithm's sampling efficiency. For each sample $\theta_k^{(i)}$ we perform a series of successive MH steps with target distributions

$$\hat{p}(\mu_k^{j,(i)} \mid \theta_k^{/j,(i)}, e_k^{(i)}, z_{1:k}) \tag{21}$$

for all $\{j \mid e_k^{j,(i)} = 1, \quad j = 1, \dots, n\}$ where the superscript $j, (i)$ denotes the jth component of the ith particle, and $\theta_k^{/j,(i)} := \{\theta_k^{l,(i)}\}_{l=1}^n / \{\mu_k^{j,(i)}\}$. Here $\hat{p}(\cdot)$ denotes the empirical approximation of the underlying pdf (i.e., based on its particle representation). The refined joint sample $\theta_k^{(i)}$ is then taken as the output of a Gibbs routine.

In practice sampling from the conditionals in (21) is carried out based on the following factorization of $\hat{p}(\mu_k^j \mid \theta_k^{/j}, e_k, z_{1:k})$

$$\begin{aligned} &\hat{p}(\mu_k^j \mid \theta_k^{/j}, e_k, z_{1:k}) \propto \\ &\quad p(z_k \mid \mu_k^j, \theta_k^{/j}, e_k, m_k)\hat{p}(\mu_k^j \mid \theta_k^{/j}, e_k, z_{1:k-1}) \\ &= p(z_k \mid \mu_k^j, \theta_k^{/j}, e_k, m_k)\frac{\hat{p}(\mu_k^j, \theta_k^{/j}, e_k \mid z_{1:k-1})}{\int \hat{p}(\mu_k^j, \theta_k^{/j}, e_k \mid z_{1:k-1}) d\mu_k^j} \\ &\propto p(z_k \mid \mu_k^j, \theta_k^{/j}, e_k, m_k)\hat{p}(\mu_k^j, \theta_k^{/j}, e_k \mid z_{1:k-1}) \end{aligned} \tag{22}$$

Equation (22) facilitates the application of a secondary MH sampling for drawing samples from (21). The overall refinement procedure is based on a Metropolis-within-Gibbs scheme for which the acceptance probability of a new candidate $\bar{\mu}_k^j$ conditioned on the previously accepted one μ_k^j, is given by

$$\frac{p(z_k \mid \bar{\mu}_k^j, \theta_k^{/j}, e_k, m_k)}{p(z_k \mid \mu_k^j, \theta_k^{/j}, e_k, m_k)} \frac{\hat{p}(\bar{\mu}_k^j, \theta_k^{/j}, e_k \mid z_{1:k-1}) q(\mu_k^j)}{\hat{p}(\mu_k^j, \theta_k^{/j}, e_k \mid z_{1:k-1}) q(\bar{\mu}_k^j)} \tag{23}$$

assuming (23) is not greater than 1. Notice that the likelihood term in (23) is identical to $p(z_k \mid \theta_k, e_k, m_k)$ of which an explicit expression is given in (3).

The instrumental density $q(\mu_k^j)$ greatly affects the efficiency of the refinement stage. In this work we have used the observation set $z_k = \{y_k(i)\}_{i=1}^{m_k}$ to construct a smart proposal density. The underlying idea here is rather simple and is based on the fact that the observations themselves are typically concentrated in the vicinity of the unknown clusters' means. Following this our proposal density is composed as a regularized pdf

$$q(\mu_k^j) \propto \sum_{i=1}^{m_k} \mathcal{N}(\mu_k^j \mid y_k(i), R) \tag{24}$$

for which R is the regularization intensity covariance.

An alternative rather efficient construction of the proposal density $q(\mu_k^j)$ consists of pruning out unlikely observations $y_k(i)$ that might not originate from the jth cluster. This can be accomplished by defining a set of feasible indices in the sense

$$G^j := \{i \mid \| y_k(i) - \mu_k^l \|_2 \ge d, \quad \forall l \ne j, \quad i = 1, \dots, m_k\} \tag{25}$$

where d is some threshold distance pertaining to the maximal allowable cluster size. The summation in (24) is then performed over the set G^j rather than over the entire m_k observations. We would like to point out that the primary purpose of this approach is to facilitate generation of distinguishable cluster mean candidates (i.e., properly separated) which thereby reduces the chance of having multiple detections of the same cluster. In this regard one could easily come up with other gating techniques for constructing various proposals.

In practice sampling from $q(\mu_k^j)$ is performed by simply picking an observation index i uniformly at random either from $[1, m_k]$ or G^j and setting $\mu_k^j \sim \mathcal{N}(\cdot \mid y_k(i), R)$.

4.2 Computing Point Estimates and Posterior Moments

As distinct from random vectors, random sets cannot be averaged owing to their property of being invariant to element permutations. This fact in turn implies that the conventional approach of computing the empirical moments of $p(\theta_k, e_k \mid z_{1:k})$ based on the MCMC output $\{\theta_k^{(i)}, e_k^{(i)}\}_{i=1}^N$ is strictly inadequate (this is otherwise known as the label switching problem [24]). For that reason, in what follows we propose two approaches for obtaining valid estimates using the empirical approximation.

4.2.1 Transforming the Set Particles Into Vectors

The first technique relies on reordering the elements of each and every particle $(\theta_k^{(i)}, e_k^{(i)})$ with respect to some reference point estimate (θ_{ref}, e_{ref}). The permuted particles thus obtained are regarded as vectors for which the entries at fixed locations correspond to the same cluster. Consequently the realizations of say, the mean of the jth cluster at time k, μ_k^j, are given by all particles $\{\mu_k^{j,(i)}\}_{i=1}^N$. This approach facilitates the application of conventional methods for computing statistical moments (e.g., the sample mean and covariance).

The reference point which plays a vital role here might be a mode of $p(\theta_k, e_k \mid z_{1:k}) = p(\theta_k \mid z_{1:k}, e_k)p(e_k \mid z_{1:k})$, or analogously when using the empirical approximation $\hat{p}(\theta_k, e_k \mid z_{1:k})$, some highly probable particle from $\{(\theta_k^{(i)}, e_k^{(i)}) \mid \sum_{j=1}^n e_k^{j,(i)} = \sum_{j=1}^n e_{ref}^j\}$, where e_{ref} is, similarly, a highly probable particle from $\hat{p}(e_k \mid z_{1:k})$. Note that as distinct from the importance sampling approach, which is in the heart of conventional particle filtering, MCMC essentially facilitates straightforward computation of proper estimates such as the maximum a posteriori (MAP), however, over some fixed dimension space (See [10] and Section 4 in [6]). Based on (θ_{ref}, e_{ref}) we can now compose the following set of particles

$$T := \left\{ (\theta_k^{(i)}, e_k^{(i)}) \middle| \sum_{j=1}^n e_k^{j,(i)} = \sum_{j=1}^n e_{ref}^j, \quad i = 1, \ldots, N \right\} \tag{26}$$

that is, we pick all particles with the same number of active entities as in e_{ref}, i.e., all elements in T have a fixed dimension $\sum_{j=1}^n e_{ref}^j$. Further transforming the set particles in T into vectors, the entries of each individual particle $(\theta_k, e_k) \in T$ are reordered such that $\| \bar{\theta}_k - \theta_{ref} \|_2$ is minimized, where $\bar{\theta}_k$ denotes the permuted particle. At this stage, the random vectors in T are of fixed dimension and as such can be used for computing empirical estimates. Thus, the sample mean and covariance over T are given by

$$\hat{\theta}_k^{MSE} = \frac{1}{|T|} \sum_{\theta_k \in T} \theta_k \tag{27a}$$

$$\frac{1}{|T|} \sum_{\theta_k \in T} \left[\theta_k - \hat{\theta}_k^{MSE}\right] \left[\theta_k - \hat{\theta}_k^{MSE}\right]^T \tag{27b}$$

where $|T|$ denotes the cardinality of T (i.e., the total number of particles in T).

