Efficient Delay-tolerant Particle Filtering

Xuan Liu

Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

October 2010

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Engineering.

Abstract

Tracking is frequently performed using multiple sensor platforms, with measurements being relayed to a fusion centre over a wireless network. This can lead to some measurements being delayed when adverse environmental conditions cause packet losses to occur. These delayed measurements are called "out-of-sequence measurements (OOSMs)". Simply discarding the delayed OOSMs can waste important information and lead to much poorer tracking performance. This thesis proposes a novel algorithm for delay-tolerant particle filtering that is computationally efficient and has limited memory requirements. The algorithm estimates the informativeness of the OOSMs and immediately discards uninformative measurements. More informative measurements are then processed using the storage efficient particle filter, which is relatively computationally simple. If the measurement induces a dramatic change in the current filtering distribution, the particle filter is re-run to increase the accuracy. From our simulation results, we observe that our novel algorithm only processes a relatively small portion of the OOSMs, but it performs almost as well as much more computationally-complex techniques that have larger storage requirements.

Abrégé

Le suivi est souvent effectué à l'aide de plates-formes composées de multiples capteurs où les mesures sont retransmises à un centre de fusion via un réseau sans fil. Lorsque des conditions environnementales défavorables entraînent des pertes de paquets, la transmission de ces mesures peut être retardée. Ces dernières sont appelées mesures déclassées (OOSM). Jeter ces OOSMs peut gaspiller des informations importantes et peut affecter négativement la performance de l'algorithme de suivi. Cette thèse propose un nouvel algorithme de filtrage de particules tolérantes au délai (delay-tolerant) qui n'est pas gourmand ni en temps de calcul, ni en mémoire. L'algorithme estime la quantité d'information des OOSMs et rejette immédiatement les mesures inutiles. Les mesures contenant suffisamment d'information sont ensuite traitées à l'aide au filtre à particules. Si la mesure induit un changement radical dans la distribution de filtrage actuelle, le filtre à particules est ré exécuter pour augmenter la précision. Nos résultats de simulation indiquent que notre nouvel algorithme ne traite qu'une petite fraction des OOSMs, mais il performe presque aussi bien que de nombreuses techniques qui requièrent des calculs plus complexes et qui ont de plus importants besoins de stockage.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Mark Coates, for his indispensable guidance and invaluable advice throughout my graduate studies at McGill University. He is an intelligent and conscientious supervisor with long-term visions.

I am very grateful to Dr. Boris Oreshkin for his brilliant ideas and inspiration for this thesis and important contribution to the simulations. I sincerely thank my fellow colleagues of the Computer Networks Research Lab for their support and friendship, especially to Deniz, Fariba, Yvan, Hong, Mohammad, Frederic, Daniel, Konstantin, Alex, Abhay, Salim, Judith, Zhe and Xi. Special thanks to Fred for translating the abstract into French; and to Boris, Fred and Deniz for reviewing the thesis and helpful comments. I would also like to thank all my friends I met at McGill for their help in these three years.

Finally, I am greatly grateful to my dear parents for everything they did for me. I would like to thank my cousin Yu for her encouragement. Also, great thanks to all my family and friends in China.

Contents

List of Figures

List of Tables

List of Acronyms

OOSM	out-of-sequence measurements
i.i.d.	independent and identically distributed
pdf	probability density function
KF	Kalman Filter
EKF	Extended Kalman Filter
UKF	Unscented Kalman Filter
MC	Monte Carlo
SMCM	Sequential Monte Carlo Methods
PF	Particle Filter
SIS	Sequential Importance Sampling
SIR	Sequential Importance Resampling
ASIR	Auxiliary Sampling Importance Resampling
RPF	Regularized Particle Filter
MSE	Mean-Square Error
LMMSE	Linear Minimum Mean-Square Error
MCMC	Markov Chain Monte Carlo
SEPF	Storage Efficient Particle Filter
EKS	Extended Kalman Smoother
UKS	Unscented Kalman Smoother
PS	Particle Smoother
OOSM-rerun	OOSM Re-run Particle Filter
OOSM-PS	OOSM Particle Filter using Particle Smoothing
OOSM-GARP	OOSM Gaussian Approximation Re-run Particle filter
MI	Mutual Information

KL	Kullback-Leibler
EDPF-SP	Efficient Delay-tolerant Particle Filter through Selective Processing of OOSMs
CT	Coordinated Turn
CV	Constant Velocity
CRLB	Cramér-Rao Lower Bound
RMS	Root Mean-Squared
RTAMS	Root Time-Averaged Mean Square
GPB	Generalized Pseudo-Bayesian
IMM	Interacting Multiple Model
MMPF	Multiple Model Particle Filter
AUX-MMPF	Auxiliary Multiple Model Particle Filter
JMS-PF	Jump Markov System Particle Filter
AS	Augmented State
PDA	Probabilistic Data Association
OOSM-rerunMM	OOSM Re-run Multiple Model Particle Filter
OOSM-GARP-MM	OOSM Gaussian Approximation Re-run Multiple Model Particle Filter
MMPF-IMM-EKS	Multiple Model Particle Filter with IMM-EKS
EDMMPF-SP	Efficient Delay-tolerant Multiple Model Particle Filter through Selective Processing of OOSMs

Chapter 1

Introduction

1.1 Motivation

Wireless sensor networks consist of spatially distributed sensors that are able to interact with their environment by sensing or controlling physical parameters, such as temperature, sound, motion or pollutants. Usually, these sensors have to cooperatively work to fulfill their tasks, thus they use wireless communication to enable this collaboration [1, 2]. Wireless sensor networks are now widely used in many industrial and civilian applications, including environment monitoring [3], habitat monitoring [4], medicine and healthcare applications [5, 6], fire detection, and traffic monitoring [2, 7].

Target tracking is one of the most important tasks in wireless sensor networks that perform functions such as surveillance, guidance or obstacle avoidance. For example, these networks can be used to track a robot over a wide wild area. For traffic management, tracking techniques can help the traffic police track and catch vehicles involved in traffic offences. Tracking algorithms take their input measurements from sensors and then process the measurements together to estimate the state of a target (or multiple targets) at each point in time. The state represents important attributes of the target such as position and velocity. Successive estimates at regular intervals provide the real-time tracks which describe the trajectory of the desired target(s).

1.2 Thesis Problem Statement

In many multi-sensor tracking systems, all the sensors send their measurements to a fusion centre that executes the tracking algorithm. If a measurement can arrive at the fusion centre in the same interval when it is measured, the fusion centre can process it and determine the current estimated state. However, in some cases, some measurements do not arrive at the fusion centre on time; they are received after a few intervals delay. Such delayed measurements are called "out-of-sequence measurements" (OOSMs) in the literature. They can be caused by a number of specific effects. Wireless communication delays can arise when packets are transmitted from the sensor to the fusion centre. Delays can also arise due to different data processing times at different sensors (for example, the sensor could be a rotating radar with measurement-specific time stamps).

The problem is how to incorporate efficiently these out of sequence measurements into the current state estimates. Several algorithms have been previously proposed for addressing this problem [8–12]. These methods differ in their tracking performance and modeling assumptions, but also in their memory requirements and computational complexity. Our goal is to find an algorithm with the best trade-off between performance and computational and memory requirements.

1.3 Thesis Contribution and Organization

Chapter 2 provides the background necessary to understand tracking problems in wireless sensor networks. We first introduce Bayesian tracking, a general theoretical framework for tracking problems and describe two popular practical methods: the Kalman filter and the particle filter. Then we focus on the issue of out-of-sequence measurements and review several proposed particle filtering algorithms addressing the problem of OOSMs.

Chapter 3 proposes two novel particle filtering algorithms addressing the OOSM problem. These two algorithms both reduce the memory requirements by storing the Gaussian approximation (the mean and covariance matrix) of the particles. The first algorithm re-runs the particle filter from the time step when the OOSM is measured. The second algorithm is a novel delay-tolerant particle filtering algorithm that is more computationally efficient. We propose a selection rule to estimate the informativeness of each OOSM and immediately discard any measurements which are deemed uninformative. The more

informative measurements are processed using the storage efficient particle filter proposed by Orguner et al. in [12]. This algorithm is usually accurate, but it can fail when a delayed measurement is extremely informative. Our algorithm applies a second test to detect these highly-informative OOSMs, and incorporates them by re-running the particle filter from the time-step when the OOSM was produced. At the end of Chapter 3, we describe the results of Matlab simulations that compare the performance of the proposed algorithms.

Chapter 4 focuses on multiple model tracking with the problem of OOSMs. We first review the proposed algorithms for multiple model tracking and describe two representative methods: interacting multiple model (IMM) filters and multiple model particle filter (MMPF). Then we follow the idea of the proposed algorithms in Chapter 3 and propose four particle filtering algorithms for multiple model tracking with OOSM problems. At the end of this chapter, we analyze the performance of our proposed algorithms using Matlab simulations of a manoeuvring target.

Chapter 5 summarizes our work and discusses potential future work.

1.4 Published Work

A paper based on some of the content presented in this thesis will be published in the following conference proceedings.

- Xuan Liu, Boris N. Oreshkin and Mark J. Coates, Efficient delay-tolerant particle filtering through selective processing of out-of-sequence measurements, in *Proceedings of 13th International Conference on Information Fusion*, Edinburgh, UK, Jul. 2010, accepted.

Chapter 2

Literature Review

Tracking techniques have been widely used in many fields, such as traffic management, security surveillance systems and military systems. As a result of their widespread practical applications, there has been considerable amount of research on developing tracking algorithms. This chapter provides the background necessary to understand tracking problems in wireless sensor networks. We first present a mathematical model of tracking problems, and then we briefly describe a selection of efficient tracking algorithms and discuss the advantages and disadvantages of each. We focus in particular on particle filters, which have been demonstrated to achieve good tracking performance in wireless sensor networks, even when the dynamics of the moving targets are highly non-linear or the noise in the system is far from Gaussian. Section 2.1 formulates the tracking problem in a state-space framework and describes the Bayesian approach. The second half of the chapter provides a more detailed discussion of the previous literature that has addressed the problem of out-of-sequence measurements (OOSMs) in tracking. The primary objective of this thesis is to introduce new methods for incorporating measurements that are delayed due to the network operating conditions, so this literature is particularly relevant. Section 2.2 provides a mathematical formulation of the OOSM problem, discusses the potential impact on the performance of standard tracking algorithms and examines several approaches that have appeared in the literature for making better use of the delayed measurements.

2.1 Bayesian Tracking

The tracking problem is essentially an estimation task. The objective is to estimate over time the evolving state of one or more moving targets, where the state represents properties of interest such as position and velocity. The Bayesian framework provides a systematic, general methodology for dynamic state estimation problems. The basic idea of Bayesian approaches is to estimate the posterior probability density function (pdf) of the state vector based on all of the past states and the available observations. The optimal estimation of the current state can be achieved using this pdf, which summarizes all of the available knowledge about the state vector.

In many cases, the state dynamics can be captured well using a linear model with innovations governed by Gaussian noise. The observations can be accurately modeled as a linear function of the state with Gaussian noise. For this scenario, the posterior distribution is a Gaussian and can thus be represented with the mean and covariance matrix. The Kalman filter [13] provides an analytic solution for Bayesian tracking when these types of models are applicable. However, for many practical applications, estimation algorithms for nonlinear/non-Gaussian systems are needed. The Extended Kalman Filter (EKF) [14] and Unscented Kalman Filter (UKF) [15] can be used when mildly nonlinear models are accurate.

These methods begin to fail when the nonlinearity becomes more pronounced or the noise (observation or innovation) is highly non-Gaussian. Particle filters are capable of successfully tracking in systems with such properties. These techniques, which belong to the more general category of Sequential Monte Carlo Methods, comprise a set of flexible simulation-based methods used to sequentially estimate Bayesian models. The key idea is to simulate the required pdf using a weighted pointwise approximation, with each random sample being considered as a "particle". State estimates can then be formulated in terms of the state-values of these samples and the associated weights. If the number of samples is large enough, this Monte Carlo approximation becomes an accurate representation of the posterior pdf and thus leads to good estimates. There are numerous particle filtering algorithms [16], but the underlying principles are the same. In this section we describe the Sequential Importance Sampling (SIS) filter [17], one of the simplest particle filtering algorithms. We briefly discuss the limitations of this algorithm and review an algorithm — Sampling Importance Resampling (SIR) filter [18], that has been proposed to address

the deficiencies.

2.1.1 Problem Statement

We employ a state-space model to capture the dynamics of the target(s) and the nature of the observations. We assume that the system is Markovian, but the dynamics may be nonlinear and the noises may be non-Gaussian.

The state vector $\{x_k; k \in \mathbb{N}\}$ ($\mathbb{N}$ denotes non-negative integers) is represented as a Markov process with transition probability $p(x_k|x_{k-1})$. The initial distribution of $\{x_k\}$ is $p(x_0)$. The measurements $\{y_k; k \in \mathbb{N}^*\}$ ($\mathbb{N}^*$ denotes positive integers) are assumed to be conditionally independent given the states $\{x_k; k \in \mathbb{N}\}$. The model can be represented by a state transition equation (i.e., the system dynamics) and a state measurement equation (i.e., the system observation):

$$x_k = f_{k|k-1}(x_{k-1}, v_{k|k-1}) \tag{2.1}$$

$$y_k = h_k(x_k, s_k), \tag{2.2}$$

where $y_k \in \mathbb{R}^{n_y}$ denotes the output measurement at time k, $x_k \in \mathbb{R}^{n_x}$ denotes the state vector of the system at time k, $v_{k|k-1} \in \mathbb{R}^{n_v}$ is the process noise and $s_k \in \mathbb{R}^{n_s}$ is the measurement noise. n_x, n_y, n_v, n_s are the dimensions of the respective vectors. The mappings $f_{k|k-1} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \to \mathbb{R}^{n_x}$ and $h_k : \mathbb{R}^{n_x} \times \mathbb{R}^{n_s} \to \mathbb{R}^{n_y}$ represent the transition process and measurement model.

We denote by $x_{0:k} \triangleq \{x_0, \ldots, x_k\}$ the state vector trajectory (a set of state vectors) and $y_{1:k} \triangleq \{y_1, \ldots, y_k\}$ the measurement vector trajectory up to time k. In the Bayesian framework, our aim is to estimate recursively in time the posterior distribution $p(x_{0:k}|y_{1:k})$, or its marginal distribution $p(x_k|y_{1:k})$ (known as the filtering distribution). Also of interest are expectations of the form

$$I(g_k) = \int g_k(x_{0:k}) p(x_{0:k}|y_{1:k}) \, \mathrm{d}x_{0:k}, \tag{2.3}$$

where g_k is some function of interest. From Bayes' theorem, the posterior distribution can be represented as

$$p(x_{0:k}|y_{1:k}) = \frac{p(y_{1:k}|x_{0:k}) p(x_{0:k})}{\int p(y_{1:k}|x_{0:k}) p(x_{0:k}) \, \mathrm{d}x_{0:k}}. \tag{2.4}$$

Due to the Markov property of x_k and the conditional independence of y_k , we can derive a recursive formula for this joint distribution $p(x_{0:k}|y_{1:k})$.

$$p(x_{0:k}|y_{1:k}) = p(x_{0:k-1}|y_{1:k-1})\frac{p(y_k|x_k)p(x_k|x_{k-1})}{p(y_k|y_{1:k-1})}. \tag{2.5}$$

For our tracking applications, the interest lies primarily in estimating the state x_k conditioned on all of the measurements $y_{1:k}$. Thus, we focus on the marginal posterior distribution $p(x_k|y_{1:k})$. We develop a recursive algorithm, so we assume that we have (an estimate of) $p(x_{k-1}|y_{1:k-1})$ available. Initiating the algorithm requires the knowledge of $p(x_0)$. The estimation procedure consists of two steps: a prediction step and an updating step. In the first step, we predict the current state x_k by forming an estimate of the predictive posterior $p(x_k|y_{1:k-1})$. This can be achieved by combining the posterior from the previous iteration $p(x_{k-1}|y_{1:k-1})$ and the transition probability $p(x_k|x_{k-1})$ as in Equation (2.6). The second step incorporates the information from the new measurement y_k. The posterior distribution $p(x_k|y_{1:k})$ can be updated by multiplying the prediction distribution $p(x_k|y_{1:k-1})$ by the likelihood function $p(y_k|x_k)$ and normalizing, as in Equation (2.7). The combined process of recursive state estimation (filtering) is then:

$$\text{Prediction} : p(x_k|y_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1})\,\mathrm{d}x_{k-1}, \tag{2.6}$$

$$\text{Updating} : p(x_k|y_{1:k}) = \frac{p(y_k|x_k)p(x_k|y_{1:k-1})}{\int p(y_k|x_k)p(x_k|y_{1:k-1})\,\mathrm{d}x_k}. \tag{2.7}$$

The above recursive formulae suggest the possibility of sequential estimation, but in most cases the integrals involved in the expressions are not analytically tractable.

The Kalman filter [13] provides an analytical solution when (i) the new state x_k is a linear function of the previous state x_{k-1} with additive Gaussian noise (this implies that the update distribution $p(x_k|x_{k-1})$ is Gaussian); and (ii) the observation y_k is a linear function of x_k with additive Gaussian noise (this implies that the likelihood is Gaussian). In practical tracking scenarios, it is often the case that these conditions do not hold. The Extended Kalman Filter (EKF) [14] and the Unscented Kalman Filter (UKF) [15] were introduced to address the case where non-linear models were required to adequately capture the dynamics or observations. The following section provides a brief summary of the Kalman filter and the Extended Kalman filter.

2.1.2 Kalman Filter and Extended Kalman Filter

The Kalman Filter [13] applies when a linear/Gaussian model can accurately capture the system dynamics and observations:

$$x_k = F_{k|k-1}x_{k-1} + v_{k|k-1} \tag{2.8}$$
$$y_k = H_k x_k + s_k. \tag{2.9}$$

Here $F_{k|k-1}$ and H_k are matrices representing linear functions, $v_{k|k-1}$ and s_k are Gaussian noises with distribution $\mathcal{N}(v_{k|k-1}; 0, V_{k|k-1})$ and $\mathcal{N}(s_k; 0, Q_k)$. Since the posterior density at every time step is assumed to be Gaussian, it can be parameterized by a mean and covariance matrix. These can be updated analytically at each time step. The prediction and updating steps for the mean and covariance matrix are:

$$\text{prediction}: \quad x_{k|k-1} = F_{k|k-1}x_{k-1|k-1} \tag{2.10}$$
$$P_{k|k-1} = F_{k|k-1}P_{k-1|k-1}F_{k|k-1}^T + V_{k|k-1} \tag{2.11}$$
$$\text{updating}: \quad S_k = H_k P_{k|k-1}H_k^T + Q_k \tag{2.12}$$
$$K_k = P_{k|k-1}H_k^T S_k^{-1} \tag{2.13}$$
$$x_{k|k} = x_{k|k-1} + K_k(y_k - H_k x_{k|k-1}) \tag{2.14}$$
$$P_{k|k} = (I - K_k H_k)P_{k|k-1}, \tag{2.15}$$

where $x_{k|k-1}$ and $P_{k|k-1}$ denote the predicted mean and covariance of state x_k (conditioned on all past observations), and $x_{k|k}$ and $P_{k|k}$ denote the updated mean and covariance.

The Kalman Filter is an optimal solution to the tracking problem and its calculation is very efficient. Unfortunately, the modeling assumptions are overly restrictive for many practical tracking tasks. The Extended Kalman Filter (EKF) [14] is a generalization that is applicable to nonlinear models with additive Gaussian noises. The EKF uses a first order Taylor series expansion of the nonlinear transition and measurement functions to approximate the current state and measurement. The transition and measurement models are described with nonlinear functions $f_{k|k-1}$ and h_k as follows:

$$x_k = f_{k|k-1}(x_{k-1}) + v_{k|k-1} \tag{2.16}$$
$$y_k = h_k(x_k) + s_k. \tag{2.17}$$

The derivation of the EKF requires that both $f_{k|k-1}$ and h_k are differentiable functions. Let $F_{k|k-1}$ and H_k denote the Jacobian matrices of partial derivatives of $f_{k|k-1}$ and h_k with respect to x as follows:

$$F_{k|k-1} = \frac{\partial f_{k|k-1}(x)}{\partial x}|x = x_{k-1|k-1} \tag{2.18}$$

$$H_k = \frac{\partial h_k(x)}{\partial x}|x = x_{k|k-1}. \tag{2.19}$$

Then we can derive the prediction and updating steps of the Extended Kalman Filter:

$$\text{prediction} : x_{k|k-1} = f_{k|k-1}(x_{k-1|k-1}) \tag{2.20}$$

$$P_{k|k-1} = F_{k|k-1}P_{k-1|k-1}F_{k|k-1}^T + V_{k|k-1} \tag{2.21}$$

$$\text{updating} : \quad S_k = H_k P_{k|k-1} H_k^T + Q_k \tag{2.22}$$

$$K_k = P_{k|k-1}H_k^T S_k^{-1} \tag{2.23}$$

$$x_{k|k} = x_{k|k-1} + K_k(y_k - h_k(x_{k|k-1})) \tag{2.24}$$

$$P_{k|k} = (I - K_k H_k)P_{k|k-1}. \tag{2.25}$$

The EKF preserves the light computational overhead of the Kalman filter and it is more generally applicable and useful in many real-world problems. There are however, important tracking problems in which the modeling assumptions do not hold. The EKF requires that the posterior distribution $p(x_k|y_{1:k})$ is Gaussian or can be accurately approximated as such. If this condition doesn't hold, for example, the true distribution is bi-modal or skewed, then the EKF will perform poorly. If the non-linearities are not mild (so that the first order Taylor series expansion is not a good approximation) or they are non-differentiable, then it is necessary to employ more computationally intensive techniques, such as the particle filters described in the following section.

2.1.3 Sequential Monte Carlo Methods (Particle Filters)

To address nonlinear/non-Gaussian tracking problems, one approach is to employ approximations for the distributions in equations (2.6) and (2.7) using Monte Carlo methods. Monte Carlo methods are computational algorithms that approximate the distributions of interest by using a large set of random samples. When we have a large enough number

of samples drawn from the required posterior distributions, it is possible to approximate the intractable integrals appearing in equations (2.6) and (2.7) using summations over the sampled state values.

Real-time tracking applications require that the estimation at each time step can be performed on-line. Sequential Monte Carlo Methods employ a recursive formulation, building on the computations that have been performed during previous time steps in order to reduce the computational overhead. These methods are also called particle filters, with each sample from the posterior being labeled as a "particle". In this section, we introduce a few representative methods from the developing history of Sequential Monte Carlo Methods.

Sequential Importance Sampling (SIS) Method

The Sequential Importance Sampling (SIS) algorithm [17] is an intrinsic component of many of the more advanced Sequential Monte Carlo methods. It is an extended version of the "importance sampling method" [19]. In general, it is not possible to sample directly from the posterior distribution. The SIS algorithm circumvents this problem by drawing random samples (particles) from an alternative "importance" distribution. An importance weight is then associated with the particle; this weight represents the ratio of the probability of drawing the sample from the posterior distribution to the probability of drawing it from the importance distribution. State estimates can then be expressed as weighted summations over the collection of particles. If the number of samples is large enough to accurately represent the posterior pdf, the SIS filter approaches the optimal Bayesian estimate [16].

The importance sampling procedure is formulated in a sequential fashion. Denote the importance sampling distribution at time step k by $\pi(x_{0:k}|y_{1:k})$. Each particle has an importance weight:

$$\omega_k = \frac{p(x_{0:k}|y_{1:k})}{\pi(x_{0:k}|y_{1:k})}. \tag{2.26}$$

We can formulate the following expression for $I(g_k)$:

$$I(g_k) = \frac{\int g_k(x_{0:k})\omega_k\pi(x_{0:k}|y_{1:k})\,\mathrm{d}x_{0:k}}{\int \omega_k\pi(x_{0:k}|y_{1:k})\,\mathrm{d}x_{0:k}}. \tag{2.27}$$

Then, if we can simulate N i.i.d. particles $\{x_{0:k}^{(i)}; i = 1, \ldots, N\}$ for each state according to

$\pi(x_{0:k}|y_{1:k})$, a possible Monte Carlo approximation of $I(g_k)$ is

$$\hat{I}_N(g_k) = \frac{\frac{1}{N}\sum_{i=1}^{N} g_k(x_{0:k}^{(i)})\omega_k^{(i)}}{\frac{1}{N}\sum_{i=1}^{N}\omega_k^{(i)}} = \sum_{i=1}^{N} g_k(x_{0:k}^{(i)})\tilde{\omega}_k^{(i)}, \tag{2.28}$$

where the normalized importance weights $\tilde{\omega}_k^{(i)}$ are given by

$$\tilde{\omega}_k^{(i)} = \frac{\omega_k^{(i)}}{\sum_{i=1}^{N}\omega_k^{(i)}}. \tag{2.29}$$

From the discussion in [16], we can see that $\hat{I}_N(g_k)$ is biased (ratio of two estimates) when N is finite, but asymptotically, under weak assumptions, the strong law of large numbers applies, that is, $\hat{I}_N(g_k) \xrightarrow[N\to\infty]{a.s} I(g_k)$ [19].

Returning to the sequential case, we usually acquire the new measurement y_k at time k, so we need a recursive formula for updating $\pi(x_{0:k}|y_{1:k})$ and $\omega_k^{(i)}$. Hence the importance distribution is chosen to satisfy

$$\pi(x_{0:k}|y_{1:k}) = \pi(x_{0:k-1}|y_{1:k-1})\pi(x_k|x_{0:k-1}, y_{1:k}) \tag{2.30}$$

$$= \pi(x_0)\prod_{m=1}^{k}\pi(x_m|x_{0:m-1}, y_{1:m}). \tag{2.31}$$

By substituting (2.5) and (2.30) into (2.26), it is clear that this importance function allows us to evaluate the importance weights recursively in time:

$$\omega_k^{(i)} \propto \frac{p(x_{0:k-1}^{(i)}|y_{1:k-1})p(y_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})}{\pi(x_{0:k-1}^{(i)}|y_{1:k-1})\pi(x_k^{(i)}|x_{0:k-1}^{(i)}, y_{1:k})} \tag{2.32}$$

$$\propto \omega_{k-1}^{(i)}\frac{p(y_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})}{\pi(x_k^{(i)}|x_{0:k-1}^{(i)}, y_{1:k})}. \tag{2.33}$$

SIS is an attractive method, but unfortunately, a common problem is the degeneracy phenomenon. Practically, after several time steps, very few particles have non-negligible importance weights. It can be shown that the variance of the importance weights can only increase over time [20]. The algorithm, consequently, fails to represent the posterior distributions of interest adequately after some number of time steps. The degeneracy can

be mitigated to some extent by choosing a good importance function (one that matches, as closely as possible, the posterior). The other common approach is to introduce a resampling step, which is employed periodically to eliminate particles with small weights and replicate particles with large weights.

