Stochastic Bayesian estimation using efficient particle filters for vehicle tracking applications

Giorgos Kravaritis

A thesis submitted for the degree of Doctor of Philosophy.
The University of Edinburgh.
July 2006

Abstract

Most dynamic real-world processes exhibit a stochastic behaviour which, due to unavoidable observation imperfections, cannot be accurately measured. The fields of state estimation and probability theory address problems dealing with this class of systems. In our work we concentrate in particular on particle filtering and focus on nonlinear and non-Gaussian tracking applications. The particle filters are numerical methods based on the sequential Monte Carlo framework which provide powerful, though computational intensive, solutions. They rely on the Bayesian analysis and employ state samples, or particles, which when propagated appropriately over time characterise the evolution of the posterior state probability distribution. Based on that distribution they estimate key statistical characteristics of the observed process. The central focus of our work is on designing particle filters which use more efficiently their particles by seeding them in state space areas with greater significance and/or by varying their number.

We begin by introducing the auxiliary local linearization particle filter (ALLPF) whose importance sampling density brings together the auxiliary sequential importance resampling technique and the local linearization particle filter (LLPF). A simulation study assesses it suitability for tracking manoeuvring targets. We next incorporate the prediction mechanism of the LLPF within a multi-target algorithm. The proposed particle filter (A-MLLPF) performs simultaneously the functions of measurement-to-track assignment and particle prediction while employing an adaptive number of prediction particles. Compared to the equivalent standard multi-target particle filter, we show that the A-MLLPF performs better both in terms of tracking accuracy and measurement association.

The remaining of the thesis is devoted to vehicle tracking which exploits information from the road map. We first focus on the variable-structure multiple model particle filter (VSMMPF) from the literature and we enhance it with a varying particle scheme for using adaptively fewer particles when the vehicle travels on the road. Simulation results show that the proposed variation results in a similar performance but with significant decrease of the particle usage. We then incorporate a gating and a joint probabilistic data association logic into the VSMMPF and use the resulting algorithm (MGTPF) to track multiple vehicles. Simulations demonstrate the suitability of the MGTPF in the multiple vehicle environment and quantify the performance improvement compared to a standard particle filter with an analogous association logic.

Returning to the single-vehicle tracking problem, we introduce lastly the variable mass particle filter (VMPF). The VMPF uses a varying number of particles which allocates efficiently to its propagation modes according to the modes' likelihood and difficulty. For compensating for the resulting statistical irregularities, it assigns to the particles appropriate masses which scale their weights. Other novel features of the proposed algorithm include an on-road propagation mechanism which uses just one particle and a technique for dealing with random road departure angles. Simulation results demonstrate the improved efficiency of the VMPF, since it requires in general fewer particles than the VSMMPF for achieving a better estimation accuracy.

Acknowledgements

Firstly, I would like to thank my supervisor Bernie Mulgrew for the help and guidance he offered throughout the last three years. I would like also to thank my second supervisor Steve McLaughlin for his comments and suggestions on my early work. I am grateful as well to Peter Fielding and Jim McBribe from BAE Systems for all the stimulating discussions on my research from the very beginning. Lastly, I would like to thank my colleagues Yannis Kopsinis and Elias Aboutanios for showing an interest on my work and for sharing their insight with me.

This work was financial supported by BAE Systems and SELEX Sensors and Airborne Systems.

Contents

List of figures

List of tables

Acronyms and abbreviations

AA	auction algorithm
ALLPF	auxiliary local linearization particle filter
A-MLLPF	adaptive multitarget local linearization particle filter
ASIR	auxiliary sequential importance resampling
cdf	cumulative density function
CPU	central processing unit
CT	coordinated turn
CV	constant velocity
EO	electro-optical
EKF	extended Kalman filter
FOV	field of view
GMTI	ground moving target indicator
GNN	global nearest neighbour
GPS	global positioning system
HF	high frequency
IMM	interacting multiple model
IR	infrared
JMPD	joint multitarget probability density
JPDA	joint probabilistic data association
KF	Kalman filter
LLPF	local linearization particle filter
MC	Monte Carlo
MHT	multiple hypothesis tracking
MGTPF	multitarget ground tracking particle filter
MLLPF	multitarget local linearization particle filter
MMPF	multiple model particle filter
MMSE	minimum mean square error
MSDA	multiple scan data association
MSIR	multitarget sequential importance resampling

MTT	multiple target tracking
PDA	probabilistic data association
pdf	probability density function
PF	particle filter
PHD	probability hypothesis density
RV	random variable
RMS	root mean square
SIR	sequential importance resampling
SIS	sequential importance sampling
SMC	sequential Monte Carlo
SR	systematic resampling
SSDA	single scan data association
UKF	unscented Kalman filter
UT	unscented transformation
VMPF	variable mass particle filter
VP-VSMMPF	varying particle variable structure multiple mode particle filter
VSIMM	variable structure interacting multiple models
VSMMPF	variable structure multiple mode particle filter

Nomenclature

a, A	variable
$\mathbf{a}$	vector
$\mathbf{A}$	matrix
$\mathbf{A}^{-T}$	matrix transpose
$\mathbf{A}^{-1}$	matrix inverse
$E\{\cdot\}$	expectation value
$\sigma\{\cdot\}$	standard deviation
$p(\cdot)$	probability density function
$p(\cdot\|\cdot)$	conditional probability density function
$\delta(\cdot)$	Dirac function
$\mathcal{N}(\cdot,\cdot)$	Gaussian probability distribution
$\mathcal{U}(\cdot,\cdot)$	uniform probability distribution
$\setminus\{\}$	set difference
$\|\{\}\|$	set cardinality
$\mathbb{R}$	set of real numbers
$\mathbb{N}$	set of natural numbers
$\|\cdot\|_2$	Euclidean norm
$\max\{\}$	maximum value
$\min\{\}$	maximum value
x_k	x-axis position at time k
y_k	y-axis position at time k
z_k	z-axis position at time k
r_k	range at time k
θ_k	azimuth angle at time k
ϵ_k	elevation angle at time k

Chapter 1
Introduction

1.1 Background

Most dynamic systems in nature, albeit causal, due to their complexity cannot be deterministically modelled. Moreover, inherent imprecision and physical limitations of the measuring sensors do not allow for accurate system observations. The approach throughout the scientific literature, from the areas of physics to economics, is to model that kind of systems as random processes. This is also the practise used in the target tracking problem that we study. In our work we rely particularly on the fields of state estimation and probability theory - specifically on particle filtering - which provide powerful tools for inferring the states of a stochastic system from a set of noisy measurements.

Back in the 60's the introduction of the Kalman filter (KF) provided the optimal state estimator for the case of linear and Gaussian systems. The KF is a recursive algorithm with a prediction-correction structure, in which the states at every time step are first predicted using the system model and then corrected using the measurements. The filter found immediately numerous applications in areas such as navigation, radar tracking and satellite orbit determination. The extended KF (EKF) and the unscented KF (UKF) are popular variations of the original algorithm which generalised the method for dealing with nonlinear systems.

The particle filter (PF) was introduced in the beginning of the 90's and offered a more powerful solution to the nonlinear and non-Gaussian estimation problem. The PF follows the Bayesian analysis and relies on numerical integration techniques to calculate discrete approximations of the probabilistic distribution of the states. Its estimation power stems not only from the efficient way it deals with the nonlinearity and non-Gaussianity, but from the fact as well that available non-standard information can be easily incorporated within its structure to account for application-specific requirements and characteristics.

The focus of this thesis is mainly on track estimation, a fundamental function of the broader target tracking problem. We assume that we have probabilistic models of the target and the

measurement sensor, which we use along with the measurements to derive the target's state estimates. Usually these states consist of the position, velocity and/or acceleration of the target, but other features like the angular orientation or identity can be found in different applications. Historically, KF-based algorithms have been used for track estimation extensively over the years. However over the last decade, and mainly due to the higher modern computational capabilities, we witnessed an increasing interest in exploiting particle filtering techniques for the problem.

1.2 Motivation of work

Particle filtering currently seems to be the most promising technique for addressing the difficult nonlinear/non-Gaussian Bayesian estimation problem. The particle filters using Monte Carlo integration and importance sampling techniques, employ a set of weighted samples or *particles* of the state density which they propagate over time to provide a sequential discrete approximation of the posterior state distribution. The number of particles for achieving a certain performance varies significantly across the applications but the higher the dimensionality and the difficulty of the system, the more particles are required. It is clear that the increasingly high computational demands in realistic complex applications soon become a problem.

It has been shown that the number of the particles along with the computational requirements can be reduced considerably with the choice of an appropriate particle prediction technique. In fact most of the algorithmic variations that have been proposed differ primarily on their prediction mechanism, the choice of which once more can be heavily application-specific. Focusing on an efficient propagation scheme is thus essential not only for enhancing the filter performance but also for lowering the computational requirements.

In our work we concentrate mainly on the problem of tracking a vehicle while exploiting available information from the road maps. We focus on the specific problem not only due to the increasing interest it has received over the last years, but also because its increased complexity make it particularly suitable for contrasting different tracking approaches. We base our work mostly on the variable-structure multiple model particle filter (VSMMPF) aiming to improve its particle efficiency. We focus specifically on reducing its particle requirements and enhancing furthermore its tracking performance using improved particle propagation mechanisms. The idea of exploiting adaptive algorithmic structures which vary the number of their particles is

also central to our work and is applied to both the single- and the multiple-target problem.

1.3 Thesis organisation

The second chapter introduces the tracking problem and offers background material upon which the rest of the thesis is based. It presents basic concepts of state estimation, target tracking and data association. It concentrates particularly on Bayesian estimation, linear and nonlinear Kalman filtering, target and radar modelling and measurement-to-track assignment using the nearest neighbour and the joint probabilistic data association (JPDA) algorithms. Short simulation studies contrast a number of algorithms that we present.

The third chapter focuses on particle filtering. It describes its generic principles and presents popular filter variations. A method for tracking a dynamically changing target, which combines elements from the auxiliary and the local linearization particle filters is also presented, along with a brief simulation analysis which investigates its performance. Moreover, a novel algorithm is introduced which incorporates the local linearization propagation mechanism into the multitarget problem and varies the number of its association particles to reduce the computational requirements. Simulation results demonstrate the improved estimation and measurement-assignment capability of the proposed approach.

The remaining chapters concentrate on vehicle tracking with particle filters which exploit road maps to constrain the vehicle motion. The fourth chapter describes briefly the problem and presents the single-target VSMMPF from the literature. A novel mechanism is introduced which varies the number of the on-road particles exploiting the fact that the state uncertainty when the vehicle travels on the road is smaller. A simulation study is presented which suggests that the performance degradation can be considered negligible considering the lighter particle usage. An novel architecture is also described which incorporates a gating function and a JPDA logic to the VSMMPF, enhancing the standard algorithm with clutter rejection and measurement-to-track assignment features. A simulation study analyses experimentally the association function of the algorithm.

The fifth chapter describes a different approach to deal with the inherent multi-modality of the vehicle tracking problem. It introduces a novel particle filtering structure in which the particles are assigned with variable masses. This approach allows for the particles to be efficiently allocated across the propagation models using additional information without biasing the estimation

process. A computationally inexpensive Kalman filtering mechanism for the on-road part of the prediction is also described. Simulation results show that the proposed variable mass particle filter can achieve better performance, while using fewer particles and less computational power compared to the standard VSMMPF.

Finally, the sixth chapter summarises the achievements of our work, presents a number of identified limitations and proposes possible areas for further research.

Chapter 2
Nonlinear tracking and data association

The second chapter introduces target tracking and offers background material upon which the rest of the thesis is based. The first part is dedicated to basic concepts of state estimation and tracking. It concentrates on Bayesian estimation, Kalman filtering and tracking modelling and presents briefly a comparison of two nonlinear Kalman-based trackers. The remaining chapter focuses on the problem of data association. It describes the nearest neighbour and the probabilistic data association algorithms and shows simulation results comparing their performances.

2.1 Bayesian estimation

Within the heart of target tracking [1] lies the estimation or filtering problem. The aim of that problem is to estimate sequentially the state of a dynamic system by processing a sequence of noisy measurements of the system [2]. Since the systems that we study exhibit stochastic behaviours, we use for modelling and analysis a probabilistic state-space representation [3, 4]. The notation that we use throughout the thesis is bold uppercase roman letters for matrices ($\mathbf{A}$), bold lowercase roman letters for vectors ($\mathbf{a}$) and italic letters for scalars (A, a). The transpose of the matrix $\mathbf{A}$ is denoted as $\mathbf{A}^T$ and its inverse as $\mathbf{A}^{-1}$.

Consider a system with state vector $\mathbf{x}_k \in \mathbb{R}^{n_x}$, where n_x is the dimensionality of the state-space, $\mathbb{R}$ is the set of real numbers and k a time index which takes its values from $k \in \mathbb{N}^+$, $\mathbb{N}^+$ being the set of positive natural numbers. Assume that the state transition equation over time is:

$$\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \tag{2.1}$$

where $\mathbf{f}_k(\cdot)$ is the system function and $\mathbf{u}_k$ is the process noise vector, both at time k. The equation that relates the states with the measurements for every k is:

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{v}_k) \tag{2.2}$$

where $\mathbf{z}_k \in \mathbb{R}^{n_z}$ is the measurement vector, n_z is the dimensionality of the measurement-space (usually smaller than that of the state-space), $\mathbf{h}_k(\cdot)$ is the measurement function and $\mathbf{v}_k$ is the measurement noise vector.

If we define $\mathbf{Z}_k$ as the set of all measurements up to time k, the aim of the filtering problem is to construct the posterior state probability density function (pdf) based on the previous measurements:

$$p(\mathbf{x}_k|\mathbf{Z}_k) = p(\mathbf{x}_k|\{\mathbf{z}_i, i = 1, \ldots, k\}) \tag{2.3}$$

where notation $p(\mathcal{A}|\mathcal{B})$ stands for the conditional probability density of event $\mathcal{A}$ assuming event $\mathcal{B}$.

Before deriving $p(\mathbf{x}_k|\mathbf{Z}_k)$, we need to define first some fundamental relations. Consider the random variables (RV) $\alpha_1, \alpha_2, \alpha_3$ and α_4. By definition the conditional probability of α_1, α_2 given α_3, α_4 is:

$$p(\alpha_1, \alpha_2|\alpha_3, \alpha_4) = \frac{p(\alpha_1, \alpha_2, \alpha_3, \alpha_4)}{p(\alpha_3, \alpha_4)} \tag{2.4}$$

Another important relation is the Chapman-Kolmogorov identity, which can relate the conditional probability distributions of RVs from different sets of coordinates. For RVs α_1, α_2 and α_3, it has the form:

$$p(\alpha_1|\alpha_3) = \int_{-\infty}^{\infty} p(\alpha_1|\alpha_2, \alpha_3)p(\alpha_2|\alpha_3)d\alpha_2 \tag{2.5}$$

Finally, the basic property of conditional probabilities is called Bayes' rule and for the RVs α_1, α_2 and α_3 becomes:

$$p(\alpha_1|\alpha_2, \alpha_3) = \frac{p(\alpha_2|\alpha_1, \alpha_3)p(\alpha_1|\alpha_3)}{p(\alpha_2|\alpha_3)} \tag{2.6}$$

Consider again the filtering problem described before. The Chapman-Kolmogorov equation from (2.5) using the states $\mathbf{x}_k, \mathbf{x}_{k-1}$, and the measurement set $\mathbf{Z}_{k-1}$ gives us:

$$p(\mathbf{x}_k|\mathbf{Z}_{k-1}) = \int_{-\infty}^{\infty} p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{Z}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1})d\mathbf{x}_{k-1} \tag{2.7}$$

In our analysis we consider Markovian systems, that is systems whose states at k depend just on the states at $k-1$, i.e.:

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{x}_{k-2}, \ldots, \mathbf{x}_0) = p(\mathbf{x}_k|\mathbf{x}_{k-1}) \tag{2.8}$$

From (2.8) we get $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{Z}_{k-1}) = p(\mathbf{x}_k|\mathbf{x}_{k-1})$, therefore (2.7) becomes:

$$p(\mathbf{x}_k|\mathbf{Z}_{k-1}) = \int_{-\infty}^{\infty} p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1}) d\mathbf{x}_{k-1} \tag{2.9}$$

where the transitional density $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ is implied from the state transition equation:

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}) \Leftrightarrow \mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \mathbf{v}_{k-1}) \tag{2.10}$$

For deriving the posterior state density, we make use of the Bayes' rule and we get:

$$\begin{aligned} p(\mathbf{x}_k|\mathbf{Z}_k) &= p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{Z}_{k-1}) \\ &= \frac{p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{Z}_{k-1}) p(\mathbf{x}_k|\mathbf{Z}_{k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{k-1})} \\ &= \frac{p(\mathbf{z}_k|\mathbf{x}_k) p(\mathbf{x}_k|\mathbf{Z}_{k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{k-1})} \end{aligned} \tag{2.11}$$

where the normalising factor at the denominator can be written using (2.5) as:

$$p(\mathbf{z}_k|\mathbf{Z}_{k-1}) = \int_{-\infty}^{\infty} p(\mathbf{z}_k|\mathbf{x}_k) p(\mathbf{x}_k|\mathbf{Z}_{k-1}) d\mathbf{x}_k \tag{2.12}$$

and the pdf of the measurement given the state can be directly deduced from the measurement equation:

$$p(\mathbf{z}_k|\mathbf{x}_k) \Leftrightarrow \mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{v}_k) \tag{2.13}$$

The pdf $p(\mathbf{z}_k|\mathbf{x}_k)$ is often called the likelihood and $p(\mathbf{x}_k|\mathbf{Z}_{k-1})$ the a-priori state density. By computing the likelihood (2.13), the prior (2.9) and the normalising factor (2.12), we can calculate the a-posteriori state density and thus construct the optimal estimate of $\mathbf{x}_k$ in the minimum mean square error estimate (MMSE) sense. The MMSE problem is to determine an estimate $\hat{\mathbf{x}}_k$ using the data $\mathbf{Z}_k$, such that the resulting mean square error:

$$e = \mathrm{E}\{[\mathbf{x}_k - \hat{\mathbf{x}}_k]^2\} \tag{2.14}$$

is minimum. Notation $\mathrm{E}\{\cdot\}$ stands for the expectation value. The error e is minimum if $\hat{\mathbf{x}}_k$ equals the conditional mean of $\mathbf{x}_k$ assuming the data, therefore:

$$\hat{\mathbf{x}}_k = \mathrm{E}\{\mathbf{x}_k|\mathbf{Z}_k\} = \int_{-\infty}^{\infty} \mathbf{x}_k p(\mathbf{x}_k|\mathbf{Z}_k) d\mathbf{x}_k \tag{2.15}$$

where $p(\mathbf{x}_k|\mathbf{Z}_k)$ is the posterior given by (2.11). For futher reading, textbooks [5, 6] offer a comprehensive analysis on probability theory and the principles of Bayesian filtering.