4.2.2 The PHD Surface (PHDS) Estimator

The previously described approach involves an intermediate stage for alleviating the label switching problem associated with the random set particles. An elegant way for obtaining estimates based on random sets would formally seek to avoid element reordering. This insight has motivate us to consider estimators other than the commonly used conditional expectation.

The PHDS (defined below) can be regarded as an extension of the notion of probability hypothesis density (PHD) [17] , which indicates the estimated target rate (i.e., the number of targets per spatial region). The PHDS is essentially a pdf that corresponds to the normalized target rate over the region of interest. As distinct from the conventional PHD, which effectively quantifies the absolute number of targets, this estimator provides a convenient measure of target concentrations which in turn renders the PHDS adequate for cluster and extended object tracking. In virtue of its unique formulation, this estimator is invariant to set permutations which thereby circumvents the necessity of element reordering.

We define the PHDS over a spatial region A at time k as

$$\text{PHDS}_k(A) = E_{\theta_k, e_k | z_{1:k}} \left[p(A \mid \theta_k, e_k) \right] \tag{28}$$

where $p(A \mid \theta_k, e_k) = \int_{y \in A} p(y \mid \theta_k, e_k) dy$. Here the likelihood pdf $p(y \mid \theta_k, e_k)$ is identical to the one used in specifying (1) and (3). Note that the PHDS estimator is in itself a pdf, as $\int_A \text{PHDS}_k(A) dA = 1$. Following (28), an empirical estimate of the PHDS at every given point $y \in A$ is computed based on the particles approximation as

$$\text{PHDS}_k(y \in A) \approx N^{-1} \sum_{i=1}^N p(y \mid \theta_k^{(i)}, e_k^{(i)}) \tag{29}$$

It turns out that in addition to its abovementioned qualities, the PHDS is also optimal in the sense of minimizing the mean relative entropy (MRE) with respect to the likelihood $p(y \mid \theta_k, e_k)$

$$\begin{aligned} J\left[p(y \mid \theta_k, e_k), \hat{p}(y)\right] &:= \\ &E_{\theta_k, e_k, z_{1:k}} \left\{ \text{KL}\left[p(y \mid \theta_k, e_k) \parallel \hat{p}(y)\right] \right\} \end{aligned} \tag{30}$$

where $\text{KL}\left[p(y) \parallel q(y)\right]$ denotes the Kullback-Leibler divergence between the two comparable pdf's, $p(y)$ and $q(y)$. This observation is established in the following theorem.

Theorem 1 (Minimum MRE Estimator) *A $z_{1:k}$-measurable estimator $\hat{p}(y)$ that minimizes the MRE in* (30) *is given by*

$$\hat{p}(y) = \text{PHDS}_k(y) = E_{\theta_k, e_k \mid z_{1:k}} \left[p(y \mid \theta_k, e_k) \right] \tag{31}$$

The proof is deferred to the Appendix for improved reading.

In order to better apprehend the idea conveyed by Theorem 1 one should bear in mind that the target rate can be represented via a pdf for which the support is the spatial region of interest. Hence it readily follows from both (30) and Theorem 1 that the PHDS pdf yields the lowest discrepancy with respect to the actual (unknown) target rate pdf, $p(A \mid \theta_k, e_k)$, in the sense of the mean Kullback-Leibler divergence.

4.3 Basic MCMC Algorithm Summary

A single cycle of the basic MCMC filtering algorithm is summarized in Algorithms 1, 2 and 3.

Algorithm 1 Basic MCMC Filtering Algorithm

```
(1) Given previous time samples $\{\theta_{k-1}^{(i)}, e_{k-1}^{(i)}\}_{i=1}^{N}$ perform the following steps
(2) for $i = 1, \dots, N + N_{Burn-in}$
(3)   for $j = 1, \dots, n$
(4)     Simulate $(\dot{\mu}_{k-1}^{j,(i)}, \mu_{k-1}^{j,(i)}) \longrightarrow (\dot{\mu}_{k}^{j,(i)}, \mu_{k}^{j,(i)})$ using (6), (7) and (8).
(5)     Draw $\Sigma_k^{j,(i)}, w_k^{j,(i)}, \rho_k^{j,(i)}$ according to (9), (10) and (11), respectively.
(6)   end for
(7)   Perform MCMC move (Algorithm 2).
(8)   Draw $\bar{e}_k^{j,(i)}$, $j = 1, \dots, n$ for the accepted move according to (12).
(9)   Optional: sample two distinct cluster indices $j_1, j_2$ uniformly at random. Perform either merging or splitting of $(\theta_k^{j_1,(i)}, \bar{e}_k^{j_1,(i)})$ and $(\theta_k^{j_2,(i)}, \bar{e}_k^{j_2,(i)})$ as described in Section 3.4.
(10)  If the above step is omitted then set $e_k^{(i)} = \bar{e}_k^{(i)}$.
(11)  Perform Gibbs refinement (Algorithm 3).
(12) end for
```

Algorithm 2 MCMC Move

```
(1) Compute the MH acceptance probability $\alpha$ of the new move using (19).
(2) Draw $u \sim U[0, 1]$
(3) if $u < \alpha$
(4)   Accept $s^{(i)} = (\theta_k^{(i)}, e_k^{(i)}, \theta_{k-1}^{(i)}, e_{k-1}^{(i)})$ as the next sample of the realized chain.
(5) else
(6)   Retain $s^{(i)} = s^{(i-1)}$.
(7) end if
```

4.4 Online Estimation of Tuning Parameters

The modeling parameters governing the clusters' motion (e.g., the diffusion coefficients σ_1, σ_2. see Section 3.2)

Algorithm 3 Particles Refinement (Metropolis within Gibbs)

```
(1) for $j = 1, \dots, n$
(2)   if $e_k^{j,(i)} = 1$
(3)     for $l = 1, \dots, N_{\text{MH Steps}}$
(4)       Draw an observation $\bar{y}$ uniformly at random from either $\{y_k(r)\}_{r=1}^{m_k}$ or $G^j$ in (25).
(5)       Propose a move $\bar{\mu}_k^j \sim \mathcal{N}(\cdot \mid \bar{y}, R)$.
(6)       Compute the MH acceptance probability $\bar{\alpha}$ of the new move using (23).
(7)       Draw $u \sim U[0, 1]$
(8)       if $u < \bar{\alpha}$
(9)         Accept the new move by setting $\mu_k^{j,(i)} = \bar{\mu}_k^j$.
(10)      else
(11)        Retain previous $\mu_k^{j,(i)}$.
(12)      end if
(13)    end for
(14)  end if
(15) end for
```