Choice of Importance Distribution

The optimal choice of the importance distribution is that minimizes the variance of the importance weights. Doucet et al. discuss the choice of importance distributions in [20], and explain that the optimal choice is $p(x_k|x_{k-1}^{(i)}, y_k)$, as introduced in [21]:

$$\begin{aligned}\pi(x_k|x_{0:k-1}^{(i)}, y_{1:k})_{optimal} &= p(x_k|x_{k-1}^{(i)}, y_k) \\ &= \frac{p(y_k|x_k, x_{k-1}^{(i)})p(x_k|x_{k-1}^{(i)})}{p(y_k|x_{k-1}^{(i)})}.\end{aligned} \tag{2.34}$$

Substituting (2.34) into (2.33), we obtain the following update equation for the weights:

$$\omega_k^{(i)} \propto \omega_{k-1}^{(i)} p(y_k|x_{k-1}^{(i)}) \tag{2.35}$$

$$\propto \omega_{k-1}^{(i)} \int p(y_k|x_k^{'}) p(x_k^{'}|x_{k-1}^{(i)})\, \mathrm{d}x_k^{'}. \tag{2.36}$$

This choice has two major drawbacks in practical implementation. It requires the ability to sample from $p(x_k|x_{k-1}^{(i)}, y_k)$ and to calculate the integral in (2.36), which will have no analytic form in general cases. There are two cases when the optimal importance distribution can definitely be used. The first case is when x_k is a member of a finite set such that the integral in (2.36) becomes a sum of finite members and sampling from $p(x_k|x_{k-1}^{(i)}, y_k)$ is possible. This approach is illustrated in [22]. The second case is the example of Gaussian state space model with non-linear transition function and linear measurement function, as described in [20].

However, it is impossible to get such analytic evaluation for many other models with nonlinear measurement functions. In such cases, an approach that performs well is to construct a suboptimal importance distribution by using local linearization [20].

In practical settings, one must consider whether the performance improvement obtained by choosing a complicated importance distribution justifies the additional computational

cost of constructing it. The most common method is to adopt the prior distribution as the importance distribution. This generally leads to a simple sampling operation and an efficient procedure for updating the weights,

$$\pi(x_{0:k}|y_{1:k}) = p(x_{0:k}) = p(x_0) \prod_{m=1}^{k} p(x_m|x_{m-1}) \tag{2.37}$$

$$\omega_k^{(i)} \propto \omega_{k-1}^{(i)} p(y_k|x_k^{(i)}). \tag{2.38}$$

Sequential Importance Resampling (SIR) Method

Introducing an additional selection step, resampling, counteracts the degeneracy phenomenon. This leads to the Sequential Importance Resampling method [18]. The key idea of resampling is to eliminate particles with small weights and to concentrate on particles with large weights. Douc et al. describe four popular resampling methods [23]: multinomial resampling [17], residual resampling [24], stratified resampling [25] and systematic resampling [25], and also discuss the advantages of these methods. From the analysis in [23], we can see that systematic resampling is the simplest to implement. But residual and stratified resampling methods are shown to have lower conditional variance for all weights. Moreover, central limit theorems hold with the residual resampling approach in more general cases than stratified resampling [23].

Thus, the residual resampling scheme is widely used in particle filtering applications. Here, we describe it in Algorithm 1. We also use it in our simulations in the following chapters.

Algorithm 1: Residual Resampling

1 Recalculate the weights

$$\bar{\omega}^i = \frac{N\omega^i - \lfloor N\omega^i \rfloor}{N - R}, i = 1, \ldots, N \tag{2.39}$$

where N is the number of particles, $\lfloor \quad \rfloor$ denote the integer part, and $R = \sum_{i=1}^{N} \lfloor N\omega^i \rfloor$.

2 Calculate the deterministic component

$$N_d^i = \lfloor N\bar{\omega}^i \rfloor. \tag{2.40}$$

3 Calculate the multinomial component: $\{N_m^i\}$ are distributed according to the multinomial distribution $Mult(N - R; \bar{\omega}^1, \ldots, \bar{\omega}^n)$.

4 $N^i = N_d^i + N_m^i$, N^i is the number of copies of particle $x_k^{(i)}$.

Basic Particle Filters

In order to provide a more complete illustration of how a particle filter operates, we now provide a complete algorithmic description. We choose the bootstrap filter [17], which is an SIR algorithm that employs the prior distribution as the importance function. Algorithm 2 describes the operation of a basic bootstrap particle filter. Note that in equation (2.41), $\omega_{k-1}^{(i)}$ does not appear because the particles $x_{k-1}^{(i)}$ have uniform weights after the resampling step at time $k - 1$.

In conclusion, particle filters are methods to solve complex nonlinear, non-Gaussian online estimation problems. Due to their statistical nature, these methods are applicable to a very large class of models and are straightforward to implement in practical applications.

2.2 Out of Sequence Measurement Problem

Wireless sensor networks can be used in many application domains for target tracking. In multi-sensor tracking systems, all the sensors send their time-stamped measurements to the fusion centre, which combines all the information available at the current time step and determines the current estimated state. However, wireless transmissions frequently experi-

Algorithm 2: Bootstrap Filter

Initialization, $k = 0$.
Sample the particles $x_0^{(i)} \sim p(x_0), (i = 1, \ldots, N)$;
for $k = 1 \ldots T$ **do**
- Propagate particles $\tilde{x}_k^{(i)} \sim p(x_k | x_{k-1}^{(i)}), (i = 1, \ldots, N)$;
- Evaluate the importance weights:

$$\omega_k^{(i)} \propto p(y_k | \tilde{x}_k^{(i)}); \tag{2.41}$$

- Normalize the importance weights:

$$\tilde{\omega}_k^{(i)} = \frac{\omega_k^{(i)}}{\sum_{i=1}^{N} \omega_k^{(i)}}; \tag{2.42}$$

- Resample N particles $(x_k^{(i)}; i = 1, \ldots, N)$ from the set $(\tilde{x}_k^{(i)}; i = 1, \ldots, N)$ according to the importance weights $\tilde{\omega}_k^{(i)}$;

endfor

ence significant delays, especially in adverse environmental conditions. This problem can easily lead to situations where measurements from some sensors arrive at the fusion centre with substantial delay, which means the collected measurements are out of sequence. Such "out-of-sequence measurements" (OOSMs) are common in practical multi-sensor tracking systems. Simply discarding such measurements can waste important information and lead to much poorer tracking performance than if they are incorporated into the tracking algorithm [9, 12].

Several strategies and algorithms have been proposed in the literature for incorporating out-of-sequence measurements when the fusion centre receives them. These vary in their tracking performance but also in their computational and memory requirements. In this section, we introduce some related work on the task of incorporating OOSMs and investigate existing particle filtering algorithms. We discuss the advantages and disadvantages of each approach.

2.2.1 Related Work

There has been a substantial amount of research addressing tracking with out-of-sequence measurements in the last twenty years. The simplest approach is to record all of the filter variables and the measurements for the window of time over which OOSMs are considered useful. The filter can then be restarted at the time step immediately prior to the time step associated with an OOSM and re-run to the current time step. This approach can be used for the Kalman filter, EKF, or the particle filter. The undesirable aspect of this method is that there is a substantial overhead, both in memory and computation. Much of this computation appears wasteful, because a filter has already processed most of the measurements. In a real-time tracking scenario with a resource-constrained tracking platform, it may be impossible to adopt this "re-run" filtering approach, especially for the particle filter which already has significant computational and memory requirements. Nevertheless, the performance of such an algorithm provides a benchmark for other methods that strive to reduce the storage requirements or the computation time.

The initial work on this topic focuses on tracking systems with linear state and measurement models with one-lag OOSMs. The challenge is how to efficiently include a measurement produced at the last time step $k-1$ into a track that has just been updated with current measurement at time step k. The simplest approach is presented in [26], which solves this problem approximately by "backward prediction" or "retrodiction", neglecting the process noise for simplicity. This approach "predicts" the state at time $k-1$ from the current state at time k by using the backward transition equation without process noise. Then the residual error between the prediction and the delayed measurement (OOSM) is used to update the state at the current time step k.

Hilton et al. propose an algorithm that accounts partially for the process noise in [8] (see [27] for an outline of the algorithm and further discussion). Bar-Shalom describes an algorithm that provides exact compensation of the process noise in [28], but this is only suitable for the one-lag OOSM problem. Mallick et al. present an extension of the approach of [28] to multiple lags in [29]. Subsequently, Bar-Shalom et al. propose a one-step solution [9] for the general multiple lags problem, which involves replacing all the measurements by an equivalent measurement. This brings the benefit of lower storage requirements but leads to a very small degradation of mean-square error (MSE) performance. Another approach, which addresses the multi-lag OOSM updating by augmented state smoothing, is discussed

in [30, 31]. Zhang et al. propose two general optimal algorithms to process OOSMs for the cases with linear state and measurement functions in [32]. Both algorithms are optimal in the linear minimum mean-square error (LMMSE) sense and they use different storage information: the first algorithm stores all necessary information but the second one only stores the information available at the current time.

OOSM problems were first considered in connection to particle filters and more general filtering problems in [11]. Orton et al. propose an approach that employs the sets of particles before and after the time step of the delayed measurement to update the current weights of particles. This method is improved with a Markov chain Monte Carlo (MCMC) smoothing step to mitigate the potential problem of degeneracy in [33]. The OOSM particle filters proposed in [11, 33] need to store all of the particles of the last l steps, where l is the predetermined maximum number of lags. When a large number of particles is needed for accurate tracking, this can lead to an excessive consumption of storage resources. To address this, Mallick et al. propose an approximate OOSM particle filter that only stores the mean and covariance matrix of all the particles [34]. When the filter receives an OOSM, it retrodicts (predicts backwards) the particles to the time step generating the OOSM and calculates the likelihood to update the weights of current particles.

These three OOSM particle filtering algorithms are only applicable if the system dynamics can be modeled by a linear system. The methods rely on "backwards prediction" to predict the state [11, 33] or particles [34] at the time step associated with the OOSM from the particles at subsequent time steps. This requires an inversion of the system dynamics, which is only possible for linear state dynamics. In subsequent work, both for OOSM processing [12] and particle smoothing [35], researchers have identified procedures for approximate inversion for some non-linear systems; it is possible to incorporate these techniques to extend the applicability of the algorithms in [11, 33, 34].

In [12], Orguner et al. focus on developing strategies to reduce both the memory requirements and computational complexity of OOSM particle filters. They propose a number of "storage efficient particle filters" for out-of-sequence measurement processing. These particle filters only store statistics (single mean and covariance) of the particle set, rather than the particles themselves, at previous time steps. Auxiliary fixed point smoothers are then employed to determine the likelihood of the delayed measurement conditioned on each particle in the current set, and this likelihood is used to update the weight of each particle. The authors compare the performance of the algorithms using three types of

smoothers: Extended Kalman Smoother (EKS), Unscented Kalman Smoother (UKS) and Particle Smoother (PS). The authors observe in their simulation experiments that the EKS performs at least as well as the more computationally complex PS, so advocate its use. These experiments address highly non-linear filtering problems where the extended Kalman filter often fails to track the target. The EKS is successful because the particle filter provides it with very valuable side-information.

2.2.2 Problem Statement

We now provide a mathematical formulation of the filtering problem when OOSMs are possible. We consider the scenario when the state dynamics are Markovian and (possibly) non-linear with additive Gaussian noise. The observations are described by (possibly) non-linear functions of the current state with additive Gaussian noise. At each time step k, there is an active set of distributed sensors, $\mathcal{V}_k$, that make measurements. These measurements are relayed to the fusion centre. A subset of them $\mathcal{S}_k$ experience minimal delay and can be processed at time k. Other measurements are delayed and only become available for processing at later timesteps. Measurements that are delayed by more than l timesteps are considered to be lost and are ignored.

The system is described by the following equations:

$$x_k = f_{k|k-1}(x_{k-1}) + v_{k|k-1} \tag{2.43}$$

$$y_k^j = h_k^j(x_k) + s_k^j \quad (\forall j \in \mathcal{V}_k) \tag{2.44}$$

$$\mathcal{Y}_k = \{y_k^{\mathcal{S}_k} : \mathcal{S}_k \subseteq \mathcal{V}_k\} \tag{2.45}$$

$$\mathcal{Z}_k = \{y_{k-l}^{\mathcal{D}_{k-l,k}}, y_{k-l+1}^{\mathcal{D}_{k-l+1,k}}, \ldots, y_{k-1}^{\mathcal{D}_{k-1,k}}\}. \tag{2.46}$$

Here $\{x_k\}$ denotes the state sequence, which is a Markov process with initial distribution $x_0 \sim p(x_0)$, and $\{y_k^j\}$ denotes the measurement sequence at the j-th sensor. $v_{k|k-1}$ is the transition noise with Gaussian distribution $\mathcal{N}(0, V_{k|k-1})$, and s_k^j is the measurement noise with Gaussian distribution $\mathcal{N}(0, Q_k^j)$. The functions $f_{k|k-1}(.)$ and $h_k^j(.)$ are the transition and measurement functions. $\mathcal{Y}_k$ denotes the set of non-delayed measurements received at time k. $\mathcal{Z}_k$ denotes the set of OOSMs received at time k. The set $\mathcal{D}_{\tau,k}$ is the subset of active sensors at time τ whose measurements are received at time step k; $y_{k-l}^{\mathcal{D}_{k-l,k}}$ is the set of measurements made at time $k-l$ that arrive at the fusion centre at time k.

Let $\widetilde{\mathcal{W}}_k^{i:j}$ denotes the set of measurements generated in the interval $[i, j]$ available at the fusion centre by time k. This includes all the non-delayed measurements $\mathcal{Y}_{i:j} = \cup_{m=i}^{j}\mathcal{Y}_m$ and OOSMs $\{y_\tau^{\mathcal{D}_{\tau,m}} \in \mathcal{Z}_m : \tau \in [i, j], m \in [1, k]\}$. We also denote $\mathcal{W}_k^{i:j} = \widetilde{\mathcal{W}}_k^{i:j} \setminus \mathcal{Z}_k$, i.e. the set of all measurements available at time k except those in $\mathcal{Z}_k$. Lastly, let $\mathcal{W}_k^{j} \equiv \mathcal{W}_k^{j:j}$ and $\widetilde{\mathcal{W}}_k^{j} \equiv \widetilde{\mathcal{W}}_k^{j:j}$.

The OOSM filtering task is to form an estimate of the posterior distribution $p(x_k|\widetilde{\mathcal{W}}_k^{1:k})$ (and hence an estimate of the state x_k). Fig.2.1 illustrates the OOSM tracking problem. The top line represents the time sequence when measurements are produced, and the bottom line represents the time sequence when measurements are received by the fusion centre. The measurement y_τ produced at time τ arrives at the fusion centre at time k. Thus, y_τ is called the "out-of-sequence measurement" in this case.

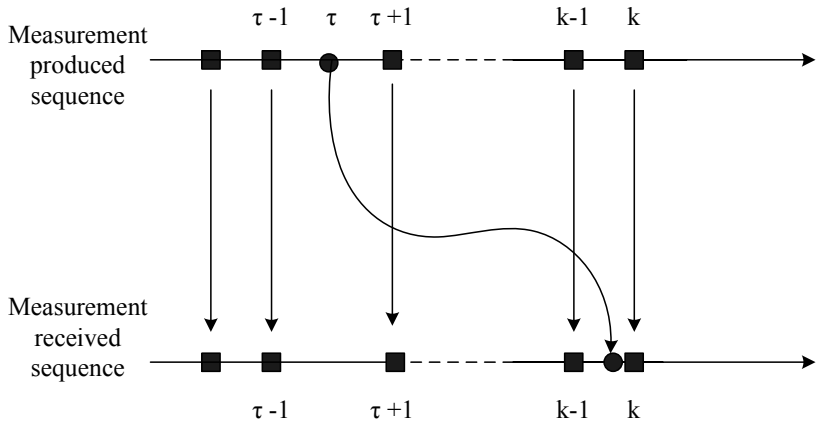


Fig. 2.1 An illustration of the OOSM tracking problem.

2.2.3 OOSM Particle Filters

Algorithm 3 summarizes a generic OOSM particle filtering algorithm. If there are no OOSMs at time k, we write $\mathcal{Z}_k = \emptyset$. Denote, respectively, by $\xi_k \equiv \{x_k^{(i)}, (i = 1, \ldots, N)\}$, $\omega_k \equiv \{\omega_k^{(i)}, (i = 1, \ldots, N)\}$ the sets of the values and weights of particles at time k. We save the particles or their statistics in the last l time steps into the stored set Ω_k. The received measurements available up to time k are stored in $\widetilde{\mathcal{W}}_k^{k-l:k}$, which includes $\mathcal{Z}_k$ and $\mathcal{Y}_k$.

Algorithm 3: Generic OOSM Particle Filter

At time k

Input: ξ_{k-1}, ω_{k-1}, Ω_{k-1}, $\widetilde{\mathcal{W}}_k^{k-l:k}$

if $\mathcal{Z}_k = \emptyset$ **then**

    $(\xi_k, \omega_k) \leftarrow$ `ParticleFilter(`$\mathcal{Y}_k$, ξ_{k-1}, ω_{k-1}`)` ;

    $(\Omega_k) \leftarrow$ `Save(`ξ_k, ω_k, Ω_{k-1}`)` ;

else

    $(\xi_k, \omega_k, \Omega_k\ ) \leftarrow$ `ProcessOOSM(`ξ_{k-1}, ω_{k-1}, Ω_{k-1}, $\widetilde{\mathcal{W}}_k^{k-l:k}$`)` ;

endif

In the above algorithm, the function `ParticleFilter` can be any SIR filter. As mentioned before, several methods have been proposed for addressing the OOSM problem using augmented particle filters. These techniques differ in storage requirements (function `Save` and the stored set Ω_k) and also in how they incorporate the out-of-sequence measurements $\mathcal{Z}_k$, i.e. in the nature of the function `ProcessOOSM`. Function `Save` stores the particles or the statistics of current particles into the stored set Ω_k. Function `ProcessOOSM` updates the current particles and the stored set by the OOSMs in $\mathcal{Z}_k$. In the following sections, we provide more detailed description of three candidate methods. The first is the most intuitive method, which we call the "OOSM re-run particle filter", which reprocesses all the measurements since the associated time-step of the OOSM. The other two algorithms operate by using OOSMs to adjust the weights of the current particles. The backwards prediction algorithm is proposed in [11] and extended in [12]. The storage efficient particle filter is proposed in [12].

OOSM Re-run Particle Filter (OOSM-rerun)

Re-running the particle filter is the most obvious method to process OOSMs. This particle filter needs to save all the particles $\xi_{k-l-1:k}$ in the last $l+1$ time steps into Ω_k and all the available measurements in the last l time steps into $\widetilde{\mathcal{W}}_k^{k-l:k}$, where l is the predetermined maximum number of delayed steps. When it receives $\mathcal{Z}_k \neq \emptyset$ at time k, it returns to the time step $\widetilde{\tau}_k - 1$ (let $\widetilde{\tau}_k$ denotes the earliest time step of all OOSMs in $\mathcal{Z}_k$), then it propagates the particles from $\xi_{\widetilde{\tau}_k - 1}$ to the time step $\widetilde{\tau}_k$ and runs the filter as a standard particle filter using all stored measurements $\widetilde{\mathcal{W}}_k^{\widetilde{\tau}_k:k}$. At each time step, it updates all the

particles stored in Ω_k.

Algorithm 4 provides pseudocode for the function `ProcessOOSM` of the OOSM re-run particle filter. In this algorithm, the function `ParticleFilter` can be any SIR filter. Thus we also assume that all the particles are resampled at the end of each time step (all the particles have equal weights) and we can only store the values of particles into Ω_k. The function `SaveParticles` stores the updated particles into the stored set Ω_k.

The OOSM re-run particle filter has high storage requirements ($l*N$ particles) and high computational cost. On the other hand, it exhibits the best tracking performance among on-line OOSM particle filters. It thus acts as a useful benchmark for other methods that reduce computational overhead or memory requirements.

Algorithm 4: ProcessOOSM by OOSM-rerun

Input: ξ_{k-1}, ω_{k-1}, Ω_{k-1}, $\widetilde{\mathcal{W}}_k^{k-l:k}$

1 $\widetilde{\tau}_k = \min_{\tau}\{\tau : y_\tau \in \mathcal{Z}_k\}$;
2 $\xi_{\widetilde{\tau}_k-1}$ from Ω_{k-1} ;
3 $\omega_{\widetilde{\tau}_k-1}^{(i)} = 1/N,\ i = 1 \dots N$;
4 **for** $j = \widetilde{\tau}_k, \dots, k$ **do**
5 $(\xi_j, \omega_j) \leftarrow$ `ParticleFilter(`$\widetilde{\mathcal{W}}_k^j$, ξ_{j-1}, ω_{j-1}`)`;
6 $(\Omega_k) \leftarrow$ `SaveParticles(`ξ_j`)` ;
7 **endfor**

OOSM Particle Filter using Particle Smoothing (OOSM-PS)

This approach is proposed in [11] for systems with linear dynamics and extended in [12] to (mildly) nonlinear systems. In this case, we consider only a single OOSM arriving at time k, $\mathcal{Z}_k = \{y_\tau \equiv y_\tau^{\mathcal{D}_{\tau,k}}\}$. The basic idea is that only the particles immediately before and after the time step τ of the OOSM are used to form an estimate of the distribution of x_τ instead of reprocessing all the particles from τ to k. The details are as follows.

Consider a scenario where all measurements from time 1 to time k have been received, except for y_τ. Denote this measurement set as $y_{1:k,\bar{\tau}}$. The joint posterior density of the target trajectory $p(x_{0:k}|y_{1:k})$ can be written into factorized form due to the Markovian

property of x_k and the conditional independence of y_k:

$$p(x_{0:k}|y_{1:k}) = p(x_0)p(x_1|x_0, y_1)p(x_2|x_1, y_2)\dots p(x_k|x_{k-1}, y_k) \tag{2.47}$$

$$= p(x_0)\prod_{m=1}^{k} p(x_m|y_m, x_{m-1}). \tag{2.48}$$

Denote by a and b the time-steps immediately before and after τ, respectively. Employing Bayes' Theorem, we can derive the following expression:

$$\begin{aligned} p(x_{0:k,\bar{\tau}}|y_{1:k,\bar{\tau}}) = &\left\{ p(x_0)\prod_{m=1}^{b-1} p(x_m|y_m, x_{m-1}) \right\} p(x_b|y_b, x_{b-1}) \\ &\times p(x_a|y_a, x_b)\left\{ \prod_{m=a+1}^{k} p(x_m|y_m, x_{m-1}) \right\} \end{aligned} \tag{2.49}$$

$$\begin{aligned} = &\, p(x_{0:b-1}|y_{1:b-1})\frac{p(y_b|x_b)p(x_b|x_{b-1})}{p(y_b)} \\ &\times \frac{p(y_a|x_a)p(x_a|x_b)}{p(y_a)} p(x_{a+1:k}|y_{a+1:k}). \end{aligned} \tag{2.50}$$

Here $x_{0:k,\bar{\tau}}$ denotes the state sequence from time 0 to k, skipping over the state value at τ. Similarly, the posterior distribution $p(x_{0:k}|y_{1:k})$, which incorporates the measurement and state at time τ, can be expanded to the following form:

$$p(x_{0:k}|y_{1:k}) = p(x_{0:b}|y_{1:b})\frac{p(y_\tau|x_\tau)p(x_\tau|x_b)}{p(y_\tau)}\frac{p(y_a|x_a)p(x_a|x_\tau)}{p(y_a)} p(x_{a+1:k}|y_{a+1:k}) \tag{2.51}$$

$$= \frac{p(y_\tau|x_\tau)p(x_\tau|x_b)p(x_a|x_\tau)}{p(x_a|x_b)p(y_\tau)} p(x_{0:b}|y_{1:b})p(x_{a:k}|y_{a:k}) \tag{2.52}$$

$$= \frac{p(y_\tau|x_\tau)p(x_\tau|x_b)p(x_a|x_\tau)}{p(x_a|x_b)p(y_\tau)} p(x_{0:k,\bar{\tau}}|y_{1:k,\bar{\tau}}) \tag{2.53}$$

$$\propto p(x_\tau|x_b, x_a)p(y_\tau|x_\tau)p(x_{0:k,\bar{\tau}}|y_{1:k,\bar{\tau}}). \tag{2.54}$$

If a particle filter was executed using the measurements $y_{1:k,\bar{\tau}}$, then the result would be a particle distribution at time k with weights $\omega^{(i)}_{k,\bar{\tau}}$. In [11], Orton et al. proposed a procedure for updating these weights to take into account y_τ, based on (2.54). Recall that $\pi(x_{0:k}|y_{1:k})$

denotes the importance distribution. The updating formula for the particle weights is then:

$$\omega_k^{(i)} = \omega_{k,\bar{\tau}}^{(i)} \frac{p(y_\tau|x_\tau^{(i)})p(x_\tau^{(i)}|x_b^{(i)}, x_a^{(i)})}{\pi(x_\tau^{(i)}|x_{0:k}^{(i)}, y_{1:k})}. \tag{2.55}$$

Based on the discussion in [20], $p(x_\tau|x_b^{(i)}, x_a^{(i)}, y_\tau)$ is the optimal importance distribution for this update, in the sense that it minimizes the variance of the importance weights. However, this choice of importance function does not lead to a practical sampling scheme. The choice of $p(x_\tau|x_b^{(i)}, x_a^{(i)})$ leads to a simple weight update formula:

$$\omega_k^{(i)} = \omega_{k,\bar{\tau}}^{(i)} \, p(y_\tau|x_\tau^{(i)}). \tag{2.56}$$

The remaining challenge is sampling from $p(x_\tau|x_b^{(i)}, x_a^{(i)})$. Orton et al. present a method in [11, 33] for the case of linear state updates and Gaussian noise, where the transition can be described as:

$$x_k = F_{k|k-1}x_{k-1} + V_{k|k-1}^{1/2}u_k. \tag{2.57}$$

Here $F_{k|k-1}$ is the transition matrix from time $k-1$ to k, $V_{k|k-1}$ is the covariance matrix for the transition noise, and u_k is a Gaussian vector. The special structure of these dynamics allow for the construction of reverse dynamics, so that we can write:

$$x_\tau = F_B x_b + F_A x_a + V_\tau^{1/2}u_\tau, \tag{2.58}$$

where

$$V_\tau = (V_{\tau|b}^{-1} + F_{a|\tau}^T V_{a|\tau}^{-1} F_{a|\tau})^{-1} \tag{2.59}$$

$$F_B = V_\tau V_{\tau|b}^{-1} F_{\tau|b} \tag{2.60}$$

$$F_A = V_\tau F_{a|\tau}^T V_{a|\tau}^{-1}. \tag{2.61}$$

Generating samples from $p(x_\tau|x_b^{(i)}, x_a^{(i)})$ can then be easily achieved by applying (2.58).

Orguner et al. propose a method for generalizing this scheme in [12]. The approach is suitable for systems with dynamics of the form:

$$x_k = f_{k|k-1}(x_{k-1}) + v_{k|k-1},$$

where $f_{k|k-1}$ is a non-linear continuously differentiable function and $v_{k|k-1}$ is the innovation noise. If $v_{k|k-1}$ is Gaussian, i.e., of the form $v_{k|k-1} = V_{k|k-1}^{1/2} u_k$, then the EKF or the UKF can be used to generate samples x_τ. The technique can be applied even if $v_{k|k-1}$ is not Gaussian, but this then involves approximating the distribution using a Gaussian, introducing another potential source of approximation error.