2.2 Kalman filter

The Kalman filter (KF) [7, 8] is the *optimal* MMSE state estimator, which can be applied to linear and Gaussian stochastic systems. It was introduced in the 60's, revolutionising the field of recursive linear filtering and finding numerous applications particularly in the area of autonomous or assisted navigation. Figure 2.1 shows an example of noise filtering using a KF for a two-state dynamic system. In what follows we concentrate just on the discrete realisation of the filter.

Suppose that a measurement $\mathbf{z}_k$ has been made at time k, which is to be used in updating the estimate of the state $\mathbf{x}_k$ at time k of a linear system with transition equation:

$$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{u}_{k-1} \tag{2.16}$$

where $\mathbf{F}_k$ is the system matrix and $\mathbf{u}_k$ is the process noise vector. We assume that the measurement $\mathbf{z}_k$ is linearly related to the state with the following measurement equation:

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \tag{2.17}$$

where $\mathbf{H}_k$ is the measurement matrix and $\mathbf{v}_k$ the measurement noise vector. The noise vectors $\mathbf{u}_k$ and $\mathbf{v}_k$ are assumed to be zero-mean, mutually independent, Gaussian with respective covariances $\mathbf{Q}_k$ and $\mathbf{R}_k$.

$$\mathbf{u}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k),\ \mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k) \tag{2.18}$$

where notation $\mathcal{N}(\mathcal{A}, \mathcal{B})$ stands for the Normal or Gaussian distribution with mean $\mathcal{A}$ and covariance $\mathcal{B}$.

The KF consists of a prediction and an update step. The algorithm first predicts the a-priori state pdf and subsequently updates it with the measurement for deriving the posterior pdf.

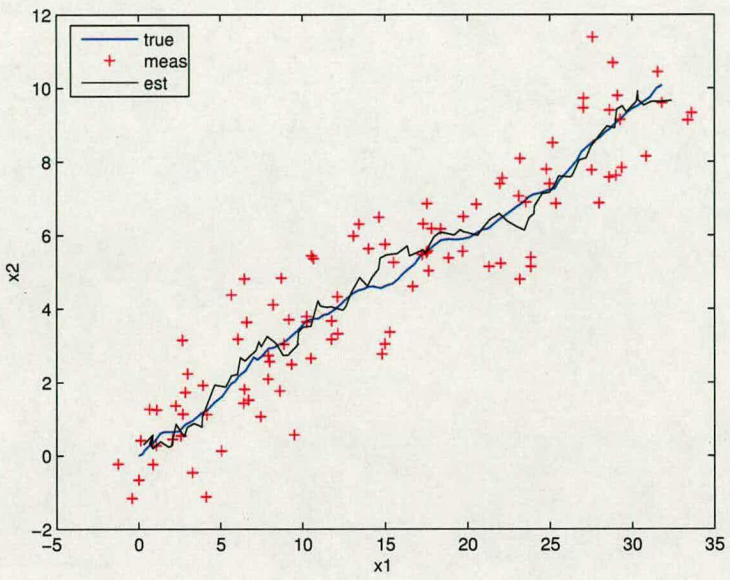


Figure 2.1: *Noise filtering for a linear dynamic system with states $x1$ and $x2$, using a KF.*

A. Prediction step:

$$\mathbf{x}_k^- = \mathbf{F}_{k-1}\mathbf{x}_{k-1} \tag{2.19}$$

$$\mathbf{P}_k^- = \mathbf{F}_{k-1}\mathbf{P}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1} \tag{2.20}$$

B. Update step:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T[\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k]^{-1} \tag{2.21}$$

$$\mathbf{x}_k = \mathbf{x}_k^- + \mathbf{K}_k[\mathbf{z}_k - \mathbf{H}_k\mathbf{x}_k^-] \tag{2.22}$$

$$\mathbf{P}_k = [\mathbf{I} - \mathbf{K}_k\mathbf{H}_k]\mathbf{P}_k^- \tag{2.23}$$

where $\mathbf{P}_k$ is the state covariance matrix, $\mathbf{I}$ is the identity matrix, $\mathbf{x}_k^-$ and $\mathbf{P}_k^-$ are the prior state and covariance and $\mathbf{K}_k$ is the Kalman matrix which corrects the predicted prior state pdf for deriving the posterior. Although equations (2.19-2.23) are the most commonly used, different forms exist when numerical and stability issues arise [8]. For example, for overcoming ill-conditioning when computing the covariance in (2.23), the Joseph form can be used [9]; or for robustifying the filter against round-off errors the information form of the KF can be used [10].

2.3 Extended Kalman filter

The extended Kalman filter (EKF) [11] is a variant of the KF designed specifically for nonlinear systems. It is particularly appropriate for systems that undergo smooth nonlinearities which locally can be approximated as linear for small perturbations of the states. The EKF is the most well-studied and widely used recursive nonlinear estimator, with many textbook chapters and papers dedicated to it [8, 12, 13].

Suppose that a discrete stochastic system can be represented by the following nonlinear process and measurement equations:

$$\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}) + \mathbf{u}_{k-1} \tag{2.24}$$

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k \tag{2.25}$$

where the functions $\mathbf{f}_k(\cdot)$ and $\mathbf{h}_k(\cdot)$ are continuously differentiable nonlinear functions of the state $\mathbf{x}_k$. The algorithm linearizes the system using a first-order Taylor series expansion [14] of the system function at the previous state estimate and of the measurement function at the current state estimate. The EKF first predicts the state using the system model and updates then the estimate with the measurement, using at every time instant the linearized quantities.

A. Prediction step:

$$\hat{\mathbf{F}}_{k-1} = \left.\frac{\partial \mathbf{f}_{k-1}(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k=\mathbf{x}_{k-1}} \tag{2.26}$$

$$\mathbf{x}_k^- = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}) \tag{2.27}$$

$$\mathbf{P}_k^- = \hat{\mathbf{F}}_{k-1}\mathbf{P}_{k-1}\hat{\mathbf{F}}_{k-1}^T + \mathbf{Q}_{k-1} \tag{2.28}$$

B. Update step:

$$\hat{\mathbf{H}}_k = \left.\frac{\partial \mathbf{h}_k(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k=\mathbf{x}_k^-} \tag{2.29}$$

$$\mathbf{K}_k = \mathbf{P}_k^-\hat{\mathbf{H}}_k^T[\hat{\mathbf{H}}_k\mathbf{P}_k^-\hat{\mathbf{H}}_k^T + \mathbf{R}_k]^{-1} \tag{2.30}$$

$$\mathbf{x}_k = \mathbf{x}_k^- + \mathbf{K}_k[\mathbf{z}_k - \mathbf{h}_k(\mathbf{x}_k^-)] \tag{2.31}$$

$$\mathbf{P}_k = [\mathbf{I} - \mathbf{K}_k\hat{\mathbf{H}}_k]\mathbf{P}_k^- \tag{2.32}$$

where $\hat{\mathbf{F}}_k$ and $\hat{\mathbf{H}}_k$ are the Jacobian matrices of the nonlinear functions $\mathbf{f}_k$ and $\mathbf{h}_k$ evaluated

respectively at $\mathbf{x}_{k-1}$ and $\mathbf{x}_k^-$. Convergence to a reasonable estimate may not be obtained if the linearization is inadequate to describe the system [15]. However, the EKF can improve its performance, at the expense of computational load, by keeping the second- or higher-order terms during the linearization [12, 16].

2.4 Unscented Kalman filter

The unscented Kalman filter (UKF) is another nonlinear filter based on the KF and the unscented transformation (UT) [17]. It approximates the state pdf as Gaussian and it represents it through a set of deterministically selected sample points. These sample points capture the exact prior mean and covariance of the state distribution, and when propagated through the nonlinear systems dynamics, represent the posterior mean and covariance accurately to at least the second-order of the Taylor series expansion. In contrast to the EKF there is no need to evaluate Jacobian matrices for linearization and thus the UKF can be also applied to non-differentiable nonlinear systems.

2.4.1 Unscented transformation

The UT calculates the statistics of a random variable which experiences a nonlinear transformation. Consider propagating an n-dimensional RV $\mathbf{x}$ through a nonlinear function $\mathbf{f}(\cdot)$ for obtaining $\mathbf{y}$:

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) \tag{2.33}$$

Assume that $\mathbf{x}$ has mean $\bar{\mathbf{x}}$ and covariance $\mathbf{P}^{xx}$. To calculate the statistics of $\mathbf{y}$, we form a weighted set $\{\mathbf{X}, W\} = \{\mathbf{X}_i, W_i\}_{i=0}^{2n}$ consisting of $2n+1$ *sigma* points $\mathbf{X}_i$, each n-dimensional, and their associated weights W_i:

$$\begin{aligned}
\mathbf{X}_0 &= \bar{\mathbf{x}} &,\ W_0 &= \kappa/(n+\kappa) \\
\mathbf{X}_i &= \bar{\mathbf{x}} + \left(\sqrt{(n+\kappa)\mathbf{P}^{xx}}\right)_i &,\ W_i &= 1/2(n+\kappa) \\
\mathbf{X}_{i+n} &= \bar{\mathbf{x}} - \left(\sqrt{(n+\kappa)\mathbf{P}^{xx}}\right)_i &,\ W_{i+n} &= 1/2(n+\kappa)
\end{aligned} \tag{2.34}$$

where $\left(\sqrt{(n+\kappa)\mathbf{P}^{xx}}\right)_i$ is the ith row of the matrix square root of $(n+\kappa)\mathbf{P}^{xx}$. The matrix square root can be computed using numerically efficient and stable methods such as Cholesky decomposition [18]. The variable $\kappa \in \mathbb{R}$ is used to fine tune the higher order moments of the

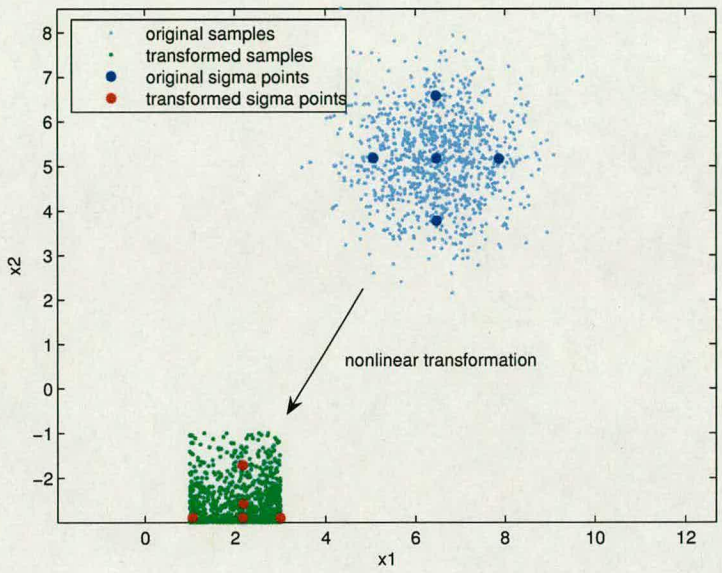


Figure 2.2: *Unscented transformation on a two-dimensional plane for variable* $\mathbf{x} = [x1 \ x2]^T$.

nonlinear approximation and to reduce the approximation errors. When the distribution of $\mathbf{x}$ is Gaussian, a practical heuristic [17] is to set:

$$n + \kappa = 3 \tag{2.35}$$

In the UT algorithm, first we propagate each sigma point through the nonlinear function $\mathbf{f}$:

$$\mathbf{Y}_i = \mathbf{f}(\mathbf{X}_i) \tag{2.36}$$

we obtain then the estimated mean $\bar{\mathbf{y}}$ by the weighted average of the transformed sigma points:

$$\bar{\mathbf{y}} = \sum_{i=0}^{2n} W_i \mathbf{Y}_i \tag{2.37}$$

and finally we compute the covariance by the weighted outer product of the transformed sigma points:

$$\mathbf{P}^{yy} = \sum_{i=0}^{2n} W_i [\mathbf{Y}_i - \bar{\mathbf{y}}][\mathbf{Y}_i - \bar{\mathbf{y}}]^T \tag{2.38}$$

Figure 2.2 demonstrates graphically the basic principle of the UT method.

2.4.2 UKF algorithm

For implementing the UKF [17, 19], we consider a system with equations (2.24-2.25), in which the state vector is augmented with the process noise term $\mathbf{u}_k$:

$$\mathbf{x}_k^a = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix} \tag{2.39}$$

whose dimensions will be $n_a = n_x + n_u$, where n_x and n_u are respectively the dimensions of $\mathbf{x}_k$ and $\mathbf{u}_k$. We re-define the system model as a function of $\mathbf{x}_k^a$:

$$\mathbf{x}_k^a = \mathbf{f}_{k-1}^a(\mathbf{x}_{k-1}^a) \tag{2.40}$$

The unscented transform uses $2n_a + 1$ sigma points $\{\mathbf{X}_{i,k}^a\}_{i=0}^{2n_a}$ as defined in (2.34), whose covariance matrix is now augmented and has the form:

$$\mathbf{P}_k^a = \begin{bmatrix} \mathbf{P}_k & \mathbf{P}_k^{xv} \\ \mathbf{P}_k^{xv} & \mathbf{Q}_k \end{bmatrix} \tag{2.41}$$

where $\mathbf{P}_k$ and $\mathbf{Q}_k$ are respectively the covariance matrices of the state and the process noise and $\mathbf{P}_k^{xv}$ the correlation across them. The measurement function is altered accordingly to account for the augmented dimensionality of the sigma points:

$$\mathbf{Z}_{i,k} = \mathbf{h}_k^a(\mathbf{X}_{i,k}^a), \;\; i = 0, \ldots, 2n_a \tag{2.42}$$

The UKF consists of a prediction and an update step. In the prediction step first we propagate the sigma points one step ahead using the augmented form of the system function (2.40). From the resulting $\{\mathbf{X}_{i,k}^a\}_{i=0}^{2n_a}$ points we obtain the a-priori estimates of the states $\hat{\mathbf{x}}_k^-$ and covariance $\hat{\mathbf{P}}_k^-$. In the update step that follows, we use the measurement for deriving the Kalman matrix $\mathbf{K}_k$ and correcting the prior estimates, in the usual KF fashion. For computing $\mathbf{K}_k$, first we propagate the sigma points through the measurement function (2.42) to get $\{\mathbf{Z}_{i,k}\}_{i=0}^{2n_a}$. From the transformed points we compute the measurement estimate $\hat{\mathbf{z}}_k$, the covariance $\mathbf{P}_k^{zz}$ and the cross-correlation matrix $\mathbf{P}_k^{xz}$ between the states and the measurement. The algorithm equations

are presented next:

A. Prediction step:

$$\mathbf{X}^a_{i,k} = \mathbf{f}^a_{k-1}(\mathbf{X}^a_{i,k-1}),\ i = 0, \ldots, 2n_a \tag{2.43}$$

$$\hat{\mathbf{x}}^-_k = \sum_{i=0}^{2n_a} \mathbf{W}_i \mathbf{X}^a_{i,k} \tag{2.44}$$

$$\hat{\mathbf{P}}^-_k = \mathbf{Q}_{k-1} + \sum_{i=0}^{2n_a} \mathbf{W}_i [\mathbf{X}^a_{i,k} - \hat{\mathbf{x}}^-_k][\mathbf{X}^a_{i,k} - \hat{\mathbf{x}}^-_k]^T \tag{2.45}$$

$$\tag{2.46}$$

B. Update step:

$$\mathbf{Z}_{i,k} = \mathbf{h}^a_k(\mathbf{X}^a_{i,k}),\ i = 0, \ldots, 2n_a \tag{2.47}$$

$$\hat{\mathbf{z}}_k = \sum_{i=0}^{2n_a} \mathbf{W}_i \mathbf{Z}_{i,k} \tag{2.48}$$

$$\mathbf{P}^{zz}_k = \sum_{i=0}^{2n_a} \mathbf{W}_i [\mathbf{Z}_{i,k} - \hat{\mathbf{z}}_k][\mathbf{Z}_{i,k} - \hat{\mathbf{z}}_k]^T \tag{2.49}$$

$$\mathbf{P}^{xz}_k = \sum_{i=0}^{2n_a} \mathbf{W}_i [\mathbf{X}^a_{i,k} - \hat{\mathbf{x}}^-_k][\mathbf{Z}_{i,k} - \hat{\mathbf{z}}_k]^T \tag{2.50}$$

$$\mathbf{K}_k = \mathbf{P}^{xz}_k [\mathbf{P}^{zz}_k + \mathbf{R}_k]^{-1} \tag{2.51}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}^-_k + \mathbf{K}_k [\mathbf{z}_k - \hat{\mathbf{z}}_k] \tag{2.52}$$

$$\hat{\mathbf{P}}_k = \hat{\mathbf{P}}^-_k - \mathbf{K}_k [\mathbf{P}^{zz}_k + \mathbf{R}_k] \mathbf{K}^T_k \tag{2.53}$$

where $\hat{\mathbf{x}}_k$ and $\hat{\mathbf{P}}_k$ are the posterior state and covariance estimates.