can be learned online during the filtering process. The resulting adaptive MCMC scheme estimates the underlying parameters as part of its augmented state. In the Numerical Study section of this work we demonstrate the performance of such an algorithm where each individual cluster state consists of the following additional quantities $B_k^j = \{\rho_k^j, \sigma_1^j, \sigma_2^j, \nu^j, \gamma_0^j, \gamma_1^j\}, \quad j = 1, \dots, n$. That is, apart from the turning radius, the set B_k^j in (2) now comprises of the diffusion coefficients σ_1, σ_2, the turning radius switching probability ν, and the existence indicator transition parameters γ_0, γ_1, constituting (6), (11), and (12), respectively. These parameters govern the transition kernel (5) which now takes the form

$$\begin{aligned} &p(\theta_k, e_k \mid \theta_{k-1}, e_{k-1}, B_k^1, \dots, B_k^n) = \\ &\prod_{j=1}^{n} p(\mu_k^j \mid \mu_{k-1}^j, \dot{\mu}_k^j, \sigma_2^j) p(\dot{\mu}_k^j \mid \dot{\mu}_{k-1}^j, \rho_k^j, \sigma_1^j) p(\Sigma_k^j \mid \Sigma_{k-1}^j) \\ &\times \prod_{j=1}^{n} p(w_k^j \mid w_{k-1}^j) p(e_k^j \mid e_{k-1}^j, \gamma_0^j, \gamma_1^j) p(\rho_k^j \mid \rho_{k-1}^j, \nu^j) \end{aligned} \tag{32}$$

The priors $p(B_k^j)$, $j = 1, \dots, n$ can be conveniently set as uniform distributions over some predetermined intervals, or alternatively they may be used to reflect relevant presumptions on the values of these parameters.

For monitoring purposes, the point estimates $\{\hat{B}_k^j\}_{k \geq 0}$ can be computed individually for each cluster following the approach described in Section 4.2.

5 Numerical Study

The new MCMC filtering algorithm is numerically tested in both synthetic and realistic tracking scenarios. The synthetic example relies on the kinematic and behavioral models derived in Section 3, and is aimed to demonstrate the performance of the Gaussian mixture MCMC (GM-MCMC) algorithm in an ideal setting where the actual and the filter's assumed behavioral models coincide. The realistic scenarios, on the other hand, consist of two distinct video tracking examples in which the actual behavioral models are much complex and unpredictable.

5.1 *Algorithm Settings: Parameter Tuning and Burn-in Period Determination*

The values of the various tuning parameters used throughout this section are detailed in Table 1. An exception is made, in this respect, in Section 5.2.1 where some of the parameters are estimated online. The burn-in periods were mostly determined based on trail and error. We nevertheless provide trace plots of a rather rigorous convergence metric, known as the Gelman-Rubin diagnostic [7].

The Gelman-Rubin diagnostic refers to an analysis of variance among two or more chain realizations started from different overdispersed initial values. The key idea behind it is to search for multimodality in the parameter space. Using the mean of the empirical variance within each chain and the empirical variance over all chains, the Gelman-Rubin diagnostic computes a shrinkage factor, R_T for a chain of length T. As $R_T \to 1$ it indicates convergence of the chain to stationarity.

The Gelman-Rubin diagnostic is used for obtaining the trace plots in Fig. 2. This figure illustrates the convergence of R_T based on 1000 chain realizations in a single cycle of the GM-MCMC. As the performance of our approach depends on the instantaneous dimensionality of the sample space (i.e., depending on the indicators e_k^j), Fig. 2 depicts the behaviour of R_T for various number of clusters. We maintain that the maximal number of clusters should be taken into account upon setting the burn-in period based on the Gelman-Rubin diagnostic or alike.

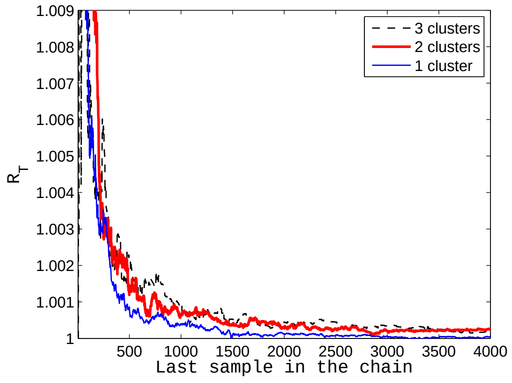


Fig. 2. The Gelman-Rubin diagnostic R_T for various number of clusters.

In addition to the above, we would like to elucidate the rationale underlying the setting of other primary tuning parameters. As distinct from most of the tunable modeling factors in Table 1, which essentially reflect the clusters' motion, both the diffusion coefficient σ_1 and the regularization intensity R regulate the particles diversity which consequently affects the algorithm's mixing properties. Having relatively small values for either of these factors might significantly deteriorate the tracking performance. We recommend setting both these parameters as some percentage of the maximum allowable cluster size (represented by one of its standard deviations). In our runs we have set the maximum cluster size as $\sqrt{40} \approx 6.5$ and both σ_1 and R as roughly 15% of this value, yielding $\sigma_1 = 1$ and $R = I_{2\times 2}$. Lastly, both factors d and $d_{\min}$, which represent a threshold distance for either observation gating or cluster merging, are taken as nearly the maximum cluster size (albeit larger).

5.2 *Synthetic Experiments*

In the first example the GM-MCMC filtering algorithm is employed for tracking a varying number of Gaussian clusters each of which produces a random number of observations varying uniformly between 5 and 20. The cluster means follow a stochastic nonuniform circular motion trajectory (see Section 3) using parametric values similar to those in Table 1 (scenario 1). The total number of clusters involved in the scenario is varied at random times during which either birth or death moves are taking place. The clutter observations are uniformly spread over the instantaneous field of view (defined to include the entire set of point observations). The number of clutter points is set as 1.5 times the total number of non-clutter observations. The MCMC tuning parameters are set as detailed in Table 1 (scenario 1).

The performance of the GM-MCMC filtering scheme is illustrated in Figs. 3–4. The actual Gaussian mixture trajectories are depicted in Fig. 3a using level plots. The intensities in this figure vary according to the values of the actual Gaussian mixture over the entire time interval, that is

$$\sum_{k=1}^{T} p(y \mid \theta_k, e_k) = \sum_{k=1}^{T} \sum_{j=1}^{n} e_k^j \tilde{w}_k^j \mathcal{N}(y \mid \mu_k^j, \Sigma_k^j) \qquad (33)$$

for every point y in the sensor's field of view (FOV). The cluster mean trajectories are shown in this figure via (red) lines. In addition, the mixture covariance ellipses are depicted at various time points along the cluster paths. The corresponding PHDS computed based on the filtering algorithm outputs is shown in the companion panel Fig. 3b. The intensities in this figure corre-

Scenario	Parameters
1	$n = 5$, $\sigma_1 = 1, \sigma_2 = 0$, $\nu_0 = 10^{-7}$, $\rho_0^j \sim U[-20, 20]$, $\dot{\mu}_0^j \sim U[-10, 10] \times [-10, 10]$, $\Sigma_0^j \sim \mathcal{W}(20I_{2\times 2}, 2, 2)$ $w_0^j \sim \Gamma(0.25, 2)$, $\nu = 0.4$, $\gamma_0 = 0.3$, $\gamma_1 = 0.7$, $n_1 = 2, n_2 = 2$, $d_{\min} = 10$, $N = 5000$, $N_{Burn-in} = 500$ $N_{\text{MH Steps}} = 30$, $\{\{\mu_0^{j,(i)}\}_{j=1}^n\}_{i=1}^N = \times_{j=1}^n \{\text{Perm}(z_0)\}$, $R = I_{2\times 2}$, $d = 10$
2	Same as above, where $d = 20$, $d_{\min} = 20$, $N_{Burn-in} = 4000$, $N_{\text{MH Steps}} = 3$
3	Same as above, where $d = 20$, $d_{\min} = 20$, $N_{Burn-in} = 2000$, $N_{\text{MH Steps}} = 3$

Table 1: MCMC particle algorithm settings. The designation Perm(z_0) above stands for a random permutation of observations in z_0.

spond to the values of the empirical estimate

$$N^{-1} \sum_{k=1}^{T} \sum_{i=1}^{N} p(y \mid \theta_k^{(i)}, e_k^{(i)}) \tag{34}$$

The actual cluster mean trajectories and covariances are repeated in this figure for comparison.