In the case where $v_{k|k-1}$ is Gaussian, the following relationship holds:

$$p(x_\tau|x_b^{(i)}) = \mathcal{N}(x_\tau; f_{\tau|b}(x_b^{(i)}), R_{\tau|b}^{(i)}) \tag{2.62}$$

$$R_{\tau|b}^{(i)} = F_{\tau|b}^{(i)} V_{\tau|b} {F_{\tau|b}^{(i)}}^T + V_{\tau|b} \tag{2.63}$$

$$F_{\tau|b}^{(i)} = \frac{\partial}{\partial x} f_{\tau|b}(x)|_{x=x_b^{(i)}}. \tag{2.64}$$

Following [12], this allows one to derive the following expression based on EKF update equations:

$$p(x_\tau|x_b^{(i)}, x_a^{(i)}) = \mathcal{N}(x_\tau; \mu_\tau^{(i)}, R_\tau^{(i)}), \tag{2.65}$$

where

$$\mu_\tau^{(i)} = f_{\tau|b}(x_b^{(i)}) + K_\tau^{(i)}(x_a^{(i)} - f_{a|\tau}(f_{\tau|b}(x_b^{(i)}))) \tag{2.66}$$

$$R_\tau^{(i)} = R_{\tau|b}^{(i)} - K_\tau^{(i)}(F_{a|\tau}^{(i)} R_{\tau|b}^{(i)} {F_{a|\tau}^{(i)}}^T + V_{a|\tau}) {K_\tau^{(i)}}^T \tag{2.67}$$

$$K_\tau^{(i)} = R_{\tau|b}^{(i)} {F_{a|\tau}^{(i)}}^T (F_{a|\tau}^{(i)} R_{\tau|b}^{(i)} {F_{a|\tau}^{(i)}}^T + V_{a|\tau})^{-1} \tag{2.68}$$

$$F_{a|\tau}^{(i)} = \frac{\partial}{\partial x} f_{a|\tau}(x)|_{x=f_{\tau|b}(x_b^{(i)})}. \tag{2.69}$$

Sampling from the Gaussian distribution in (2.65) is then a relatively simple task.

The OOSM particle filter using particle smoothing requires the storage of all particles over the last $l+1$ steps. Thus it needs the same amount of storage as OOSM re-run particle filter. The computational requirements are approximately equivalent to running a particle filter for two time steps, so there is a reduction compared to the re-run particle filter, where each OOSM requires a particle filter to be run for $k - \tau + 1$ time steps.

Storage Efficient Particle Filter with Extended Kalman Smoother (SEPF-EKS)

In [12], Orguner et al. propose "storage efficient OOSM particle filters". These particle filters only store the statistics (single mean and covariance) of the previous particle sets, rather than the particles themselves. Denote, respectively, by ξ_k, ω_k the sets of the values and weights of particles at time k, and let μ_k, R_k denote their mean and covariance. The stored information then include all the available measurement up to current time step k, $\mathcal{W}_k^{k-l:k}$ and

$$\Omega_k = \{\mu_{k-l-1:k}, R_{k-l-1:k}\}, \tag{2.70}$$

which stores the means and covariances of previous particle sets for $l+1$ time steps (where l is the predetermined maximum number of delayed steps).

The computational requirements are also reduced, compared to the re-run particle filter, by using an auxiliary fixed point smoother to update the weights of current particles by the OOSMs. Orguner et al. explore the use of three kinds of smoothers: the Extended Kalman Smoother (EKS) [36], the Unscented Kalman Smoother (UKS) [37] and the Particle Smoother (PS) [12]. Based on the simulation results reported in [12] and our own experiments, the EKS approach is the best choice, achieving good tracking performance and not imposing a substantial computational burden. We now review the storage efficient OOSM particle filter that employs EKS, as presented in [12]. In this case, we consider only a single OOSM arriving at time k, $\mathcal{Z}_k = \{y_\tau \equiv y_\tau^{\mathcal{D}_{\tau,k}}\}$. Thus all measurements from time 1 to time k have been received, except for y_τ. Denote this measurement set as $y_{1:k,\bar{\tau}}$.

The approach is based on the following expression for the posterior:

$$p(x_k|y_{1:k}) = \frac{p(y_\tau|x_k, y_{1:k,\bar{\tau}})}{p(y_\tau|y_{1:k,\bar{\tau}})} p(x_k|y_{1:k,\bar{\tau}}). \tag{2.71}$$

Substituting the particle approximation of $p(x_k|y_{1:k,\bar{\tau}})$ into it, we have:

$$p(x_k|y_{1:k}) = \sum_{i=1}^{N} \frac{p(y_\tau|x_k^{(i)}, y_{1:k,\bar{\tau}})}{p(y_\tau|y_{1:k,\bar{\tau}})} \omega_{k,\bar{\tau}}^{(i)} \delta(x_k - x_k^{(i)}) \tag{2.72}$$

$$= \sum_{i=1}^{N} \omega_k^{(i)} \delta(x_k - x_k^{(i)}), \tag{2.73}$$

$$\text{where} \qquad \omega_k^{(i)} \propto p(y_\tau|x_k^{(i)}, y_{1:k,\bar{\tau}}) \omega_{k,\bar{\tau}}^{(i)}. \tag{2.74}$$

The remaining challenge is to form an approximation of the likelihood $p(y_\tau|x_k^{(i)}, y_{1:k,\bar{\tau}})$. This likelihood can be expressed as:

$$p(y_\tau|x_k^{(i)}, y_{1:k,\bar{\tau}}) = \int p(y_\tau|x_\tau) p(x_\tau|x_k^{(i)}, y_{1:k,\bar{\tau}}) \, \mathrm{d}x_\tau \tag{2.75}$$

$$= \int p(y_\tau|x_\tau) p(x_\tau|x_k^{(i)}, y_{1:k-1,\bar{\tau}}) \, \mathrm{d}x_\tau. \tag{2.76}$$

Here, (2.75) follows from (2.76) because $x_k^{(i)}$ already incorporates the information provided by y_k.

Assume that $p(x_\tau|x_k^{(i)}, y_{1:k-1,\bar{\tau}})$ can be approximated by a single Gaussian (this assumption is likely to hold in many cases because the density is conditioned on future measurements as well as a future state value, which often acts to eliminate any multi-modal structure). Denote this approximating Gaussian as:

$$p(x_\tau|x_k^{(i)}, y_{1:k-1,\bar{\tau}}) \approx \mathcal{N}(x_\tau; \mu^x_{\tau|1:k-1,\bar{\tau};k^{(i)}}, R^x_{\tau|1:k-1,\bar{\tau};k^{(i)}}). \tag{2.77}$$

Here $\mu^x_{\tau|1:k-1,\bar{\tau};k^{(i)}}$ and $R^x_{\tau|1:k-1,\bar{\tau};k^{(i)}}$ denote, respectively, the mean and covariance of the approximating Gaussian (the subscript notation is used to denote that these are based on measurements $y_{1:k-1,\bar{\tau}}$ and the state (particle) $x_k^{(i)}$).

An EKF approximation of $p(y_\tau|x_\tau)$ can now be employed to construct an estimate of the likelihood:

$$p(y_\tau|y_{1:k,\bar{\tau}}, x_k^{(i)}) = \mathcal{N}(y_\tau; \mu^y_{\tau|1:k,\bar{\tau};k^{(i)}}, R^y_{\tau|1:k,\bar{\tau};k^{(i)}}), \tag{2.78}$$

where

$$\mu^y_{\tau|1:k,\bar{\tau};k^{(i)}} = h_\tau(\mu^x_{\tau|1:k-1,\bar{\tau};k^{(i)}}) \tag{2.79}$$

$$R^y_{\tau|1:k,\bar{\tau};k^{(i)}} = H_\tau R^x_{\tau|1:k-1,\bar{\tau};k^{(i)}} H_\tau^T + Q_\tau \tag{2.80}$$

$$H_\tau = \frac{\partial}{\partial x} h_\tau(x)|_{x=\mu^x_{\tau|1:k-1,\bar{\tau};k^{(i)}}}. \tag{2.81}$$

The final task is then to obtain the approximation in (2.77). By partitioning $y_{1:k-1,\bar{\tau}}$, we can write $p(x_\tau|x_k^{(i)}, y_{1:k-1,\bar{\tau}})$ as:

$$p(x_\tau|x_k^{(i)}, y_{1:k-1,\bar{\tau}}) = \frac{p(x_k^{(i)}, y_{\tau+1:k-1}|x_\tau)}{p(x_k^{(i)}, y_{\tau+1:k-1}|y_{1:\tau-1})} p(x_\tau|y_{1:\tau-1}). \tag{2.82}$$

We can interpret (2.82) as a Bayesian update equation where both $y_{\tau+1:k-1}$ and the particle $x_k^{(i)}$ are considered as measurements related to the state x_τ.

Recall that $\mathcal{W}_k^{\tau:m}$ is the set of measurements made at time steps ranging from τ to m that have been received at the fusion centre by time step k except for the OOSMs received at time k (in this single case, just y_τ). The storage efficient particle filter maintains a Gaussian approximation of $p(x_m|\mathcal{W}_k^{\tau:m})$ for all m in the range $k-l-1,\ldots,k$. We denote these distributions by $\mathcal{N}(\mu_{m|m}, R_{m|m})$. Here the standard predictive/update notation is employed, i.e. $\mu_{m|m}$ is the estimate of the mean at time m using $\mathcal{W}_k^{\tau:m}$.

Since we store the means and covariances of previous particles in Ω_k, with these Gaussian approximations in hand, the storage efficient particle filter can readily form an approximation of $p(x_\tau|y_{1:\tau-1})$ by applying the state equations describing the dynamics of the system. Incorporating the information in $\{x_k^{(i)}, y_{\tau+1:k-1}\}$ to obtain $p(x_\tau|x_k^{(i)}, y_{1:k-1,\bar{\tau}})$ can now be identified as a fixed point smoothing problem, and standard techniques from the literature can be applied.

In Appendix A.1, we present a complete algorithmic description of function `ProcessOOSM` for the storage-efficient OOSM particle filter that employs the Extended Kalman Smoother (EKS) approach, which was originally described in [36]. The Unscented Kalman Smoother (UKS) or the Particle Smoother (PS) are alternatives and their application is described in [12].

This approach saves much storage by representing the distribution of particles by Gaussian approximation. Moreover, it reprocesses the mean and covariance matrix instead of

reprocessing all the particles. Thus, the computational complexity is less than the OOSM re-run particle filter in most cases. However, it is difficult to efficiently extend the SEPF-EKS algorithm to process batches of OOSMs. We have to use loops to process them separately. In each cycle, we can get the updated weights based on only one OOSM. Therefore, the computational savings diminish when it is common for multiple OOSMs to arrive in a given time step.

Another point worth noting is that this algorithm is vulnerable to highly informative OOSMs. After processing an OOSM that should lead to a major change in the filtering distribution, the effective number of particles (measured by $1/\sum(\omega^{(i)})^2$) can be greatly diminished. This reduces sample diversity in the particle filter and can cause significant performance deterioration, which will be discussed in more detail in the next chapter.

Chapter 3

Efficient Delay-tolerant Particle Filter through Selective Processing of Out-of-sequence Measurements (OOSMs)

In this chapter, we propose two new OOSM particle filters. The first algorithm uses Gaussian approximations to construct a storage-efficient version of the re-run particle filter. The second algorithm uses Gaussian approximations to reduce the memory requirements and employs selective processing of OOSMs to reduce the high computational cost. We propose a selection rule based on mutual information to estimate the informativeness of the OOSMs and immediately discard uninformative measurements. The more informative measurements are processed using the storage efficient particle filter proposed by Orguner et al. in [12]. This filter sometimes fails when a measurement is extremely informative, so our algorithm applies a second test to detect these highly-informative OOSMs, and incorporates them by re-running the particle filter from the time-step of the OOSM.

Subsequently, we use a simulation scenario to compare the performance of the proposed algorithms and some of the methods surveyed in Chapter 2. Our simulation experiments provide an example tracking scenario where the proposed algorithm processes only 30-40% of all OOSMs using the storage efficient particle filter and 1-3% of OOSMs by rerunning the particle filter. By doing so, it requires less computational resources but achieves greater

accuracy than the storage efficient particle filter proposed in [12]. Finally, we examine the computational complexity of each algorithm by comparing the running time in MATLAB.

3.1 OOSM Gaussian Approximation Re-run Particle filter (OOSM-GARP)

This particle filter is a storage-efficient version of the OOSM re-run particle filter. To reduce memory requirements, we use Gaussian approximations to represent the estimated posterior distributions of particles at previous time steps, in a similar way as the storage efficient particle filter of [12]. Thus, this particle filter only saves the mean μ_m and covariance matrix R_m of the particles $\xi_m = \{x_m^{(i)}, i = 1, \ldots, N\}$ at each time step from $k-l-1$ to k, where l is the predefined maximum delay. The stored information then includes all the available measurements up to current time step k, $\mathcal{W}_k^{k-l:k}$, and Ω_k, where

$$\Omega_k = \{\mu_{k-l-1:k}, R_{k-l-1:k}\}. \tag{3.1}$$

The algorithm needs much less storage resources than the OOSM re-run particle filter, which must store all the particles from $k-l-1$ to k. Recall that $\mathcal{Z}_k$ denotes the set of received OOSMs at time k. When OOSM-GARP receives $\mathcal{Z}_k$ at time k, it returns to the time step $\widetilde{\tau}_k - 1$ (let $\widetilde{\tau}_k$ denotes the earliest time step of all OOSMs in $\mathcal{Z}_k$). It samples particles from $\mathcal{N}(\mu_{\widetilde{\tau}_k-1}, R_{\widetilde{\tau}_k-1})$, propagates them to the time step $\widetilde{\tau}_k$ and runs the filter as standard particle filter using $\widetilde{\mathcal{W}}_k^{\widetilde{\tau}_k:k}$, which denotes all the available measurements up to time k and generated in the interval $[\widetilde{\tau}_k, k]$. At each step, it updates the mean and covariance matrix in the stored set Ω_k as described in Algorithm 5. In this algorithm, the function `ParticleFilter` can be any standard SIR filter. If the set of non-delayed measurements $\mathcal{Y}_k = \emptyset$, `ParticleFilter` only propagates the particles and skips the measurement processing step. The function `SaveGauss` uses the maximum likelihood estimator of the mean and covariance given the weighted sample set ξ_k, ω_k:

$$\mu_k = \sum_{i=1}^{N} \omega_k^{(i)} x_k^{(i)} \tag{3.2}$$

$$R_k = \sum_{i=1}^{N} \omega_k^{(i)} (x_k^{(i)} - \mu_k)(x_k^{(i)} - \mu_k)^T. \tag{3.3}$$

Algorithm 5: ProcessOOSM-GARP

Input: Ω_{k-1}, $\widetilde{\mathcal{W}}_k^{k-l:k}$

$\widetilde{\tau}_k = \min_{\tau}\{\tau : y_\tau \in \mathcal{Z}_k\}$;

$\{x^{(i)}_{\widetilde{\tau}_k-1}\}_{i=1}^N \sim \mathcal{N}(x_{\widetilde{\tau}_k-1}, \mu_{\widetilde{\tau}_k-1}, R_{\widetilde{\tau}_k-1})$;

$\omega^{(i)}_{\widetilde{\tau}_k-1} = 1/N,\ i = 1 \dots N$;

for $j = \widetilde{\tau}_k, \dots, k$ **do**

    $(\xi_j, \omega_j) \leftarrow$ `ParticleFilter(`$\widetilde{\mathcal{W}}_k^j$, ξ_{j-1}, ω_{j-1}`)`;

    $(\Omega_k) \leftarrow$ `SaveGauss(`ξ_j, ω_j`)` ;

endfor

In most cases, Gaussian approximation can adequately represent the statistics of particle clouds [1], so OOSM-GARP can achieve similar performance as the OOSM re-run particle filter but with much less storage. However, OOSM-GARP still requires high computational cost since it needs to reprocess all the particles from the time step when the earliest OOSM was measured.

3.2 Efficient Delay-tolerant Particle Filter through Selective Processing of OOSMs (EDPF-SP)

In [12], Orguner et al. mentioned that the storage efficient particle filters have one critical problem that, at some times, an OOSM update would cause significant performance deterioration when it greatly reduced the effective number of particles (measured by $1/\sum(\omega^{(i)})^2$). In [12], the storage efficient particle filters choose to discard such OOSMs. However, after more investigation by simulation experiments, we find that such OOSMs are highly informative. When the fusion center always can get informative measurements from sensors (see 3.3.3 Example 1), this operation of discarding an OOSM is reasonable; the performance of the SEPF-EKS is similar to that of the OOSM re-run particle filter. However, when the fusion center cannot always get sufficient measurements (see 3.3.3 Example 2), discarding such OOSMs leads to a significant deterioration in performance. The storage efficient particle filter can fail when a measurement is extremely informative, because it does not change the locations of particles but just updates their weights. It thus cannot address

[1] It is important to remember that the Gaussian is only used as an initialization distribution of the re-running process.

situations in which the current particle filter distribution should be changed significantly because of the new out-of-sequence measurement. In such settings, it is better to use the re-run particle filter to reprocess the particles.

Algorithm 6: ProcessOOSM-SP

Input: Ω_{k-1}, $\widetilde{\mathcal{W}}_k^{k-l:k}$

$(\xi_k, \omega_k) \leftarrow$ `ParticleFilter` $(\mathcal{Y}_k, \xi_{k-1}, \omega_{k-1})$;
$(\Omega_k) \leftarrow$ `SaveGauss`(ξ_k, ω_k) ;
EKSfailed = 0;
for $y_\tau \in \mathcal{Z}_k$ **do**
    $I_{y_\tau} \leftarrow$ `CalcMI`$(y_\tau, \mu_\tau, R_\tau, H_\tau)$;
    if $I_{y_\tau} < \gamma_1$ **then**
        discard y_τ ;
    else
        $N_{\text{eff}}^{\text{prior}} = 1/\sum_{i=1}^{N}(\omega_k^{(i)})^2$;
        $(\xi_k, \omega_k) \leftarrow$ `ProcessOOSM-EKS`$(y_\tau, \xi_k, \omega_k, \Omega_k)$;
        $N_{\text{eff}}^{\text{post}} = 1/\sum_{i=1}^{N}(\omega_k^{(i)})^2$;
        if $N_{\text{eff}}^{\text{post}}/N_{\text{eff}}^{\text{prior}} < \gamma_2$ **then**
            EKSfailed = 1 ;
            break ;
        else
            $(\Omega_k) \leftarrow$ `SaveGauss`(ξ_k, ω_k) ;
        endif
    endif
endfor
if *EKSfailed* **then**
    $(\xi_k, \omega_k, \Omega_k) \leftarrow$ `ProcessOOSM-GARP`$(\widetilde{\mathcal{W}}_k^{k-l:k}, \Omega_k)$;

On the other hand, some OOSMs are uninformative — discarding them does not affect the performance of the algorithm. Thus, we think it is worthwhile to process the OOSMs according to their informativeness. We now propose a simple but effective two-stage algorithm that processes only the informative OOSMs, leading to significantly increased computational efficiency. The first stage of the algorithm estimates the informativeness of an OOSM and discards those deemed uninformative. In the second stage, the informa-

tive OOSMs are processed by SEPF-EKS (Section 2.2.3). If significant reduction of the effective sample size is detected after the application of SEPF-EKS, we choose to apply OOSM-GARP. The approach combines the advantages of OOSM-GARP and SEPF-EKS in order to achieve a satisfactory tradeoff between performance and complexity. We describe the proposed approach `ProcessOOSM-SP` in Algorithm 6. Here, we assume that there are potentially multiple OOSMs in $\mathcal{Z}_k$, any $y_\tau \equiv y_\tau^{\mathcal{D}_{\tau,k}} \in \mathcal{Z}_k$.

In this algorithm, the function `CalcMI` is used to estimate the informativeness of a measurement, and it is discussed in more detail below. The thresholds γ_1 and γ_2 govern the trade-off between computational complexity and accuracy. The first threshold γ_1 determines the proportion of OOSMs that are declared uninformative and immediately discarded. The more informative OOSMs are processed by the function `ProcessOOSM-EKS` (Appendix A.1). The second threshold γ_2 defines the proportion of informative OOSMs that are processed using `ProcessOOSM-GARP` (Algorithm 5), which reruns the particle filter from the time τ when the OOSM was measured. In our experiments we observe that `ProcessOOSM-GARP` can be invoked rarely and yet this substantially improves the overall quality of tracking in some cases.

3.2.1 OOSM Selection Rule

We propose two metrics for assessing the "informativeness" of OOSMs, both based on information-theoretic concepts. For the first metric, the OOSM is treated as a random variable Y_τ, so the metric and decision do not depend on the actual measured value $y_\tau \equiv y_\tau^{\mathcal{D}_{\tau,k}} \in \mathcal{Z}_k$. The first metric is the mutual information between the OOSM Y_τ and the state X_k, $I(Y_\tau, X_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}})$. The mutual information is conditioned on all received measurements in $\mathcal{W}_k^{1:k}$ and the recently received OOSMs in $\mathcal{Z}_k$ except y_τ, denoted by $\mathcal{Z}_{k,\bar{\tau}} = \mathcal{Z}_k \setminus \{y_\tau\}$.

The second metric is the Kullback-Leibler divergence (KL-divergence) [38] between the distribution at time k, conditioned on all measurements except for y_τ, and the distribution at time k conditioned on all measurements including y_τ. This KL-divergence is denoted by $D(p(x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}})\|p(x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_k))$.

Our goal is to estimate the metric quickly and relatively accurately in order to decide whether to process the OOSM. We therefore employ Gaussian approximations to the distributions of interest and use the extended Kalman filter to calculate their parameters. We

now discuss the individual metrics and the procedures used for their estimation.

Mutual Information Metric

The mutual information is defined between measurement Y_τ and state X_k as follows:

$$\begin{aligned} &I(Y_\tau, X_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}}) \\ &= \int \log(\frac{p(y_\tau, x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}})}{p(y_\tau|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}})p(x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}})})p(y_\tau, x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}})\,\mathrm{d}y_\tau\,\mathrm{d}x_k. \end{aligned} \tag{3.4}$$

Thus to calculate the mutual information based test statistic it is sufficient to know the joint distribution $p(y_\tau, x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}})$. The mutual information can also be expressed in terms of conditional entropies H:

$$I(Y_\tau, X_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}}) = H(X_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}}) - H(X_k|Y_\tau, \mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}}). \tag{3.5}$$

We choose to approximate the joint distribution by a Gaussian distribution:

$$p(y_\tau, x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}}) \approx \mathcal{N}(\begin{pmatrix} x_k \\ y_\tau \end{pmatrix}; \begin{pmatrix} \mu_{x_k} \\ \mu_{y_\tau} \end{pmatrix}, \begin{pmatrix} R_{x_k} & R_{x_k y_\tau} \\ R_{y_\tau x_k} & R_{y_\tau} \end{pmatrix}). \tag{3.6}$$

Let us define $R_{x_k|y_\tau} = R_{x_k} - R_{x_k y_\tau} R_{y_\tau}^{-1} R_{y_\tau x_k}$. Standard Gaussian marginalization and conditioning formula lead to the following relationships:

$$H(X_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}}) = \frac{1}{2}\log|2\pi e R_{x_k}| \tag{3.7}$$

$$H(X_k|Y_\tau, \mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}}) = \frac{1}{2}\log|2\pi e R_{x_k|y_\tau}| \tag{3.8}$$

$$I(Y_\tau, X_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}}) = \frac{1}{2}\log\frac{|R_{x_k}|}{|R_{x_k} - R_{x_k y_\tau} R_{y_\tau}^{-1} R_{y_\tau x_k}|}. \tag{3.9}$$

We can devise the following technique for estimating $I(Y_\tau, X_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}})$. We assume that the measurement equation at time τ can be reasonably accurately linearized around the estimate of the state. Defining $H_\tau = \frac{\partial}{\partial x} h_\tau(x)|_{x=\mu_{x_\tau}}$, this implies that $y_\tau \approx H_\tau x_\tau + s_\tau$. See [39] for further discussion about this assumption.

Commencing with the saved distribution at time τ, $p(x_\tau|\mathcal{W}_k^{1:\tau}) \approx \mathcal{N}(x_\tau; \mu_\tau, R_\tau)$ and using the linearization assumption, we can calculate the Gaussian approximation of the

joint distribution at time τ:

$$p(y_\tau, x_\tau | \mathcal{W}_k^{1:\tau}) \approx \mathcal{N}(\begin{pmatrix} x_\tau \\ y_\tau \end{pmatrix}; \begin{pmatrix} \mu_{x_\tau} \\ \mu_{y_\tau} \end{pmatrix}, \begin{pmatrix} R_{x_\tau} & R_{x_\tau y_\tau} \\ R_{y_\tau x_\tau} & R_{y_\tau} \end{pmatrix}), \tag{3.10}$$

where we set $\mu_{x_\tau} = \mu_\tau$, $\mu_{y_\tau} = h_\tau(\mu_\tau)$; $R_{x_\tau} = R_\tau$, $R_{y_\tau} = H_\tau R_\tau H_\tau^T + Q_\tau$, $R_{x_\tau y_\tau} = R_\tau H_\tau^T$, $R_{y_\tau x_\tau} = R_{x_\tau y_\tau}^T$.

We now apply a forward EKF recursion from τ to $k-1$, augmenting the state x with the measurement y_τ, denoted by z with its covariance P. The EKF recursion consists of a prediction step:

$$z_{m+1|m} = \begin{pmatrix} \mu_{x_{m+1|m}} \\ \mu_{y_\tau} \end{pmatrix} = \begin{pmatrix} f_m(\mu_{x_m}) \\ \mu_{y_\tau} \end{pmatrix} \tag{3.11}$$

$$P_{m+1|m} = \begin{pmatrix} F_m & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} R_{x_m} & R_{x_m y_\tau} \\ R_{y_\tau x_m} & R_{y_\tau} \end{pmatrix} \begin{pmatrix} F_m^T & 0 \\ 0 & I \end{pmatrix} + \begin{pmatrix} V_{m+1|m} & 0 \\ 0 & 0 \end{pmatrix}, \tag{3.12}$$

and an update step:

$$r_{m+1} = \widetilde{y}_{m+1} - \widetilde{h}_{m+1}(\mu_{x_{m+1|m}}) \tag{3.13}$$

$$K_{m+1} = P_{m+1|m} \widetilde{H}_{m+1}^T (\widetilde{H}_{m+1} P_{m+1|m} \widetilde{H}_{m+1}^T + \widetilde{Q}_{m+1})^{-1} \tag{3.14}$$

$$z_{m+1} = z_{m+1|m} + K_{m+1} r_{m+1} \tag{3.15}$$

$$P_{m+1} = (I - K_{m+1} \widetilde{H}_{m+1}) P_{m+1|m}. \tag{3.16}$$

In these update equations,

$$\widetilde{y}_{m+1} = \begin{pmatrix} \cdots \\ y_{m+1}^j \\ \cdots \end{pmatrix}$$

is the vector of varying dimensionality that contains stacked measurements from time $m+1$ available at time k, i.e. $\widetilde{y}_{m+1}$ contains y_{m+1}^j if $y_{m+1}^j \in \mathcal{W}_k^{m+1} \cup \mathcal{Z}_k$. Likewise, $\widetilde{h}_{m+1}(\cdot)$ is the corresponding non-linear vector function defined similarly to $\widetilde{y}_{m+1}$. However, $\widetilde{H}_{m+1} = \frac{\partial}{\partial x} \widetilde{h}_{m+1}(x)|_{x=z_{m+1|m}}$ is its linearization with augmented state. $\widetilde{Q}_{m+1}$ is the block-diagonal matrix which describes noise terms corresponding to components y_{m+1}^j of vector $\widetilde{y}_{m+1}$.

Repeated application of the recursion up to time k permits estimation of the joint dis-

tribution $p(y_\tau, x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}})$ and this allows us to estimate the mutual information metric $I(Y_\tau, X_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}})$ using the expressions above.