Various modifications can be made to the standard algorithm, to account for specific details of a given application. For example, if the observation noise is introduced in a nonlinear fashion, or is correlated, then the augmented vector is expanded to include the observation terms [17]. The scaled UKF introduced in [20], and the square-root UKF, introduced in [21], are more efficient and stable implementations of the UKF. The UKF has been studied in a variety of problems, from visual tracking to parameter estimation and neural networks [22–25], where it has been proved that consistently performs better than the EKF while having the additional benefit of an easy implementation.

2.5 Target and sensor modelling

This section presents some basic concepts of target modelling and tracking. A comprehensive introduction to tracking can be found in the classic textbooks [1, 26]. The objective of target tracking is to form tracks of targets of interest by processing a set of observations collected from a measuring device. Throughout the thesis we are concentrating on tracking algorithms that use kinematic measurements obtained from a radar. These measurements lie in the spherical space and usually consist of the target's range and azimuth and elevation angles. Other measuring devices used commonly in the literature include electro-optical (EO) and infrared (IR) sensors and passive and active sonar systems. When multiple targets exist in the field of view (FOV) and/or erroneous clutter returns are collected from the radar and are classified as possible targets, data association techniques (see later sections) should be exploited before tracking.

2.5.1 Constant velocity model

For modelling the target dynamics we either use the white noise *constant velocity* (CV) or *coordinated turn* (CT) state-space models [1]. Consider a target which moves in one dimension with constant velocity. The difference equations describing its motion over time are:

$$x_k = x_{k-1} + \dot{x}_{k-1}T \tag{2.54}$$

$$\dot{x}_k = \dot{x}_{k-1} \tag{2.55}$$

where x_k is the target's position, $\dot{x}_k$ its velocity (notation $\dot{x}$ stands for the first derivative of x) and T the elapsed time between time step $k-1$ and k. By setting the state vector as $\mathbf{x}_k = [x_k \ \dot{x}_k]^T$, the state-space equivalent of (2.54-2.55) becomes:

$$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} \tag{2.56}$$

whose open form is:

$$\begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{k-1} \\ \dot{x}_{k-1} \end{bmatrix} \tag{2.57}$$

where the matrix $\mathbf{F}_k$ is the system matrix. For accounting for model uncertainty and for possible random unknown disturbances on the motion, we augment (2.56) with a *process noise* term

$\mathbf{u}_k = [u_{x,k} \;\; u_{\dot{x},k}]^T$, forming thus a *stochastic* system representation:

$$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{u}_{k-1} \tag{2.58}$$

where $\mathbf{u}_k$ is drawn from a white zero-mean Gaussian pdf $\mathbf{u}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$, which has a diagonal covariance $\mathbf{Q}_k$ of the form:

$$\mathbf{Q}_k = \begin{bmatrix} \sigma\{u_{x,k}\}^2 & 0 \\ 0 & \sigma\{u_{\dot{x},k}\}^2 \end{bmatrix} \tag{2.59}$$

where notation $\sigma\{\cdot\}$ stands for the standard deviation. The terms $\sigma\{u_{x,k}\}$ and $\sigma\{u_{\dot{x},k}\}$ are respectively the standard deviation of the process noise of the position and velocity. Although a CV target can be assumed to move independently along each of its moving directions, we see next that for the CT case that motion is highly correlated.

2.5.2 Coordinated turn model

In a similar fashion we derive the system equations for the CT model. We focus on a target that nominally executes a coordinated turn with a constant turn rate on the 2-D x-y plane. For the target it holds that:

$$\dot{x}_k = s_k \cos\psi_k \tag{2.60}$$

$$\dot{y}_k = s_k \sin\psi_k \tag{2.61}$$

where $\dot{x}_k$ and $\dot{y}_k$ are the x-y velocity components of it, $s_k = \sqrt{\dot{x}_k^2 + \dot{y}_k^2}$ its speed and ψ_k its heading angle. The turn rate is given by the first derivative of the heading angle, $\omega_k = \dot{\psi}_k$, and is considered to be fixed:

$$\omega_k = \omega_{k-1} \tag{2.62}$$

The equations that describe the x- and y-position for a constant ω_k are:

$$\begin{aligned} x_k &= x_{k-1} + \int_{t_{k-1}}^{t_k} \dot{x}_{\tau|t_{k-1}} d\tau \\ &= x_{k-1} + s_{k-1} \int_0^T \cos(\psi_{k-1} + \omega_{k-1}\tau) d\tau \\ &= x_{k-1} + \dot{x}_{k-1} \frac{\sin(\omega_{k-1}T)}{\omega_{k-1}} - \dot{y}_{k-1} \frac{1 - \cos(\omega_{k-1}T)}{\omega_{k-1}} \end{aligned} \tag{2.63}$$

$$\begin{aligned} y_k &= y_{k-1} + \int_{t_{k-1}}^{t_k} \dot{y}_{\tau|t_{k-1}} d\tau \\ &= y_{k-1} + s_{k-1} \int_0^T \sin(\psi_{k-1} + \omega_{k-1}\tau) d\tau \\ &= y_{k-1} + \dot{x}_{k-1} \frac{1 - \cos(\omega_{k-1}T)}{\omega_{k-1}} + \dot{y}_{k-1} \frac{\sin(\omega_{k-1}T)}{\omega_{k-1}} \end{aligned} \tag{2.64}$$

where (x_k, y_k) is the position of the target at the x-y plane and T is the elapsed time between $k-1$ and k. Similarly for the velocities from (2.60-2.61) we get:

$$\dot{x}_k = s_{k-1} \cos(\psi_{k-1} + \omega_{k-1}T) = \dot{x}_{k-1} \cos(\omega_{k-1}T) - \dot{y}_{k-1} \sin(\omega_{k-1}T) \tag{2.65}$$

$$\dot{y}_k = s_{k-1} \sin(\psi_{k-1} + \omega_{k-1}T) = \dot{x}_{k-1} \sin(\omega_{k-1}T) + \dot{y}_{k-1} \cos(\omega_{k-1}T) \tag{2.66}$$

Equations (2.62-2.66) describe a target performing a circular turn on a 2-D plane. In contrast to the CV model, the system is now *nonlinear* and its states are *correlated* across the moving dimensions. By defining the state vector $\mathbf{x}_k = [x_k \ \dot{x}_k \ y_k \ \dot{y}_k \ \omega_k]^T$ and introducing the process noise vector $\mathbf{u}_k = [u_{x,k} \ u_{\dot{x},k} \ u_{y,k} \ u_{\dot{y},k} \ u_{\omega,k}]^T$, we form the state transition equation:

$$\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}) + \mathbf{u}_{k-1} \tag{2.67}$$

where $\mathbf{f}_k(\cdot)$ is the nonlinear function:

$$\mathbf{f}(\mathbf{x}_k) = \begin{bmatrix} x_k + \dot{x}_k \frac{\sin(\omega_k T)}{\omega_k} - \dot{y}_k \frac{1-\cos(\omega_k T)}{\omega_k} \\ \dot{x}_k \cos(\omega_k T) - \dot{y}_k \sin(\omega_k T) \\ y_k + \dot{x}_k \frac{1-\cos(\omega_k T)}{\omega_k} + \dot{y}_k \frac{\sin(\omega_k T)}{\omega_k} \\ \dot{x}_k \sin(\omega_k T) + \dot{y}_k \cos(\omega_k T) \\ \omega_k \end{bmatrix} \tag{2.68}$$

and $\mathbf{u}_k$ is a white zero-mean Gaussian term with covariance $\mathbf{Q}_k$:

$$\mathbf{Q}_k = \sigma\{a_k\}^2 \cdot \begin{bmatrix} T^4/4 & T^3/2 & 0 & 0 & 0 \\ T^3/2 & T^2 & 0 & 0 & 0 \\ 0 & 0 & T^4/4 & T^3/2 & 0 \\ 0 & 0 & T^3/2 & T^2 & 0 \\ 0 & 0 & 0 & 0 & \sigma\{\omega_k\}^2 T^2 / \sigma\{a_k\}^2 \end{bmatrix} \tag{2.69}$$

where $\sigma\{a_k\}$ and $\sigma\{\omega_k\}$ are respectively the standard deviations of the random acceleration and the turning rate. From (2.68-2.59) we see that there is a strong correlation across the x and y components of the target motion.

2.5.3 Sensor model

In this section we relate the target states to the sensor measurements. The measuring device that we consider throughout the thesis is radar. From the vast literature on radars, we recommend textbooks [27, 28] for an in-depth analysis of their principles and applications, and [29] for an up-to-date thorough list of term and definitions.

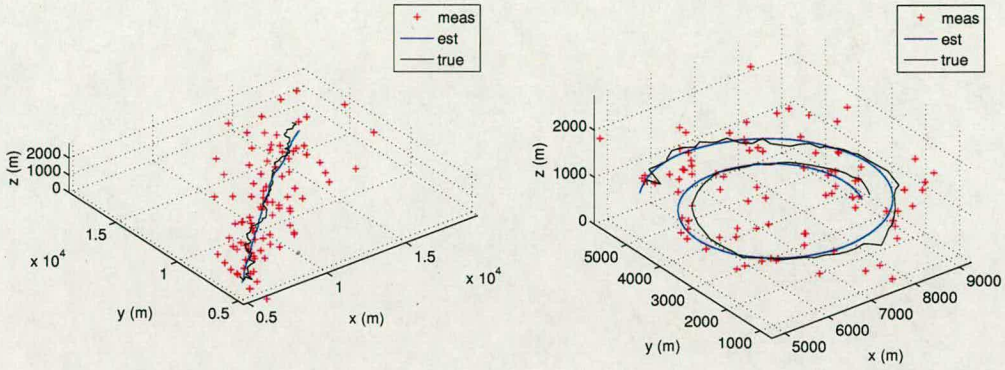
The radar is an active sensor which by emitting and collecting radio waves, measures kinematic quantities of scatterers of interest. The emitting frequency can be from as low as 3MHz (HF radio band) to higher than 300 GHz (millimetre radars). The choice of the frequency is affected primarily by the aperture size, range resolution, path loss, radar cross section and hardware constraints. In general, the higher the frequency the smaller the aperture and the better the range resolution, but atmospheric effects attenuate the signal and hardware problems (especially concerning the power consumption) arise. The dimensions of the scatterers within the tracking environment, relative to the wavelength size, affect also the radar cross section of the target and therefore the target detectability.

In the tracking literature the radar is usually modelled in a high level way, as a *black-box* device which feeds the tracking algorithms with noisy spherical target measurements. In the general case, the radar operates in the three dimensions generating measurements $\mathbf{z}_k$ as every time instant k:

$$\mathbf{z}_k = \begin{bmatrix} r_k \\ \theta_k \\ \epsilon_k \end{bmatrix} \tag{2.70}$$

where r_k is the range, θ_k the azimuth angle and ϵ_k the elevation angle of the target. Assuming that the target's states consist of the x-y-z components of its position and velocity $\mathbf{x}_k = [x_k \ \dot{x}_k \ y_k \ \dot{y}_k \ z_k \ \dot{z}_k]^T$, the equation that relates the measurements with the states is:

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k \tag{2.71}$$



(a) *Constant velocity model* **(b)** *Coordinated turn model*

Figure 2.3: *Tracking targets with different kinematics models using an EKF.*

where $\mathbf{h}_k(\cdot)$ is the following nonlinear function:

$$\mathbf{h}(\mathbf{x}_k) = \begin{bmatrix} \sqrt{x_k^2 + y_k^2 + z_k^2} \\ \arctan(y_k/x_k) \\ \arctan(\sqrt{x_k^2 + y_k^2}/z_k) \end{bmatrix} \tag{2.72}$$

and $\mathbf{v}_k = [v_{r,k} \;\; v_{\theta,k} \;\; v_{\epsilon,k}]^T$ is the measurement noise vector which models the range, azimuth and elevation angle inaccuracy of the radar. The measurement noise is white zero-mean Gaussian $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$, and has a diagonal covariance matrix $\mathbf{R}_k$:

$$\mathbf{R}_k = \begin{bmatrix} \sigma\{v_{r,k}\}^2 & 0 & 0 \\ 0 & \sigma\{v_{\theta,k}\}^2 & 0 \\ 0 & 0 & \sigma\{v_{\epsilon,k}\}^2 \end{bmatrix} \tag{2.73}$$

where $\sigma\{v_{r,k}\}, \sigma\{v_{\theta,k}\}$ and $\sigma\{v_{\epsilon,k}\}$ are the standard deviation of the measurement noise on the range, azimuth and elevation. Equations (2.71-2.72) can be altered accordingly to account for a different state representation and measurement dimensionality [1].

2.6 Tracking comparison

In this section we compare the performance of the EKF and the UKF for tracking a CV and a CT airborne target (see figure 2.3 for two track examples). We consider a static radar which

lies on the ground at the origin of the plane and we use relations (2.71-2.72) for modelling it. The radar feeds the algorithms at every $T = 1$s with noisy spherical target measurements $\mathbf{z}_k = [r_k \;\; \theta_k \;\; \epsilon_k]^T$. The standard deviations of the measurement noise are 60m for the range and 4^o for the azimuth and elevation angles. For both target models, the target travels in the 3-D x-y-z space with initial position (5,5,1) km and velocity (500,500,0.1) km/h.

For the CV case, the target states are $\mathbf{x}_k = [x_k \;\; \dot{x}_k \;\; y_k \;\; \dot{y}_k \;\; z_k \;\; \dot{z}_k]^T$. The system function is linear (2.58) and has a system matrix of the form:

$$\mathbf{F}_k = \begin{bmatrix} 1 & T & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & T & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.74}$$

The standard deviation of the process noise is 15m/s for the position and 15km/h for the velocity for all x-y-z directions.

For the EKF we use equations (2.26-2.32). The Jacobian of the linearized measurement function in (2.29) is:

$$\begin{aligned} \hat{\mathbf{H}}_k &= \left.\frac{\partial \mathbf{h}_k(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k=\mathbf{x}_k^-} \\ &= \left[\begin{array}{cccccc} \frac{\partial r_k}{\partial x_k} & \frac{\partial r_k}{\partial \dot{x}_k} & \frac{\partial r_k}{\partial y_k} & \frac{\partial r_k}{\partial \dot{y}_k} & \frac{\partial r_k}{\partial z_k} & \frac{\partial r_k}{\partial \dot{z}_k} \\ \frac{\partial \theta_k}{\partial x_k} & \frac{\partial \theta_k}{\partial \dot{x}_k} & \frac{\partial \theta_k}{\partial y_k} & \frac{\partial \theta_k}{\partial \dot{y}_k} & \frac{\partial \theta_k}{\partial z_k} & \frac{\partial \theta_k}{\partial \dot{z}_k} \\ \frac{\partial \epsilon_k}{\partial x_k} & \frac{\partial \epsilon_k}{\partial \dot{x}_k} & \frac{\partial \epsilon_k}{\partial y_k} & \frac{\partial \epsilon_k}{\partial \dot{y}_k} & \frac{\partial \epsilon_k}{\partial z_k} & \frac{\partial \epsilon_k}{\partial \dot{z}_k} \end{array}\right]_{\mathbf{x}_k=\mathbf{x}_k^-} \\ &= \left[\begin{array}{cccccc} \frac{x_k}{\sqrt{x_k^2+y_k^2+z_k^2}} & 0 & \frac{y_k}{\sqrt{x_k^2+y_k^2+z_k^2}} & 0 & \frac{z_k}{\sqrt{x_k^2+y_k^2+z_k^2}} & 0 \\ -\frac{y_k}{x^2+y^2} & 0 & \frac{x_k}{x^2+y^2} & 0 & 0 & 0 \\ -\frac{x_k z_k}{(x_k^2+y_k^2+z_k^2)\sqrt{x_k^2+y_k^2}} & 0 & -\frac{z_k y_k}{(x_k^2+y_k^2+z_k^2)\sqrt{x_k^2+y_k^2}} & 0 & \frac{\sqrt{x_k^2+y_k^2}}{x_k^2+y_k^2+z_k^2} & 0 \end{array}\right]_{\mathbf{x}_k=\mathbf{x}_k^-} \end{aligned} \tag{2.75}$$

For the UKF we use (2.43-2.53), setting $\kappa = -11$ from (2.35).

For the CT case, the state-space is augmented with the turn rate ω_k, which has an initial value of -5°/s. The standard deviation of the process noise is 1m/s^2 for the random accelerations and 0.1°/s for the turn rate. We assume that due to the executed turn the dynamics are correlated

just across the x-y plane. Based on that, we design both trackers with decoupled structures, within which a separate uncorrelated filter is used for estimating the vertical component of the motion. In general, decoupled filters require less computational power[1] and exhibit enhanced estimation performance [30].