Figure 3 clearly demonstrates a good tracking performance of the GM-MCMC filtering algorithm as the actual and estimated tracks over time show an excellent correspondence. This is further illustrated in Fig. 4 where the actual and estimated cluster means and the (cluttered) observations are individually depicted for the X and Y components. In both Figs. 4c and 4d the obtained mean estimates are computed using (27a).

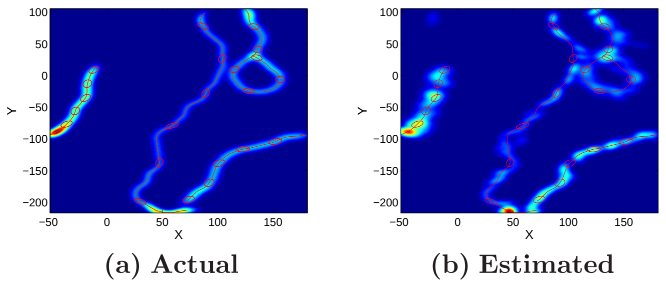


Fig. 3. Actual and estimated Gaussian mixture trajectories (PHDS). Actual means and covariances are marked via red lines and ellipses in both figures.

5.2.1 *Adaptive GM-MCMC*

The GM-MCMC is made adaptive following the approach described in Section 4.4. This variant of the algorithm estimates the modeling parameters, $\{\sigma_1^j, \sigma_2^j, \nu^j, \gamma_0^j, \gamma_1^j\}$, individually for each cluster $j = 1, \ldots, n$. We examine the tracking behavior of the adaptive scheme in a simple scenario consisting of two birth moves occurring at two distinct (albeit fixed) times. In this example the total number of clusters at any given time does not exceed 3.

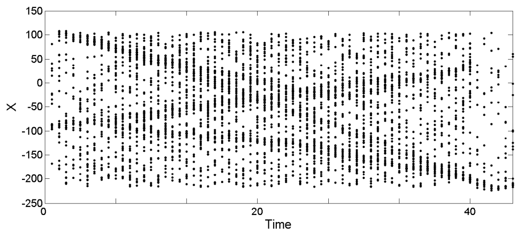
The performance of the adaptive scheme is compared with that of the non adaptive GM-MCMC for which the underlying parameters are set according to Table 1. In the adaptive version these parameters are considered to be random with predetermined priors. Thus, the initial set of corresponding particles is sampled according to $\sigma_1^j \sim U[0.5, 2]$, $\sigma_2^j \sim U[0.5, 4]$, $\nu^j \sim U[0.1, 0.8]$, $\gamma_0^j \sim U[0.1, 0.8]$, $\gamma_1^j \sim U[0.1, 0.8]$.

From this point onwards, the tracking accuracy of the various methods is evaluated by means of the Hausdorff distance. This metric is defined here between the set of actual cluster means, $S_k = \{\mu_k^j \mid e_k^j = 1, \quad j = 1, \ldots, n\}$ and its corresponding estimate $\hat{S}_k^{MSE} = \{\hat{\mu}_k^j\}$ at time k (computed by the GM-MCMC using (27a)). Hence

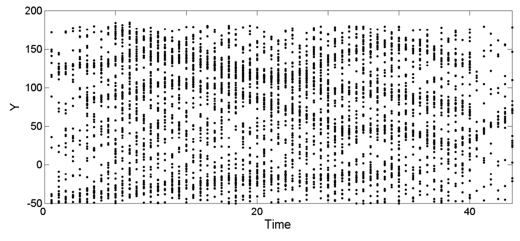
$$d_H(S_k, \hat{S}_k^{MSE}) := \max \left\{ \max_{\mu_k^{j_1} \in S_k} \min_{\hat{\mu}_k^{j_2} \in \hat{S}_k^{MSE}} D\left(\mu_k^{j_1}, \hat{\mu}_k^{j_2}\right), \max_{\hat{\mu}_k^{j_2} \in \hat{S}_k^{MSE}} \min_{\mu_k^{j_1} \in S_k} D\left(\mu_k^{j_1}, \hat{\mu}_k^{j_2}\right) \right\} \tag{35}$$

with $D(\mu_k^{j_1}, \hat{\mu}_k^{j_2}) := \| \mu_k^{j_1} - \hat{\mu}_k^{j_2} \|_2$. Notice that the left-hand-side term inside the brackets in (35) yields relatively large values whenever there is a misdetection of a distant cluster. Similarly, the remaining (right-hand-side) term yields large values for a false detection of a distant cluster. For that reason, large values of d_H are normally attributed to failures in the tracking process.

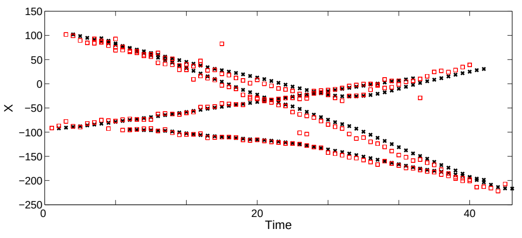
The averaged tracking performance over 20 Monte Carlo runs of both the non adaptive and adaptive GM-MCMC schemes is shown in the left panel in Fig. 5. In this panel, the adaptive GM-MCMC demonstrates a tracking performance comparable with that of the carefully tuned non adaptive tracker. The two prominent spikes in this figure are due to instantaneous misdetections of new born clusters. The remaining (right) panel shows the estimated parameters for the 1st cluster in a typical run. These values have been obtained using the technique described in Section 4.2. We note that the converged values of the indicator transition probabilities obey $\gamma_1 > \gamma_0$ which essentially concurs with the higher birth rate in this specific scenario.



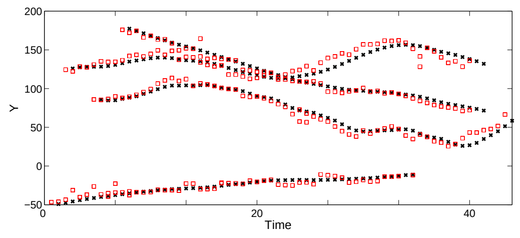
(a) X observations



(b) Y observations

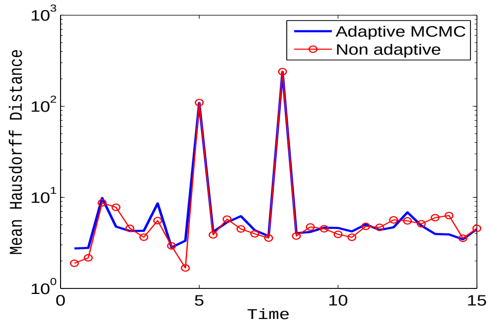


(c) X components

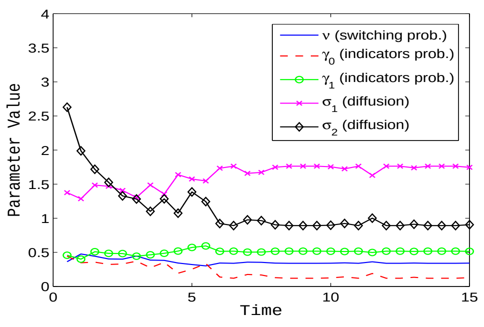


(d) Y components

Fig. 4. Point observations and the actual (black crosses) and estimated (red squares) Gaussian mixture means. Showing X and Y components over time.