KL-divergence Metric

The KL-divergence between $p(x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}})$ and $p(x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_k)$ is calculated using the following formula:

$$\begin{aligned} &D(p(x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}})\|p(x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_k)) \\ &= \frac{1}{2}\log\left(\frac{|\widehat{R}_k|}{|R_k|} + \mathrm{tr}(\widehat{R}_k^{-1}R_k) + (\widehat{\mu}_k - \mu_k)^T\widehat{R}_k^{-1}(\widehat{\mu}_k - \mu_k) - d_x\right), \end{aligned} \tag{3.17}$$

where d_x is the dimensionality of the state x_k. In Algorithm 6 we use the symmetrized KL-divergence:

$$I_{y_\tau} = \frac{D(p(x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}})\|p(x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_k)) + D(p(x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_k)\|p(x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}}))}{2}. \tag{3.18}$$

We estimate the KL-divergence using Gaussian approximations:

$$p(x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}}) \approx \mathcal{N}(x_k, \mu_k, R_k) \tag{3.19}$$

$$p(x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_k) \approx \mathcal{N}(x_k, \widehat{\mu}_k, \widehat{R}_k). \tag{3.20}$$

Both distributions are obtained by applying forward EKF recursions starting from time τ. To obtain the latter distribution, we first apply a measurement update step at time τ and then calculate standard EKF recursion. The former distribution is obtained by calculating the standard EKF recursion and excluding the measurement y_τ.

3.3 Simulations

In this section, we present a performance comparison of several OOSM particle filters described in previous sections. We consider the scenario of bearings-only target tracking and examine tracking performance in two cases, one where the OOSMs are informative and another where they are substantially less informative.

3.3.1 Simulation Scenario

We use same experimental scenario as in [12]. In this two-dimensional scenario, a single target makes a clockwise coordinated turn of radius $500m$ with a constant speed $200km/h$. It starts in the y-direction with initial position $[-500m, 500m]$ and is tracked for 40 seconds. The true trajectory is displayed in Fig. 3.1.

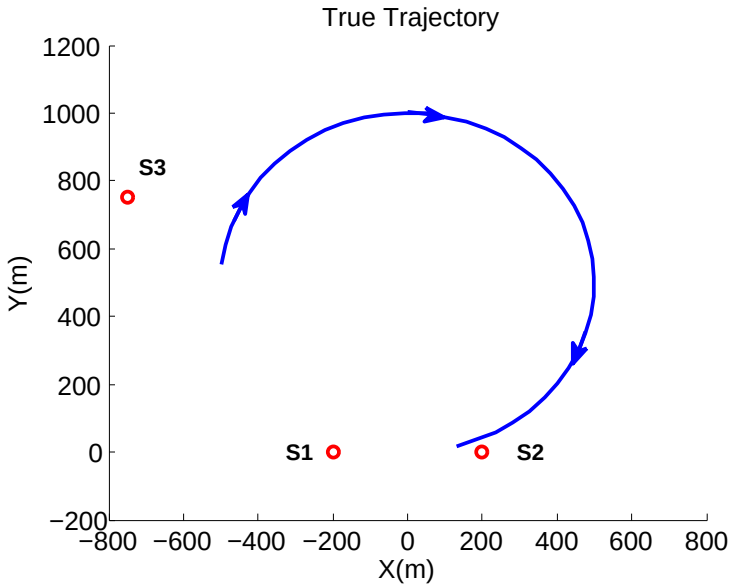


Fig. 3.1 Example 1 and 2 : Target trajectory and the sensors.

The target motion is modeled in the filters by the nearly coordinated turn model [40] with unknown constant turn rate and cartesian velocity. The state of the target is given as $x_k = [p_k^x, p_k^y, v_k^x, v_k^y, \omega_k]^T$, where p, v and ω denote the position, velocity and turn rate respectively. In the simulations, we select the standard deviations for the position, speed and turn rate as $\sigma_p = 30m$, $\sigma_v = 10m/sec$, $\sigma_\omega = 0.1rad/sec$ [2]. At the beginning, we assume that all the filters know very little about the initial state of the target and therefore they are initialized with the state value $x_0 = [0, 0, 0, 0, 0]^T$ and a large covariance $R_0 = \text{diag}([1000^2, 1000^2, 30^2, 30^2, 0.1^2])$ in order to cover the real position of the target.

[2] These values represent reasonable, if somewhat large, levels of uncertainty given the true motion of the target.

The dynamic model is:

$$x_{k+1} = f_{k+1|k}(x_k) + v_{k+1|k}, \tag{3.21}$$

where the coordinated turn (CT) $f_{k+1|k}(.)$ is

$$x_{k+1} = \begin{pmatrix} 1 & 0 & \frac{\sin(\omega_k \Delta t)}{\omega_k} & \frac{\cos(\omega_k \Delta t)-1}{\omega_k} & 0 \\ 0 & 1 & \frac{1-\cos(\omega_k \Delta t)}{\omega_k} & \frac{\sin(\omega_k \Delta t)}{\omega_k} & 0 \\ 0 & 0 & \cos(\omega_k \Delta t) & -\sin(\omega_k \Delta t) & 0 \\ 0 & 0 & \sin(\omega_k \Delta t) & \cos(\omega_k \Delta t) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} x_k \tag{3.22}$$

and $v_{k+1|k}$ is Gaussian process noise with a distribution $\mathcal{N}(0, V_{k+1|k})$. The covariance matrix $V_{k+1|k} = \text{diag}([30^2, 30^2, 10^2, 10^2, 0.1^2])$ for any k. $\Delta t = 1$ is the sampling period.

There are three sensors S1, S2 and S3 sending measurements to a common fusion centre in this scenario. The locations of the three sensors are $[S_1^x, S_1^y] = [-200, 0]$, $[S_2^x, S_2^y] = [200, 0]$, $[S_3^x, S_3^y] = [-750, 750]$ (Fig. 3.1). The measurements are with additive Gaussian noise with zero mean and standard deviation $\sigma_m = 0.05 rads$ for all sensors with the sampling period $\Delta t = 1$. The observation model with measurement function of bearings-only tracking is

$$y_k = h_k(x_k) + s_k \tag{3.23}$$

$$h_k(x_k) = \arctan(\frac{p_k^y - S_j^y}{p_k^x - S_j^x}), \qquad j = 1, 2, 3 \tag{3.24}$$

where $s_k \sim \mathcal{N}(0, Q_k)$ and $Q_k = \text{diag}([\sigma_m^2, \sigma_m^2, \sigma_m^2])$ for any k .

An OOSM arrives at the fusion center with probability p_{osm} and delay t_d. The probability p_{osm} characterizes the reliability of OOSM delivery (a portion of the OOSMs are lost on the way to the fusion centre). The delay t_d is uniformly distributed in the interval $[0, l]$, where l is the predefined maximum delay.

3.3.2 Cramér-Rao Lower Bound

In tracking problems, the Cramér-Rao lower bound (CRLB) provides an indication of performance limitations [41]. In our results, we construct a Cramér-Rao lower bound

based on the derivation in [41], which presented a simple derivation of the posterior CRLB for discrete-time multidimensional nonlinear filtering problems.

Let $\hat{x}_k$ be an estimate of a real state vector x_k. The Cramér-Rao bound specifies a lower bound on the covariance of $\hat{x}_k$:

$$\mathbb{E}[(\hat{x}_k - x_k)(\hat{x}_k - x_k)^T] \geq J_k^{-1}. \tag{3.25}$$

Here J_k is the $d \times d$ Fisher information matrix (d is the dimensionality of the state x_k),

$$J_k = \mathbb{E}[(\nabla_{x_k} \log p(x_k, y_k))(\nabla_{x_k} \log p(x_k, y_k))^T]. \tag{3.26}$$

The Fisher information matrix can be evaluated using the following recursive equation in [41]:

$$J_{k+1} = D_k^{22} - D_k^{21}(J_k + D_k^{11})^{-1} D_k^{12}, \tag{3.27}$$

where D_k^{ij} are given by

$$\begin{aligned} D_k^{11} &= \mathbb{E}\{F_{k+1|k}^T V_{k+1|k}^{-1} F_{k+1|k}\} \\ D_k^{12} &= -\mathbb{E}\{F_{k+1|k}^T\} V_{k+1|k}^{-1} = (D_k^{21})^T \\ D_k^{22} &= V_{k+1|k}^{-1} + \mathbb{E}\{H_{k+1}^T Q_{k+1}^{-1} H_{k+1}\}. \end{aligned} \tag{3.28}$$

Here $F_{k+1|k}$ and H_{k+1} are the Jacobian matrices of the transition function $f_{k+1|k}(x)$ and measurement function $h_{k+1}(x)$, respectively, and $V_{k+1|k}$ and Q_{k+1} are covariance matrices of process noise and measurement noise. The Jacobian of the coordinated turn (CT) model is

$$F_{k+1|k} = \begin{pmatrix} 1 & 0 & \frac{\sin(\omega_k \Delta t)}{\omega_k} & \frac{\cos(\omega_k \Delta t) - 1}{\omega_k} & \frac{\partial p_{k+1}^x}{\partial \omega_k} \\ 0 & 1 & \frac{1 - \cos(\omega_k \Delta t)}{\omega_k} & \frac{\sin(\omega_k \Delta t)}{\omega_k} & \frac{\partial p_{k+1}^y}{\partial \omega_k} \\ 0 & 0 & \cos(\omega_k \Delta t) & -\sin(\omega_k \Delta t) & \frac{\partial v_{k+1}^x}{\partial \omega_k} \\ 0 & 0 & \sin(\omega_k \Delta t) & \cos(\omega_k \Delta t) & \frac{\partial v_{k+1}^y}{\partial \omega_k} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.29}$$

$$\frac{\partial p_{k+1}^x}{\partial \omega_k} = \frac{\omega_k \Delta t \cos(\omega_k \Delta t) - \sin(\omega_k \Delta t)}{\omega_k^2} v_k^x - \frac{\omega_k \Delta t \sin(\omega_k \Delta t) + \cos(\omega_k \Delta t) - 1}{\omega_k^2} v_k^y \quad (3.30)$$

$$\frac{\partial p_{k+1}^y}{\partial \omega_k} = \frac{\omega_k \Delta t \sin(\omega_k \Delta t) + \cos(\omega_k \Delta t) - 1}{\omega_k^2} v_k^x - \frac{\omega_k \Delta t \cos(\omega_k \Delta t) - \sin(\omega_k \Delta t)}{\omega_k^2} v_k^y \quad (3.31)$$

$$\frac{\partial v_{k+1}^x}{\partial \omega_k} = -\Delta t \sin(\omega_k \Delta t) v_k^x - \Delta t \cos(\omega_k \Delta t) v_k^y \quad (3.32)$$

$$\frac{\partial v_{k+1}^y}{\partial \omega_k} = -\Delta t \cos(\omega_k \Delta t) v_k^x - \Delta t \sin(\omega_k \Delta t) v_k^y. \quad (3.33)$$

The Jacobian of $h_k(x)$ is

$$H_k = \begin{pmatrix} \frac{-(p_k^y - S_1^y)}{(p_k^x - S_1^x)^2 + (p_k^y - S_1^y)^2} & \frac{p_k^x - S_1^x}{(p_k^x - S_1^x)^2 + (p_k^y - S_1^y)^2} & 0 & 0 & 0 \\ \frac{-(p_k^y - S_2^y)}{(p_k^x - S_2^x)^2 + (p_k^y - S_2^y)^2} & \frac{p_k^x - S_2^x}{(p_k^x - S_2^x)^2 + (p_k^y - S_2^y)^2} & 0 & 0 & 0 \\ \frac{-(p_k^y - S_3^y)}{(p_k^x - S_3^x)^2 + (p_k^y - S_3^y)^2} & \frac{p_k^x - S_3^x}{(p_k^x - S_3^x)^2 + (p_k^y - S_3^y)^2} & 0 & 0 & 0 \end{pmatrix}. \quad (3.34)$$

Our simulations are carried out using a fixed trajectory and thus the expectation operators in (3.28) vanish and the required Jacobians can be calculated using the true trajectory. The recursion in (3.27) is initialized by $J_0 = R_0^{-1}$, where R_0 is the initial covariance matrix of the state x_0.

3.3.3 Simulation Results

We have implemented seven different particle filters, all based on the Sampling Importance Resampling (SIR) filtering paradigm [17]. The prior distribution is used as the importance function. The filters were implemented in MATLAB and the code was highly optimized.

- *PFall*: a SIR particle filter which collects all the measurements from all the sensors. There are no OOSMs in this case.
- *PFmis*: a SIR particle filter which discards all the OOSMs and therefore only processes the measurements that are not delayed.
- *OOSM-rerun*: a SIR particle filter which processes the OOSMs by re-running the particle filter starting using the saved particle cloud at the time step producing the OOSM. (See Section 2.2.3)

- *OOSM-GARP* : a SIR particle filter which processes the OOSMs by re-running the particle filter starting with the saved Gaussian approximation of the particle cloud at the time step producing the OOSM. (See Section 3.1)
- *SEPF-EKS*: a SIR particle filter equipped with the extended Kalman smoother proposed in [12], which stores all the Gaussian approximations of the particle clouds. (See Section 2.2.3)
- *SP-MI*: Selective OOSM processing based on the mutual information metric. (See Section 3.2)
- *SP-KL*: Selective OOSM processing based on the KL-divergence metric. (See Section 3.2)

We use the root mean-squared (RMS) position error to compare the performance of particle filters. Let $(p_k^{x(i)}, p_k^{y(i)})$ and $(\widehat{p_k^x}^{(i)}, \widehat{p_k^y}^{(i)})$ denote the true and estimated target positions at time step k for the i-th of M Monte-Carlo runs. The RMS position error at k is calculated as

$$RMS_k = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(\widehat{p_k^x}^{(i)} - p_k^{x(i)})^2 + (\widehat{p_k^y}^{(i)} - p_k^{y(i)})^2}. \tag{3.35}$$

The CRLB indicates the best possible performance that we can expect for a given scenario and a set of parameters. Let $J_k^{-1}[i,j]$ denotes the ij-th element of the inverse information matrix. The corresponding CRLB for the RMS position error as in [42] is given by

$$CRLB(RMS_k) = \sqrt{J_k^{-1}[1,1] + J_k^{-1}[2,2]}. \tag{3.36}$$

We compare the computational complexity of each filter by computing average running time of a Monte-Carlo run for tracking 40s in MATLAB. The complexity versus accuracy trade-off can be tuned by adjusting the thresholds γ_1 and γ_2. We illustrate this in our experiment, where we vary the computational complexity of the proposed algorithms *SP-MI* and *SP-KL* by varying the respective thresholds and plot the RMS vs. Average running time curves measured in MATLAB. The values of γ_1 will be shown in each example and γ_2 is fixed to 2.5% for *SP-MI* and *SP-KL* in both examples. We show the relationship

between complexity and performance for the proposed algorithms *SP-MI* and *SP-KL* with ten values of the first stage threshold γ_1 and results of 10 simulations for other algorithms. Each simulation involves 1000 Monte Carlo runs. We compare the performance of all particle filters when they use 2000 particles. The results are run on a Dell laptop with Genuine Intel(R) CPU T2400 1.83GHz, 0.99GB RAM and Win-XP OS.

The overall performance of a filter can be evaluated using the root time-averaged mean square (RTAMS) error as [42]. This is defined as

$$RTAMS = \sqrt{\frac{1}{(t_{max} - t_a) \times M} \sum_{k=t_a+1}^{t_{max}} \sum_{i=1}^{M} (\widehat{p}_k^{x(i)} - p_k^{x(i)})^2 + (\widehat{p}_k^{y(i)} - p_k^{y(i)})^2}, \tag{3.37}$$

where t_{max} is the duration of tracking time and t_a is a time index after which the averaging is carried out. M is the total number of Monte Carlo runs.

Example 1

S1 and S2 are standard sensors without OOSMs, which means that all the measurements from S1 and S2 arrive at the fusion centre without delay. The sensor S3 generates OOSMs. The OOSMs arrive at the fusion centre with probability p_{osm}. If they do arrive, they arrive with delay t_d. In our simulations, the delay t_d is uniformly distributed in $[0, 5]$ and p_{osm} is set to 0.7. In this example, the ten values of first stage thresholds are $\gamma_1 = 0 : 0.06 : 0.54$ for *SP-MI* and $\gamma_1 = 0 : 0.15 : 1.35$ for *SP-KL*.

We first plot RMS position errors of *PFall*, *PFmis*, *OOSM-rerun*, *OOSM-GARP*, *SEPF-EKS* and CRLB, which is shown in Fig. 3.2. We can see that by processing OOSMs, *SEPF-EKS*, *OOSM-rerun* and *OOSM-GARP* all improve tracking performance significantly. Moreover, *SEPF-EKS* can achieve almost similar performance compared to the two re-run OOSM particle filters. The performance of *PFall* is close to the CRLB, which indicates that 3 sensors is sufficient to give satisfactory performance in this scenario.

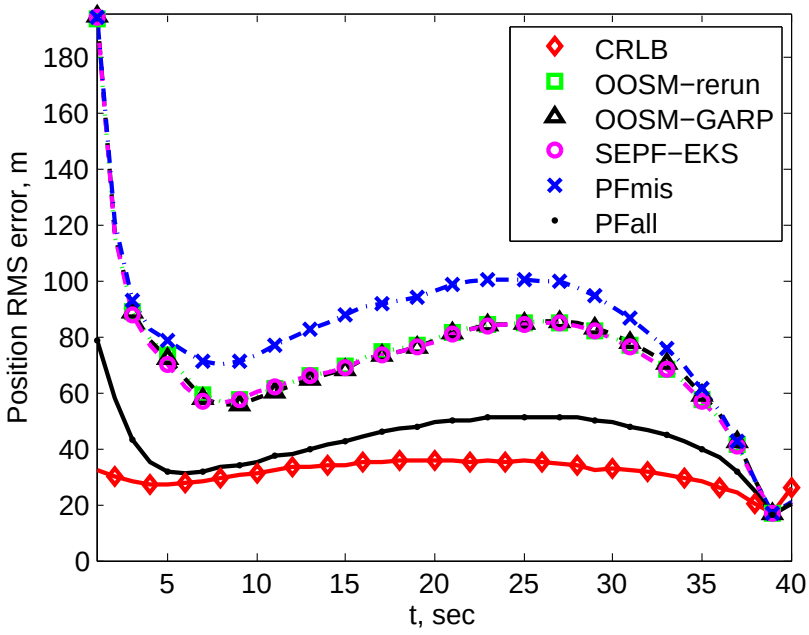
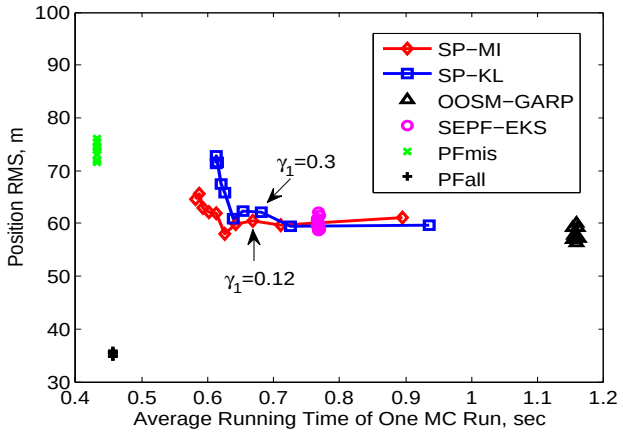
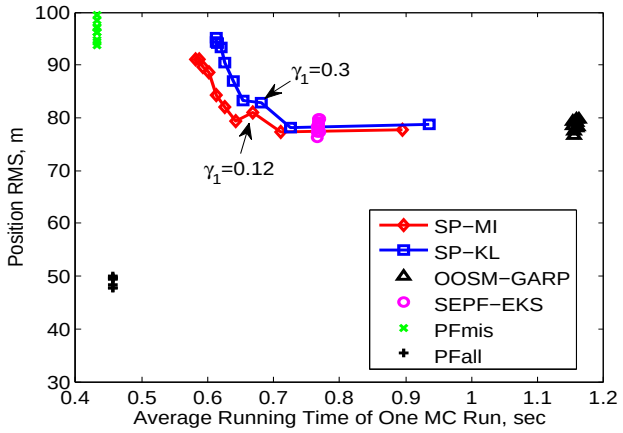


Fig. 3.2 Example 1: RMS position errors for *PFall*, *PFmis*, *OOSM-rerun*, *OOSM-GARP* and *SEPF-EKS*.

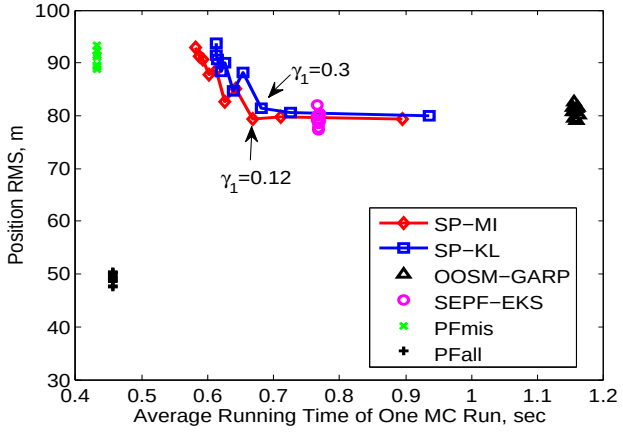
The RMS at time step k vs Average running time of one MC run curves are shown in Fig. 3.3. In this example, ten diamond points on the red curve represent the results of *SP-MI* with different γ_1 from 0 to 0.54 with step size 0.06 (from right to left). Ten square points on the blue curve represent the results of *SP-KL* with different γ_1 from 0 to 1.35 with step size 0.15 (from right to left). When the thresholds are chosen so that the selective processing filters have the same fixed RMS error performance as *SEPF-EKS*, the selective processing algorithms reduce the computation time by approximately 15%.



(a) $N = 2000$, $k = 10$



(b) $N = 2000$, $k = 20$



(c) $N = 2000$, $k = 30$

Fig. 3.3 Example 1: RMS vs Average running time of one MC run from 10 simulations with different γ_1 for *SP-MI* (from 0 to 0.54) and *SP-KL* (from 0 to 1.35). We select three timesteps, $k = 10, 20, 30$ for filters with 2000 particles.

From Fig. 3.3, *SP-MI* with $\gamma_1 = 0.12$ and *SP-KL* with $\gamma_1 = 0.3$ can achieve similar performance as re-run filters but with least average running time. In Fig. 3.4, we plot the RMS position performance for $40s$ of the algorithms with these settings. *SP-MI* and *SP-KL* can achieve similar performance as *SEPF-EKS* and *OOSM-GARP*. From the results, the number of individual OOSMs processed by the EKS after the first threshold γ_1 is 32.77% for *SP-MI* and and 22.91% for *SP-KL*. After the second threshold the number of most informative OOSMs processed by rerunning the particle filter is 0.11% for *SP-MI* and 0.16% for *SP-KL*. Though the selective processing algorithms cannot improve the performance of *SEPF-EKS*, they still reduce the complexity of it by discarding a portion of OOSMs. The *SEPF-EKS* rarely fails, since in this situation only sensor S3 generates OOSMs. The received OOSMs are rarely highly informative measurements because the non-delayed measurements from sensors S1 and S2 have provided substantial information for the estimation.

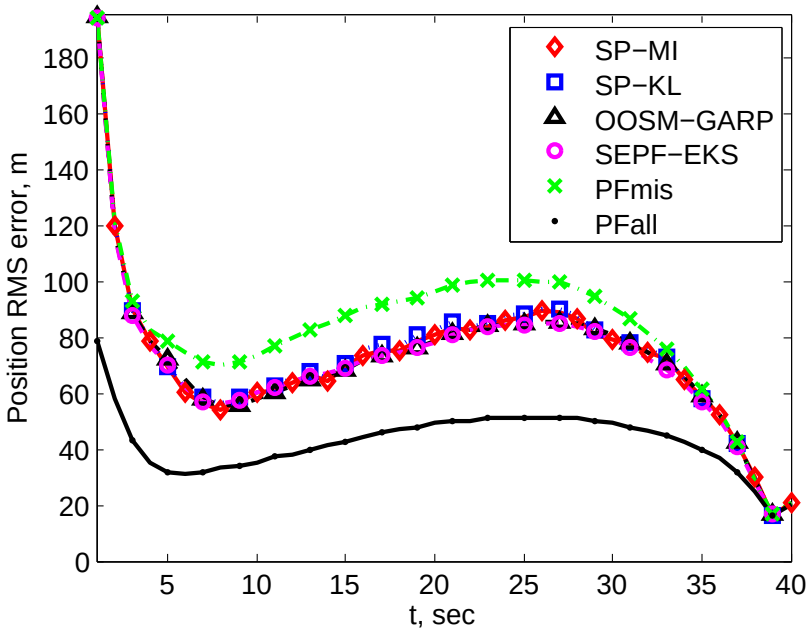


Fig. 3.4 Example 1: RMS position errors with $\gamma_1 = 0.12$ for *SP-MI* and $\gamma_1 = 0.3$ for *SP-KL*.

The boxplot figures are shown in Fig. 3.5. This figure shows the variation of position RMS error for *SEPF-EKS*, *OOSM-GARP*, *SP-MI* and *SP-KL*. The box has lines at the

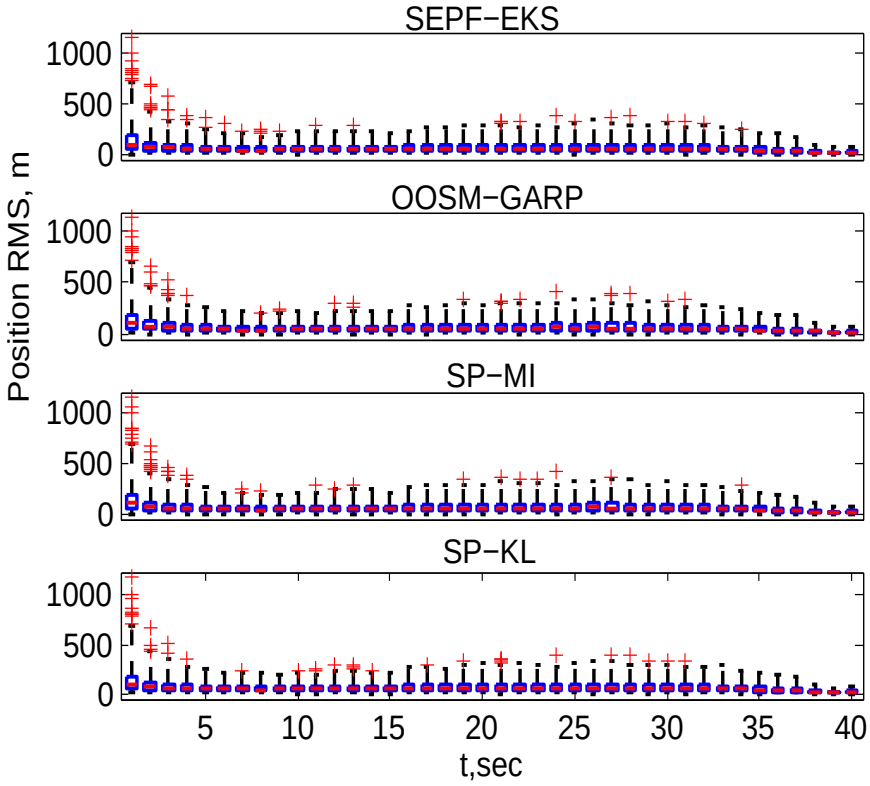


Fig. 3.5 Example 1: Errorbars showing the variation of position RMS errors for *SEPF-EKS*, *OOSM-GARP*, *SP-MI*($\gamma_1 = 0.12$) and *SP-KL*($\gamma_1 = 0.3$).

lower quartile, median, and upper quartile values. An outlier (marked by '+' sign) is defined as a value that is more than 4 times the interquartile range away from the top or bottom of the box. The distributions of errors are similar for the four OOSM particle filters, implying that they have similar robustness to outlying measurements.