Using this approach, both EKF and UKF consist of two *subfilters*: one accounting for the manoeuvre across the x-y plane with states $\mathbf{x}_k = [x_k \;\; \dot{x}_k \;\; y_k \;\; \dot{y}_k \;\; \omega_k]^T$, and another for the motion along the z-axis with states $\mathbf{x}_k = [z_k \;\; \dot{z}_k]^T$. The two Jacobians of the EKF's x-y subfilter have the form:

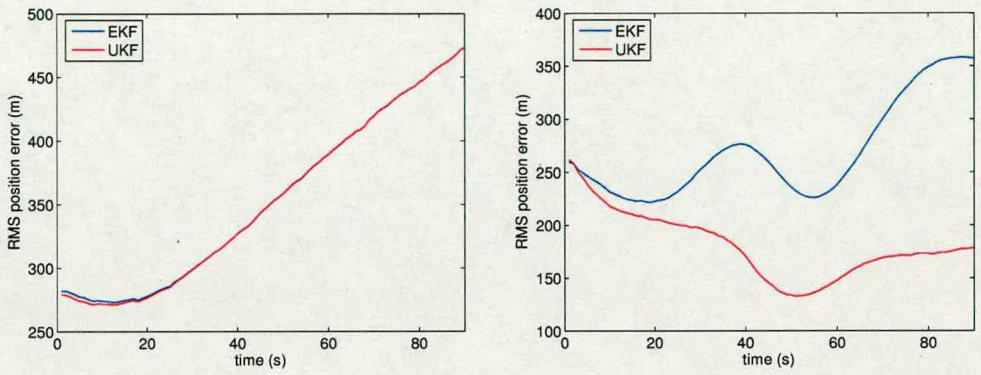
$$\begin{aligned}
\hat{\mathbf{F}}_{k-1} &= \left.\frac{\partial \mathbf{f}_{k-1}(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k=\mathbf{x}_{k-1}} \\
&= \left.\begin{bmatrix}
\frac{\partial x_k}{\partial x_k} & \frac{\partial x_k}{\partial \dot{x}_k} & \frac{\partial x_k}{\partial y_k} & \frac{\partial x_k}{\partial \dot{y}_k} & \frac{\partial x_k}{\partial \omega_k} \\
\frac{\partial \dot{x}_k}{\partial x_k} & \frac{\partial \dot{x}_k}{\partial \dot{x}_k} & \frac{\partial \dot{x}_k}{\partial y_k} & \frac{\partial \dot{x}_k}{\partial \dot{y}_k} & \frac{\partial \dot{x}_k}{\partial \omega_k} \\
\frac{\partial y_k}{\partial x_k} & \frac{\partial y_k}{\partial \dot{x}_k} & \frac{\partial y_k}{\partial y_k} & \frac{\partial y_k}{\partial \dot{y}_k} & \frac{\partial y_k}{\partial \omega_k} \\
\frac{\partial \dot{y}_k}{\partial x_k} & \frac{\partial \dot{y}_k}{\partial \dot{x}_k} & \frac{\partial \dot{y}_k}{\partial y_k} & \frac{\partial \dot{y}_k}{\partial \dot{y}_k} & \frac{\partial \dot{y}_k}{\partial \omega_k} \\
\frac{\partial \omega_k}{\partial x_k} & \frac{\partial \omega_k}{\partial \dot{x}_k} & \frac{\partial \omega_k}{\partial y_k} & \frac{\partial \omega_k}{\partial \dot{y}_k} & \frac{\partial \omega_k}{\partial \omega_k}
\end{bmatrix}\right|_{\mathbf{x}_k=\mathbf{x}_{k-1}} \\
&= \left.\begin{bmatrix}
1 & \frac{\sin(\omega_k T)}{\omega_k} & 0 & -\frac{1-\cos(\omega_k T)}{\omega_k} & \phi_1 \\
0 & \cos(\omega_k T) & 0 & -\sin(\omega_k T) & \phi_2 \\
0 & \frac{1-\cos(\omega_k T)}{\omega_k} & 1 & \frac{\sin(\omega_k T)}{\omega_k} & -\dot{x}_k T\sin(\omega_k T) - \dot{y}_k T\cos(\omega_k T) \\
0 & \sin(\omega_k T) & 0 & \cos(\omega_k T) & \dot{x}_k T\cos(\omega_k T) - \dot{y}_k T\sin(\omega_k T) \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}\right|_{\mathbf{x}_k=\mathbf{x}_{k-1}}
\end{aligned} \tag{2.76}$$

where:

$$\begin{aligned}
\phi_1 &= \left.\frac{\partial x_k}{\partial \omega_k}\right|_{\mathbf{x}_k=\mathbf{x}_{k-1}} \\
&= \left.\frac{-\dot{y}_k(\omega_k T\sin(\omega_k T) + \cos(\omega_k T) - 1) + \dot{x}_k(\omega_k T\cos(\omega_k T) - \sin(\omega T))}{\omega_k^2}\right|_{\mathbf{x}_k=\mathbf{x}_{k-1}}
\end{aligned} \tag{2.77}$$

$$\begin{aligned}
\phi_2 &= \left.\frac{\partial \dot{x}_k}{\partial \omega_k}\right|_{\mathbf{x}_k=\mathbf{x}_{k-1}} \\
&= \left.\frac{\dot{x}_k(\omega_k T\sin(\omega_k T) + \cos(\omega_k T) - 1) + \dot{y}_k(\omega_k T\cos(\omega_k T) - \sin(\omega T))}{\omega_k^2}\right|_{\mathbf{x}_k=\mathbf{x}_{k-1}}
\end{aligned} \tag{2.78}$$

[1] As a rule of thumb, the computational complexity increases almost proportionally with n_x^3 [1], where n_x is the dimensionality of the states. Therefore for our example, a decoupled 5+2 state filter compared to a fully-coupled 7 state filter, results in a complexity reduction by about a factor of $(5^3 + 2^3)/7^3 \approx 0.38$.



(a) *Constant velocity model* **(b)** *Coordinated turn model*

Figure 2.4: *Comparison of the RMS position error of the EKF and the UKF.*

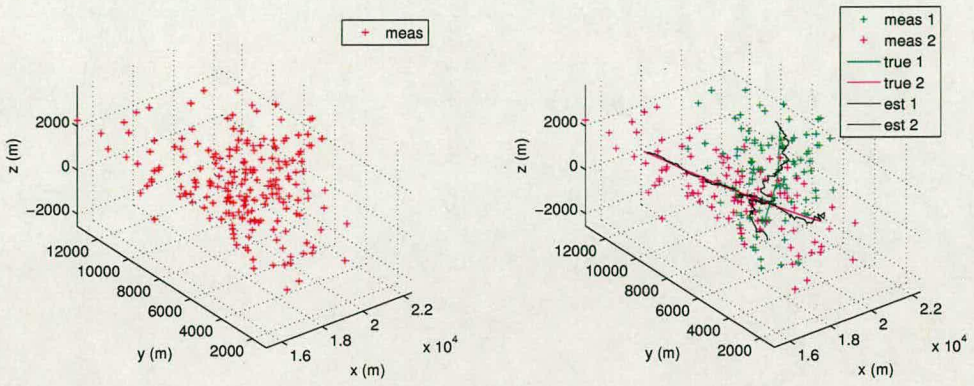
and:

$$\hat{\mathbf{H}}_k = \left.\frac{\partial \mathbf{h}_k(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k=\mathbf{x}_k^-} = \left[\begin{array}{ccccc} \frac{\partial r_k}{\partial x_k} & \frac{\partial r_k}{\partial \dot{x}_k} & \frac{\partial r_k}{\partial y_k} & \frac{\partial r_k}{\partial \dot{y}_k} & \frac{\partial r_k}{\partial \omega_k} \\ \frac{\partial \theta_k}{\partial x_k} & \frac{\partial \theta_k}{\partial \dot{x}_k} & \frac{\partial \theta_k}{\partial y_k} & \frac{\partial \theta_k}{\partial \dot{y}_k} & \frac{\partial \theta_k}{\partial \omega_k} \end{array}\right]_{\mathbf{x}_k=\mathbf{x}_k^-}$$

$$= \left[\begin{array}{ccccc} \frac{x_k}{\sqrt{x_k^2+y_k^2}} & 0 & \frac{y_k}{\sqrt{x_k^2+y_k^2}} & 0 & 0 \\ -\frac{y_k}{x^2+y^2} & 0 & \frac{x_k}{x^2+y^2} & 0 & 0 \end{array}\right]_{\mathbf{x}_k=\mathbf{x}_k^-} \tag{2.79}$$

The uncorrelated z-axis filter is designed as a 2-D version of the CV EKF. The decoupled CT UKF is formed in a similar fashion to the decoupled EKF (again $\kappa = -11$). Once more, equations (2.26-2.32) and (2.43-2.53) describe respectively the EKF and the UKF algorithms.

The simulation results were obtained after 5000 Monte Carlo (MC) runs for each model. The tracking time for each simulation was 90s. Figure 2.4 present the room mean square (RMS) position error of the algorithms for both cases. Although the algorithms' performance is almost identical when tracking the CV target, for the CT case the superiority of the UKF is evident. The second-order linearization accuracy of the UKF helped the filter to cope more efficiently with the more difficult dynamics of the nonlinear CT model, and resulted on average in a 34% smaller position error than that of the EKF.



(a) *Raw measurements* **(b)** *Associated measurements and tracks*

Figure 2.5: *Data association and tracking for two crossing targets.*

2.7 Data association

The data association problem arises when multiple scatterers exist in the field of view of the measuring sensor. The resulting multiple measurements can either originate from targets of interest or from background clutter. For this class of problems a data association function is employed for assigning the measurements of interest to new or existing target tracks (see figure 2.5 for an association example). In a realistic tracking system, complex association structures are used which include subfunctions for gating, clutter rejection and track initiation, confirmation and deletion.

There are two major approaches on the data association problem: the single scan data association (SSDA) and the multiple scan data association (MSDA). The SSDA methods use data just from the current scan and deduce "hard" decisions on the current association. The MSDA techniques exploit data from consecutive scans, normally using simultaneously more than one association hypothesis and derive "soft" or probabilistic assignment decisions at every time step. The three most well studied algorithms in the literature are the global nearest neighbour (GNN), the joint probabilistic data association (JPDA) and the multiple hypothesis tracking (MHT), the first two based on SSDA and the third on MSDA.

The conventional GNN [31] seeks the single most likely association relying upon the statistical distance between the measurements and the tracks. The JPDA [32, 33] selects the association hypothesis that maximises the joint association probability. The MHT [34] processes multiple

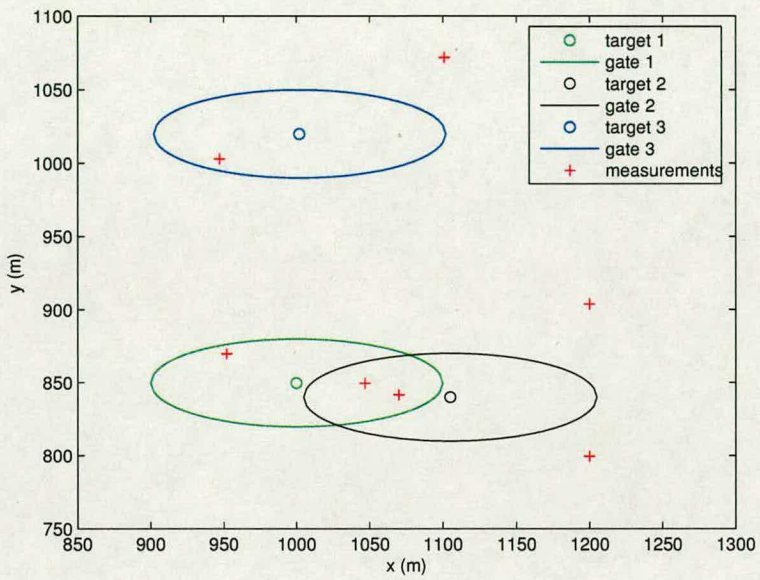


Figure 2.6: *Gates are formed about the predicted targets' position to eliminate unlikely measurement-to-target associations. A data association algorithm should be used between targets 1 and 2 and the measurements that fall within their gates, for resolving the assignment ambiguity.*

association hypotheses which either confirms, merges or deletes, propagating the surviving ones to the next scan. In what follows we focus on SSDA methods since they provide at every scan a definite association decision and they are significantly computationally lighter.

For an efficient data association implementation, before the association algorithm a gating technique [1, 35, 36] should be exploited. Gates are formed about the predicted target positions and just the measurements that fall within each gate are assignment candidates. The association algorithm therefore is used only when ambiguous conflict situations arise (see figure 2.6). The shape and the volume of the gates depends on the application, but usually they are ellipsoidal and have volume proportional to the combined uncertainty of the measurement and the position of the target. The bigger the gate, the more likely the actual target measurement will be included in the gate, but since it's more probable that on average more measurements will be considered, the computational load increases.

2.8 Global nearest neighbour

The GNN [31] is simple to implement and has light computational requirements, features that make it particularly appropriate for non-demanding low-complexity systems. Its basic principle is to identify the most likely association hypothesis, by assigning costs to the possible measurement-to-track pairings and then by solving the resulting *assignment problem.*

The classic implementation of the GNN algorithm uses the statistical distance of the measurement to the predicted target position, as the cost of the association problem. Assume that at time k (after gating) we have $n_{t,k}$ unassociated targets and $n_{m,k}$ measurements $\{\mathbf{z}_k^j\}_{j=1}^{n_{m,k}}$. A conventional approach is to form an $n_{t,k} \times n_{m,k}$ assignment matrix $\mathbf{A}_k$ with elements $a_{ij,k}$ given as the inverse of the costs $c_{ij,k}$:

$$a_{ij,k} = (c_{ij,k})^{-1} = (\tilde{\mathbf{z}}_k^{ij\,T} \mathbf{S}_k^{ij\,-1} \tilde{\mathbf{z}}_k^{ij} + \ln|\mathbf{S}_k^{ij}|)^{-1} \qquad (2.80)$$
$$\forall i = 1, \ldots, n_{t,k}, \;\; \forall j = 1, \ldots, n_{m,k}$$

where $\tilde{\mathbf{z}}_k^{ij}$ and $\mathbf{S}_k^{ij}$ are respectively the measurement residual vector and covariance resulted from:

$$\tilde{\mathbf{z}}_k^{ij} = \mathbf{z}_k^j - \mathbf{h}_k(\mathbf{x}_k^{i-}) \qquad (2.81)$$

in which $\mathbf{h}_k(\cdot)$ is the measurement function as defined in (2.72) and $\mathbf{x}_k^{i-}$ is the prior estimate of the position of the $i\text{-}th$ target. The cost function in (2.80) includes the term $\ln|\mathbf{S}_k^{ij}|$ for penalising tracks with greater estimation uncertainty.

A typical solution to the assignment problem is to find the pairings that minimise the overall cost, while using the maximum number of assignments. Although in simple cases this can be done by exhaustive enumeration, more efficient techniques [1] are available for when the size of $\mathbf{A}_k$ increases. For our GNN implementation we rely on the auction algorithm (AA) [37, 38] which currently seems to be the most efficient approach to the assignment problem.

For assigning the tracks to the measurements, in analogy with real-life auctions, the AA uses a competitive bidding approach until the total assignment "gain" is maximised. The AA introduces a *price* $p_{i,k}$ for each track i, and a measurement-to-track *profit margin*:

$$\pi_{j,k} = \max_{i=1,\ldots,n_{t,k}} \{a_{ij,k} - p_{i,k}\} \qquad (2.82)$$

where $a_{ij,k}$ is given in (2.80). By relaxing the complementary slackness condition (from linear programming theory [39]) it seeks the near-optimal solution under the condition:

$$\pi_{j,k} - (a_{i_j j,k} - p_{i_j,k}) \leq \epsilon \tag{2.83}$$

for every track i_j. The user-defined parameter ϵ determines the "optimality" of the algorithm, where a smaller value results in a solution closer to the optimal (at the expense of more iterations). The AA first initialises all measurements as unassigned and sets the track prices to zero. Within the iterative phase that follows, the algorithm selects an unassigned measurement j, tentatively assigns it with track i_j for which:

$$a_{i_j j,k} - p_{i_j,k} = \pi_{j,k} \tag{2.84}$$

and adjusts the track's price:

$$p_{i_j,k} = p_{i_j,k} + y_{j,k} + \epsilon \tag{2.85}$$

where $y_{j,k}$ is the difference between the two best assignment values $a_{ij,k}$ for measurement j. The AA re-iterates, until all the possible assignments are computed.

2.9 Joint probabilistic data association

The JPDA [33] is another SSDA method which at every scan forms all the possible association hypotheses and selects the one that maximises the joint association probability. The JPDA generalised the probabilistic data association algorithm [32], which was essentially a clutter rejection method, so as to explicitly take under consideration the presence of multiple targets.

Consider again that at time k after gating we have $n_{t,k}$ unassociated targets and $n_{m,k}$ measurements $\{\mathbf{z}_k^j\}_{j=1}^{n_{m,k}}$. Following the analysis of [33], central to the JPDA is the computation of the conditional probabilities of the joint events $\chi_k = \bigcap_{j=1}^{n_{m,k}} \chi_{i_j j,k}$, where $\chi_{i_j j,k}$ is the event that the $j\text{-}th$ measurement originated from the $i_j\text{-}th$ target. The feasible events are the joint events in which just one measurement originates from each target. For notational purposes we define

the following binary functions:

$$\iota_{j,k}(\chi_k) = \begin{cases} 1, & \text{if } i_j > 0 \\ 0, & \text{if } i_j = 0 \end{cases} \tag{2.86}$$

$$\delta_{i,k}(\chi_k) = \begin{cases} 1, & \text{if } i_j = i \text{ for some j} \\ 0, & \text{if } i_j \neq i \text{ for all j} \end{cases} \tag{2.87}$$

$$\omega_{ij,k}(\chi_k) = \begin{cases} 1, & \text{if } \chi_{ij} \text{ occurs} \\ 0, & \text{otherwise} \end{cases} \tag{2.88}$$

where $\iota_{j,k}(\chi_k)$ indicates whether the $j\text{-}th$ measurement is associated with any target in event χ_k, $\delta_{i,k}(\chi_k)$ whether any measurement is associated with the $i\text{-}th$ target in event χ_k and $\omega_{ij,k}(\chi_k)$ whether an association between the $j\text{-}th$ measurement and the $i\text{-}th$ target is assumed in event χ_k. For i and j it holds that $i = 1, \ldots, n_{t,k}$, $j = 1, \ldots, n_{m,k}$.