(a) Mean Hausdorff Distance



(b) Estimated Parameters (1st cluster)

Fig. 5. Performance of the adaptive GM-MCMC scheme.

5.2.2 Comparison with Other Methods

In this example the GM-MCMC is compared with two other cluster/extended object tracking algorithms. These refer to: (a) independent SIR particle filters, and (b) a recently introduced extended object PHD filter. The former tracking approach was suggested in [16] for reducing the complexity associated with the joint filtering pdf of the multi-target state. It essentially involves employing a filtering algorithm (SIR-PF) individually for each target rather than a single joint tracker. The performance of this technique heavily relies on the reliability of the data association scheme, which is aimed at assigning observations exclusively for each filter. In our implementation the data association is carried out following the simple gating approach of Section. 4.1.2 and the number of particles used by the individual PFs is identical to that of the GM-MCMC.

The second tracking method considered here amends the acclaimed PHD filter for extended objects or clusters [13] (The original `MATLAB`® code of this PHD variant can be found at `http://www.control.isy.liu.se/ karl`). In virtue of its semi-analytical form, the extended target PHD filter (termed here ET-PHD), and in particular its Gaussian mixture implementation, requires significantly less computational resources than any MCMC-based tracking scheme. Nevertheless, this specific variant of the PHD filter turns out to be highly sensitive to its primary tuning parameters, namely the detection and survival probabilities. In our experiments and especially in the heavily cluttered ones, we have frequently experienced difficulties in setting these factors for attaining reasonable tracking performance. In this regard we have included here the most successful runs, essentially omitting the seemingly non convergent ones.

We have compared the performance of the abovementioned methods with our GM-MCMC in both mildly and heavily cluttered scenarios. The total number of clusters in any scenario may not exceed 3. The clusters themselves appear and disappear at random times with no merging nor splitting. The values of the remaining parameters associated with this example are specified in Table 1 (scenario 2). The corresponding point observations in a typical run are shown for both scenarios in Fig. 6.

The performance of the various methods is illustrated in a typical run for the mildly and heavily cluttered scenarios in Figs. 7 and 8, respectively. Both these figures show the actual (black crosses) and estimated (red squares) cluster mean trajectories for the 3 different tracking methods. Hence, it can be readily recognized that the GM-MCMC exhibits a superior tracking accuracy essentially yielding the least number of false detections. This observation is further emphasized in Fig. 9 where both the corresponding cluster identification capabilities and tracking errors are depicted. The upper panel in this figure shows the actual (line) and estimated number of clusters (markers) of the various methods for both the mildly and heavily cluttered scenarios (left and right sub-figures, respectively). Thus, it can be seen that in the mildly cluttered scenario there are precisely 7 occasions in which the ET-PHD fails to correctly identify the actual number of clusters. An even worse behavior is exhibited by the independent PF's approach which yields many more false detections. The GM-MCMC, on the other hand, concludes with no more than a single false detection and 3 misdetections at those times when either birth or death has taken place. The same depiction repeats itself in the heavily cluttered case (right sub-figure in the same panel) with the only difference of much more failures of both the ET-PHD and the independent PF's. The GM-MCMC, unexceptionally, maintains a good tracking performance having no more than 2 false detections and 2 misdetections.

The remaining (bottom) panel in Fig. 9 illustrates the tracking errors (i.e., Hausdorff distance) for both the mildly and heavily cluttered scenarios (left and right sub-figures, respectively). We note that, correspondingly to the upper panel in Fig. 9, the GM-MCMC attains an excellent tracking accuracy in both scenarios. The large spikes, which appear in 3-4 occasions, correspond to the detection failures in the upper panel in this figure. The remaining tracking methods, the ET-PHD and the independent PF's, exhibit an inferior performance compared to the GM-MCMC.

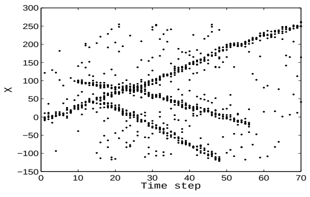
The above observations are further illustrated in Table 2, where both the tracking performance and the mean cycle time of all methods are shown. In addition to the abovementioned tracking algorithms we have included here multiple chain implementations of the GM-MCMC, which are referred to as population GM-MCMC and evolutionary GM-MCMC. Whereas the former approach consists of mixing samples from multiple (non-tempered) chain realizations, the latter method includes an additional evolutionary interaction stage that operates on those samples. Both these implementations are discussed in [5]. The values appearing in Table 2 are computed based on 100 Monte Carlo heavily cluttered runs in which the point observations and diffusion noises were randomly sampled according to the specifications in Table 1 (scenario 2). For obtaining these values in a reasonable time we have considered only a small fraction of the entire time interval, specifically, ranging from 0 to 11. Within this interval the total number of clusters does not exceed 2 as shown in Fig. 9. For that reason, all the particle-based approaches (the MCMC and SIR-PF) are set accordingly with a maximum of mixture components $n = 2$ and a total of 1000 particles. The burn-in length of all MCMC implementations is set as 100, and the multiple chain MCMC variants utilize 2 realizations. The estimation errors in this table refer to the Hausdorff metric (35) averaged over all times in which the number of clusters is correctly identified.

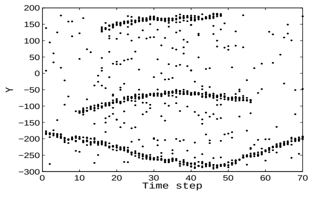
Table 2 indicates that possibly the best tradeoff between tracking accuracy and computational complexity is maintained by the single chain GM-MCMC, having 133 failures (i.e., either misdetections or false detections) over 100 runs, and mean estimation error of 4.39 with a corresponding standard deviation of 1.04. The only other method attaining a better tracking performance is the evolutionary GM-MCMC, which yields an estimation error of 4.04 with a standard deviation of 0.97 while keeping the same number of failures. The population GM-MCMC, however, exhibits an inferior tracking performance compared with the other two MCMC implementations. The remaining non-MCMC tracking approaches, the ET-PHD and the independent PF's, suffer from relatively high failure rates of 373 and 499, respectively.

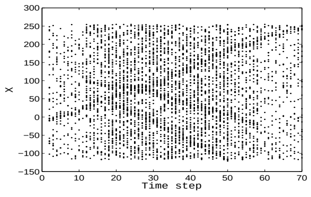
The last column in Table 2 provides the mean cycle time of the various tracking methods (a single cycle comprises of the time propagation and measurement update stages). The numbers appearing in this column refer to the elapsed time as measured in `MATLAB`® environment running under Windows® on an Intel's Core Duo 1.6GHz platform. Thus, it can be seen that in this specific example the computational overhead of the single chain GM-MCMC is slightly less than that of the independent PF's. As expected, the multiple chain MCMC approaches are the most demanding in this regard.