We show the RTAMS error, average running time of each filter tracking 40s and the number of divergent tracks in Table 3.1. In this case, t_a in Equation (4.40) is set to 5, since the averaging is carried out by the timestep 5. By defining the track with RTAMS error larger than $150m$ as a divergent track, we can calculate the number of divergent tracks out of 1000 tracks for each filter. In this example, *SEPF-EKS*, *SP-MI* and *SP-KL* can achieve similar RMS performance as *OOSM-GARP* and *OOSM-rerun*, but with almost half the complexity. There are no divergent tracks for any of the algorithms since we have two standard sensors S1 and S2, which provide non-delayed measurements to the fusion centre.

Table 3.1 Example 1: Performance comparison of RTAMS error, running time and the number of divergent tracks

Algorithm	Time (s)	RTAMS (m)	Divergent tracks
PFall	0.4558	42.1909	0
PFmis	0.4327	81.5491	0
OOSM-rerun	1.2204	68.9698	0
OOSM-GARP	1.1582	69.2617	0
SEPF-EKS	0.7662	68.5650	0
SP-MI($\gamma_1 = 0.12$)	0.6680	69.6772	0
SP-KL($\gamma_1 = 0.3$)	0.6810	70.91	0

Example 2

In the second example, we test the particle filters in a different situation from Example 1– all 3 sensors generate OOSMs. As in Example 1, the OOSMs arrive at the fusion centre with probability p_{osm}. If they do arrive, they arrive with delay t_d. In the simulations, t_d is uniformly distributed in the range $[0, 5]$ and p_{osm} is set to 0.7. In this example, the ten values of first stage thresholds are $\gamma_1 = 0 : 0.2 : 1.8$ for *SP-MI* and $\gamma_1 = 0 : 0.5 : 4.5$ for *SP-KL*.

We first plot RMS position errors of *PFall*, *PFmis*, *OOSM-rerun*, *OOSM-GARP*, *SEPF-EKS* and CRLB, which is shown in Fig. 3.6. In this figure, *SEPF-EKS* performs much worse than *OOSM-rerun* and *OOSM-GARP*. The reason is that more measurements arrive at the fusion centre with delay compared to Example 1. If the fusion centre receives an OOSM after missing measurements for a few time steps, this OOSM can be very informative and will greatly change the distributions of particles. Then *SEPF-EKS* fails to process such highly informative OOSMs because it does not change the locations of particles but just updates their weights.

The RMS at time step k vs Average running time of one MC run curves are shown in Fig. 3.7. In this example, ten diamond points on the red curve represent the results of *SP-MI* with different γ_1 from 0 to 1.8 with step size 0.2 (from right to left). Ten square points on the blue curve represent the results of *SP-KL* with different γ_1 from 0 to 4.5 with step size 0.5 (from right to left). When the thresholds are chosen so that the selective processing

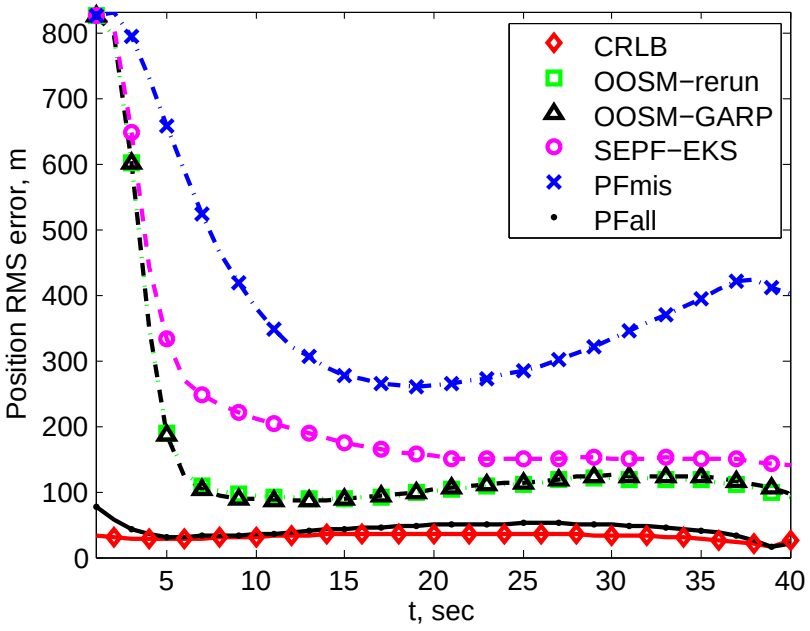
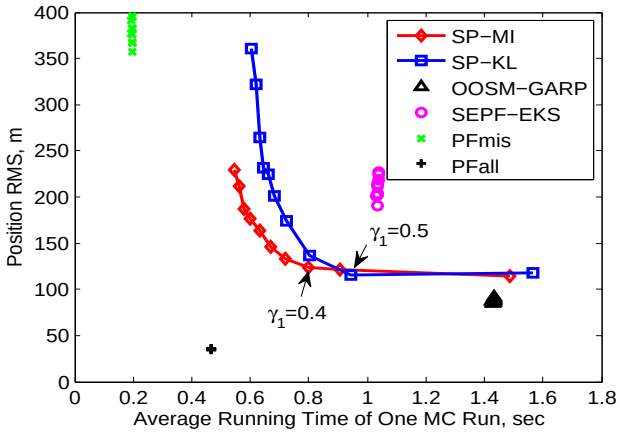
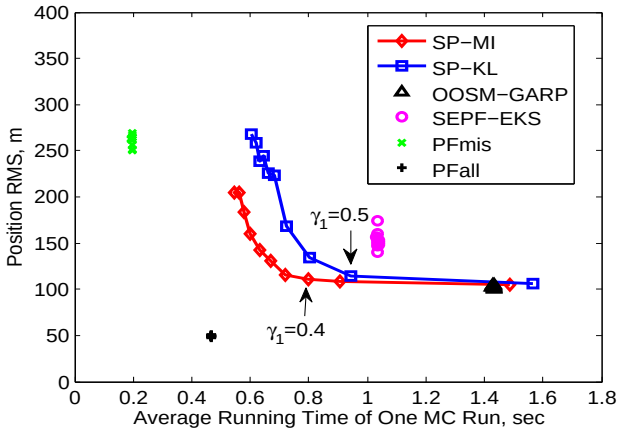


Fig. 3.6 Example 2: RMS position errors for *PFall*, *PFmis*, *OOSM-rerun*, *OOSM-GARP* and *SEPF-EKS*.

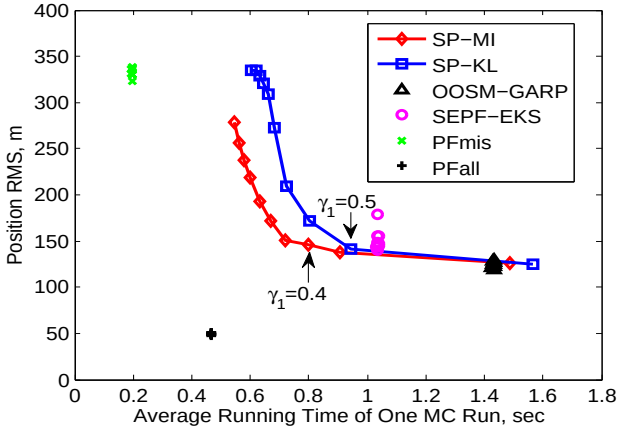
filters have the same computational complexity as *SEPF-EKS* they achieve significantly better tracking performance. Alternatively, for the same fixed RMS error performance, the selective processing algorithms reduce the computation time by approximately 40%.



(a) $N = 2000$, $k = 10$



(b) $N = 2000$, $k = 20$



(c) $N = 2000$, $k = 30$

Fig. 3.7 Example 2: RMS vs Average running time of one MC run from 10 simulations with different γ_1 for *SP-MI* (from 0 to 1.8) and *SP-KL* (from 0 to 4.5). We select three timesteps, $k = 10, 20, 30$ for filters with 2000 particles.

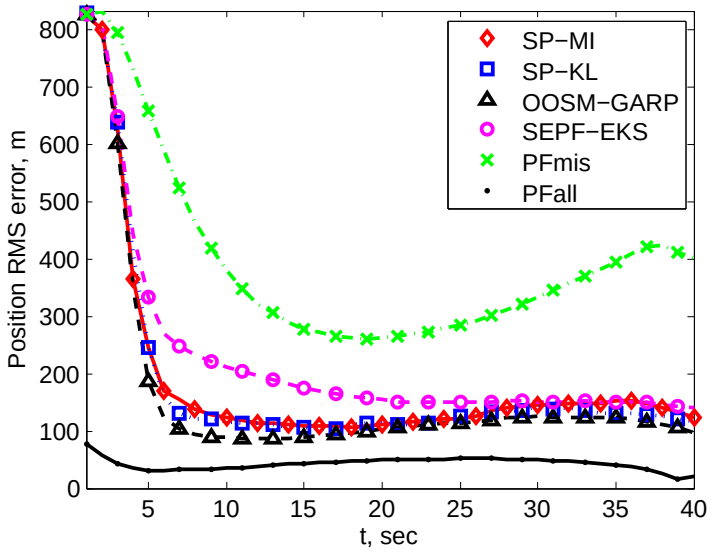


Fig. 3.8 Example 2: RMS position errors with $\gamma_1 = 0.4$ for *SP-MI* and $\gamma_1 = 0.5$ for *SP-KL*.

From Fig. 3.7, *SP-MI* with $\gamma_1 = 0.4$ and *SP-KL* with $\gamma_1 = 0.5$ can achieve similar performance as re-run filters but with least average running time. In Fig. 3.8, we plot the RMS position performance for $40s$ of the algorithms with these settings. In this example, *SEPF-EKS* performs much worse than *OOSM-rerun* and *OOSM-GARP*, but *SP-MI* and *SP-KL* can achieve similar performance as *OOSM-rerun* and *OOSM-GARP*. The selective processing algorithms process highly informative OOSMs by *OOSM-GARP* in order to avoid the deterioration. This additional operation only leads to a slight increase in complexity since it only occurs for a very small portion of the OOSMs. From the results, the fraction of individual OOSMs processed by the *SEPF-EKS* after the first threshold γ_1 is 30.57% for *SP-MI* and 35.67% for *SP-KL*. After the second threshold, the fraction of most informative OOSMs processed by rerunning the particle filter is 1.42% for *SP-MI* and 3.13% for *SP-KL*. Comparing the two proposed algorithms, *SP-MI* performs a little more efficient than *SP-KL*, because *SP-MI* processes less OOSMs but achieves similar performance as *SP-KL*.

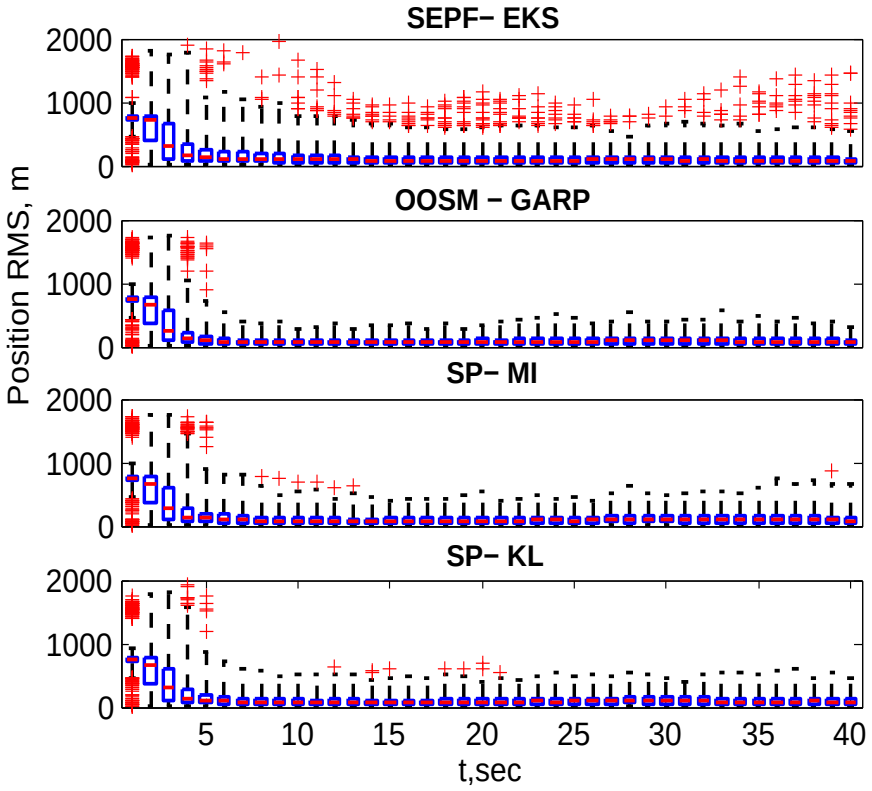


Fig. 3.9 Example 2: Errorbars showing the variation of position RMS errors for *SEPF-EKS*, *OOSM-GARP*, *SP-MI*($\gamma_1 = 0.4$) and *SP-KL*($\gamma_1 = 0.5$).

The boxplot figures are shown in Fig. 3.9. This figure shows the variation of position RMS error for *SEPF-EKS*, *OOSM-GARP*, *SP-MI* and *SP-KL*. In these figures, an outlier (marked by '+' sign) is defined as a value that is more than 6 times the interquartile range away from the top or bottom of the box. In this example, the distributions of the errors of *SEPF-EKS* are severely diffused, whereas the other three particle filters have stable performance. For *SEPF-EKS*, we can deduce that there are probably some divergent tracks in the simulation (see Table 3.2). Comparing *SEPF-EKS* and selective processing algorithms, the operation of reprocessing particles by *OOSM-GARP* prevents the divergent trend of *SEPF-EKS* and results in a major performance improvement.

We show the RTAMS error, average running time of each filter tracking 40s and the number of divergent tracks in Table 3.2. In this case, t_a in Equation (4.40) is set to 5, since the averaging is carried out by the timestep 5. By defining the track with RTAMS error

larger than $500m$ as a divergent track, we can calculate the number of divergent tracks out of 1000 tracks for each filter. In this example, *SP-MI* and *SP-KL* can achieve similar RMS performance as *OOSM-GARP* and *OOSM-rerun*, but with less complexity. *SEPF-EKS* performs worse than the re-run particle filters and there are some divergent tracks in the simulations. There are no divergent tracks for *SP-MI* and *SP-KL* since the second stage detects the situation when *SEPF-EKS* fails to process highly informative OOSMs.

Table 3.2 Example 2: Performance comparison of RTAMS error, running time and the number of divergent tracks

Algorithm	Time (s)	RTAMS (m)	Divergent tracks
PFall	0.4654	42.6357	0
PFmis	0.196	365.0253	43
OOSM-rerun	1.5671	109.3645	0
OOSM-GARP	1.4301	110.6551	0
SEPF-EKS	1.0349	178.3266	14
SP-MI($\gamma_1 = 0.4$)	0.8002	135.342	0
SP-KL($\gamma_1 = 0.5$)	0.9425	128.1042	0

Chapter 4

Multiple Model Tracking with OOSMs

In practical scenarios, we often cannot accurately capture the dynamics using a single model. For example, a target may move along a straight-line with constant velocity, or it may execute a manoeuvre for a short period to turn to another direction. Therefore, we usually need multiple models to capture the dynamics of manoeuvring targets. Multiple model tracking poses a more complicated problem, since the system can switch between different models. This implies the need for some form of adaptive filter model. In this chapter, we first introduce related work on multiple model tracking with OOSMs. Then we propose four candidate algorithms for multiple model cases based on the algorithms introduced in Chapter 3 for single model cases: the OOSM re-run multiple model particle filter, the OOSM Gaussian approximation re-run multiple model particle filter, the multiple model particle filter with interacting multiple model extended Kalman smoother (IMM-EKS), and the efficient delay-tolerant multiple model particle filter through selective processing of OOSMs. Finally, we use a scenario with a manoeuvring target to test the performance of our proposed methods.

4.1 Related Work

The multiple model tracking approach was originally presented in [43] for Gaussian state-space models; the algorithm is referred to as the *static multiple model estimator* in [40]. The static multiple model estimator operates several Kalman filters in parallel. Each filter is

matched to a possible motion model and then the estimates of the state and the covariance matrix are calculated as a weighted sum of the estimate from each filter. The weights are determined by the posterior probability of each model, which is attained from the likelihood of the current measurements for each filter. This algorithm does not adequately address the switching behavior of manoeuvring targets, where one of a number of candidate models accurately portrays the dynamics at any given time instant. This mode switching behavior is frequently modeled as a Markov process with known transition probabilities. The Markov switching of models is discussed in [44–46]. In [40], the *dynamic multiple model estimator* is proposed; it achieves optimal performance, since it keeps track of all the possible branches of mode history. It is obviously impractical owing to the exponential growth with time in the number of possible branches. In order to develop practical algorithms, one can resort to a suboptimal technique that involves maintaining a constant number of Kalman filters. At each time step, only those with the largest probabilities are retained, and the mode probabilities associated with the retained filters are renormalized so that they sum up to unity.

The generalized pseudo-Bayesian (GPB) approaches are proposed in [47, 48]. GPB approaches only consider all possible branches in the last several time steps. The first-order GPB1 (described in [40]) considers all possible models for one time step only. It then combines the state estimates after the measurement update and uses the combined estimate as the input for all the filters in the next step. The second-order version GPB2 [40, 49] considers all the possible models for two time steps. Thus GPB1 and GPB2 require r and r^2 filters respectively to operate in parallel, where r is the number of possible models.

The interacting multiple model filter (IMM-filter) [50, 51] was proposed by Blom et al. by merging the estimates after the hypothesis branching step and using r hypotheses as the inputs of r filters. Therefore, mixing of the estimates before entering into filters is the key idea that yields r hypotheses with r filters, rather than r^2 filters as in the GPB2 algorithm. The IMM-filter is computationally efficient and in many cases performs well. In [52], there is a detailed survey of existing IMM methods for maneuvering tracking problems.

In order to address multiple-model behavior for highly nonlinear and non-Gaussian systems, McGinnity et al. extended the bootstrap particle filter to the multiple-model estimation problem in [53]. Their algorithm adds a variable associated with the index of model into the state vector of each particle, and employs resampling method to implement the mode transition. In [54], Ristic et al. develop a multiple model particle filter (MMPF)

for angle-only tracking following the idea in [53] and compare its performance with the IMM extended Kalman filter (IMM-EKF) and the IMM unscented Kalman filter (IMM-UKF) in an example with CT and CV models[1]. Arulampalam and Ristic conduct a comparison of the MMPF, the auxiliary MMPF (AUX-MMPF) [55] and the jump Markov system PF (JMS-PF) in [42].

There are only a few papers about out-of-sequence measurements in multiple model tracking problems. Recall Section 2.2.1, the Bl1 approach proposed in [9] is the one-step implementation of the update with a multistep lagged OOSM. Bar-Shalom et al. then incorporate OOSMs into the nonlinear IMM estimator via the Bl1 approach. The algorithm described in [10] uses state retrodiction within the IMM estimator in a decoupled manner for each of the models using the procedure of [9]. When multiple OOSMs in clutter are presented, the probabilistic data association (PDA) technique can be used to calculate the association probabilities for each validated measurement at the current time to the target of interest. The AS-IMM-PDA algorithm [56] incorporates the probabilistic data association (PDA) technique into an augmented state IMM (AS-IMM) filter for multiple model target tracking in clutter using OOSM. The AS-IMM-PDA algorithm is an extension of AS-PDA algorithm in [30], which addresses the multi-lag OOSM updating by augmented state smoothing for single model cases. In [57], Maskell et al. review several existing approaches in a common Bayesian framework for multi-target multi-model tracking with OOSMs.

These approaches are all based on the IMM Kalman filter or IMM-EKF and IMM-UKF for linear or mildly non-linear models. To deal with more practical tracking problems with highly non-linear models and OOSMs, we need new algorithms to handle OOSMs in multiple model particle filters.

4.2 Problem Statement

In the multiple model system, we use a general parameterized state-space representation:

$$x_k = f^{\alpha}_{k|k-1}(x_{k-1}) + v_{k|k-1} \tag{4.1}$$

$$y_k = h^{\alpha}_{k}(x_k) + s_k, \tag{4.2}$$

[1]CT: coordinated turn model; CV: constant velocity model.

where $f^{\alpha}_{k|k-1}$ and h^{α}_{k} are the parameterized state transition function and measurement function for model α at time k, respectively. x_k denotes the state vector of the system at time k. y_k denotes the output measurement at time k. $v_{k|k-1}$ is the process noise and s_k is the measurement noise. Suppose there are r possible models in the system, then α denotes the index of model at time k. The model transition is modeled as a Markov process with the initial distribution $p(\alpha_0)$ and transition probability $p_{\beta|\alpha}$ (from model α to model β), for $\alpha, \beta \in \{1, \ldots, r\}$.

4.3 Multiple Model Filters

In this section, we introduce the interacting multiple model extended Kalman filter (IMM-EKF) [50, 51] and the multiple model particle filter (MMPF) [53]. IMM-EKF is computationally efficient because it executes only r extended Kalman filters by merging the states and covariance matrices estimates after the hypothesis branching step. This avoids the exponential growth with time in the number of possible branches. MMPF is a relatively simple extension of the particle filter for multiple model problems.

4.3.1 Interacting Multiple Model Extended Kalman Filter (IMM-EKF)

The IMM-filter is summarized in [40] and [58]. We describe this method again here for completeness. It consists of three major steps: interaction(mixing), filtering and combination. In the interaction step, we obtain the mixing probability and combined inputs for the current step. This step accounts for the possibility of model transition. In the filtering step, the input of each filter is processed by the corresponding filter separately and then we obtain the estimates of the state and covariance from each filter. In the final step, we compute the likelihoods of the measurements to evaluate the probability of each model. We formulate a final estimate by calculating a weighted combination of all the estimates from all the filters. The weights are equal to the posterior probabilities of the model associated with each filter. The interacting multiple model extended Kalman filter (IMM-EKF) uses EKFs in the filtering step to allow for mild nonlinearities. The details of each step of the IMM-EKF are as follows and also illustrated in Fig. 4.1:

1. Interaction(mixing): The mixing probabilities $\varphi^{\beta|\alpha}_{k}, (\alpha, \beta = 1, \ldots, r)$ denote the prob-

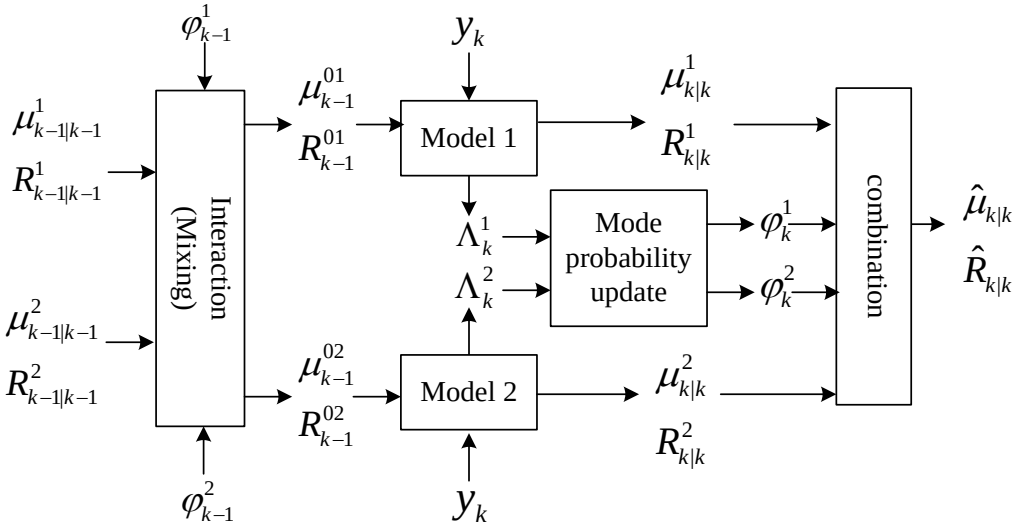


Fig. 4.1 The IMM estimator with two models (one cycle).

abilities of transition from model α to model β and are calculated as:

$$\varphi_k^{\beta|\alpha} = \frac{1}{\bar{c}_\beta} p_{\beta|\alpha} \varphi_{k-1}^{\alpha} \tag{4.3}$$

$$\bar{c}_\beta = \sum_{\alpha=1}^{r} p_{\beta|\alpha} \varphi_{k-1}^{\alpha}, \tag{4.4}$$

$$\tag{4.5}$$

where φ_{k-1}^{α} is the probability of model α at the time step $k-1$ and $\bar{c}_\beta$ is a normalizing factor.

Then we can calculate the mixed inputs for each filter :

$$\mu_{k-1}^{0\beta} = \sum_{\alpha=1}^{r} \varphi_k^{\beta|\alpha} \mu_{k-1|k-1}^{\alpha} \tag{4.6}$$

$$R_{k-1}^{0\beta} = \sum_{\alpha=1}^{r} \varphi_k^{\beta|\alpha} \times \{R_{k-1|k-1}^{\alpha} + [\mu_{k-1|k-1}^{\alpha} - \mu_{k-1}^{0\beta}][\mu_{k-1|k-1}^{\alpha} - \mu_{k-1}^{0\beta}]^T\}, \tag{4.7}$$

where $\mu^{\alpha}_{k-1|k-1}$ and $R^{\alpha}_{k-1|k-1}$ are the estimates of state and covariance for model α at time step $k-1$.

2. Filtering: in this step, the inputs of r filters are processed by each filter separately.

$$[\mu^{\beta}_{k|k-1}, R^{\beta}_{k|k-1}] = EKF_p(\mu^{0\beta}_{k-1}, R^{0\beta}_{k-1}) \tag{4.8}$$

$$[\mu^{\beta}_{k|k}, R^{\beta}_{k|k}] = EKF_u(\mu^{\beta}_{k|k-1}, R^{\beta}_{k|k-1}, y_k), \tag{4.9}$$

where EKF_p and EKF_u denote the prediction and update steps of extended Kalman filter (see details in Section 2.1.2). In addition to the above operation, we also compute the likelihood of the measurement for each filter:

$$\Lambda^{\beta}_k = \mathcal{N}(y_k; h^{\beta}_k(\mu^{\beta}_{k|k-1}), S^{\beta}_k), \tag{4.10}$$

where S^{β}_k is the covariance of the measurement residual.

3. Combination: in the final step, we first compute the probability of each model at time step k :

$$\varphi^{\beta}_k = \frac{1}{c}\Lambda^{\beta}_k \bar{c}_{\beta} \tag{4.11}$$

$$c = \sum_{\beta=1}^{r} \Lambda^{\beta}_k \bar{c}_{\beta}, \tag{4.12}$$

where c is a normalizing factor. Then we combine the estimates from all the filters together:

$$\hat{\mu}_{k|k} = \sum_{\beta=1}^{r} \varphi^{\beta}_k \mu^{\beta}_{k|k} \tag{4.13}$$

$$\hat{R}_{k|k} = \sum_{\beta=1}^{r} \varphi^{\beta}_k \times \{R^{\beta}_{k|k} + [\mu^{\beta}_{k|k} - \hat{\mu}_{k|k}][\mu^{\beta}_{k|k} - \hat{\mu}_{k|k}]^T\}. \tag{4.14}$$

4.3.2 Multiple Model Particle Filter (MMPF)

The key idea of the MMPF [53] is to add a discrete variable A_k into the state vector, which is augmented as $\{x_k, A_k\}$. x_k is the original state vector and A_k labels the index of the

current model. Therefore, the propagation step of particle filter can be described as:

$$p(x_k, A_k|x_{k-1}, A_{k-1}) = p(x_k|A_k, x_{k-1}, A_{k-1})p(A_k|A_{k-1}). \tag{4.15}$$

Since $\{A_k\}$ is a Markov process with transition probability $p(A_k|A_{k-1})$, we can obtain the index $A_k^{(i)}$ for the i-th particle from $p(A_k|A_{k-1}^{(i)})$. Then we can propagate the particles to current states $x_k^{(i)}$ by corresponding models determined by $A_k^{(i)}$. We compute the likelihood function of the measurements $p(y_k|x_k^{(i)}, A_k^{(i)})$ to update the weight of each particle and resample the particles at the end of each recursion. The details are shown in Algorithm 7.