The probability $p_{ij,k}$ that the $j\text{-}th$ measurement belongs to the $i\text{-}th$ target is obtained by summing over all feasible events χ_k for which the assignment occurs:

$$p_{ij,k} = \sum_{\chi_k} P\{\chi_k|\mathbf{Z}_k\}\omega_{ij}(\chi_k) \tag{2.89}$$

$$p_{i0,k} = 1 - \sum_{j=1}^{n_{m,k}} p_{ij,k} \tag{2.90}$$

where $P\{\chi_k|\mathbf{Z}_k\}$ is the probability of event χ_k[2] assuming the set of all measurements $\mathbf{Z}_k$, given by:

$$P\{\chi_k|\mathbf{Z}_k\} = \frac{C^\phi}{c_k} \prod_{j:\iota_{j,k}=1} \frac{\exp\left(-\frac{1}{2}\tilde{\mathbf{z}}_k^{ij\,jT}\mathbf{S}_k^{i_j}\tilde{\mathbf{z}}_k^{ij\,j}\right)}{(2\pi)^{n_{m,k}/2}\sqrt{|\mathbf{S}_k^{i_j}|}} \prod_{i:\delta_{i,k}=1} P_D^i \prod_{i:\delta_{i,k}=0} (1 - P_D^i) \tag{2.91}$$

in which C and ϕ are respectively the density and the number of false alarms (Poisson distributed), P_D is the probability of detection and $c_k = p(\mathbf{Z}_k|\mathbf{Z}_{k-1})$ is the normalisation factor from Bayes equation (2.11). The measurement residual $\tilde{\mathbf{z}}_k^{ij\,j}$ (with $\mathbf{S}_k^{i_j}$ being its associated covariance matrix) is given by:

$$\tilde{\mathbf{z}}_k^{ij\,j} = \mathbf{z}_k^j - \mathbf{H}_k\mathbf{x}_k^{i_j-} \tag{2.92}$$

[2]The first product in (2.91) refers to the association likelihood and the two others refer to the prior detection probabilities.

where $\mathbf{h}_k$ is the measurement function and $\mathbf{x}_k^{i_j-}$ the a-priori predicted position of the i_j-th target (the term $\mathbf{H}_k\mathbf{x}_k^{i_j-}$ becomes equivalently $\mathbf{h}_k(\mathbf{x}_k^{i_j-})$ in the nonlinear case).

The assignment probabilities $p_{ij,k}$ are used to modify the state update equation of the KF[3], which now uses for every target i a weighted sum of the residuals from all measurements:

$$\mathbf{x}_k^i = \mathbf{x}_k^{i-} + \mathbf{K}_k^i \tilde{\mathbf{z}}_k^i \tag{2.93}$$

where:

$$\tilde{\mathbf{z}}_k^i = \sum_{j=1}^{n_{m,k}} p_{ij,k} \tilde{\mathbf{z}}_k^{ij} \tag{2.94}$$

in which $\tilde{\mathbf{z}}_k^{ij}$ is given in (2.92). The Kalman gain is computed using (2.21) in the normal KF fashion. The covariance update equation is also modified and becomes:

$$\mathbf{P}_k^i = \mathbf{P}_k^{i\,\prime} + \boldsymbol{\pi}_k^i \tag{2.95}$$

where $\mathbf{P}_k^{i\,\prime}$ is the covariance that would be computed if a single measurement were present, and $\boldsymbol{\pi}_k^i$ is an increment accounting for the association uncertainty:

$$\mathbf{P}_k^{i\,\prime} = p_{i0,k}\mathbf{P}_k^{i-} + (1 - p_{i0,k})\big[[\mathbf{I} - \mathbf{K}_k^i\mathbf{H}_k]\mathbf{P}_k^{i-}\big] \tag{2.96}$$

$$\boldsymbol{\pi}_k^i = \mathbf{K}_k^i\big[\sum_{j=1}^{n_{m,k}} p_{ij}\tilde{\mathbf{z}}_k^{ij}\tilde{\mathbf{z}}_k^{ijT} - \tilde{\mathbf{z}}_k^i\tilde{\mathbf{z}}_k^{iT}\big]\mathbf{K}_k^{i\,T} \tag{2.97}$$

Track initiation and deletion functions [43] can be added to the JPDA method, to account for a varying number of targets. Overall, the JPDA method is expected to improve the performance over the simple GNN, since it makes use of all available hypotheses to reach an association decision.

2.10 Data association comparison

In this section we present a brief comparison of the performance of the GNN and the JPDA, when dealing with three couples of closely separated CV airborne targets. For our study we do not use any gating scheme for not inflicting gate-related biases to the association. For tracking

[3]Here we follow the analysis from the original JPDA which was based on the Kalman filtering framework. Currently several implementations have been proposed in which different tracking filters are used [40–42].

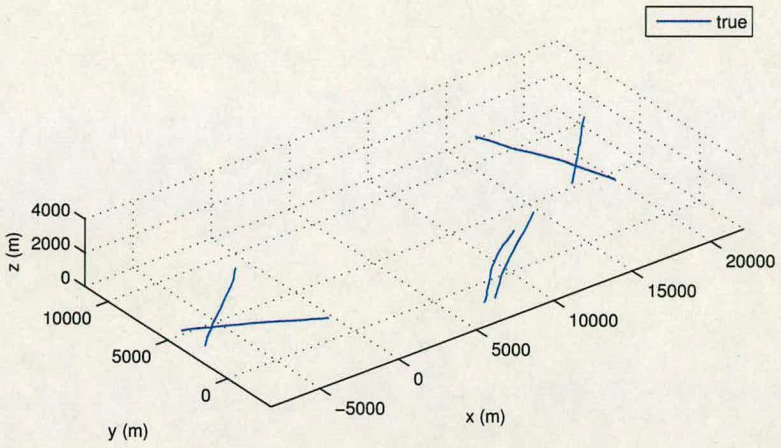


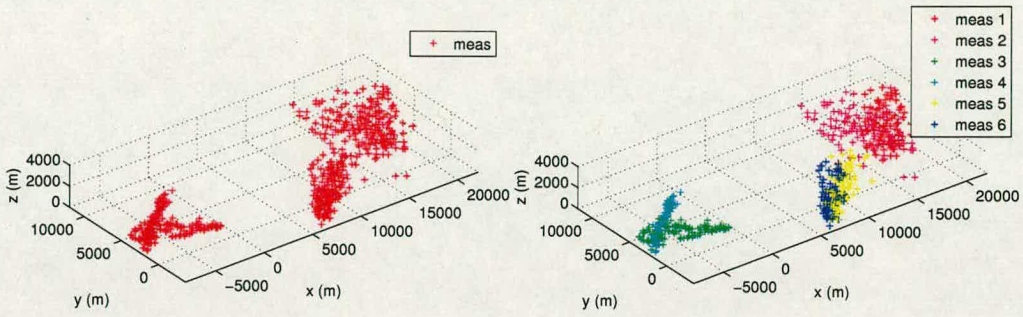
Figure 2.7: *An example of multiple target tracks.*

we employ EKFs. The radar lies on the ground at point (0,0) and every 1s provides the tracker with noisy spherical measurements of the position of the targets. The standard deviation of the measurement noise is 100m for range and 4° for both azimuth and elevation angles. We set the probability of false alarms to zero and the probability of detection to one.

The standard deviation of the process noise for all targets is 5m for their position and 5km/h for their velocity across the x-y plane, and respectively 1m and 1km/h along their z-axis. Each simulation lasts 100s. Figure 2.7 shows a set of randomly perturbed target tracks from our scenario. We particularly chose these track patterns so as to force the algorithms to solve difficult association problems. Figure 2.8 shows the raw and the associated measurements for

	Average disassociations (per run)	Overall track losses (5000 runs)	Percentage of track losses (per run)
JPDA	4.09	32	0.64%
GNN	6.06	177	3.54%

Table 2.1: *Comparison of the association performance of the GNN and the JPDA.*



(a) *Raw measurements* **(b)** *Associated measurements*

Figure 2.8: *Raw and associated measurements of multiple targets (tracks at figure 2.7).*

the same tracking example.

The simulation results presented in table 2.1 were obtained after 5000 MC runs. The metrics for judging GNN and JPDA association performance were the disassociations, i.e. when the associations were wrong, and the track losses, i.e. when due to a series of disassociations the tracker swapped the tracks between the targets. From table 2.1 we see that as expected the JPDA clearly exhibits a superior performance, having on average 32% less disassociations and 82% fewer track losses.

2.11 Chapter summary

In this chapter we presented a short introduction to target tracking and data association. First, after briefly discussing the basic principles of Bayesian estimation, we focused on the Kalman filter and its extended and unscented nonlinear variations. We described basic models for target and sensor modelling and showed experimentally the superiority of the UKF for difficult nonlinear tracking problems. Next we introduced the problem of data association. We concentrated on the popular global nearest neighbour and joint probabilistic data association algorithms and we demonstrated through a short comparative study that we normally expect the (more computationally complex) JPDA to attain a better association performance.

Chapter 3
Tracking with particle filters

The third chapter focuses on particle filtering. It introduces its generic principles and presents popular filter variations. After a brief comparison of the basic particle filter and the extended Kalman filter in a difficult chaotic problem, a particle filter for tracking a manoeuvring target is introduced and studied experimentally. The remaining of the chapter is dedicated in a new efficient multitarget tracking particle filter that uses a measurement-coupled prediction and association phase in which the number of the particles varies according to the association difficulty. Simulation results demonstrate the performance gains of the proposed approach.

3.1 Introduction to particle filtering

The particle filters (PF) are powerful numerical methods which address the nonlinear/non-Gaussian Bayesian estimation problem. Based on the concepts of Monte Carlo integration and importance sampling, they use a set of weighted samples or *particles* of the state density, which when propagated over time provide a sequential discrete approximation of the posterior state distribution. Although there was a certain body of work on sequential Monte Carlo (SMC) methods during the 70's and 80's (see [44] and the references therein), it was only until the 90's where due to the increased computing capabilities the field started to receive systematic attention. Arguably reference [45] in 1993 laid the foundation of modern particle filtering by incorporating a resampling step to the algorithm. Textbook [46], report [44] and papers [47–49], from which we draw in this chapter, offer a comprehensive analysis and literature review on SMC methods and particle filtering.

We start our analysis by introducing Monte Carlo integration. Suppose we want to compute the expectation of function $\mathbf{f}(\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^{n_x}$ is a continuous random variable with pdf $\pi(\mathbf{x})$; we want thus to evaluate:

$$I = \int \mathbf{f}(\mathbf{x})\pi(\mathbf{x})d\mathbf{x} \tag{3.1}$$

The MC estimate $\hat{I}_{N_s}$ of that interval is computed as the mean of N_s samples $\mathbf{x}^i$ which are

drawn from the density $\pi(\mathbf{x})$ and are propagated through function $\mathbf{f}(\mathbf{x})$:

$$\hat{I}_{N_s} = \frac{1}{N_s} \sum_{i=1}^{N_s} \mathbf{f}(\mathbf{x}^i) \tag{3.2}$$

Under the strong law of large number the MC estimate almost surely converges to the real integral, when the number of the samples tends to infinity, i.e. $P\{\hat{I}_{N_s} \rightarrow I\}|_{N_s \rightarrow \infty} = 1$. Moreover, the rate of convergence of the estimate is independent of the dimensionality n_x of the integrand. The MC solution of relation (3.1), is of great importance in the Bayesian context, since for example by setting $\mathbf{f}(\mathbf{x}) = \mathbf{x}$ and $\pi(\mathbf{x})$ as the posterior state density, theoretically we could directly compute the MMSE state estimate (2.15). In practise we cannot directly draw samples from the posterior density, usually being multivariate and nonstandard, and therefore we rely on *importance sampling* for obtaining samples' approximations.

Importance sampling is a technique for sampling indirectly from a given pdf $\pi(\mathbf{x})$. Under this method we obtain samples from another similar pdf $q(\mathbf{x})$ (the importance density) and we correct them by weighting them appropriately. The rather weak similarity condition for $\pi(\mathbf{x})$ and $q(\mathbf{x})$ is both to be positive for all values of $\mathbf{x}$. Using the importance density, we can now rewrite (3.1) as:

$$I = \int \left(\mathbf{f}(\mathbf{x}) \frac{\pi(\mathbf{x})}{q(\mathbf{x})} \right) q(\mathbf{x}) d\mathbf{x} \tag{3.3}$$

The MC estimate of this integral is the weighted mean of N_s samples $\{\mathbf{x}^i\}_{i=1}^{N_s}$ drawn from the importance density $q(\mathbf{x})$:

$$\hat{I}_{N_s} = \frac{1}{N_s} \sum_{i=1}^{N_s} \mathbf{f}(\mathbf{x}^i) \tilde{w}^i \tag{3.4}$$

where the *importance* weights are given by:

$$\tilde{w}^i = \frac{\pi(\mathbf{x}^i)}{q(\mathbf{x}^i)} \tag{3.5}$$

In the Bayesian framework, the normalisation factor of $\pi(\mathbf{x})$ (the state posterior) usually cannot be expressed in closed form. To circumvent, this problem it can be shown that the MC estimate becomes:

$$\hat{I}_{N_s} = \sum_{i=1}^{N_s} \mathbf{f}(\mathbf{x}^i) w^i \tag{3.6}$$

if the importance weights are normalised to sum to one, i.e. $\sum_{i=1}^{N_s} w^i = 1$:

$$w^i = \frac{\tilde{w}^i}{\sum_{j=1}^{N_s} \tilde{w}^j} \tag{3.7}$$

Based on (3.6-3.7) we derive next a sequential Bayesian solution. Following section 2.1 we define $\mathbf{X}_k$ and $\mathbf{Z}_k$ as the sets of respectively all states and measurements up to time k and $p(\mathbf{X}_k|\mathbf{Z}_k)$ as the posterior state pdf (to be estimated) assuming the measurements. The principle is to draw at every k from a known importance density $q(\mathbf{X}_k|\mathbf{Z}_k)$ a set of N_s particles $\{\mathbf{x}_k^i\}_{i=1}^{N_s}$ and to assign them appropriate importance weights $\{w_k^i\}_{i=1}^{N_s}$ to characterise the posterior state distribution. Following (3.5) the weights are proportional to the posterior over the importance density:

$$\tilde{w}_k^i \propto \frac{p(\mathbf{X}_k^i|\mathbf{Z}_k)}{q(\mathbf{X}_k^i|\mathbf{Z}_k)} = \frac{p(\mathbf{X}_k^i|\mathbf{Z}_k)}{q(\mathbf{x}_k^i|\mathbf{X}_{k-1}^i, \mathbf{Z}_k) q(\mathbf{X}_{k-1}^i|\mathbf{Z}_{k-1})} \tag{3.8}$$

for a $q(\mathbf{X}_k^i|\mathbf{Z}_k)$ chosen to factorise as above. We apply Bayes' rule to the numerator and obtain:

$$\begin{aligned} p(\mathbf{X}_k^i|\mathbf{Z}_k) &= \frac{p(\mathbf{z}_k|\mathbf{X}_k^i, \mathbf{Z}_{k-1}) p(\mathbf{X}_k^i|\mathbf{Z}_{k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{k-1})} \\ &= \frac{p(\mathbf{z}_k|\mathbf{X}_k^i, \mathbf{Z}_{k-1}) p(\mathbf{x}_k^i|\mathbf{X}_{k-1}^i, \mathbf{Z}_{k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{k-1})} p(\mathbf{X}_{k-1}^i|\mathbf{Z}_{k-1}) \\ &= \frac{p(\mathbf{z}_k|\mathbf{x}_k^i) p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{p(\mathbf{z}_k|\mathbf{Z}_{k-1})} p(\mathbf{X}_{k-1}^i|\mathbf{Z}_{k-1}) \\ &\propto p(\mathbf{z}_k|\mathbf{x}_k^i) p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i) p(\mathbf{X}_{k-1}^i|\mathbf{Z}_{k-1}) \end{aligned} \tag{3.9}$$

By using (3.9) and exploiting the Markovian property, we can rewrite (3.8) as:

$$\tilde{w}_k^i \propto \frac{p(\mathbf{z}_k|\mathbf{x}_k^i) p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i) p(\mathbf{X}_{k-1}^i|\mathbf{Z}_{k-1})}{q(\mathbf{x}_k^i|\mathbf{X}_{k-1}^i, \mathbf{Z}_k) q(\mathbf{X}_{k-1}^i|\mathbf{Z}_{k-1})} = w_{k-1}^i \frac{p(\mathbf{z}_k|\mathbf{x}_k^i) p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{z}_k)} \tag{3.10}$$

Therefore the posterior density at every k can be approximated as the following discrete sequence:

$$p(\mathbf{x}_k|\mathbf{Z}_k) \approx \sum_{i=1}^{N_s} w_k^i \delta(\mathbf{x}_k - \mathbf{x}_k^i) \tag{3.11}$$

where $\delta(\cdot)$ is the Dirac function. The state estimate finally becomes:

$$\hat{\mathbf{x}}_k = \sum_{i=1}^{N_s} \mathbf{x}_k^i w_k^i \tag{3.12}$$

The weights from (3.11-3.12) are obtained from (3.10) and are normalised as in (3.7).

3.2 Filter degeneracy and resampling

Based on the analysis above, the sequential importance sampling (SIS) particle filter is given as a pseudocode below. For the specific filter it has been proved [50] that the variance of its importance weights can only increase over time. This means that eventually all but one weights become negligible, i.e. the SIS effectively uses just one particle (see figure 3.1), and thus the filter diverges and collapses. This so-called *degeneracy* phenomenon can be controlled by employing a resampling scheme at the end of every estimation cycle as proposed in [45]. The key idea is to eliminate small weight particles by replacing them with others which have larger weights, before propagating them to the future.