5.2.3 *Multiple Chain and Evolutionary MCMC*

As clearly conveyed by Table 2, the evolutionary GM-MCMC somewhat improves the tracking accuracy with respect to the other MCMC implementations. It is worthwhile noting that the this approach is nearly 4-


(a) X (mild)


(b) Y (mild)


(c) X (heavy)

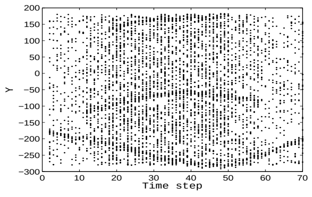
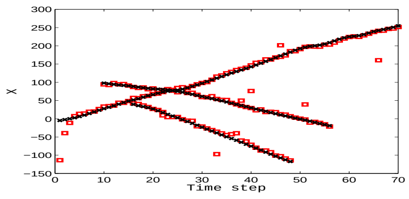
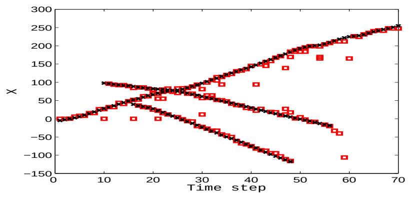
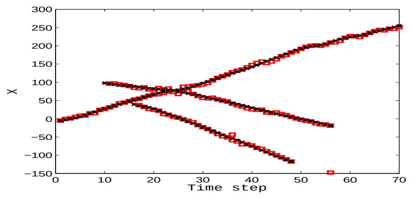
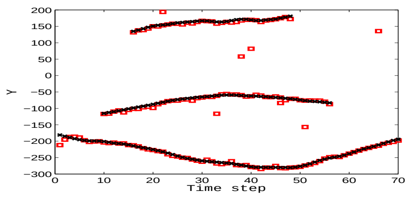

(d) Y (heavy)

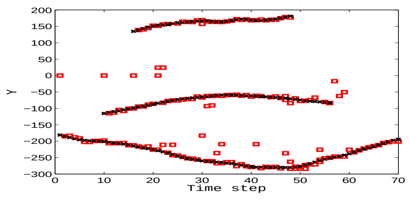
Fig. 6. Point observations over time in typical mildly and heavily cluttered scenarios.


(a) ET-PHD


(b) Independent PFs


(c) GM-MCMC


(d) ET-PHD


(e) Independent PFs

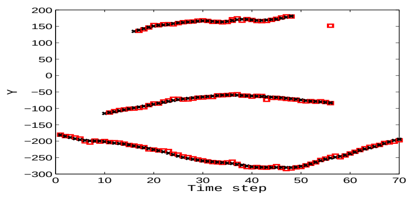
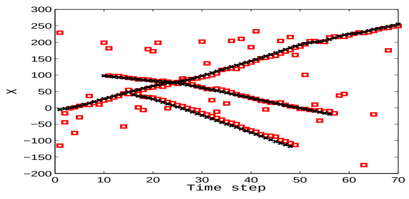
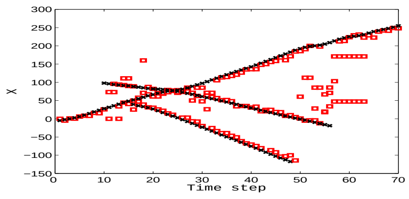
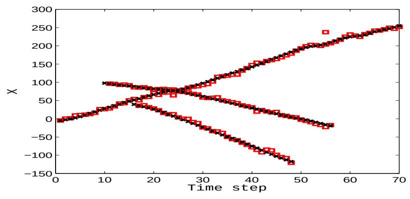
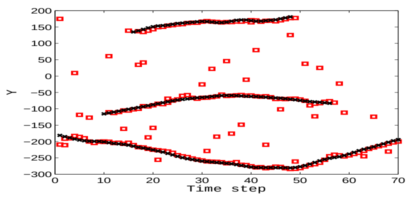

(f) GM-MCMC

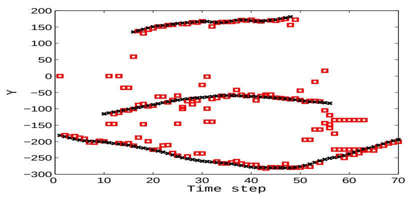
Fig. 7. Tracking performance of the various methods in a typical mildly cluttered run. Showing the actual (black crosses) and estimated (red squares) Gaussian mixture means over time (X and Y components are shown in the top and bottom panels respectively).


(a) ET-PHD


(b) Independent PFs


(c) GM-MCMC


(d) ET-PHD


(e) Independent PFs

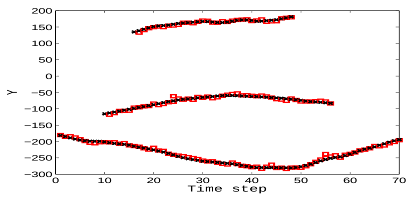

(f) GM-MCMC

Fig. 8. Tracking performance of the various methods in a typical heavily cluttered run. Showing the actual (black crosses) and estimated (red squares) Gaussian mixture means over time (X and Y components are shown in the top and bottom panels respectively).

fold slower than the single chain MCMC, which essentially renders the multiple chain MCMC a subordinate method at most in the context of this work. We have, nevertheless, conducted an experimental study illus-

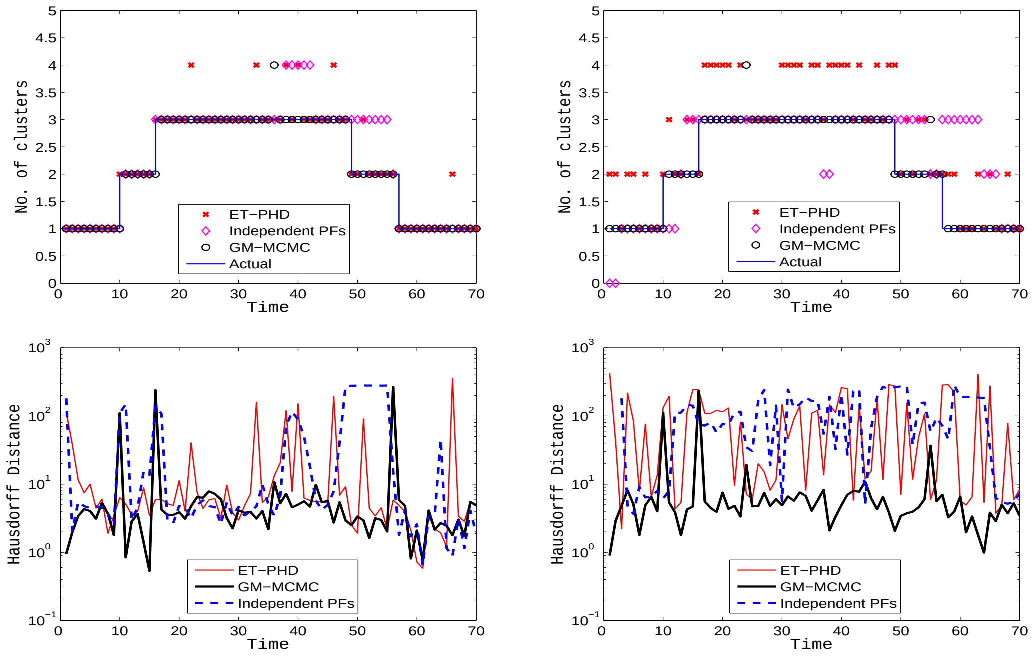


Fig. 9. The Hausdorff distance (lower panel) and actual and estimated number of clusters (upper panel) in a single typical run.