Algorithm 7: Multiple Model Particle Filter

1 **Initialization**, $k = 0$.

2 Sample the particles $x_0^{(i)} \sim p(x_0), (i = 1, \ldots, N)$ and $A_0^{(i)} \sim p(\alpha_0)$;

3 **for** $k = 1 \ldots T$ **do**

4 Model switching : draw $A_k^{(i)} \sim p(A_k|A_{k-1}^{(i)})$;

5 Propagate particles :

$$x_k^{(i)} = f_{k|k-1}^{A_k^{(i)}}(x_{k-1}^{(i)}) + v_{k|k-1}; \tag{4.16}$$

6 Update and normalize weights:

$$\omega_k^{(i)} = p(y_k|x_k^{(i)}, A_k^{(i)}) \tag{4.17}$$

$$\tilde{\omega}_k^{(i)} = \frac{\omega_k^{(i)}}{\sum_{i=1}^{N} \omega_k^{(i)}}; \tag{4.18}$$

7 Resampling Step: resample N particles $(x_k^{(i)}, A_k^{(i)}; i = 1, \ldots, N)$ according to the importance weights ;

8 **endfor**

4.4 OOSM Multiple Model Particle Filters (OOSM-MMPF)

Here, we investigate OOSM problems in multiple model particle filter and propose four kinds of OOSM multiple model particle filter: the OOSM re-run multiple model particle filter, the OOSM Gaussian approximation re-run multiple model particle filter, the multiple model particle filter with interacting multiple model extended Kalman smoother

(IMM-EKS), and the efficient delay-tolerant multiple model particle filter through selective processing of OOSMs.

4.4.1 OOSM Re-run Multiple Model Particle Filter (OOSM-rerunMM)

OOSM-rerunMM is an extension of the OOSM-rerun algorithm for multiple model cases. This method is a close-to-optimal solution which simply reprocesses all the OOSMs in an ordered sequence using the MMPF. However, this method needs to save all the particles $\xi_{k-l-1:k}$ in the last $l+1$ time steps into Ω_k and all the available measurements in the last k time steps into $\widetilde{\mathcal{W}}_k^{k-l:k}$, where l is the predetermined maximum number of delayed steps. Here, the particles include the state vectors and indices of models, which are denoted as $\xi_k \equiv \{x_k^{(i)}, A_k^{(i)}; (i = 1, \ldots, N)\}$. Thus the requirement of memory and computation can be excessive.

Algorithm 8 provides pseudocode for the function `ProcessOOSM` of the OOSM re-run multiple model particle filter, which is very similar to Algorithm 4 in Section 2.2.3. The only difference is the use of function `MMParticleFilter` (refers to Algorithm 7) to process particles in each step. The function `SaveParticles` stores the updated particles into the stored set Ω_k.

Algorithm 8: ProcessOOSM by OOSM-rerunMM

Input: ξ_{k-1}, ω_{k-1}, Ω_{k-1}, $\widetilde{\mathcal{W}}_k^{k-l:k}$

1 $\widetilde{\tau}_k = \min_{\tau}\{\tau : y_\tau \in \mathcal{Z}_k\}$;
2 $\xi_{\widetilde{\tau}_k-1}$ from Ω_{k-1} ;
3 $\omega_{\widetilde{\tau}_k-1}^{(i)} = 1/N,\ i = 1 \ldots N$;
4 **for** $j = \widetilde{\tau}_k, \ldots, k$ **do**
5 $(\xi_j, \omega_j) \leftarrow$ `MMParticleFilter`$(\widetilde{\mathcal{W}}_k^j, \xi_{j-1}, \omega_{j-1})$;
6 $(\Omega_k) \leftarrow$ `SaveParticles`(ξ_j) ;
7 **endfor**

The OOSM re-run multiple model particle filter involves a high memory and high computational cost. On the other hand, it exhibits the best tracking performance among on-line OOSM multiple model particle filters. It acts as a useful benchmark for other methods that reduce computational overhead or memory requirements.

4.4.2 OOSM Gaussian Approximation Re-run Multiple Model Particle Filter (OOSM-GARP-MM)

Based on OOSM-GARP in Section 3.1, the OOSM Gaussian approximation re-run MMPF is a storage-efficient version of the OOSM re-run MMPF, which only saves the statistics of the particles. For state vectors $x_k^{(i)}$, only the mean μ_k and covariance matrix R_k are saved. For the variable $A_k^{(i)}$, only the probability of each model $p(A_k = \alpha), \alpha = 1, \ldots, r$ are stored as

$$p(A_k = \alpha) \approx \frac{n(A_k^{(i)} = \alpha)}{N}, \tag{4.19}$$

where $n(A_k^{(i)} = \alpha)$ denotes the number of particles with $A_k^{(i)}$ equal to α and N is the total number of particles. Therefore, the stored information needs to be $\Omega_k = \{\mu_{k-l-1:k}, R_{k-l-1:k}\}$, the probability of each model from time step $k-l-1$ to time step k, $\{p(A_m = \alpha)\}_{m=k-l-1:k}^{\alpha=1:r}$ and all the available measurements up to current time step $\widetilde{\mathcal{W}}_k^{k-l:k}$. The details of this algorithm are described in Algorithm 9. When the filter is re-run from a previous time step upon arrival of an OOSM, it first needs to sample the state vector for each particle using the mean and covariance. It also samples the index of the model associated with each particle. The function `SaveMMGauss` computes the stored statistics using Equation (3.2), (3.3) and (4.19). Here, we only store the mean and covariance of all the particles (not for each model) for simplicity.

Algorithm 9: ProcessOOSM-GARP-MM

Input: Ω_{k-1}, $\widetilde{\mathcal{W}}_k^{k-l:k}$

$\widetilde{\tau}_k = \min_{\tau}\{\tau : y_\tau \in \mathcal{Z}_k\}$;

$\{x_{\widetilde{\tau}_k-1}^{(i)}\}_{i=1}^N \sim \mathcal{N}(x_{\widetilde{\tau}_k-1}, \mu_{\widetilde{\tau}_k-1}, R_{\widetilde{\tau}_k-1})$;

Sample $A_{\widetilde{\tau}_k-1}^{(i)}$ from $\{p(A_{\widetilde{\tau}_k-1} = \alpha)\}_{\alpha=1:r}$;

$\omega_{\widetilde{\tau}_k-1}^{(i)} = 1/N,\ i = 1 \ldots N$;

for $j = \widetilde{\tau}_k, \ldots, k$ **do**

    $(\xi_j, \omega_j) \leftarrow$ `MMParticleFilter(`$\widetilde{\mathcal{W}}_k^j$, ξ_{j-1}, ω_{j-1}`)`;

    $(\Omega_k, p(A_k = \alpha)) \leftarrow$ `SaveMMGauss(`ξ_j, ω_j`)` ;

endfor

In most cases, OOSM-GARP-MM can achieve performance similar to that of the OOSM re-run MMPF but with much less storage. The two filters have similar computational complexity since the basic steps of processing the OOSMs are very similar.

4.4.3 Multiple Model Particle Filter with IMM-EKS (MMPF-IMM-EKS)

The multiple model particle filter with interacting multiple model extended Kalman smoother (MMPF-IMM-EKS) is an extension of the storage efficient particle filter with extended Kalman smoother in [12]. It runs a multiple model particle filter and deals with the OOSMs using IMM-EKS. This filter also only stores the statistics of particles, similar to OOSM-GARP-MM, to reduce the memory requirement. To reduce the computational complexity, we use IMM-EKS to reprocess the mean and covariance of particles from the time step generating the OOSM to the current time step for updating the weights of current particles. The detailed description of the algorithm is provided in Appendix A.2. After an OOSM update using IMM-EKS, we check the effective number of particles before and after the update, denoted by $N_{\text{eff}}^{\text{post}}$ and $N_{\text{eff}}^{\text{prior}}$ respectively. If $N_{\text{eff}}^{\text{post}} \ll N_{\text{eff}}^{\text{prior}}$, this update will be discarded (in our experiments we check the condition $N_{\text{eff}}^{\text{post}} < N_{\text{eff}}^{\text{prior}}/100$).

4.4.4 Efficient Delay-tolerant Multiple Model Particle Filter through Selective Processing of OOSMs (EDMMPF-SP)

In some cases, the multiple model particle filter with IMM-EKS has the same weakness as the storage efficient particle filter with EKS (see Section 2.2.3). MMPF-IMM-EKS can discard some highly-informative measurements because they lead to an OOSM update greatly reducing the effective number of particles. In this section, we extend the efficient delay-tolerant particle filter through selective processing of OOSMs (EDPF-SP) (see Section 3.2) to the multiple model case. The key idea of this algorithm is to process the OOSMs according to their informativeness. The first stage of the algorithm estimates the informativeness of an OOSM (two metrics below) and discards those considered uninformative. In the second stage, the informative OOSMs are processed by MMPF-IMM-EKS. If the OOSM update by MMPF-IMM-EKS significantly reduces the effective sample size, we employ OOSM-GARP-MM. The approach reduces the computational complexity by the first stage and improves the performance by applying OOSM-GARP-MM as a backup when MMPF-IMM-EKF fails. We describe the proposed approach `ProcessOOSM-MMSP` in

Algorithm 10. Here, we assume that there are potentially multiple OOSMs in $\mathcal{Z}_k$, any $y_\tau \equiv y_\tau^{\mathcal{D}_{\tau,k}} \in \mathcal{Z}_k$.

Algorithm 10: ProcessOOSM-MMSP

Input: Ω_{k-1}, $\widetilde{\mathcal{W}}_k^{k-l:k}$

$(\xi_k, \omega_k) \leftarrow$ `MMParticleFilter` $(\mathcal{Y}_k, \xi_{k-1}, \omega_{k-1})$;
$(\Omega_k, p(A_k = \alpha)) \leftarrow$ `SaveMMGauss`(ξ_k, ω_k) ;
EKSfailed = 0;
for $y_\tau \in \mathcal{Z}_k$ **do**
    $I_{y_\tau} \leftarrow$ `CalcMI`$(y_\tau, \mu_\tau, R_\tau, H_\tau)$;
    if $I_{y_\tau} < \gamma_1$ **then**
        discard y_τ ;
    else
        $N_{\text{eff}}^{\text{prior}} = 1/\sum_{i=1}^{N}(\omega_k^{(i)})^2$;
        $(\xi_k, \omega_k) \leftarrow$ `ProcessOOSM-IMMEKS`$(y_\tau, \xi_k, \omega_k, \Omega_k)$;
        $N_{\text{eff}}^{\text{post}} = 1/\sum_{i=1}^{N}(\omega_k^{(i)})^2$;
        if $N_{\text{eff}}^{\text{post}}/N_{\text{eff}}^{\text{prior}} < \gamma_2$ **then**
            EKSfailed = 1 ;
            break ;
        else
            $(\Omega_k, p(A_k = \alpha)) \leftarrow$ `SaveMMGauss`(ξ_k, ω_k) ;
        endif
    endif
endfor
if *EKSfailed* **then**
    $(\xi_k, \omega_k, \Omega_k) \leftarrow$ `ProcessOOSM-GARP-MM`$(\widetilde{\mathcal{W}}_k^{k-l:k}, \Omega_k)$;

In this algorithm, the function `MMParticleFilter` (Algorithm 7) is to process non-delayed measurements in $\mathcal{Y}_k$. The function `SaveMMGauss` computes the stored statistics using Equation (3.2), (3.3) and (4.19). The function `CalcMI` is used to estimate the informativeness of a measurement in multiple model case, and it is discussed in more detail below. The thresholds γ_1 and γ_2 govern the trade-off between computational complexity and accuracy. The first threshold γ_1 determines the proportion of OOSMs that are declared uninformative and immediately discarded. The informative OOSMs are first processed by function `ProcessOOSM-IMMEKS` (Section A.2). The second threshold γ_2 defines the proportion of informative OOSMs that are processed using `ProcessOOSM-GARP-MM` (Algorithm 9) which reruns the filter from the time τ when the OOSM is measured. In our experiments

we observed that `ProcessOOSM-GARP-MM` can be invoked rarely and yet this substantially improves the quality of tracking in some cases.

OOSM Selection Rule

We extend our proposed two metrics in the previous chapter. The first metric is the mutual information between the OOSM Y_τ and the state X_k, $I(Y_\tau, X_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}})$. The mutual information is conditioned on all received measurements in $\mathcal{W}_k^{1:k}$ and the recently received OOSMs in $\mathcal{Z}_k$ except y_τ, denoted $\mathcal{Z}_{k,\bar{\tau}} = \mathcal{Z}_k \setminus \{y_\tau\}$. The second metric is the Kullback-Leibler divergence (KL-divergence) [38] between the distribution at time k, conditioned on all measurements except for y_τ, and the distribution at time k conditioned on all measurements including y_τ, denoted by $D(p(x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_{k,\bar{\tau}})\|p(x_k|\mathcal{W}_k^{1:k}, \mathcal{Z}_k))$. In the multiple-model case, we use IMM-EKF to calculate the mutual information instead of EKF.

4.5 Simulations

4.5.1 Simulation Model — Example 3

In this scenario, the target starts at initial position $[-200m, 0m]$. It performs a straight-line motion with constant velocity 60m/s for 5 seconds, then an anti-clockwise coordinated turn of radius $320m$ with a constant speed for another 30 seconds. It then continues with straight-line motion for 5 seconds, makes a clockwise coordinated turn of radius $320m$ with a constant speed for another 30 seconds and then continues with straight-line motion for the last 5 seconds. The true trajectory is displayed in Fig. 4.2.

Therefore, we have to model the target motion in the filters by two models : the constant velocity (CV) model and the nearly coordinated turn (CT) model [40] with unknown constant turn rate and cartesian velocity. The initial model distribution is set to $[0.9, 0.1]$, and the Markovian transition probabilities for model switching [2] are

$$P_{switch} = \begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix}. \tag{4.20}$$

The state of the target is given as $x_k = [p_k^x, p_k^y, v_k^x, v_k^y, \omega_k]^T$, where p, v and ω denote the

[2]The parameters of model transition are chosen following the tradition for this problem in the literature, such as [16].

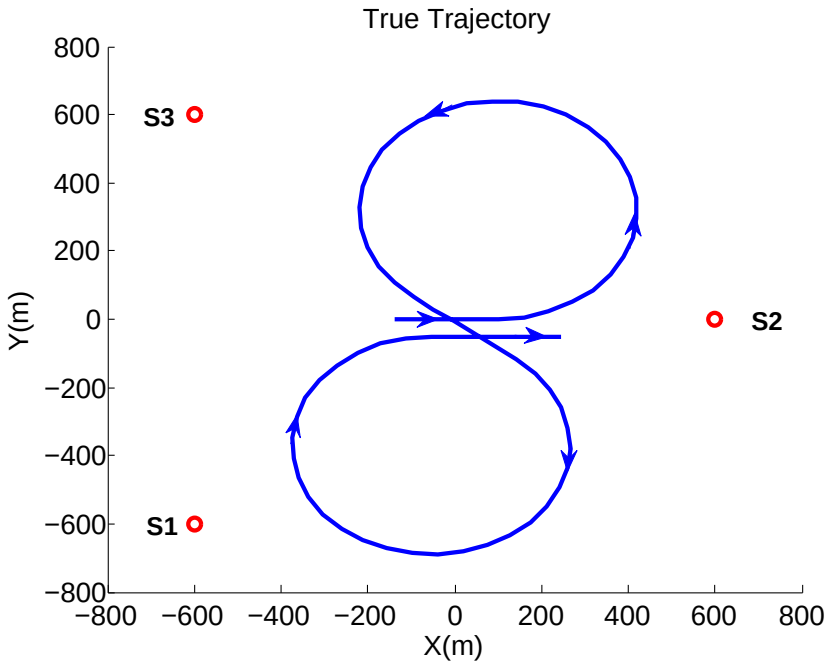


Fig. 4.2 Example 3: Target trajectory and the sensors.

position, velocity and turn rate respectively. In the simulations, we select the standard deviations for the position, speed and turn rate as $\sigma_p = 30m$, $\sigma_v = 10m/sec$, $\sigma_\omega = 0.1rad/sec$. At the beginning, all the filters to be run are assumed to know very little about the initial state of the target and therefore they are initialized with the state value $x_0 = [0, 0, 0, 0, 0]^T$ and a large covariance $R_0 = \text{diag}([500^2, 500^2, 30^2, 30^2, 0.1^2])$ in order to cover the real position of the target.

The dynamic model is :

$$x_{k+1} = f^{\alpha}_{k+1|k}(x_k) + v_{k+1|k} \quad (\alpha = 1, 2). \tag{4.21}$$

The dynamic model for the constant velocity (CV) model $f^1_{k+1|k}(.)$ is

$$x_{k+1} = \begin{pmatrix} 1 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} x_k. \tag{4.22}$$

The dynamic model for the coordinated turn (CT) model $f^2_{k+1|k}(.)$ is

$$x_{k+1} = \begin{pmatrix} 1 & 0 & \frac{\sin(\omega_k \Delta t)}{\omega_k} & \frac{\cos(\omega_k \Delta t)-1}{\omega_k} & 0 \\ 0 & 1 & \frac{1-\cos(\omega_k \Delta t)}{\omega_k} & \frac{\sin(\omega_k \Delta t)}{\omega_k} & 0 \\ 0 & 0 & \cos(\omega_k \Delta t) & -\sin(\omega_k \Delta t) & 0 \\ 0 & 0 & \sin(\omega_k \Delta t) & \cos(\omega_k \Delta t) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} x_k. \tag{4.23}$$

$v_{k+1|k}$ is Gaussian process noise with a distribution $\mathcal{N}(0, V_{k+1|k})$ and the covariance matrix $V_{k+1|k} = \text{diag}([30^2, 30^2, 10^2, 10^2, 0.1^2])$ for any k. $\Delta t = 1$ is the sampling period.

There are three sensors S1, S2 and S3 sending measurements to a common fusion center in this scenario. The locations of the three sensors are $[S^x_1, S^y_1] = [-600, -600]$, $[S^x_2, S^y_2] = [600, 0]$, $[S^x_3, S^y_3] = [-600, 600]$ (See Fig. 4.2). All the sensors are measuring the angle of the target. The measurements are with additive Gaussian noise with zero mean and standard deviation $\sigma_m = 0.1 rads$ for all sensors with sampling period $\Delta t = 1$. The measurement function of bearings-only tracking is

$$y_k = h_k(x_k) + s_k \tag{4.24}$$

$$h_k(x_k) = \arctan(\frac{p^y_k - S^y_j}{p^x_k - S^x_j}), \qquad j = 1, 2, 3 \tag{4.25}$$

where $s_k \sim \mathcal{N}(0, Q_k)$ and $Q_k = \text{diag}([\sigma^2_m, \sigma^2_m, \sigma^2_m])$. An OOSM arrives at the fusion center with probability p_{osm} and delay t_d. The probability p_{osm} characterizes the reliability of OOSM delivery (a portion of the OOSMs are lost on the way to the fusion centre). The delay t_d is uniformly distributed in the interval $[0, l]$, where l is the predefined maximum delay.

4.5.2 Cramér-Rao Lower Bound

For the multiple model case, a Cramér-Rao lower bound was presented in [42] as an indication of performance limitations. We describe it here again.

This bound assumes that the true model history of the target trajectory is known *a priori*:

$$\mathcal{H}_k^* = \{A_1^*, A_2^*, \ldots, A_k^*\}. \tag{4.26}$$

Let $\hat{x}_k$ be an estimate of a real state vector x_k. The Cramér-Rao bound is specified as a bound on the covariance of $\hat{x}_k$:

$$\begin{aligned} &\mathbb{E}[(\hat{x}_k - x_k)(\hat{x}_k - x_k)^T] && (4.27)\\ \geq &\mathbb{E}[(\hat{x}_k - x_k)(\hat{x}_k - x_k)^T | \mathcal{H}_k^*] && (4.28)\\ \geq &[J_k^*]^{-1}, && (4.29) \end{aligned}$$

where the mode-history-conditioned information matrix J_k^* is

$$J_k^* = \mathbb{E}[(\nabla_{x_k} \log p(x_k, y_k))(\nabla_{x_k} \log p(x_k, y_k))^T | \mathcal{H}_k^*]. \tag{4.30}$$

Following the development in [41] as in Equation (3.27), the mode-history-conditioned information matrix can be evaluated using the following recursive equation:

$$J_{k+1}^* = D_k^{22} - D_k^{21}(J_k^* + D_k^{11})^{-1} D_k^{12}, \tag{4.31}$$

where D_k^{ij} are given by

$$\begin{aligned} D_k^{11} &= \mathbb{E}\{(F_{k+1|k}^{(A_{k+1}^*)})^T V_{k+1|k}^{-1} F_{k+1|k}^{(A_{k+1}^*)}\} \\ D_k^{12} &= -\mathbb{E}\{(F_{k+1|k}^{(A_{k+1}^*)})^T\} V_{k+1|k}^{-1} = (D_k^{21})^T \\ D_k^{22} &= V_{k+1|k}^{-1} + \mathbb{E}\{H_{k+1}^T Q_{k+1}^{-1} H_{k+1}.\} \end{aligned} \tag{4.32}$$

Here $F_{k+1|k}^{(A_k^*)}$ and H_{k+1} are the Jacobian matrices of the transition function $f_{k+1|k}^{A_{k+1}^*}(x)$ and measurement function $h_{k+1}(x)$, respectively, and $V_{k+1|k}$ and Q_{k+1} are covariance matrices of process noise and measurement noise. The Jacobian of the constant velocity (CV) model

is

$$F^{(1)}_{k+1|k} = \begin{pmatrix} 1 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \tag{4.33}$$

The Jacobian of the coordinated turn (CT) model is

$$F^{(2)}_{k+1|k} = \begin{pmatrix} 1 & 0 & \frac{\sin(\omega_k \Delta t)}{\omega_k} & \frac{\cos(\omega_k \Delta t)-1}{\omega_k} & \frac{\partial p^x_{k+1}}{\partial \omega_k} \\ 0 & 1 & \frac{1-\cos(\omega_k \Delta t)}{\omega_k} & \frac{\sin(\omega_k \Delta t)}{\omega_k} & \frac{\partial p^y_{k+1}}{\partial \omega_k} \\ 0 & 0 & \cos(\omega_k \Delta t) & -\sin(\omega_k \Delta t) & \frac{\partial v^x_{k+1}}{\partial \omega_k} \\ 0 & 0 & \sin(\omega_k \Delta t) & \cos(\omega_k \Delta t) & \frac{\partial v^y_{k+1}}{\partial \omega_k} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.34}$$

$$\frac{\partial p^x_{k+1}}{\partial \omega_k} = \frac{\omega_k \Delta t \cos(\omega_k \Delta t) - \sin(\omega_k \Delta t)}{\omega_k^2} v^x_k - \frac{\omega_k \Delta t \sin(\omega_k \Delta t) + \cos(\omega_k \Delta t) - 1}{\omega_k^2} v^y_k \tag{4.35}$$

$$\frac{\partial p^y_{k+1}}{\partial \omega_k} = \frac{\omega_k \Delta t \sin(\omega_k \Delta t) + \cos(\omega_k \Delta t) - 1}{\omega_k^2} v^x_k - \frac{\omega_k \Delta t \cos(\omega_k \Delta t) - \sin(\omega_k \Delta t)}{\omega_k^2} v^y_k \tag{4.36}$$

$$\frac{\partial v^x_{k+1}}{\partial \omega_k} = -\Delta t \sin(\omega_k \Delta t) v^x_k - \Delta t \cos(\omega_k \Delta t) v^y_k \tag{4.37}$$

$$\frac{\partial v^y_{k+1}}{\partial \omega_k} = -\Delta t \cos(\omega_k \Delta t) v^x_k - \Delta t \sin(\omega_k \Delta t) v^y_k. \tag{4.38}$$

The Jacobian of $h_k(x)$ is

$$H_k = \begin{pmatrix} \frac{-(p^y_k - S^y_1)}{(p^x_k - S^x_1)^2 + (p^y_k - S^y_1)^2} & \frac{p^x_k - S^x_1}{(p^x_k - S^x_1)^2 + (p^y_k - S^y_1)^2} & 0 & 0 & 0 \\ \frac{-(p^y_k - S^y_2)}{(p^x_k - S^x_2)^2 + (p^y_k - S^y_2)^2} & \frac{p^x_k - S^x_2}{(p^x_k - S^x_2)^2 + (p^y_k - S^y_2)^2} & 0 & 0 & 0 \\ \frac{-(p^y_k - S^y_3)}{(p^x_k - S^x_3)^2 + (p^y_k - S^y_3)^2} & \frac{p^x_k - S^x_3}{(p^x_k - S^x_3)^2 + (p^y_k - S^y_3)^2} & 0 & 0 & 0 \end{pmatrix}. \tag{4.39}$$

Our simulations are carried out using a fixed trajectory and thus the expectation operators in (4.32) vanish and the required Jacobians can be calculated using the true trajectory. The recursion in (4.31) is initialized by $J_0 = R_0^{-1}$, where R_0 is the initial covariance matrix of the state x_0.

4.5.3 Simulation Results

We have implemented nine different multiple model filters. All the particle filters are based on the Multiple Model Particle Filter (MMPF) with 2000 particles (Section 4.3.2). The filters were implemented in MATLAB and the code was highly optimized.

- *MMPFall*: a SIR multiple model particle filter which collects all the measurements from all the sensors. There are no OOSMs in this case. (See Section 4.3.2)
- *MMPFmis*: a SIR multiple model particle filter which discards all the OOSMs and therefore only processes the measurements that are not delayed. (See Section 4.3.2)
- *IMM-EKF all*: an interacting multiple model extended Kalman filter which collects all the measurements from all the sensors. There are no OOSMs in this case. (See Section 4.3.1)
- *IMM-EKF mis*: an interacting multiple model extended Kalman filter which discards all the OOSMs and therefore only processes the measurements that are not delayed. (See Section 4.3.1)
- *OOSM-rerunMM*: a SIR multiple model particle filter which processes the OOSMs by re-running MMPF starting with the saved particle cloud at the time step producing OOSM. (See Section 4.4.1)
- *OOSM-GARP-MM* : a SIR multiple model particle filter which processes the OOSMs by re-running particle filter starting with the saved Gaussian approximation of the particle cloud at the time step producing OOSM. (See Section 4.4.2)
- *MMPF-IMM-EKS*: a SIR multiple model particle filter equipped with the extended Kalman smoother, which stores all the Gaussian approximations of the particle clouds. (See Section 4.4.3)
- *EDMMPF-MI*: multiple model particle filter through selective OOSM processing based on the mutual information metric. (See Section 4.4.4)
- *EDMMPF-KL*: multiple model particle filter through selective OOSM processing based on the KL-divergence metric. (See Section 4.4.4)

In this example, all 3 sensors generate OOSMs. The OOSMs arrive at the fusion centre with probability p_{osm}. If they do arrive, they arrive with delay t_d. In the simulations, t_d is uniformly distributed in the range $[0, 10]$ and p_{osm} is set to 0.7. We set ten values of first stage thresholds as $\gamma_1 = 0 : 0.1 : 0.9$ for *EDMMPF-MI* and $\gamma_1 = 0 : 0.2 : 1.8$ for *EDMMPF-KL*. $\gamma_2 = 10\%$ for both of them. The simulation of each threshold shows the average of 1000 Monte Carlo runs.