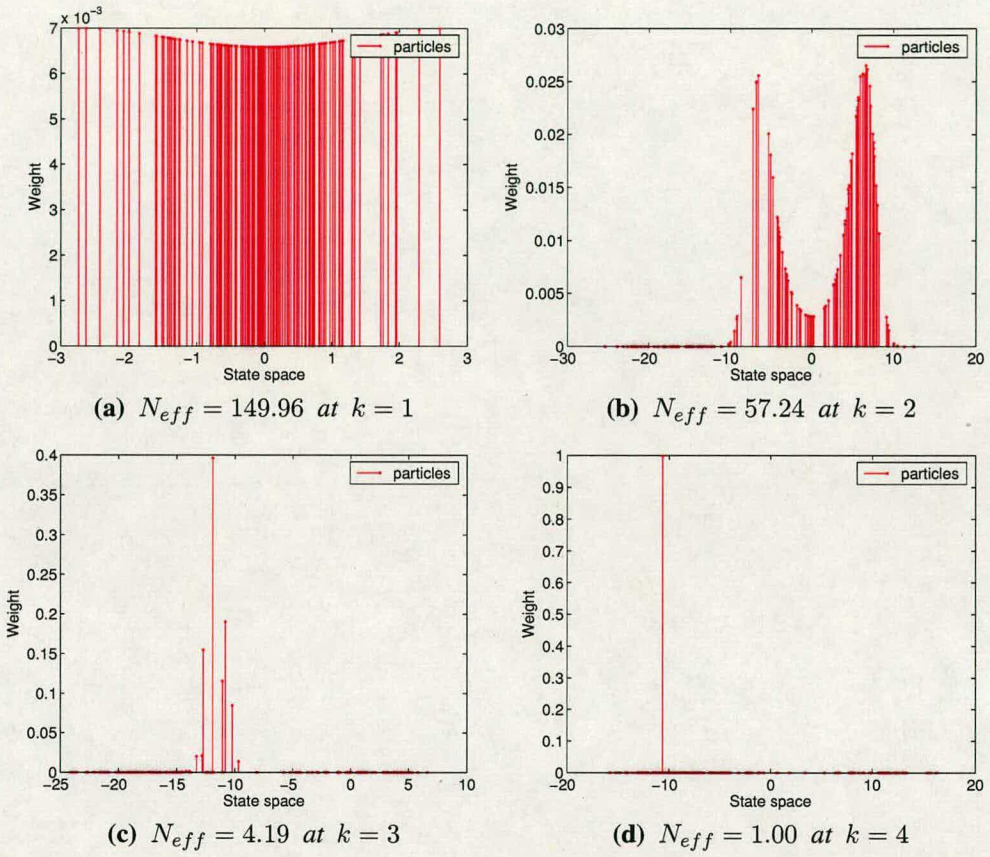
By far the most popular resampling algorithm in the particle filtering literature is the systematic resampling (SR) [51] due to its simplicity, small computational complexity of $O(N_s)$ and effectiveness in minimising the MC variation [52, 53]. The SR algorithm generates a new set of particles $\{\mathbf{x}_k^{i\star}\}_{i=1}^{N_s}$ by resampling with replacement from the discrete approximation of the posterior state pdf (3.11) so as $P\{\mathbf{x}_k^{i\star} = \mathbf{x}_k^j\} = w_k^j$. A pseudocode of the technique is given in algorithm 2 and a graphical illustration is presented in figure 3.2. As figure 3.3 illustrates, since the resulting particles obtain equal weights $w_k^i = 1/N_s, \forall i$, after resampling the posterior density is approximated just by the density of the particles (number of particles at a given point at the state-space).

When a PF uses resampling, the particles propagated to the next step $k+1$ originate just from the states of the most heavily weighted particles at k. This limits the diversity of the particles

Algorithm 1 Sequential Importance Sampling

$[\{\mathbf{x}_k^i, w_k^i\}_{i=1}^{N_s}] = \text{SIS}[\{\mathbf{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_s}, \mathbf{z}_k]$

1: **for** $i = 1 : N_s$ **do**
2: Draw particles: $\mathbf{x}_k^i \sim q(\mathbf{x}_k^i | \mathbf{x}_{k-1}^i, \mathbf{z}_k)$
3: Compute weights: $\tilde{w}_k^i = w_{k-1}^i \dfrac{p(\mathbf{z}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{z}_k)}$
4: **end for**
5: Compute total weight: $s_{w,k} = \sum_{i=1}^{N} \tilde{w}_k^i$
6: **for** $i = 1 : N_s$ **do**
7: Normalise weights: $w_k^i = \tilde{w}_k^i / s_{w,k}$
8: **end for**



(a) $N_{eff} = 149.96$ *at* $k = 1$ **(b)** $N_{eff} = 57.24$ *at* $k = 2$

(c) $N_{eff} = 4.19$ *at* $k = 3$ **(d)** $N_{eff} = 1.00$ *at* $k = 4$

Figure 3.1: *The degeneracy phenomenon when no resampling is used. After only 4 steps for the system described in (3.19-3.20), the SIS degenerates and uses just one particle. The effective number of particles is computed from (3.13).*

and it might consequently lead to sample *impoverishment* and performance degradation. The effects of the phenomenon are more intense particularly when the process noise is relatively small. For dealing with the problem, one can resample *only* when there is an indication that degeneracy is severe. The effective number of particles N_{eff} can be used for this purpose, which can be approximated [54] as:

$$N_{eff} \approx \frac{1}{\sum_{i=1}^{N_s} (w_k^i)^2} \tag{3.13}$$

Therefore, a generic PF can be structured from an SIS filter which calls the SR algorithm for resampling when the effective number of particles falls below a certain user-defined threshold. In the next sections we describe some popular variations of this basic PF.

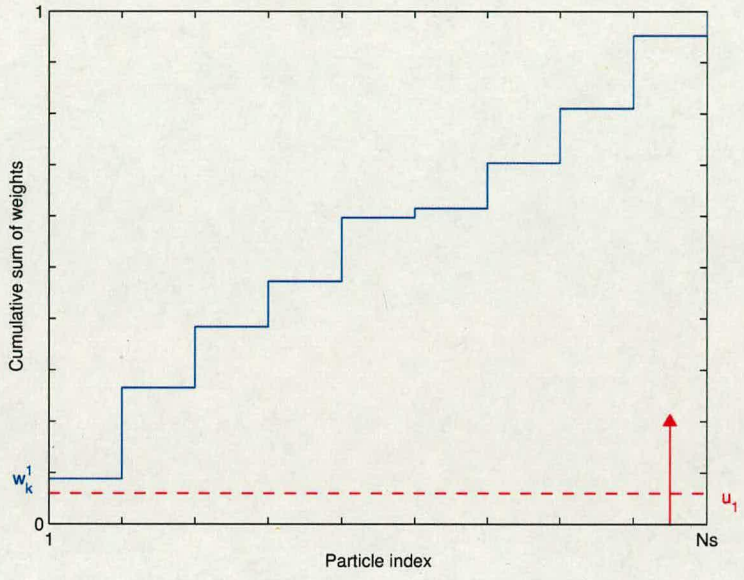


Figure 3.2: *The systematic resampling algorithm: in the graph the cumulative sum of weights is plotted over the particle indices. A random sample $u_1 \sim \mathbb{U}[0, N_s^{-1}]$ is drawn at every k which sets the particle replacement threshold. Subsequently, according to algorithm 2, all weights are compared with this threshold (which is increased by N_s^{-1} after every comparison). For the example shown, the first weight w_k^1 is larger that the threshold u_1 and thus the particle is not replaced.*

Algorithm 2 Systematic Resampling

$[\{\mathbf{x}_k^{j\star}, w_k^j, i^j\}_{j=1}^{N_s}] = \mathrm{SR}[\{\mathbf{x}_k^i, w_k^i\}_{i=1}^{N_s}]$

Initialise cumulative sum of weight (csw): $c_1 = w_k^1$
for $i = 2 : N_s$ **do**
  Compute csw: $c_i = c_{i-1} + w_k^i$
end for
Initialise index: $i = 1$
Draw a random threshold: $u_1 \sim \mathbb{U}[0, N_s^{-1}]$
for $j = 1 : N_s$ **do**
  Compare weights with threshold: $u_j = u_1 + N_s^{-1}(j-1)$
  while $u_j > c_i$ **do**
    Go to next weight: $i = i + 1$
  end while
  Assign sample: $\mathbf{x}_k^{j\star} = \mathbf{x}_k^i$
  Assign weight: $w_k^j = N_s^{-1}$
  Assign index: $i^j = i$
end for

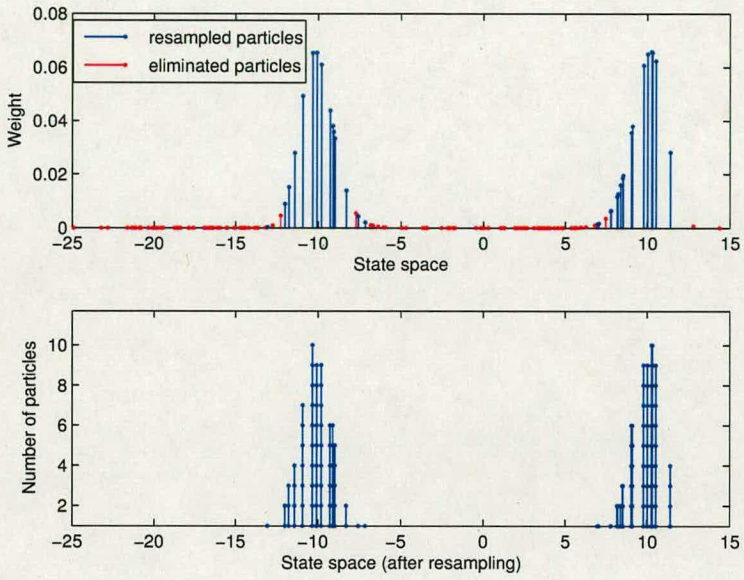


Figure 3.3: *Before resampling the approximation for the posterior state density is given by the particles' weight; after resampling by the density of the particles.*

3.3 Sequential importance resampling

The sequential importance resampling (SIR) filter was introduced in 1993 in [45]. It is a PF based on SIS which uses the state prior as the importance density:

$$q(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k) = p(\mathbf{x}_k|\mathbf{x}_{k-1}) \tag{3.14}$$

and resamples at every time step. For the specific importance density, the weight update equation (3.10) is simplified as:

$$\begin{aligned} \tilde{w}_k^i &= w_{k-1}^i \frac{p(\mathbf{z}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{z}_k)} \\ &= w_{k-1}^i \frac{p(\mathbf{z}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)} \\ &= w_{k-1}^i p(\mathbf{z}_k|\mathbf{x}_k^i) \end{aligned} \tag{3.15}$$

Because the algorithm resamples its posterior at every k, it holds that $w_{k-1}^i = 1/N_s,\ \forall i, k$. As

a result the term w_{k-1}^i cancels out during normalisation:

$$w_k^i = \frac{\tilde{w}_k^i}{\sum_{j=1}^{N_s} \tilde{w}_k^j} = \frac{w_{k-1}^i p(\mathbf{z}_k|\mathbf{x}_k^i)}{\sum_{j=1}^{N_s} w_{k-1}^j p(\mathbf{z}_k|\mathbf{x}_k^j)} = \frac{p(\mathbf{z}_k|\mathbf{x}_k^i)}{\sum_{j=1}^{N_s} p(\mathbf{z}_k|\mathbf{x}_k^j)} \tag{3.16}$$

which enables us to simply set:

$$\tilde{w}_k^i = p(\mathbf{z}_k|\mathbf{x}_k^i) \tag{3.17}$$

Since the importance density is the state-transition density, the particle generation is done by propagating each particle $\mathbf{x}_{k-1}^i$ through the system function $\mathbf{f}_{k-1}(\cdot)$ and perturbing it at the same time with a random process noise sample $\mathbf{u}_{k-1}^i$:

$$\mathbf{x}_k^i \sim p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i) \Leftrightarrow \mathbf{x}_k^i = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}^i) + \mathbf{u}_{k-1}^i \tag{3.18}$$

where $\mathbf{u}_k^i$ is drawn from the known process noise distribution.

The fact that the importance density - being the state prior $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ - does not incorporate information from the measurements, although it makes it easy to draw samples, might result in performance degradation. The phenomenon is more evident especially when the measurement noise or the modelling inaccuracies are significant. Fewer particles are seeded in areas in the state-space with bigger measurement likelihood, i.e. areas within which the particles obtain a considerable weight, and thus the resolution and consequently accuracy of the state posterior approximation deteriorates. The particle filters presented in the next section try to address this problem.

Finally, we present a brief comparison between the SIR and the conventional EKF for the

Algorithm 3 Sequential Importance Resampling

$[\{\mathbf{x}_k^i\}_{i=1}^{N_s}] = \text{SIR}[\{\mathbf{x}_{k-1}^i\}_{i=1}^{N_s}, \mathbf{z}_k]$

1: **for** $i = 1 : N_s$ **do**
2: Draw particles: $\mathbf{x}_k^i \sim p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)$
3: Compute weights: $\tilde{w}_k^i = p(\mathbf{z}_k|\mathbf{x}_k^i)$
4: **end for**
5: Compute total weight: $s_{w,k} = \sum_{i=1}^{N_s} \tilde{w}_k^i$
6: **for** $i = 1 : N_s$ **do**
7: Normalise weights: $w_k^i = \tilde{w}_k^i / s_{w,k}$
8: **end for**
9: Resample using SR: $[\{\mathbf{x}_k^i, \cdot, \cdot\}_{i=1}^{N_s}] = \text{SR}[\{\mathbf{x}_k^i, w_k^i\}_{i=1}^{N_s}]$

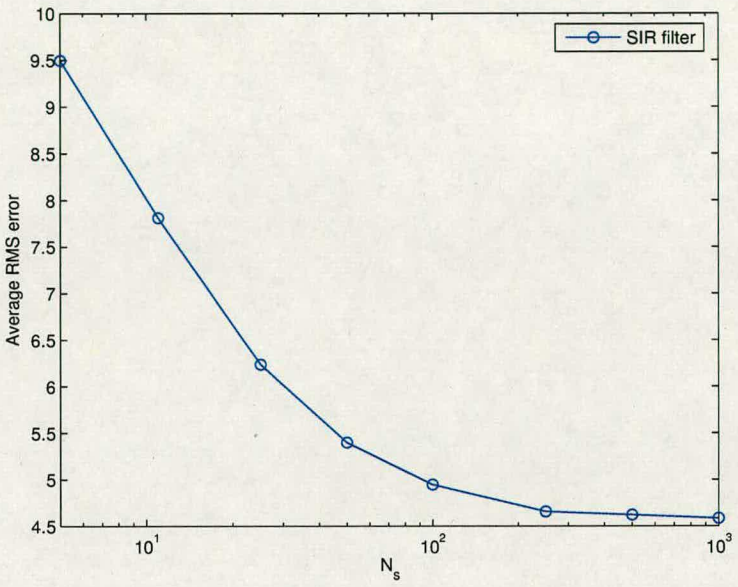


Figure 3.4: *Average RMS state error of the SIR over the number of particles N_s.*

following nonlinear system:

$$x_k = 0.5x_{k-1} + \frac{25x_{k-1}}{1+x_{k-1}^2} + 8\cos(1.2(k-1)) + u_k \tag{3.19}$$

$$z_k = \frac{x_k^2}{20} + v_k \tag{3.20}$$

where u_k and v_k are respectively the process and the measurement noise terms, which are white and Gaussian with variances $Q_k = 10$ and $R_k = 1$. For our analysis, we use eight SIR implementations each employing a different number of particles ($N_s = 5, 11, 25, 50, 100, 250, 500$ and 1000). The simulation time is 100 steps and the results are averaged after 1000 MC runs. Figure 3.4 shows the average RMS error of the SIR (between 4.58 and 9.49) over the number of the particles The error of the EKF is 19.91. The significant difference in performance is due to the severe nonlinearities of the system and the multimodal nature of the likelihood, which cannot be handled efficiency from the EKF operating under smooth-nonlinearity and Gaussianity assumptions. The cost for the improved performance of the SIR is a heavier computational complexity, which increases almost linearly with the number of the particles.

3.4 Filters approximating the optimal importance density

In what follows we present the auxiliary SIR and the local linearization particle filter. These filters approximate sampling from the optimal importance density[1] by incorporating information from the measurements in the particle prediction phase. The first algorithm indirectly by using a pre-prediction resampling scheme and the second by explicitly employing measurement-dependent nonlinear estimators in its prediction mechanism.

3.4.1 Auxiliary SIR

The auxiliary SIR (ASIR) was proposed in 1999 in [55]. The ASIR approximates sampling from the optimal $q(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k)$ by propagating through the state prior pre-selected particles which according to the measurement $\mathbf{z}_k$ have "large predictive likelihoods" [56, Ch. 13]. For the pre-selection certain *auxiliary* measures $\boldsymbol{\mu}_k^i$ associated with the particles $\mathbf{x}_{k-1}^i$ are initially predicted, weighted according to the measurement $\mathbf{z}_k$ and resampled. The particles follow the resampling outcome (i.e. they get replaced if their auxiliary measure does so) in anticipation that the resulting particle set will more likely be predicted in areas with larger likelihood.

The importance density of the ASIR is chosen as the joint density of the state $\mathbf{x}_k$ and the index i of the auxiliary measure, conditioned on the measurement $\mathbf{z}_k$[2]:

$$q(\mathbf{x}_k, i|\mathbf{z}_k) \propto p(\mathbf{z}_k|\boldsymbol{\mu}_k^i)p(\mathbf{x}_k|\mathbf{x}_{k-1}^i)w_{k-1}^i \tag{3.21}$$

where $\boldsymbol{\mu}_k^i$ can be any appropriate measure associated with $p(\mathbf{x}_k|\mathbf{x}_{k-1}^i)$ (the mean, the mode, a sample etc.). By setting $q(\mathbf{x}_k|i, \mathbf{z}_k) = p(\mathbf{x}_k|\mathbf{x}_{k-1}^i)$ we get:

$$\tilde{\lambda}_k^i := q(i|\mathbf{z}_k) \propto p(\mathbf{z}_k|\boldsymbol{\mu}_k^i)w_{k-1}^i \tag{3.22}$$

which when normalised:

$$\lambda_k^i = \frac{\tilde{\lambda}_k^i}{\sum_{t=1}^{N_s} \tilde{\lambda}_k^t} \tag{3.23}$$

[1] It has been shown [50] that the optimal importance density is:

$$q(\mathbf{x}_k|\mathbf{x}_{k-1}^i, z_k)_{optimal} = \frac{p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{x}_{k-1}^i)p(\mathbf{x}_k|\mathbf{x}_{k-1}^i)}{p(\mathbf{z}_k|\mathbf{x}_{k-1}^i)}$$

from which in general we cannot sample directly (except in certain cases as presented in [46]).