Method	**Mean estimation error (std dev)**	**Failures**	**Mean cycle time (sec)**
Single chain GM-MCMC	4.39 (1.04)	133	3.23
Population GM-MCMC	4.35 (1.15)	136	7.46
Evolutionary GM-MCMC	4.04 (0.97)	133	13.27
Independent PF's	0.32 (0.20)	499	3.44
ET-PHD	7.97 (2.50)	373	0.14

Table 2: Tracking performance of the various methods averaged over 100 Monte Carlo runs.

trating the sampling efficiency of the various MCMC schemes.

The results of our study are provided in Fig. 10. This figure depicts the mixing properties of the single and multiple chain GM-MCMC schemes where the latter scheme is implemented using either 2 or 3 chains. The mixing metric appearing in the left panel in Fig. 10 is obtained as an empirical approximation of

$$\| \Pr(x_k^{(i)}, x_k^{(i+\Delta t)}) - \Pr(x_k^{(i)}) \Pr(x_k^{(i+\Delta t)}) \|_{TV} \qquad (36)$$

where $\| \cdot \|_{TV}$ denotes the total variation norm [1]. Equation (36) essentially quantifies to what extent two far apart samples (for which the indices are i and $i+\Delta t$) can be regarded as being statistically independent (which is a straightforward interpretation of the notion of mixing). A good mixing would therefore imply an expeditious decrease of (36) with any growth of $|\Delta t|$.

In practice, (36) is approximated using histogram estimators (i.e., the norm is computed using the empirical approximations of the underlying distributions) for any of the abovementioned MCMC variants based on 1000 chain realizations with $i = 100$ and $\Delta t \in [-40, 40]$.

The depiction appearing in Fig. 10 clearly shows the advantage of using an increased population of chain realizations as the mixing consistently improves with the number of chains. The advantage of using the evolutionary stage becomes visible only in the 3 chain implementation in this figure.

5.3 *Tracking Features in a Video Sequence*

The single chain MCMC filtering algorithm is applied for tracking a crowd of people in a video sequence. This scenario consists of people walking in a corridor as seen

[1] The total variation norm between two probability distributions $P_1(z)$ and $P_2(z)$ is defined as $\|P_1(z) - P_2(z)\|_{TV} := \sup_z |P_1(z) - P_2(z)|$

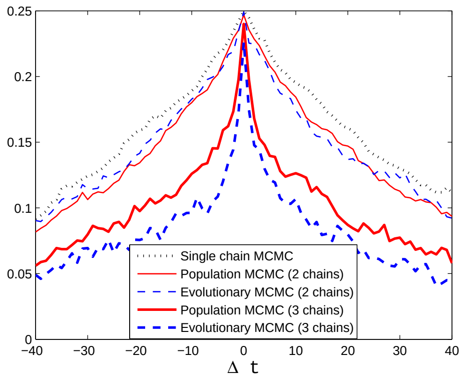


Fig. 10. Mixing properties of the population and evolutionary MCMC variants. Showing the empirical chain mixing metric computed based on 1000 realizations.

from above. The video, which is the same one used by [3], is preprocessed for obtaining both Rosten-Drummond and Tomasi-Kanade features [3]. These features are then used as the point observations z_k that are processed by the filtering algorithm. The obtained set of points is characterized by a relatively dense concentration near each person's head which in some cases extends towards the shoulders. Other points corresponding to non moving objects are considered here as clutter.

The actual scenario, the preprocessed observations and the filtered cluster means are shown for a single time point in Fig. 11. In this figure, the estimated Gaussian means are marked by yellow crosses. The corresponding point observations are plotted separately in the leftmost panel in this figure.

The tracking performance over 30 frames is illustrated in the rightmost panel in Fig. 11. This panel shows both the (manually marked) actual trajectories (green squares) and the estimated cluster means obtained by the filtering algorithm (red squares).

6 Conclusions

A novel Markov chain Monte Carlo filtering algorithm is derived for tracking multiple clusters. The cluster formation is represented using a dynamic Gaussian mixture model for which the number of active clusters is non fixed. The varying nature of the clustering structure is captured using a point process formulation that takes into account birth, death, merging and splitting moves. An additional Metropolis-within-Gibbs refinement stage that is aimed at improving the sample space exploration capabilities of the basic MCMC scheme renders the resulting filtering algorithm highly efficient and viable even when using a relatively small number of particles. An adaptive variant of the algorithm is also presented that is capable of learning the tuning and motion parameters during the filtering process.

The new filter is tested in both synthetic and realistic scenarios. In either cases the algorithm exhibits a good tracking performance as it captures the essence of the clusters behavior. Furthermore, the basic single chain MCMC tracking algorithm is compared with multiple and evolutionary MCMC schemes. An experimental study is provided which shows that even though multiple chain implementations exhibit improved mixing properties, these variants turn out to be inferior with respect to the single chain MCMC which maintains the best tradeoff between tracking accuracy and computational overhead.

7 Acknowledgments

This work was sponsored by the Data and Information Fusion Defense Technology Centre, UK, under the Tracking Cluster. The authors thank these parties for funding this work.

A Proof of Theorem 1

We show that for any $z_{1:k}$-measurable estimator $\hat{p}(y)$ the following is satisfied

$$\Delta J = J\big(p(y \mid \theta_k, e_k), \hat{p}(y)\big) - J\big(p(y \mid \theta_k, e_k), \mathrm{PHDS}_k(y)\big) \geq 0 \quad \text{(A.1)}$$

Using the smoothing property of the conditional expectation, $J(p, \hat{p})$ in (30) may be written as

$$E_{z_{1:k}}\left[E_{\theta_k, e_k \mid z_{1:k}}\left[\int_{y\in A} p(y \mid \theta_k, e_k) \log \frac{p(y \mid \theta_k, e_k)}{\hat{p}(y)} dy\right]\right] \quad \text{(A.2)}$$