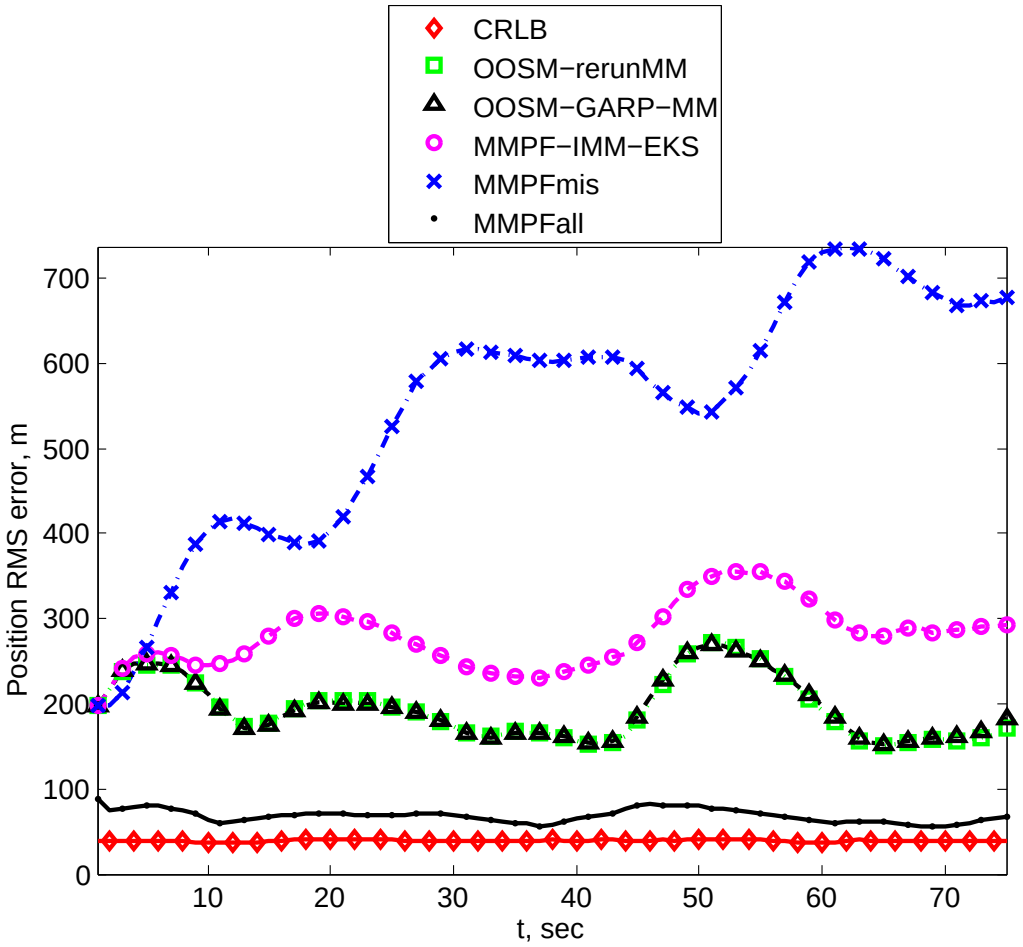


Fig. 4.3 Example 3: RMS position errors for *MMPFall*, *MMPFmis*, *OOSM-rerunMM*, *OOSM-GARP-MM* and *MMPF-IMM-EKS*.

We first plot RMS position errors of *MMPFall*, *MMPFmis*, *OOSM-rerunMM*, *OOSM-GARP-MM*, *MMPF-IMM-EKS* and CRLB, which are shown in Fig. 4.3. In this figure,

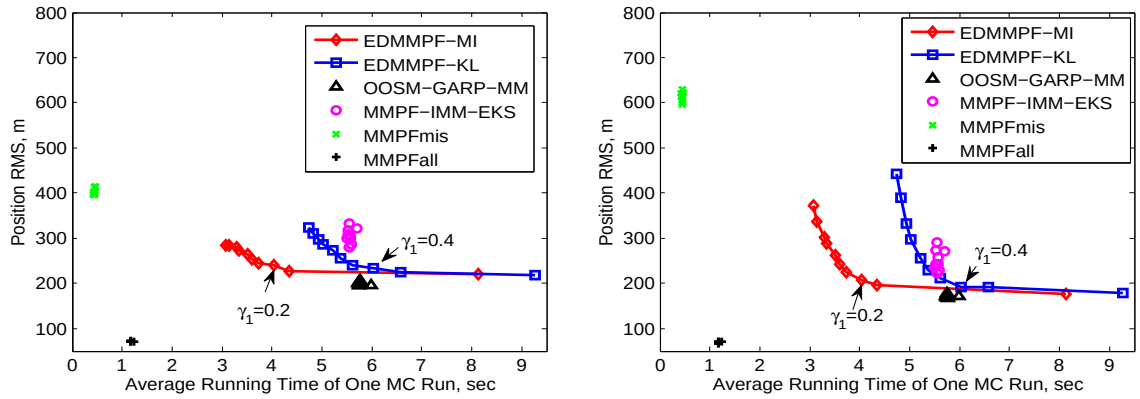
MMPF-IMM-EKS performs much worse than *OOSM-rerunMM* and *OOSM-GARP-MM*. The reason is that many measurements arrive at the fusion centre after a substantial delay. If the fusion centre receives an OOSM after missing measurements for a few time steps, this OOSM can be very informative and can greatly change the distributions of particles. *MMPF-IMM-EKS* fails to process such highly informative OOSMs because it does not change the locations of particles but just updates their weights.

The RMS at time step k vs Average running time of one MC run curves are displayed in Fig. 4.4. In this example, ten diamond points on the red curve represent the results of *SP-MI* with different γ_1 from 0 to 0.9 with step size 0.1 (from right to left). Ten square points on the blue curve represent the results of *SP-KL* with different γ_1 from 0 to 1.8 with step size 0.2 (from right to left). When the thresholds are chosen so that the selective processing filters have the same computational complexity as *MMPF-IMM-EKS*, they achieve significantly better tracking performance. Alternatively, for the same fixed RMS error performance, the selective processing algorithms reduce the computation time by approximately $20 - 40\%$.

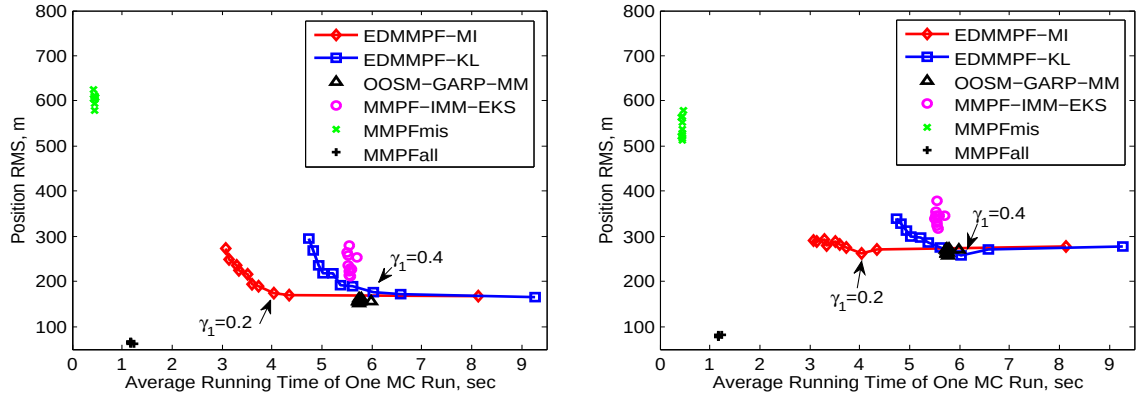
In this case, we do not show the results of *IMM-EKF mis* and *IMM-EKF all* in the figures, since they are totally divergent due to the long period of nonlinear trajectory.

From Fig. 4.4, *EDMMPF-MI* with $\gamma_1 = 0.2$ and *EDMMPF-KL* with $\gamma_1 = 0.4$ can achieve similar performance as re-run filters but with least average running time. In Fig. 4.5, we plot the RMS position performance for $75s$ of the algorithms with these settings. In this example, *MMPF-IMM-EKS* performs much worse than *OOSM-GARP-MM*, but *EDMMPF-MI* and *EDMMPF-KL* can achieve similar performance as *OOSM-GARP-MM*. The selective processing algorithms process highly informative OOSMs by *OOSM-GARP-MM* in order to avoid the deterioration. This additional operation only leads to a small increase in complexity since it only relevant for a very small portion of the OOSMs. From the results, the fraction of individual OOSMs processed by the *MMPF-IMM-EKS* after the first threshold γ_1 is 27.9% for *EDMMPF-MI* and 40.9% for *EDMMPF-KL*. After the second threshold, the fraction of informative OOSMs processed by rerunning the particle filter is 4.8% for *EDMMPF-MI* and 11.3% for *EDMMPF-KL*[3]. In the multiple model case, *EDMMPF-MI* is obviously more efficient than *EDMMPF-KL* because *EDMMPF-MI* processes less OOSMs but achieves similar performance as *EDMMPF-KL*.

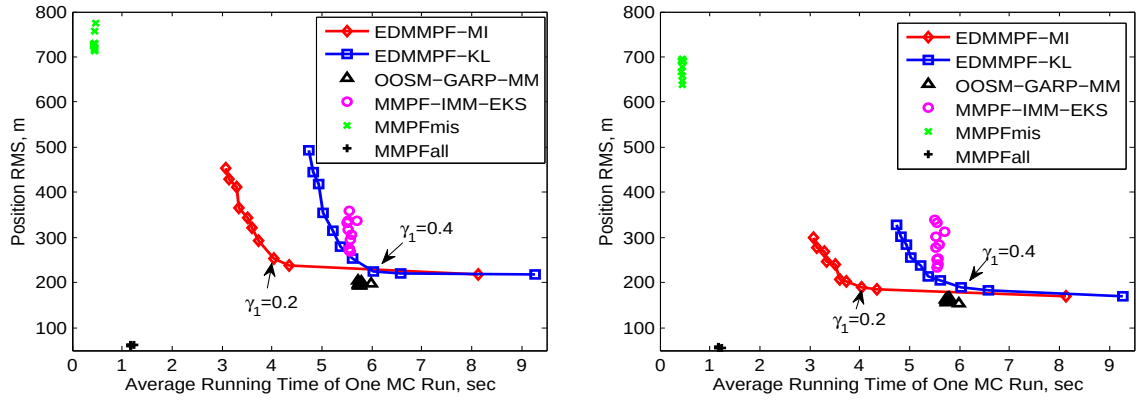
[3]This fraction means the percentage of original received OOSMs.



(a) $N = 2000$, $k = 20$ (b) $N = 2000$, $k = 30$



(c) $N = 2000$, $k = 40$ (d) $N = 2000$, $k = 50$



(e) $N = 2000$, $k = 60$ (f) $N = 2000$, $k = 70$

Fig. 4.4 Example 3: RMS vs Average running time of one MC run from 10 simulations with different γ_1 for *EDMMPF-MI* (from 0 to 0.9) and *EDMMPF-KL* (from 0 to 1.8). We select six timesteps, $k = 20, 30, 40, 50, 60, 70$ for filters with 2000 particles.

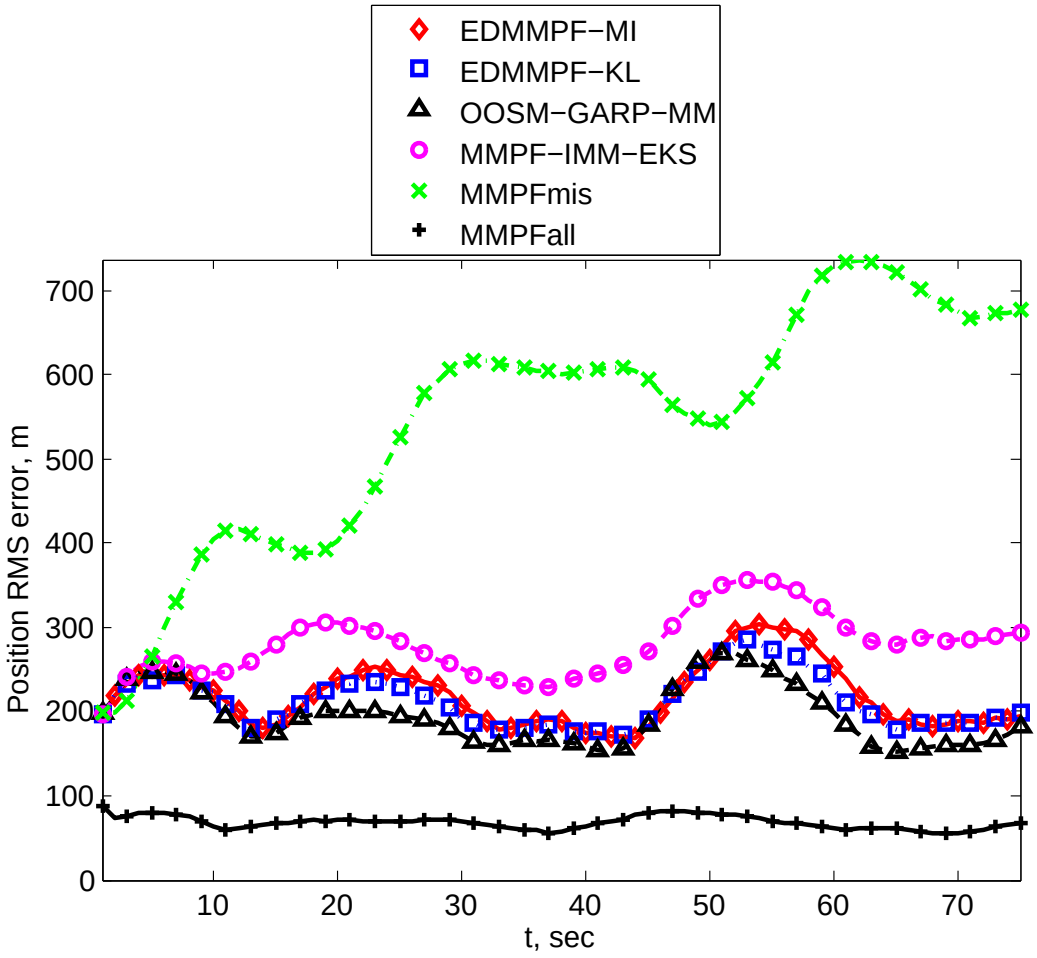


Fig. 4.5 Example 3: RMS position errors ($\gamma_1 = 0.2$ for *EDMMPF-MI* and $\gamma_1 = 0.4$ for *EDMMPF-KL*).

The boxplot figures are shown in Fig. 4.6. These figures show the variation of position RMS error for *MMPF-IMM-EKS*, *OOSM-GARP*, *EDMMPF-MI* and *EDMMPF-KL*. In these figures, an outlier (marked by '+' sign) is defined as a value that is more than 5 times the interquartile range away from the top or bottom of the box. In this example, the distributions of the errors of *MMPF-IMM-EKS* are severely diffused, whereas the other three particle filters have similarly stable performance. For *MMPF-IMM-EKS*, we can deduce that there are probably some divergent tracks in the simulation (See Table 4.1). When there are many significantly-delayed OOSMs, the distributions of current particles cannot

represent the real posterior distribution. The performance of *MMPF-IMM-EKS* becomes poorer because it does not change the locations of the particles but just updates their weights. We need to reprocess the particles and update the stored statistics of particles. Comparing *MMPF-IMM-EKS* and selective processing algorithms, the operation of reprocessing particles by *OOSM-GARP-MM* prevents the divergent trend of *MMPF-IMM-EKS* and results in a major performance improvement.

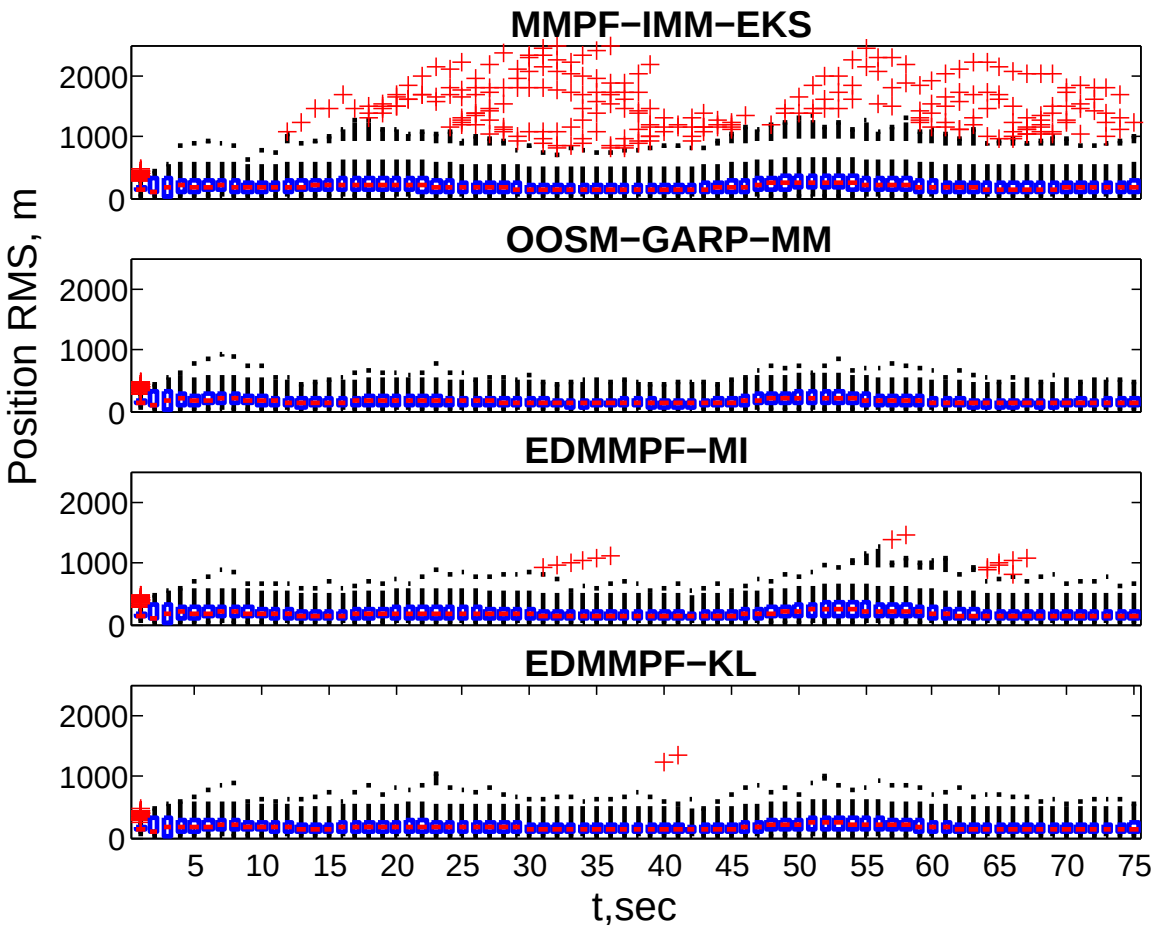


Fig. 4.6 Example 3: Errorbars showing the variation of position RMS errors for *MMPF-IMM-EKS*, *OOSM-GARP-MM*, *EDMMPF-MI*($\gamma_1 = 0.2$) and *EDMMPF-KL*($\gamma_1 = 0.4$).

The overall performance of a filter can be evaluated using RTAMS error as [42]. This is defined as

$$RTAMS = \sqrt{\frac{1}{(t_{max} - t_a) \times M} \sum_{k=t_a+1}^{t_{max}} \sum_{i=1}^{M} (\widehat{p_k^x}^{(i)} - p_k^{x(i)})^2 + (\widehat{p_k^y}^{(i)} - p_k^{y(i)})^2}, \tag{4.40}$$

where t_{max} is the duration of tracking time and t_a is a time index after which the averaging is carried out. M is the total number of Monte Carlo runs. In this case, $t_{max} = 75s$ and we select $t_a = 0$. By defining the track with RTAMS error larger than $500m$ as a divergent track, we can calculate the number of divergent tracks out of 1000 tracks for each filter. We show the RTAMS error, average running time of each filter tracking 75s and number of divergent tracks in Table 4.1. In this table, *IMM-EKF all* and *IMM-EKF mis* are almost always divergent because the EKF cannot satisfactorily track the highly nonlinear dynamics. *EDMMPF-MI* balances well the trade-off between performance and complexity. *EDMMPF-KL* consumes more time since it invokes the *OOSM-GARP-MM* more frequently than *EDMMPF-MI*. When there are frequent batches of OOSMs, the efficiency of *MMPF-IMM-EKS* decreases because it needs to individually process the OOSMs generated at different time steps. On the other hand, the efficiency of *OOSM-GARP-MM* increases since it can process each batch of OOSMs by calling the function once.

Table 4.1 Example 3: Performance comparison of RTAMS error, running time and the number of divergent tracks

Algorithm	Time (s)	RTAMS (m)	Divergent tracks
MMPFall	1.1633	68.8634	0
MMPFmis	0.4384	568.9295	246
IMM-EKF all	0.2632	1.5×10^{10}	962
IMM-EKF mis	0.1891	1.0×10^{9}	903
OOSM-rerunMM	6.3546	196.1403	0
OOSM-GARP-MM	5.7713	196.6750	0
MMPF-IMM-EKS	5.5796	263.5673	10
EDMMPF-MI($\gamma_1 = 0.2$)	4.0286	222.3772	0
EDMMPF-KL($\gamma_1 = 0.4$)	6.0261	214.9070	0

Chapter 5

Conclusion

5.1 Summary and Discussion

In this thesis, we have proposed and analyzed several particle filtering algorithms that can efficiently incorporate out-of-sequence measurements while tracking targets in wireless sensor networks. Our work focuses on balancing the trade-off between the complexity and accuracy of the methods.

Chapter 2 reviews the general Bayesian tracking framework and introduces two popular algorithms. The Kalman filter is a simple and effective algorithm for tracking problems with linear dynamics and measurement function and Gaussian process and measurement noises. Generalization of this algorithm, such as the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF), can track targets with mildly nonlinear dynamics. Particle Filters (PF), which belong to another category — Sequential Monte Carlo (SMC) methods, are applicable to more practical tracking problems with highly non-linear target dynamics and non-Gaussian noises. The latter half of the chapter focuses on the out-of-sequence measurement (OOSM) problem in tracking and provides a detailed mathematical model to describe this situation.

We review three existing methods in Chapter 2. The OOSM Re-run Particle Filter (OOSM-rerun) is an obvious approach for tracking in the presence of OOSMs, but it requires exorbitant memory resources. It must store all the past particles and has high computational complexity because it must reprocess all the particles for several time steps. The algorithm does provide an accuracy benchmark for other approaches in this thesis. The OOSM Particle Filter using Particle Smoothing (OOSM-PS) in [11] has reduced computa-

tional complexity, but it still requires large memory to store past particles. The Storage Efficient Particle Filters (SEPF) proposed in [12] reduce the memory requirements by only storing a Gaussian approximation (the mean and covariance matrix) of the particles for each past time-step. Auxiliary fixed point smoothers are employed to determine the likelihood of a delayed measurement conditioned on each particle in the current set, and this likelihood is used to update the weight of the particle. The Storage Efficient Particle Filters with Extended Kalman Smoother (SEPF-EKS) works well in many cases, especially when the fusion centre seldom receives highly informative OOSMs. However, in some cases, SEPF-EKS fails to deal with highly informative delayed measurements and this leads to performance deterioration.

In Chapter 3, we propose two candidate algorithms. The OOSM Gaussian Approximation Re-run Particle filter (OOSM-GARP) is a storage-efficient version of OOSM-rerun and can achieve similar performance. It reduces the memory requirements by storing a Gaussian approximation of the particles, but still needs to reprocess the particles for several time steps. Our novel efficient delay-tolerant particle filter through selective processing of OOSMs (EDPF-SP) is proposed to reduce the computational complexity. The most important idea of our algorithm is selective processing of OOSMs based on their information content. We propose a computationally-simple selection rule to determine the informativeness of OOSMs. The algorithm immediately discards uninformative OOSMs and then uses the relatively computationally-simple SEPF-EKS to process the informative OOSMs. If significant reduction of the effective particle number is detected after application of SEPF-EKS, we choose to apply OOSM-GARP. The approach combines the advantages of OOSM-GARP and SEPF-EKS in order to achieve a satisfactory tradeoff between accuracy and complexity. From the simulation results of two examples, EDPF-SP only processes $30-40\%$ of all OOSMs using SEPF-EKS and $1-3\%$ of OOSMs using OOSM-GARP, but it can achieve similar performance as the two re-run particle filters.

In Chapter 4, we extend our novel algorithm to the more complicated multiple model tracking case. In the Multiple Model Particle Filter with IMM-EKS (MMPF-IMM-EKS), we use multiple model particle filter (MMPF) as our standard filter to process non-delayed measurements and also employ IMM-EKS to tackle the OOSMs. We also propose an efficient delay-tolerant multiple model particle filter through selective processing of OOSMs (EDMMPF-SP) to reduce the computational complexity and improve the performance of MMPF-IMM-EKS. From a simulation example with long periods of highly nonlinear

dynamics and substantially-delayed OOSMS, we can see the advantages of EDMMPF-SP over existing methods.

5.2 Future Work

We observed that it is difficult to efficiently extend the SEPF-EKS algorithm to process batches of OOSMs, so the computational savings diminish when it is common for multiple OOSMs to arrive in a given time step. Further investigation of how to improve the efficiency of SEPF-EKS to process batches of OOSMs is warranted. On the other hand, the OOSM-GARP algorithm readily accommodates such batches of OOSMs. It is possible to evaluate the number of individual OOSMs in the received batch to decide which approach is more efficient. This detection would improve the speed of our selective processing methods, especially in the case with many batches of OOSMs.

In our algorithms, the thresholds γ_1 and γ_2 are important parameters which determine the complexity and performance of the methods. From the simulation results, the best values vary in different situations. Therefore, another issue is how to adjust these thresholds automatically and preferably on-line. We plan to develop an adaptive process that modifies the thresholds during operation until a satisfactory balance between computational load and accuracy is achieved.

Appendix A

A.1 Algorithm Description of Storage Efficient Particle Filter with Extended Kalman Smoother

In this section, we provide a detailed description of the storage efficient particle filter that employs an extended Kalman smoother, as described in [12]. The algorithm is described for the case when all measurements are available from time steps 1 to k, except for a measurement at time step τ, $\mathcal{Z}_k = \{y_\tau \equiv y_\tau^{\mathcal{D}_{\tau,k}}\}$.

There is one aspect of the following algorithm that is worth noting. The storage efficient particle filter maintains a Gaussian approximation of $p(x_m|\mathcal{W}_k^{\tau:m})$ for all m in the range $k-l-1,\ldots,k$. We denote these distributions by $\mathcal{N}(\mu_{m|m}, R_{m|m})$. When processing an OOSM y_τ at time step k using the algorithm presented below, the algorithm only updates μ_k and R_k in Ω_k, while the Gaussian approximations of $p(x_m|\mathcal{W}_k^{\tau:m})$, denoted by $\mathcal{N}(\mu_{m|m}, R_{m|m})$, are not updated. This means that the approximations at earlier time-steps $m < k$ can become significantly poorer if several informative OOSMs with generation times before m are processed by the algorithm.

Algorithm: SEPF-EKS

Function: `ProcessOOSM-EKS`

Input: ξ_{k-1}, ω_{k-1}, $\widetilde{\mathcal{W}}_k^{k-l:k}$, $\Omega_{k-1} = \{\mu_{k-l-2:k-1}, R_{k-l-2:k-1}\}$

1. $(\xi_k, \omega_k) \leftarrow$ `ParticleFilter`$(\mathcal{Y}_k, \xi_{k-1}, \omega_{k-1})$.