[2] The density $q(\mathbf{x}_k, i|\mathbf{z}_k)$ in 3.21 (which we use for keeping the notation consistent with the literature) could have had the more mathematically descriptive form $q(\mathbf{x}_k, i|\mathbf{z}_k, \mathbf{x}_{k-1}^i)$ for emphasising its dependency from $\mathbf{x}_{k-1}^i$.

form the first-stage weights. The weighted set $\{\boldsymbol{\mu}_k^i, \lambda_k^i\}_{i=1}^{N_s}$ is resampled using the SR algorithm, and the resulting resampled indices $\{i^j\}_{i=1}^{N_s}$ are used for predicting the particles:

$$\mathbf{x}_k^j \sim q(\mathbf{x}_k|i^j, \mathbf{z}_k) = p(\mathbf{x}_k|\mathbf{x}_{k-1}^{i^j}) \tag{3.24}$$

and for computing the final weights:

$$\tilde{w}_k^j = \frac{p(\mathbf{z}_k|\mathbf{x}_k^j)}{p(\mathbf{z}_k|\boldsymbol{\mu}_k^{i^j})} \tag{3.25}$$

which once more are normalised to sum to one:

$$w_k^j = \frac{\tilde{w}_k^j}{\sum_{t=1}^{N_s} \tilde{w}_k^t} \tag{3.26}$$

The ASIR performs particularly well when the process noise is relatively small. The auxiliary measures then represent more closely the density $p(\mathbf{x}_k|\mathbf{x}_{k-1}^i)$ and the particles when predicted lie indeed in areas with higher likelihood. This results in larger particle diversity and thus better characterisation of the posterior state density.

Algorithm 4 Auxiliary SIR

$[\{\mathbf{x}_k^i, w_k^i\}_{i=1}^{N_s}] = \text{ASIR}[\{\mathbf{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_s}, \mathbf{z}_k]$

for $i = 1 : N_s$ **do**
  Compute $\boldsymbol{\mu}_k^i$
  Compute first-stage weights: $\tilde{\lambda}_k^i = p(\mathbf{z}_k|\boldsymbol{\mu}_k^i) w_{k-1}^i$
end for
Compute total weight: $s_{\lambda,k} = \sum_{i=1}^{N_s} \tilde{\lambda}_k^i$
for $i = 1 : N_s$ **do**
  Normalise first-stage weights: $\lambda_k^i = \tilde{\lambda}_k^i / s_{\lambda,k}$
end for
Resample using SR: $[\{\cdot, \cdot, i^j\}_{j=1}^{N_s}] = \text{SR}[\{\boldsymbol{\mu}_k^i, \lambda_k^i\}_{i=1}^{N_s}]$
for $j = 1 : N_s$ **do**
  Draw particles: $\mathbf{x}_k^j \sim p(\mathbf{x}_k|\mathbf{x}_{k-1}^{i^j})$
  Compute weights: $\tilde{w}_k^j = p(\mathbf{z}_k|\mathbf{x}_k^j)/p(\mathbf{z}_k|\boldsymbol{\mu}_k^{i^j})$
end for
Compute total weight: $s_{w,k} = \sum_{j=1}^{N_s} \tilde{w}_k^j$
for $j = 1 : N_s$ **do**
  Normalise weights: $w_k^j = \tilde{w}_k^j / s_{w,k}$
end for

Reference [56] devises the metric:

$$\pi_k^i = \int \left(\frac{p(\mathbf{z}_k|\mathbf{x}_k)}{p(\mathbf{z}_k|\boldsymbol{\mu}_k^i)} \right)^2 d\mathbf{x}_k \tag{3.27}$$

and proves that if:

$$\sum_{i=1}^{N_s} \lambda_k^i \pi_k^i < N_s \sum_{i=1}^{N_s} (\lambda_k^i)^2 \pi_k^i \tag{3.28}$$

the efficiency (in terms of minimising the variance of the weights) of ASIR is better than the SIR. Since π_k^i varies mildly over i, approximately it holds that:

$$\sum_{i=1}^{N_s} \lambda_k^i = 1 \leq N_s \sum_{i=1}^{N_s} (\lambda_k^i)^2 \tag{3.29}$$

implying that the ASIR is in general more efficient and consequently performs better that the standard SIR.

3.4.2 Local linearization particle filter

The local linearization particle filter was introduced in 2000 in [50]. For predicting each particle $\mathbf{x}_k^i$ it first runs a nonlinear filter (e.g. an EKF) to approximate its posterior state density and then draws a sample from the resulting distribution. The resulting approximation of the optimal importance density is:

$$q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{z}_k) = \mathcal{N}(\hat{\mathbf{x}}_k^i, \hat{\mathbf{P}}_k^i) \tag{3.30}$$

where $\hat{\mathbf{x}}_k^i$ and $\hat{\mathbf{P}}_k^i$ are the estimated mean and covariance obtained from the nonlinear estimator. The weights are then computed:

$$\tilde{w}_k^i = \frac{p(\mathbf{z}_k|\mathbf{x}_k^i) p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{z}_k)} \tag{3.31}$$

and normalised:

$$w_k^i = \frac{\tilde{w}_k^i}{\sum_{j=1}^{N_s} \tilde{w}_k^j} \tag{3.32}$$

Like the SIR, the LLPF uses resampling at every time step therefore there is no need to store and propagate the weights to the future. A pseudocode describing the algorithm is given next.

The LLPF does propagate naturally the particles towards areas with higher likelihood, and

therefore compared to the original SIR it approximates better the posterior density and thus its performance improves. Its approach, that is to use directly the measurement for predicting the particles, seems to be the most powerful currently available. In [52] the UKF is used rather the EKF for prediction and it is shown to improve the performance. In [57] it is even proposed to use a particle filter like the SIR for predicting the particles; albeit powerful, the solution results in a rather heavy $O(N_s^2)$ computational complexity.

3.5 Tracking a dynamic changing target with a single-model particle filter

This section presents a brief study on a variation of the ASIR which we presented in [58] in 2004. We are interested in tracking a CV target which dynamically changes its kinematics model by executing subsequent abrupt turns; a moving pattern common in the vehicle tracking problem which we examine in the following chapters. As a simpler alternative from using a multiple model filter, we choose to employ a bank of EKFs, in the LLPF fashion, to predict the auxiliary measures $\boldsymbol{\mu}_k^i$ of the ASIR. Using the appropriate weights we indirectly bias the particle propagation mechanism with the measurement making thus the filter more sensitive to the model changes. In what follows we describe the algorithm and we contrast it with the SIR and ASIR in two simulated scenarios.

Algorithm 5 Local linearization particle filter

$[\{\mathbf{x}_k^i, \mathbf{P}_k^i\}_{i=1}^{N_s}] = \text{LLPF}[\{\mathbf{x}_{k-1}^i, \mathbf{P}_{k-1}^i\}_{i=1}^{N_s}, \mathbf{z}_k]$

1: **for** $i = 1 : N_s$ **do**
2: Run EKF: $[\{\hat{\mathbf{x}}_k^i, \hat{\mathbf{P}}_k^i\}_{i=1}^{N_s}] = \text{EKF}[\{\mathbf{x}_{k-1}^i, \mathbf{P}_{k-1}^i\}_{i=1}^{N_s}, \mathbf{z}_k]$
3: Draw particles: $\mathbf{x}_k^i \sim \mathcal{N}(\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k^i)$
4: Compute weights: $\tilde{w}_k^i = \dfrac{p(\mathbf{z}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{z}_k)}$
5: **end for**
6: Compute total weight: $s_{w,k} = \sum_{i=1}^{N_s} \tilde{w}_k^i$
7: **for** $i = 1 : N_s$ **do**
8: Normalise weights: $w_k^i = \tilde{w}_k^i / s_{w,k}$
9: **end for**
10: Resample using SR: $[\{\mathbf{x}_k^j, \cdot, i^j\}_{j=1}^{N_s}] = \text{SR}[\{\mathbf{x}_k^i, w_k^i\}_{i=1}^{N_s}]$
11: **for** $j = 1 : N_s$ **do**
12: Assign covariance: $\mathbf{P}_k^j = \hat{\mathbf{P}}_k^{i^j}$
13: **end for**

3.5.1 Auxiliary local linearization particle filter

The auxiliary local linearization particle filter (ALLPF) adopts the concept of pre-prediction resampling of the ASIR. It constructs a weighted set of auxiliary measures $\{\boldsymbol{\mu}_k^i, \lambda_k^i\}_{i=1}^{N_s}$ associated with $\{\mathbf{x}_{k-1}^i\}_{i=1}^{N_s}$, which uses to resample the particles at $k-1$. The resampled particles when predicted form $\{\mathbf{x}_k^i\}_{i=1}^{N_s}$. For making the technique more sensitive to measurements, EKFs are employed to calculate the auxiliary measures. The ALLPF, following the ASIR analysis, uses the importance density given below:

$$q(\mathbf{x}_k, i|\mathbf{z}_k) \propto q(\mathbf{z}_k|\boldsymbol{\mu}_k^i)p(\mathbf{x}_k|\mathbf{x}_{k-1}^i)w_{k-1}^i \tag{3.33}$$

in which the auxiliary measures $\boldsymbol{\mu}_k^i$ are obtained as samples from the distribution:

$$\boldsymbol{\mu}_k^i \sim \mathcal{N}(\hat{\mathbf{x}}_k^i, \hat{\mathbf{P}}_k^i) \tag{3.34}$$

where $\hat{\mathbf{x}}_k^i$ and $\hat{\mathbf{P}}_k^i$ are respectively the estimated mean and covariance from an EKF when applied on the $i\text{-}th$ particle at $k-1$ using measurement $\mathbf{z}_k$.

We factorise (3.33) as:

$$q(\mathbf{x}_k, i|\mathbf{z}_k) = q(i|\mathbf{z}_k)q(\mathbf{x}_k|i, \mathbf{z}_k) \tag{3.35}$$

and assume that the particles are propagated just according to the state prior (independently of the measurement):

$$q(\mathbf{x}_k|i, \mathbf{z}_k) = p(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{z}_k) \approx p(\mathbf{x}_k|\mathbf{x}_{k-1}^i) \tag{3.36}$$

The approximation in (3.36) biases the algorithm towards the measurement, since although the measurements were exploited from the EKFs in (3.34), they are not explicitly considered later in the weight update mechanism. With this approach we achieve indirectly greater sensitivity to possible model changes.

We continue by calculating the first-stage weights simply as the product of the likelihood and the previous weights:

$$\tilde{\lambda}_k^i := q(i|\mathbf{z}_k) = p(\mathbf{z}_k|\boldsymbol{\mu}_k^i)w_{k-1}^i \tag{3.37}$$

which are normalised to sum to one:

$$\lambda_k^i = \frac{\tilde{\lambda}_k^i}{\sum_{j=1}^{N_s} \tilde{\lambda}_k^j} \tag{3.38}$$

The set $\{\boldsymbol{\mu}_k^i, \lambda_k^i\}_{i=1}^{N_s}$ is resampled with the SR algorithm and the resulting resampled indices $\{i^j\}_{i=1}^{N_s}$ are used for predicting the particles using (3.36):

$$\mathbf{x}_k^j \sim q(\mathbf{x}_k^j | i^j, \mathbf{z}_k) = p(\mathbf{x}_k^j | \mathbf{x}_{k-1}^{ij}, \mathbf{z}_k) = \mathcal{N}(\hat{\mathbf{x}}_k^{ij}, \hat{\mathbf{P}}_k^{ij}) \tag{3.39}$$

where $\mathbf{x}_k^{ij}$ and $\hat{\mathbf{P}}_k^{ij}$ are the mean and covariance computed before. For calculating the final weights w_k^j we use equations (3.25-3.26) from the ASIR. The pseudocode of the ALLPF is given below.

The use of the EKFs for calculating the auxiliary measures in conjunction with the assumption in (3.36) naturally make the ALLPF more sensitive not only to model changes but to measurement noise and outliers. As we will see in the next chapter, a better approach to deal with this class of problems is to use algorithms which employ multiple models of operation. Nevertheless, the ALLPF provides a simple and efficient technique suitable for applications in which performance is gauged more in terms of robustness and track maintenance capability than just estimation accuracy.

Algorithm 6 Auxiliary LLPF

$[\{\mathbf{x}_k^i, \mathbf{P}_k^i, w_k^i\}_{i=1}^{N_s}] = \text{ALLPF}[\{\mathbf{x}_{k-1}^i, \mathbf{P}_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_s}, \mathbf{z}_k]$

1: **for** $i = 1 : N_s$ **do**
2: Run EKF: $[\{\hat{\mathbf{x}}_k^i, \hat{\mathbf{P}}_k^i\}_{i=1}^{N_s}] = \text{EKF}[\{\mathbf{x}_{k-1}^i, \mathbf{P}_{k-1}^i\}_{i=1}^{N_s}, \mathbf{z}_k]$
3: Draw auxiliary measures: $\boldsymbol{\mu}_k^i \sim \mathcal{N}(\hat{\mathbf{x}}_k^i, \hat{\mathbf{P}}_k^i)$
4: Compute first-stage weights: $\tilde{\lambda}_k^i = p(\mathbf{z}_k | \boldsymbol{\mu}_k^i) w_{k-1}^i$
5: **end for**
6: Compute total weight: $s_{\lambda,k} = \sum_{i=1}^{N_s} \tilde{\lambda}_k^i$
7: **for** $i = 1 : N_s$ **do**
8: Normalise first-stage weights: $\lambda_k^i = \tilde{\lambda}_k^i / s_{\lambda,k}$
9: **end for**
10: Resample using SR: $[\{\cdot, \cdot, i^j\}_{j=1}^{N_s}] = \text{SR}[\{\boldsymbol{\mu}_k^i, \lambda_k^i\}_{i=1}^{N_s}]$
11: **for** $j = 1 : N_s$ **do**
12: Assign covariance: $\mathbf{P}_k^j = \hat{\mathbf{P}}_k^{ij}$
13: Draw particles: $\mathbf{x}_k^j \sim \mathcal{N}(\hat{\mathbf{x}}_k^{ij}, \mathbf{P}_k^j)$
14: Compute weights: $\tilde{w}_k^j = p(\mathbf{z}_k | \mathbf{x}_k^j) / p(\mathbf{z}_k | \boldsymbol{\mu}_k^{ij})$
15: **end for**
16: Compute total weight: $s_{w,k} = \sum_{j=1}^{N_s} \tilde{w}_k^j$
17: **for** $j = 1 : N_s$ **do**
18: Normalise weights: $w_k^j = \tilde{w}_k^j / s_{w,k}$
19: **end for**

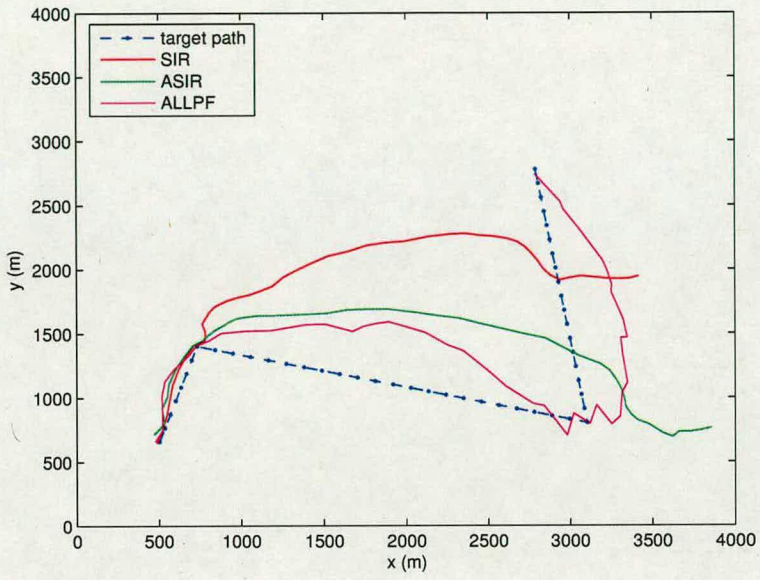


Figure 3.5: *The true track of the target and the track estimates from the particle filters for a representative example.*

3.5.2 Simulation study

In this section we test the performance of the ALLPF in two different tracking scenarios. The first scenario involves tracking a two-dimensional target (e.g. a ground target) which although nominally moves with constant velocity, executes two abrupt turns along its path. The importance density and particularly its efficiency in exploiting the measurement information is expected to play a critical role in the tracking accuracy.

For our simulations we use the target track shown at figure 3.5. The initial position of the target is $(x, y) = (504, 663)$m and its velocity $(\dot{x}, \dot{y}) = (33, 106)$m/s. The first turn is $-75°$ and the second $115°$. The particle filters consider a constant velocity target model:

$$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_{k-1} \tag{3.40}$$

where $\mathbf{x}_k = [x_k \;\; y_k \;\; \dot{x}_k \;\; \dot{y}_k]^T$ is the state vector, $\mathbf{u}_k = [u_{x,k} \;\; u_{y,k}]^T$ is the process noise vector, and:

$$\mathbf{F}_k = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{G}_k = \begin{bmatrix} T^2/2 & 0 \\ 0 & T^2/2 \\ T & 0 \\ 0 & T \end{bmatrix} \tag{3.41}$$

are the system and process noise matrices. The process noise is assumed to be zero-mean Gaussian with standard deviation 5m/s^2 for both moving directions.

The radar lies at the origin of the plane at point $(0, 0)$ and has an update rate $T = 1\text{s}$. The measurement equation is:

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k \tag{3.42}$$

where $\mathbf{z}_k = [\theta_k \;\; r_k]^T$ is the measurement vector consisting of the azimuth θ_k and range r_k, $\mathbf{h}_k(\cdot)$ is the nonlinear function:

$$\mathbf{h}(\mathbf{x}_k) = \begin{bmatrix} \arctan(y_k/x_k) \\ \sqrt{x_k^2 + y_k^2} \end{bmatrix} \tag{3.43}$$

and $\mathbf{v}_k = [v_{\theta,k} \;\; v_{r,k}]^T$ is the measurement noise vector. The radar noise is modelled as zero-mean Gaussian with standard deviation 4° for the azimuth and 20m for the range.

For tracking we use the SIR, ASIR and ALLPF. For a fair comparison, the filters depending on their computational complexity employ a different number of particles (respectively 1200, 1000 and 200) so as to finally have similar CPU-time requirements. The results presented next were obtained after 1000 MC runs. The algorithms are initialised with the true target states.