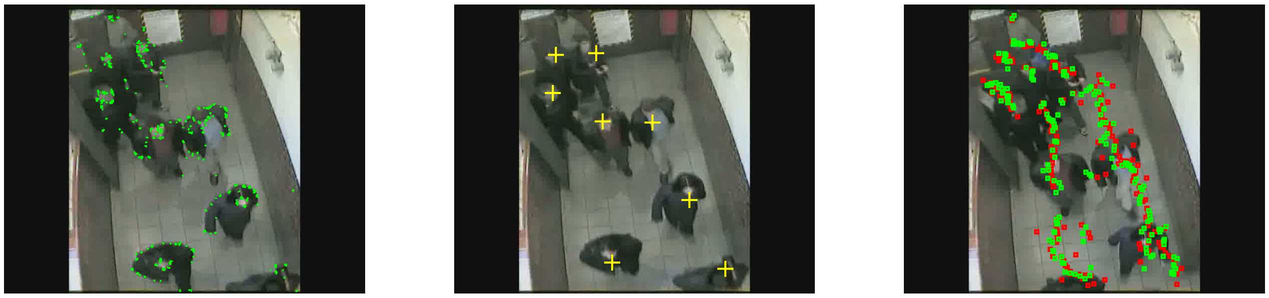
Hence,

$$\begin{aligned}\Delta J &= E_{z_{1:k}}\Bigg[E_{\theta_k, e_k \mid z_{1:k}}\Bigg[\int_{y\in A} p(y \mid \theta_k, e_k) \\ &\quad \times \left(\log \frac{p(y \mid \theta_k, e_k)}{\hat{p}(y)} - \log \frac{p(y \mid \theta_k, e_k)}{\mathrm{PHDS}_k(y)}\right) dy\Bigg]\Bigg] \\ &= E_{z_{1:k}}\left[E_{\theta_k, e_k \mid z_{1:k}}\left[\int_{y\in A} p(y \mid \theta_k, e_k) \log \frac{\mathrm{PHDS}_k(y)}{\hat{p}(y)} dy\right]\right] \\ &= E_{z_{1:k}}\left[\int_{y\in A} E_{\theta_k, e_k \mid z_{1:k}}\left[p(y \mid \theta_k, e_k) \log \frac{\mathrm{PHDS}_k(y)}{\hat{p}(y)}\right] dy\right] \\ &= E_{z_{1:k}}\left[\int_{y\in A} E_{\theta_k, e_k \mid z_{1:k}}\left[p(y \mid \theta_k, e_k)\right] \log \frac{\mathrm{PHDS}_k(y)}{\hat{p}(y)} dy\right] \\ &= E_{z_{1:k}}\left[\int_{y\in A} \mathrm{PHDS}_k(y) \log \frac{\mathrm{PHDS}_k(y)}{\hat{p}(y)} dy\right] \geq 0\end{aligned} \quad \text{(A.3)}$$

which thereby completes the proof. *QED.*

References

[1] Y. Bar-Shalom, X. Li, and T. Kirubarajan. *Estimation with Application to Tracking and Navigation.* John Wiley & Sons, Inc., New York, 2001.



(a) Observations **(b) Estimated means** **(c) Actual and estimated trajectories**

Fig. 11. Estimated cluster centers (yellow crosses in the right panel) and point observations (middle panel) obtained by the corner detector.

[2] C. Berzuini, G. Nicola, W. R. Gilks, and C. Larizza. Dynamic Conditional Independence Models and Markov Chain Monte Carlo Methods. *Journal of the American Statistical Association*, 92(440):1403–1412, 1997.

[3] G. J. Brostow and R. Cipolla. Unsupervised Bayesian Detection of Independent Motion in Crowds. pages 594–601. Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006.

[4] O. Cappé, S.J. Godsill, and E.Moulines. An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo. *Proc. IEEE*, 95(5):899–924, May 2007.

[5] A. Carmi, M. Mihaylova, F. Septier, S. K. Pang, P. Gurfil, and S. J. Godsill. Inferring leadership from group dynamics using markov chain monte carlo methods. In *Modeling, Simulation and Visual Analysis of Crowds: A Multidisciplinary Perspective.* [2] .

[6] P. M. Djuric and J. Chun. An MCMC Sampling Approach to Estimation of Nonstationary Hidden Markov Models. *IEEE Transactions on Signal Processing*, 50(5):1113–1123, 2002.

[7] A. Gelman and D. B. Rubin. Inference from iterative simulation using multiple sequences (with discussion). *Statistical Science*, 7:457–511, 1992.

[8] K. Gilholm, S. Godsill, S. Maskell, and D. Salmond. Poisson Models for Extended Target and Group Tracking. pages 230–241. Proceedings of the SPIE Conference, August 2005.

[9] K. Gilholm and D. Salmond. A Spatial Distribution Model for Tracking Extended Objects. pages 107–113. IEE Proceedings on Radar, Sonar and Navigation 140(2), 1993.

[10] S. Godsill, A. Doucet, and M. West. Maximum a posteriori inference sequence estimation using monte carlo particle filters. *Annals of the Institute of Statistical Mathematics*, 53(1):82–96, 2001.

[11] S. J. Godsill. On the relationship between Markov chain Monte Carlo methods for model uncertainty. *Journal of Comp. Graph. Stats.*, 10(2):230–248, 2001.

[12] I. Goodman, R. Mahler, and H. Nguyen. *Mathematics of Data Fusion.* Boston: Kluwer Academic Publishing Co., 1997.

[13] K. Granström, C. Lundquist, and U Orguner. A gaussian mixture phd filter for extended target tracking. In *Proceedings of the International Conference on Information Fusion*, Edinburgh, UK, 2010.

[14] P. J. Green. Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination. *Biometrika*, 82(4):711–732, December 1995.

[15] C. Hue, J. P. Le Cadre, and P. Perez. Tracking Multiple Objects with Particle Filtering. *IEEE Transactions on Aerospace and Electronic Systems*, 38:791–812, July 2002.

[16] 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*, 27(11):1805–1819, November 2005.

[17] R. Mahler. Multi-Target Bayes Filtering via First-Order Multi-Target Moments. *IEEE Transactions on Aerospace and Electronic Systems*, 39(4):1152–1178, 2003.

[18] W. Ng, J. Li, S. J. Godsill, and S. K. Pang. Multitarget Initiation, Tracking and Termination Using Bayesian Monte Carlo Methods. *Computer Journal*, 50(6):674–693, 2007.

[19] S. K. Pang, J. Li, and S. J. Godsill. Models and Algorithms for Detection and Tracking of Coordinated Groups. pages 1–17. Proceedings of the IEEE Aerospace Conference, 2008.

[20] C. E. Rasmussen. The Infinite Gaussian Mixture Model. *in Advances in Neural Information Processing Systems 12 (MIT Press)*, pages 554–560, 2000.

[21] D. Reid. An Algorithm for Tracking Multiple Targets. *IEEE Transactions on Automation and Control*, 24(6):84–90, December 1979.

[22] F. Septier, A. Carmi, S. J. Godsill, and S. K. Pang. Multiple Object Tracking Using Evolutionary and Hybrid MCMC-Based Particle Algorithms. Proceedings of the IFAC 15th Symposium on System Identification, 2009.

[23] I. Sethi and R. Jain. Finding Trajectories of Feature Points in A Monocular Image Sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:56–72, 1987.

[24] M. Stephens. Dealing with Label Switching in Mixture Models. *Journal of the Royal Statistical Society*, 2000.

[25] C. B. Storlie, T. C. M. Lee, Jan Hannig, and Douglas Nychka. Tracking of Multiple Merging and Splitting Targets with Application to Convective Systems. Proceedings of the ICSA 2008 Applied Statistics Symposium.

[26] J. Vermaak, S. J. Godsill, and P. Perez. Monte Carlo Filtering for Multi-Target Tracking and Data Association. *IEEE Transactions on Aerospace and Electronic Systems*, 41(1):309–330, January 2005.

[27] B. Vo, S. Singh, and A. Doucet. Sequential Monte Carlo Methods for Multi-Target Filtering with Random Finite Sets. *IEEE Transactions on Aerospace and Electronic Systems*, 41(4):1224–1245, October 2005.

[2] `http://dssl.technion.ac.il/DSSL//userdata/SendFile.asp?DBID=1&LNGID=1&GID=555`