2. Approximate $p(x_\tau|y_{1:\tau-1})$ as $\mathcal{N}(x_\tau; \mu_{\tau|\tau-1}, R_{\tau|\tau-1})$, where

$$\mu_{\tau|\tau-1} = f_{\tau|\tau-1}(\mu_{\tau-1|\tau-1}) \tag{A.1}$$
$$R_{\tau|\tau-1} = F_{\tau|\tau-1} R_{\tau-1|\tau-1} F_{\tau|\tau-1}^T + V_{\tau|\tau-1} \tag{A.2}$$
$$F_{\tau|\tau-1} = \frac{\partial}{\partial x} f_{\tau|\tau-1}(x)|_{x=\mu_{\tau-1|\tau-1}} \tag{A.3}$$
$$\mu_{\tau-1|\tau-1} = \mu_{\tau-1} \in \Omega_{k-1}, R_{\tau-1|\tau-1} = R_{\tau-1} \in \Omega_{k-1}. \tag{A.4}$$

3. Define the initial state and covariance of an augmented system as:

$$z_\tau = \begin{pmatrix} z_\tau^1 \\ z_\tau^2 \end{pmatrix} = \begin{pmatrix} \mu_{\tau|\tau-1} \\ \mu_{\tau|\tau-1} \end{pmatrix}, \quad P_\tau = \begin{pmatrix} R_{\tau|\tau-1} & R_{\tau|\tau-1} \\ R_{\tau|\tau-1} & R_{\tau|\tau-1} \end{pmatrix}. \tag{A.5}$$

Extended Kalman Smoother: from $m = \tau + 1$ to $m = k - 1$

Prediction:

$$z_{m|m-1} = \begin{pmatrix} f_{m|m-1}(z_{m-1}^1) \\ z_{m-1}^2 \end{pmatrix}, \quad P_{m|m-1} = F_m P_m F_m^T + \tilde{V}_m \tag{A.6}$$
$$F_m = \begin{pmatrix} \frac{\partial}{\partial x} f_{m|m-1}(x)|_{x=z_{m-1}^1} & 0 \\ 0 & I \end{pmatrix}, \quad \tilde{V}_m = \begin{pmatrix} V_{m|m-1} & 0 \\ 0 & 0 \end{pmatrix}. \tag{A.7}$$

Updating:

$\tau + 1 \leqslant m \leqslant k - 2$:

$$K_m = P_{m|m-1} H_m^T (H_m P_{m|m-1} H_m^T + \tilde{Q}_m)^{-1} \tag{A.8}$$
$$z_m = z_{m|m-1} + K_m (y_m - h_m(z_{m|m-1}^1)) \tag{A.9}$$
$$P_m = (I - K_m H_m) P_{m|m-1}, \tag{A.10}$$
$$\text{where} \quad H_m = \frac{\partial}{\partial x} h_m(x)|_{x=z_{m|m-1}^1}, \tilde{Q}_m = Q_m. \tag{A.11}$$

$m = k - 1$:

$$K_m = P_{m|m-1} H_m^T (H_m P_{m|m-1} H_m^T + \tilde{Q}_m)^{-1} \tag{A.12}$$

$$z_{k-1}^{(i)} = z_{m|m-1} + K_m \begin{pmatrix} y_m - h_m(z_{m|m-1}^1) \\ x_k^{(i)} - f_{m+1|m}(z_{m|m-1}^1) \end{pmatrix} \tag{A.13}$$

$$P_{k-1}^{(i)} = (I - K_m H_m) P_{m|m-1}, \tag{A.14}$$

$$\text{where} \quad H_m = \begin{pmatrix} \frac{\partial}{\partial x} h_m(x)|_{x=z_{m|m-1}^1} \\ \frac{\partial}{\partial x} f_{m+1|m}(x)|_{x=z_{m|m-1}^1} \end{pmatrix} \tag{A.15}$$

$$\tilde{Q}_m = \begin{pmatrix} Q_m & 0 \\ 0 & V_{k|k-1} \end{pmatrix}. \tag{A.16}$$

4. Obtain the smoothed density

$$p(x_\tau | x_k^{(i)}, y_{1:k-1,\bar{\tau}}) = \mathcal{N}(x_\tau; \mu^x_{\tau|1:k-1,\bar{\tau},k^{(i)}}, R^x_{\tau|1:k-1,\bar{\tau},k^{(i)}}) \tag{A.17}$$

$$\mu^x_{\tau|1:k-1,\bar{\tau},k^{(i)}} = z_{k-1}^{(i),2} : \text{lower block of} \quad z_{k-1}^{(i)} \tag{A.18}$$

$$R^x_{\tau|1:k-1,\bar{\tau},k^{(i)}} = P_{k-1}^{(i),22} : \text{bottom-right block of} \quad P_{k-1}^{(i)}. \tag{A.19}$$

5. Evaluate the likelihood

$$p(y_\tau | x_k^{(i)}, y_{1:k,\bar{\tau}}) = \mathcal{N}(y_\tau; \mu^y_{\tau|1:k,\bar{\tau},k^{(i)}}, R^y_{\tau|1:k,\bar{\tau},k^{(i)}}) \tag{A.20}$$

$$\text{where} \quad \mu^y_{\tau|1:k,\bar{\tau},k^{(i)}} = h_\tau(\mu^x_{\tau|1:k-1,\bar{\tau},k^{(i)}}) \tag{A.21}$$

$$R^y_{\tau|1:k,\bar{\tau},k^{(i)}} = H_\tau R^x_{\tau|1:k-1,\bar{\tau},k^{(i)}} H_\tau^T + Q_\tau \tag{A.22}$$

$$H_\tau = \frac{\partial}{\partial x} h_\tau(x) | x = \mu^x_{\tau|1:k-1,\bar{\tau},k^{(i)}}. \tag{A.23}$$

6. Update and normalize weights:

$$\omega_k{}^{(i)} = \tilde{\omega}_{k,\bar{\tau}}^{(i)} \times p(y_\tau | x_k^{(i)}, y_{1:k,\bar{\tau}}) \tag{A.24}$$

$$\tilde{\omega_k}{}^{(i)} = \frac{\omega_k{}^{(i)}}{\sum_{i=1}^{N} \omega_k{}^{(i)}} \tag{A.25}$$

$$N_{\text{eff}}^{\text{prior}} = 1/\sum_{i=1}^{N} (\tilde{\omega}_{k,\bar{\tau}}^{(i)})^2 \tag{A.26}$$

$$N_{\text{eff}}^{\text{post}} = 1/\sum_{i=1}^{N} (\tilde{\omega_k}{}^{(i)})^2. \tag{A.27}$$

If $N_{\text{eff}}^{\text{post}}/N_{\text{eff}}^{\text{prior}} > \frac{1}{100}$, the weights are updated by $\tilde{\omega_k}{}^{(i)}$, otherwise, the weights $\tilde{\omega}_{k,\bar{\tau}}^{(i)}$ are remained.

7. Resample particles according to the updated weights, calculate the mean μ_k and the covariance matrix R_k and store them into the set Ω_k .

A.2 Algorithm Description of Multiple Model Particle Filter with IMM-EKS

In this section, we provide a detailed description of the multiple model particle filter with IMM-EKS (Section 4.4.3). The algorithm is described for the case when all measurements are available from time steps 1 to k, except for a measurement at time step τ, $\mathcal{Z}_k = \{y_\tau \equiv y_\tau^{\mathcal{D}_{\tau,k}}\}$.

There is one aspect of the following algorithm that is worth noting. The MMPF-IMM-EKS maintains a Gaussian approximation of $p(x_m|\mathcal{W}_k^{\tau:m})$ for all m in the range $k-l-1,\ldots,k$. We denote these distributions by $\mathcal{N}(\mu_{m|m}, R_{m|m})$. When processing an OOSM y_τ at time step k using the algorithm presented below, the algorithm only updates μ_k and R_k in Ω_k, while the Gaussian approximations of $p(x_m|\mathcal{W}_k^{\tau:m})$, denoted by $\mathcal{N}(\mu_{m|m}, R_{m|m})$, are not updated. This means that the approximations at earlier time-steps $m < k$ can become significantly poorer if several informative OOSMs with generation times before m are processed by the algorithm.

Algorithm: MMPF-IMM-EKS

Function: `ProcessOOSM-IMMEKS`

Input: ξ_{k-1}, ω_{k-1}, $\widetilde{\mathcal{W}}_k^{k-l:k}$, $\Omega_{k-1} = \{\mu_{k-l-2:k-1}, R_{k-l-2:k-1}\}$, $\{p(A_m = \alpha)\}_{m=k-l-1:k}^{\alpha=1:r}$

1. $(\xi_k, \omega_k) \leftarrow$ `MMParticleFilter`$(\mathcal{Y}_k, \xi_{k-1}, \omega_{k-1})$.

2. Approximate $p(x_\tau|y_{1:\tau-1})$ as $\mathcal{N}(x_\tau; \mu_{\tau|\tau-1}, R_{\tau|\tau-1})$, where

$$\mu_{\tau|\tau-1}^{\alpha} = f_{\tau|\tau-1}^{\alpha}(\mu_{\tau-1|\tau-1}), (\alpha = 1,\ldots,r) \tag{A.28}$$

$$R_{\tau|\tau-1}^{\alpha} = F_{\tau|\tau-1}^{\alpha} R_{\tau-1|\tau-1} {F_{\tau|\tau-1}^{\alpha}}^T + V_{\tau|\tau-1} \tag{A.29}$$

$$F_{\tau|\tau-1}^{\alpha} = \frac{\partial}{\partial x} f_{\tau|\tau-1}^{\alpha}(x)|_{x=\mu_{\tau-1|\tau-1}} \tag{A.30}$$

$$\mu_{\tau-1|\tau-1} = \mu_{\tau-1} \in \Omega_{k-1}, R_{\tau-1|\tau-1} = R_{\tau-1} \in \Omega_{k-1} \tag{A.31}$$

$$\varphi_{\tau|\tau-1}^{\alpha|\beta} = \frac{1}{\bar{c}_\alpha} p_{\alpha|\beta} \varphi_{\tau-1}^{\beta} \tag{A.32}$$

$$\bar{c}_\alpha = \sum_{\beta=1}^{r} p_{\alpha|\beta} \varphi_{\tau-1}^{\beta}. \tag{A.33}$$

3. Define the initial state and covariance of an augmented system as:

$$z_\tau^\alpha = \begin{pmatrix} \mu_{\tau|\tau-1}^\alpha \\ \mu_{\tau|\tau-1}^\alpha \end{pmatrix}, \quad P_\tau^\alpha = \begin{pmatrix} R_{\tau|\tau-1}^\alpha & R_{\tau|\tau-1}^\alpha \\ R_{\tau|\tau-1}^\alpha & R_{\tau|\tau-1}^\alpha \end{pmatrix}, \quad \varphi_\tau^\alpha = \varphi_{\tau|\tau-1}^{\alpha|\beta}. \tag{A.34}$$

4. Run IMM-Extended Kalman Smoother from $m = \tau+1$ to $m = k-2$ with augmented dynamic and measurement systems, where ${z_{m-1}^\alpha}^1$ and ${z_{m-1}^\alpha}^2$ denote upper block and lower block in z_{m-1}^α:

$$z_m^\alpha = \begin{pmatrix} f_{m|m-1}^\alpha({z_{m-1}^\alpha}^1) \\ {z_{m-1}^\alpha}^2 \end{pmatrix} \tag{A.35}$$

$$\eta_m^\alpha = h_m^\alpha({z_m^\alpha}^1). \tag{A.36}$$

(1)Interaction(mixing): the mixing probabilities $\varphi_k^{\beta|\alpha}, (\alpha, \beta = 1, \ldots, r)$ denote the probabilities that transition from model α to model β and are calculated as:

$$\varphi_m^{\beta|\alpha} = \frac{1}{\bar{c}_\beta} p_{\beta|\alpha} \varphi_{m-1}^\alpha \tag{A.37}$$

$$\bar{c}_\beta = \sum_{\alpha=1}^{r} p_{\beta|\alpha} \varphi_{m-1}^\alpha, \tag{A.38}$$

where φ_{m-1}^α is the probability of model α at the time step $m-1$ and $\bar{c}_\beta$ is a normalizing factor.

Then we can calculate the mixed inputs for each filter :

$$z_{m-1}^{0\beta} = \sum_{\alpha=1}^{r} \varphi_m^{\beta|\alpha} z_{m-1}^\alpha \tag{A.39}$$

$$P_{m-1}^{0\beta} = \sum_{\alpha=1}^{r} \varphi_m^{\beta|\alpha} \times \{P_{m-1|m-1}^\alpha + [z_{m-1|m-1}^\alpha - z_{m-1}^{0\beta}][z_{m-1|m-1}^\alpha - z_{m-1}^{0\beta}]^T\}, \tag{A.40}$$

where z_{m-1}^α and P_{m-1}^α are the estimates of state and covariance for each filter at time step $m-1$.

(2)Filtering: in this step, the inputs of r filters are processed by each filter separately.

$$[z^{\beta}_{m|m-1}, P^{\beta}_{m|m-1}] = EKF_p(z^{0\beta}_{m-1}, P^{0\beta}_{m-1}) \tag{A.41}$$

$$[z^{\beta}_{m}, P^{\beta}_{m}] = EKF_u(z^{\beta}_{m|m-1}, P^{\beta}_{m|m-1}, \eta^{\beta}_{m}). \tag{A.42}$$

In addition to above operation, we also compute the likelihood functions corresponding to each filter by using the measurement residual and its covariance S^{β}_{m}:

$$\Lambda^{\beta}_{m} = \mathcal{N}(\eta^{\beta}_{m}; h^{\beta}_{m}(z^{\beta}_{m|m-1}{}^{1}), S^{\beta}_{m}). \tag{A.43}$$

(3)Combination: in this step, we compute the probability of each model at time step m:

$$\varphi^{\beta}_{m} = \frac{1}{c}\Lambda^{\beta}_{m}\bar{c}_{\beta} \tag{A.44}$$

$$c = \sum_{\beta=1}^{r} \Lambda^{\beta}_{m}\bar{c}_{\beta}, \tag{A.45}$$

where c is a normalizing factor. Here, we do not need combined estimates, so the step of combining estimates can be skipped.

5. At the last time step $m = k-1$, we use particles $x^{(i)}_{k}$ as part of measurements with following function:

$$\eta^{\alpha}_{m} = \begin{pmatrix} h^{\alpha}_{m}(z^{\alpha\,1}_{m}) \\ f^{\alpha}_{m+1|m}(z^{\alpha\,1}_{m}) \end{pmatrix}. \tag{A.46}$$

Operate the interaction and filtering step, then we calculate the likelihood for each model and then combine them with Λ^{β}_{k-1}. In each model with index $\beta = 1, \ldots, r$, (Here, we do not add index β for simplicity)

(1)Get the smoothed density

$$p(x_\tau|x_k^{(i)}, y_{1:k-1,\bar{\tau}}) = \mathcal{N}(x_\tau; \mu^x_{\tau|1:k-1,\bar{\tau},k^{(i)}}, R^x_{\tau|1:k-1,\bar{\tau},k^{(i)}}) \tag{A.47}$$

$$\mu^x_{\tau|1:k-1,\bar{\tau},k^{(i)}} = z^{(i),2}_{k-1} : \text{lower block of} \quad z^{(i)}_{k-1} \tag{A.48}$$

$$R^x_{\tau|1:k-1,\bar{\tau},k^{(i)}} = P^{(i),22}_{k-1} : \text{bottom-right block of} \quad P^{(i)}_{k-1}. \tag{A.49}$$

(2)Evaluate the likelihood

$$p(y_\tau|x_k^{(i)}, y_{1:k,\bar{\tau}}) = \mathcal{N}(y_\tau; \mu^y_{\tau|1:k,\bar{\tau},k^{(i)}}, R^y_{\tau|1:k,\bar{\tau},k^{(i)}}) \tag{A.50}$$

$$\text{where} \quad \mu^y_{\tau|1:k,\bar{\tau},k^{(i)}} = h_\tau(\mu^x_{\tau|1:k-1,\bar{\tau},k^{(i)}}) \tag{A.51}$$

$$R^y_{\tau|1:k,\bar{\tau},k^{(i)}} = H_\tau R^x_{\tau|1:k-1,\bar{\tau},k^{(i)}} H^T_\tau + Q_\tau \tag{A.52}$$

$$H_\tau = \frac{\partial}{\partial x} h_\tau(x)|x = \mu^x_{\tau|1:k-1,\bar{\tau},k^{(i)}}. \tag{A.53}$$

(3)Combine the likelihood of each model:

$$p(y_\tau|x_k^{(i)}, y_{1:k,\bar{\tau}}) = p^\beta(y_\tau|x_k^{(i)}, y_{1:k,\bar{\tau}}) * \Lambda^\beta_{k-1}. \tag{A.54}$$

6. Update and normalize weights:

$$\omega_k{}^{(i)} = \tilde{\omega}^{(i)}_{k,\bar{\tau}} \times p(y_\tau|x_k^{(i)}, y_{1:k,\bar{\tau}}) \tag{A.55}$$

$$\tilde{\omega_k}{}^{(i)} = \frac{\omega_k{}^{(i)}}{\sum_{i=1}^N \omega_k{}^{(i)}} \tag{A.56}$$

$$N^{\text{prior}}_{\text{eff}} = 1/\sum_{i=1}^N (\tilde{\omega}^{(i)}_{k,\bar{\tau}})^2 \tag{A.57}$$

$$N^{\text{post}}_{\text{eff}} = 1/\sum_{i=1}^N (\tilde{\omega_k}{}^{(i)})^2. \tag{A.58}$$

If $N^{\text{post}}_{\text{eff}}/N^{\text{prior}}_{\text{eff}} > \frac{1}{100}$, the weights are updated by $\tilde{\omega_k}{}^{(i)}$, otherwise, the weights $\tilde{\omega}^{(i)}_{k,\bar{\tau}}$ are remained.

7. Resample particles according to the updated weights, calculate the mean μ_k and the covariance matrix R_k and store them into the set Ω_k.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393 – 422, 2002.

[2] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, 2005.

[3] J. K. Hart and K. Martinez, "Environmental sensor networks: A revolution in the earth system science?," *Earth-Science Reviews*, vol. 78, no. 3-4, pp. 177 – 191, 2006.

[4] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, (New York, NY, USA), pp. 88–97, ACM, Sept. 2002.

[5] V. Shnayder, B. Chen, K. Lorincz, T. R. F. F. Jones, and M. Welsh, "Sensor networks for medical care," in *SenSys '05: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, (New York, NY, USA), pp. 314–314, ACM, Nov. 2005.

[6] D. Benhaddou, M. Balakrishnan, and X. Yuan, "Remote healthcare monitoring system architecture using sensor networks," in *2008 IEEE Region 5 Conference*, (Kansas City, MO), pp. 1–6, Apr. 2008.

[7] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292 – 2330, 2008.

[8] R. D. Hilton, D. A. Martin, and W. D. Blair, "Tracking with time-delayed data in multisensor systems," Tech. Rep. NSWCD/TR-93/351, Naval Surface Warfare Center, Dahlgren,VA, Aug. 1993.

[9] Y. Bar-Shalom, H. Chen, and M. Mallick, "One-step solution for the multistep out-of-sequence-measurement problem in tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, pp. 27 – 37, Jan. 2004.

[10] Y. Bar-Shalom and H. Chen, "IMM estimator with out-of-sequence measurements," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, pp. 90–98, Jan. 2005.

[11] M. Orton and W. Fitzgerald, "A Bayesian approach to tracking multiple targets using sensor arrays and particle filters," *IEEE Transactions on Signal Processing*, vol. 50, pp. 216–223, Feb. 2002.

[12] U. Orguner and F. Gustafsson, "Storage efficient particle filters for the out of sequence measurement problem," in *Proceedings of 11th International Conference on Information Fusion*, (Cologne, Germany), pp. 1–8, Jul. 2008.

[13] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME : Journal of Basic Engineering*, vol. 82 (Series D), no. 1, pp. 35–45, 1960.

[14] A. Jazwinski, *Stochastic Processes and Filtering Theory*. New York: Academic Press, 1970.

[15] S. Julier and J. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *Proceeding of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, vol. Multi Sensor Fusion, Tracking and Resoource Management II, (Orlando, FL), Apr. 1997.

[16] A. Doucet and N. de Freitas, N.and Gordon, *Sequential Monte Carlo Methods*. New York: Springer-Verlag, 2001.

[17] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEEE Proceedings F: Radar and Signal Processing*, vol. 140, pp. 107–113, Apr. 1993.

[18] D. B. Rubin, "Using the SIR algorithm to simulate posterior distributions," in *Bayesian Statistics 3* (M. H. Bernardo, K. M. Degroot, D. V. Lindley, and A. F. M. Smith, eds.), Oxford University Press, 1988.

[19] J. Geweke, "Bayesian inference in econometric models using Monte Carlo integration," *Econometrica*, vol. 57, no. 6, pp. 1317–1339, 1989.

[20] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000.

[21] V. Zaritskii, V. Svetnik, and L. Shimelevich, "Monte Carlo technique in problems of optimal data processing," *Automation and Remote Control*, vol. 12, pp. 95–103, 1975.

[22] A. Doucet, N. J. Gordon, and V. Krishnamurthy, "Particle filters for state estimation of jump Markov linear systems," *IEEE Transactions on Signal Processing*, vol. 49, pp. 613–624, Mar. 2001.

[23] R. Douc and O. Cappe, "Comparison of resampling schemes for particle filtering," in *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis(ISPA 2005.)*, (Zagreb, Croatia), pp. 64–69, Sept. 2005.

[24] J. Liu and R. Chen, "Sequential Monte Carlo methods for dynamic systems," *Journal of the American Statistical Association*, vol. 93, no. 443, pp. 1032–1044, 1998.

[25] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *Journal of Computational and Graphical Statistics*, vol. 5, no. 1, pp. 1–25, 1996.

[26] Y. Bar-Shalom and X. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*. Storrs,CT: YBS Publishing, 1995.

[27] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Norwood, MA: Artech House, 1999.

[28] Y. Bar-Shalom, "Update with out-of-sequence measurements in tracking: Exact solution," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, pp. 769 – 778, Jul. 2002.

[29] M. Mallick, S. Coraluppi, and C. Carthel, "Advances in asynchronous and decentralized estimation," in *Proceedings of IEEE Aerospace Conference*, vol. 4, pp. 1873–1888, Mar. 2001.

[30] S. Challa, R. J. Evans, and X. Wang, "A Bayesian solution and its approximations to out-of-sequence measurement problems," *Information Fusion*, vol. 4, pp. 185 – 199, Sept. 2003.

[31] S. Challa, R. J. Evans, X. Wang, and J. Legg, "A fixed-lag smoothing solution to out of sequence information fusion problems," *Communications in Information and Systems*, vol. 2, pp. 325–348, Dec. 2002.

[32] K. Zhang, X. Li, and Y. Zhu, "Optimal update with out-of-sequence measurements," *IEEE Transactions on Signal Processing*, vol. 53, pp. 1992–2004, Jun. 2005.

[33] M. Orton and A. Marrs, "Particle filters for tracking with out-of-sequence measurements," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, pp. 693–702, Apr. 2005.

[34] M. Mallick, T. Kirubarajan, and S. Arulampalam, "Out-of-sequence measurement processing for tracking ground target using particle filters," in *Proceedings of IEEE Aerospace Conference*, vol. 4, pp. 1809–1818, Mar. 2002.

[35] P. Fearnhead, D. Wyncoll, and T. Tawn, "A sequential smoothing algorithm with linear computational cost," Tech. Rep. 8844, Lancaster University, May 2008.

[36] K. K. Biswas and Mahalana.Ak, "Suboptimal algorithms for nonlinear smoothing," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 9, pp. 529–534, Jul. 1973.

[37] S. Julier, J. Uhlmann, and H. F. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Transactions on Automatic Control*, vol. 45, pp. 477–482, Mar. 2000.

[38] S. Kullback and R. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 1317–1339, 1951.

[39] J. L. Williams, J. W. Fisher, and A. Willsky, "Approximate dynamic programming for communication-constrained sensor network management," *IEEE Transactions on Signal Processing*, vol. 55, pp. 3995–4003, Aug. 2007.

[40] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. Wiley-Interscience, 1 ed., Jun. 2001.

[41] P. Tichavsky, C. Muravchik, and A. Nehorai, "Posterior Cramér-Rao bounds for discrete-time nonlinear filtering," *IEEE Transactions on Signal Processing*, vol. 46, pp. 1386–1396, May 1998.

[42] M. Arulampalam, B. Ristic, N. Gordon, and T. Mansell, "Bearings-only tracking of manoeuvring targets using particle filters," *EURASIP J. Appl. Signal Process.*, vol. 15, pp. 2351 – 2365, 2004.

[43] D. Magill, "Optimal adaptive estimation of sampled stochastic processes," *IEEE Transactions on Automatic Control*, vol. 10, pp. 434 – 439, Oct. 1965.

[44] R. L. Moose and P. L. Wang, "An adaptive estimator with learning for a plant containing semi-Markov switching parameters," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 3, pp. 277 –281, May 1973.

[45] R. Moose, "An adaptive state estimation solution to the maneuvering target problem," *IEEE Transactions on Automatic Control*, vol. 20, pp. 359 – 362, Jun. 1975.

[46] N. Gholson and R. Moose, "Maneuvering target tracking using adaptive state estimation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 13, pp. 310 –317, May 1977.

[47] G. Ackerson and K. Fu, "On state estimation in switching environments," *IEEE Transactions on Automatic Control*, vol. 15, pp. 10 – 17, Feb. 1970.

[48] A. Jaffer and S. Gupta, "Recursive Bayesian estimation with uncertain observation," *IEEE Transactions on Information Theory*, vol. 17, pp. 614 – 616, Sept. 1971.

[49] C. Chang and M. Athans, "State estimation for discrete systems with switching parameters," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 14, pp. 418–425, May 1978.

[50] H. A. P. Blom, "An efficient filter for abruptly changing systems," in *Proceedings of the 23rd IEEE Conference on Decision and Control*, vol. 23, pp. 656 –658, Dec. 1984.

[51] H. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with Markovian switching coefficients," *IEEE Transactions on Automatic Control*, vol. 33, pp. 780–783, Aug. 1988.

[52] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, "Interacting multiple model methods in target tracking: A survey," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, pp. 103–123, Jan. 1998.

[53] S. McGinnity and G. W. Irwin, "Multiple model bootstrap filter for maneuvering target tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, pp. 1006–1012, Jul. 2000.

[54] M. Ristic, B.and Arulampalam, "Tracking a manoeuvring target using angle-only measurements: algorithms and performance," *Signal Processing*, vol. 83, pp. 1223–1238, Jun. 2003.

[55] R. Karlsson and N. Bergman, "Auxiliary particle filters for tracking a maneuvering target," in *Proceedings of the 39th IEEE Conference on Decision and Control*, vol. 4, (Sydney, Australia), pp. 3891–3895, Dec. 2000.

[56] X. Wang and S. Challa, "Augmented state IMM-PDA for OOSM solution to maneuvering target tracking in clutter," in *Proceedings of the International Radar Conference*, pp. 479 – 485, Sept. 2003.

[57] S. R. Maskell, R. G. Everitt, R. Wright, and M. Briers, "Multi-target out-of-sequence data association: Tracking using graphical models," *Information Fusion*, vol. 7, pp. 434 – 447, Dec. 2006.

[58] J. Hartikainen and S. Särkkä, "Optimal filtering with Kalman filters and smoothers — a manual for Matlab toolbox EKF/UKF," tech. rep., Department of Biomedical Engineering and Computational Science, Helsinki University of Technology, Finland, Feb. 2008.