Figure 3.5 presents a representative example of the track estimates of the three different filters. For this run the ALLPF exhibits the faster response succeeding in maintaining the track, whereas both the SIR and ASIR cannot not cope with the changing dynamic behaviour of the target and diverge. Overall, the RMS position error over the simulation time can be seen in figure 3.6[3]. The average values are given in table 4.1. As expected, the particle prediction mechanism of the ALLPF being more sensitive to the model changes, enables the algorithm for a faster and more accurate transient response after each turn. From table 4.1 we obtain that for the specific example the ALLPF results respectively in 63% and 59% smaller RMS position

[3]For completeness we plot also the RMSE of the un-biased version of the ALLPF (black dashed line), for which we set $\tilde{\lambda}_k^i = p(\mathbf{z}_k|\boldsymbol{\mu}_k^i) w_{k-1}^i p(\boldsymbol{\mu}_k^i|\mathbf{x}_{k-1}^i)/\mathcal{N}(\hat{\mathbf{x}}_k^i, \mathbf{P}_k^i)$ and $\tilde{w}_k^j = p(\mathbf{z}_k|\mathbf{x}_k^j)/p(\mathbf{z}_k|\boldsymbol{\mu}_k^{i^j}) p(\mathbf{x}_k^j|\mathbf{x}_{k-1}^{i^j})/\mathcal{N}(\hat{\mathbf{x}}_k^{i^j}, \mathbf{P}_k^j)$.

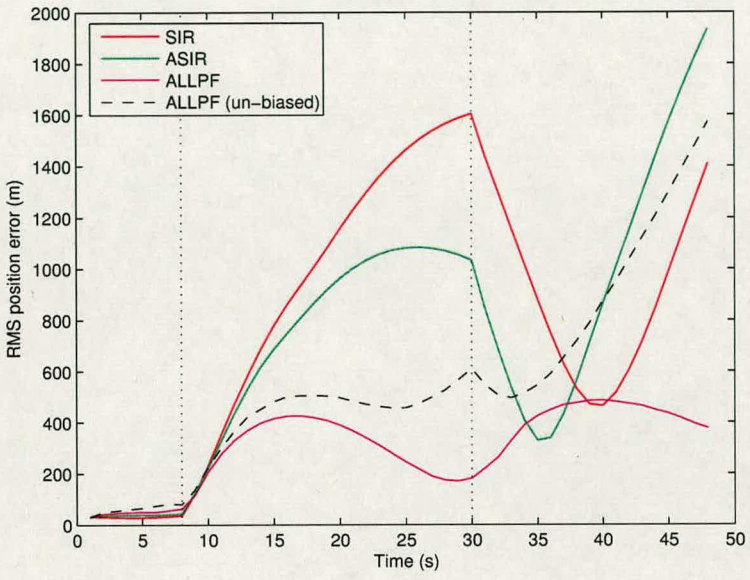


Figure 3.6: *Average RMS position error of the particle filters after 1000 MC runs. The dotted lines indicate the track turns.*

error than the SIR and the ASIR, while using significantly fewer particles. More importantly the ALLPF manages to maintain the track of the target.

In the second scenario we simply track a constant velocity target which is perturbed with random accelerations, using the same simulation parameters as before. After 1000 MC runs, we obtain that the average RMS position error is 95.88m for the SIR, 100.52m for the ASIR and 122.99m for the ALLPF. Naturally, the measurement-sensitive ALLPF in this case exhibits the worst performance and results respectively in about 28% and 22% larger error than the SIR and ASIR.

	Particles	Average RMS position error (m)	Average CPU time (s)
SIR	1200	814.45	3.61
ASIR	1000	743.29	3.58
ALLPF	200	300.39	3.63

Table 3.1: *Comparison of the tracking performance of the particle filters after 1000 MC runs.*

3.5.3 Conclusions

This section presented a particle filter which combined elements from both the ASIR and LLPF. The proposed ALLPF adopts the resampling mechanism of the ASIR for selecting the prediction particles and uses the local linearization logic of the LLPF for predicting them. Using an appropriate weight update mechanism, it biases the particles towards the measurement and thus robustifies the estimates to irregular changes of the system model.

The ALLPF was first contrasted with the SIR and the ASIR in a tracking scenario in which a target executed two unpredictable abrupt turns along its path. Simulation results demonstrated the improved capability of the ALLPF for adopting quickly to model changes. A second scenario with a non-manoeuvring target, highlighted the moderate weakness of the ALLPF to filter efficiency the measurement noise when the kinematic model was consistent. Based on the experimental results we can argue that the ALLPF can be proved suitable for tracking devices whose priority is an enhanced track-maintenance capability. For more demanding applications and more complex tracking environments a multiple-model filter like the ones described in the next chapters, or other single-model algorithms which explicitly account for manoeuvring dynamics as in [59], would be more appropriate.

3.6 Multitarget tracking with particle filters

The remaining part of the chapter focuses on multitarget tracking after a work we published in [60] in 2005. After a brief introduction and a formulation of the problem, we propose a multitarget particle filter which uses a local linearization prediction mechanism within which the number of the particles vary and we study the performance gains compared to an equivalent multitarget SIR-based particle filter.

3.6.1 Introduction

Although a big body of the work in particle filtering studies the single-target tracking problem, a certain number of algorithms have been proposed for the multitarget case. When multiple targets exist, modifications to the standard single-target algorithms should be made. There are two major approaches to address the problem: we can either incorporate all targets and measurements within the state and measurement vectors and track all targets jointly, or use first

a data association scheme to assign the measurements to the targets and then track the targets separately.

In the first approach (e.g. [61, 62]) the state-space is augmented and the joint multitarget probability density (JMPD) is used for deriving the state estimates of the targets. The main drawback of this technique is that the dimensionality of the joint state-space grows exponentially with the number of targets. This results in an increased computational complexity and the need of more particles for attain a certain performance since even a modest 3-target problem may produce a 27-dimensional space[4]. It has been shown that gating and partition sampling [61] can reduce the computational requirements for this class of problems. The probability hypothesis density (PHD) particle filter [63–65] is a sub-optimal approach that tries to address the complexity problem by using just the first-order moments of the multitarget posterior pdf. In the PHD-PF the state-dimensionality remains as in the single target case but now information from the essential higher-order statistics[5] is lost. The major weakness of all these methods is that they can only provide estimates of the states of the targets but not target tracks [66, 67].

The second class of particle filters (e.g. [68–70]) incorporate a data association logic which is used to assign the measurement to new or already existing targets. Particle filtering techniques are then used for deriving the state estimates. This can be done either by using separate trackers for each target or by forming an augmented multitarget state space and using the "hard" associations step to perform the prediction and estimation jointly for all targets. Generally, functions like gating, track initialisation/deletion and clutter rejection can be implemented and incorporated within the data association mechanism more easily and in a more computational efficient way than in the JMPD-based algorithms. If after the association separate trackers are used for the targets, the dimensions remain as in the single-target case.

Regarding the estimation mechanisms themselves, most multitarget particle filters in the literature utilise an SIR-like structure in conjunction with a more sophisticated multitarget logic. For predicting the particles, they normally use the prior state pdf as their importance density, a practise which as we have seen albeit easy and computationally cheap can limit the tracking performance. In the following sections we study the benefits of using the local linearization importance density within the multitarget problem, concentrating in parallel on a computationally

[4] If we consider the states as the targets' position, velocity and acceleration within the x-y-z space

[5] One could argue that this is one of the major flaws of the PHD technique, since these high-order moments are in general responsible for the powerful performance associated with particle filtering.

efficient filter implementation.

The particle filters that we present next employ a data association logic and operate within an augmented multitarget state- and measurement-space. The algorithms first use a joint step of data association and prediction and then calculate the posterior state estimates. For demonstrating clearly the differences in performance between the filters, we use a simple association method and a basic tracking environment which is presented next.

3.6.2 Multitarget modelling

In this section we generalise the single-target system described in 2.5.1 and 2.5.3 for the multiple-target case concentrating on two-dimensional constant velocity targets moving across the x-y plane. For simplicity in our analysis we consider that at every time scan k we have n targets and an equal number of measurements originating from them. For the observations we use a radar which measures the azimuth θ_k and range r_k of each target.

The state- and measurement spaces are augmented for accounting for the multiple targets and measurements. Therefore, the state transition equation:

$$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_{k-1} \tag{3.44}$$

has a state vector which now is given by:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_k^{(1)} \\ \vdots \\ \mathbf{x}_k^{(n)} \end{bmatrix} \tag{3.45}$$

in which $\mathbf{x}_k^{(\tau)} = [x_k^{(\tau)} \;\; y_k^{(\tau)} \;\; \dot{x}_k^{(\tau)} \;\; \dot{y}_k^{(\tau)}]^T$ is the single-target state for the τ-*th* target, where $\tau = 1, \ldots, n$. The state and process noise matrices become respectively $\mathbf{F}_k = diag\{\mathbf{F}'_k, \ldots, \mathbf{F}'_k\}$ and $\mathbf{G}_k = diag\{\mathbf{G}'_k, \ldots, \mathbf{G}'_k\}$, where notation $diag\{\mathbf{A}, \ldots, \mathbf{B}\}$ stands for the diagonal matrix with diagonal elements the sub-matrices $\mathbf{A}, \ldots, \mathbf{B}$. For our system both $\mathbf{F}_k$ and $\mathbf{G}_k$ have n diagonal elements. The $\mathbf{F}'_k$ and $\mathbf{G}'_k$ are the known single-target matrices:

$$\mathbf{F}'_k = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{G}'_k = \begin{bmatrix} T^2/2 & 0 \\ 0 & T^2/2 \\ T & 0 \\ 0 & T \end{bmatrix} \tag{3.46}$$

The process noise vector in (3.44) has the form:

$$\mathbf{u}_k = \begin{bmatrix} \mathbf{u}_k^{(1)} \\ \vdots \\ \mathbf{u}_k^{(n)} \end{bmatrix} \tag{3.47}$$

where $\mathbf{u}_k^{(\tau)} = [u_{x,k}^{(\tau)} \; u_{y,k}^{(\tau)}]^T$ is the noise vector for the τ-*th* target. The process noise is assumed to be zero-mean white Gaussian $\mathbf{u}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$, where $\mathbf{Q}_k = diag\{\mathbf{Q}'_k, \ldots, \mathbf{Q}'_k\}$ is the augmented $2n \times 2n$ covariance matrix consisting of n diagonal elements:

$$\mathbf{Q}'_k = \begin{bmatrix} \sigma\{u_{x,k}\}^2 & 0 \\ 0 & \sigma\{u_{y,k}\}^2 \end{bmatrix} \tag{3.48}$$

where $\sigma\{u_{x,k}\}$ and $\sigma\{u_{y,k}\}$ are the standard deviation of the process noise along the x and y moving directions.

The multitarget measurement equation is:

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k \tag{3.49}$$

where $\mathbf{z}_k$ is obtained after the association. The augmented vector with the raw measurements (before association) is:

$$\mathbf{z}_k^\star = \begin{bmatrix} \mathbf{z}_k^{(1)} \\ \vdots \\ \mathbf{z}_k^{(n)} \end{bmatrix} \tag{3.50}$$

where $\mathbf{z}_k^{(\mu)} = [\theta_k^{(\mu)} \; r_k^{(\mu)}]^T$ is the μ-*th* "sub-measurement" at scan k, where $\mu = 1, \ldots, n$. The nonlinear function $\mathbf{h}_k(\cdot)$ that maps the state- to the measurement-space is:

$$\mathbf{h}_k(\mathbf{x}_k) = \begin{bmatrix} \mathbf{h}'_k(\mathbf{x}_k^{(1)}) \\ \vdots \\ \mathbf{h}'_k(\mathbf{x}_k^{(n)}) \end{bmatrix} \tag{3.51}$$

where:

$$\mathbf{h}'_k(\mathbf{x}_k^{(\tau)}) = \begin{bmatrix} \arctan(y_k^{(\tau)}/x_k^{(\tau)}) \\ \sqrt{(x_k^{(\tau)})^2 + (y_k^{(\tau)})^2} \end{bmatrix} \tag{3.52}$$

for the τ-*th* target. The measurement noise vector is:

$$\mathbf{v}_k = \begin{bmatrix} \mathbf{v}_k^{(1)} \\ \vdots \\ \mathbf{v}_k^{(n)} \end{bmatrix} \tag{3.53}$$

where $\mathbf{v}_k^{(\mu)} = [v_{\theta,k}^{(\mu)} \;\; v_{r,k}^{(\mu)}]^T$ is the noise of the μ-*th* measurement. The measurement noise is considered also zero-mean white Gaussian $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$, where $\mathbf{R}_k = diag\{\mathbf{R}'_k, \ldots, \mathbf{R}'_k\}$ is the $2n \times 2n$ noise covariance matrix with n diagonal sub-matrices:

$$\mathbf{R}'_k = \begin{bmatrix} \sigma\{v_{\theta,k}\}^2 & 0 \\ 0 & \sigma\{v_{r,k}\}^2 \end{bmatrix} \tag{3.54}$$

where $\sigma\{v_{\theta,k}\}$ and $\sigma\{v_{r,k}\}$ are the standard deviation of the measurement noise of the azimuth θ_k and range r_k.

For clarity, the formulation above assumes the same type of targets and one static measuring sensor, therefore the covariance matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$ consist of identical sub-covariances $\mathbf{Q}'_k$ and $\mathbf{R}'_k$. The generalisation for different $\mathbf{Q}'^{(\tau)}_k$ and $\mathbf{R}'^{(\mu)}_k$ is straightforward.

3.7 Multitarget particle filters

Sections 3.7.1 and 3.7.2 describe two multitarget particle filters based on the SIR and the LLPF algorithms. Section 3.7.3 presents an efficient variation of the multitarget LLPF algorithm in which the number of the particles during the association phase varies.

3.7.1 Multitarget SIR

The multitarget SIR (MSIR) is based on the SIR algorithm enhanced with a basic data association function. The MSIR follows the multitarget formulation of the previous section and accounts for an environment in which an equal number n of targets and measurement exist. It consists of four steps: a prediction step in which the particles are propagated one step ahead and form the a-priori state estimate, a data association step in which the new measurements are assigned to the targets, an update step in which the posterior state distribution is estimated using the measurements and a resampling step in which the posterior particles are resampled with replacement.

The MSIR employs N_s $4n$-dimensional particles $\{\mathbf{x}_k^i\}_{i=1}^{N_s}$ and uses the transitional prior as its importance density:

$$q(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k) = p(\mathbf{x}_k|\mathbf{x}_{k-1}) \tag{3.55}$$

Therefore, the algorithm begins by predicting the particles using the state transition equation (3.44):

$$\mathbf{x}_k^i = \mathbf{F}_{k-1}\mathbf{x}_{k-1}^i + \mathbf{G}_{k-1}\mathbf{u}_{k-1}^i \tag{3.56}$$

where $\mathbf{u}_{k-1}^i$ are random noise samples drawn from $\mathbf{u}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$. The a-priori state estimate $\bar{\mathbf{x}}_k$ is computed as the mean value of the predicted particles:

$$\bar{\mathbf{x}}_k = \begin{bmatrix} \bar{\mathbf{x}}_k^{(1)} \\ \vdots \\ \bar{\mathbf{x}}_k^{(n)} \end{bmatrix} = \frac{1}{N_s}\sum_{i=1}^{N_s}\mathbf{x}_k^i \tag{3.57}$$

where $\bar{\mathbf{x}}_k^{(\tau)} = [\bar{x}_k^{(\tau)} \;\; \bar{y}_k^{(\tau)} \;\; \bar{\dot{x}}_k^{(\tau)} \;\; \bar{\dot{y}}_k^{(\tau)}]^T$ for $\tau = 1, \ldots, n$.

The data association step follows. We focus on a robust and simple-to-implement association mechanism based on the distance between the a-priori state $\bar{\mathbf{x}}_k$ and the measurement $\mathbf{z}_k$. For implementing the algorithm we construct an $n \times n!$ hypothesis matrix $\mathbf{H}$ consisting of $n!$ columns each denoted as H_α where $\alpha = 1, \ldots, n!$. Each column H_α corresponds to one of the different permutations of the set $\{1, \ldots, n\}$. For example for $n = 3$ we can use the following

3×6 hypothesis matrix:

$$\begin{matrix} & H_1 & H_2 & \dots & & H_6 \\ & \downarrow & \downarrow & & & \downarrow \end{matrix}$$
$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 2 & 2 & 3 & 3 \\ 2 & 3 & 1 & 3 & 1 & 2 \\ 3 & 2 & 3 & 1 & 2 & 1 \end{bmatrix} \tag{3.58}$$

The columns account for a different association hypothesis, e.g. for the matrix in (3.58) column H_2 shows that:

$$H_2 = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} \begin{matrix} \rightarrow & H_2(1) = 1 \\ \rightarrow & H_2(2) = 3 \\ \rightarrow & H_2(3) = 2 \end{matrix} \tag{3.59}$$

which implies that target 1 should be assigned with measurement 1 ($H_2(1) = 1$), target 2 with measurement 3 ($H_2(2) = 3$) and target 3 with measurement 2 ($H_2(3) = 2$).

We seek to find the association that minimises the following distance function:

$$d_k(\alpha) = \left|\mathbf{x}_{d,k} - \mathbf{z}_{d,k}^{\alpha}\right|_2 \tag{3.60}$$

where $\alpha = 1, \dots, n!$. Notation $|\mathbf{a} - \mathbf{b}|_2$ stands for the l^2-norm or Euclidean distance between points (vectors) $\mathbf{a}$ and $\mathbf{b}$. In the equality above, the constant term $\mathbf{x}_{d,k}$ is the Cartesian position of the a-priori state estimate:

$$\mathbf{x}_{d,k} = \begin{bmatrix} \mathbf{x}_{d,k}^{(1)} \\ \vdots \\ \mathbf{x}_{d,k}^{(n)} \end{bmatrix} \tag{3.61}$$

where $\mathbf{x}_{d,k}^{(\tau)} = [\bar{x}_k^{(\tau)} \;\; \bar{y}_k^{(\tau)}]^T$ for $\tau = 1, \dots, n$. The second term $\mathbf{z}_{d,k}^{\alpha}$ represents the multitarget measurement transformed to the Cartesian plane for all $\alpha = 1, \dots, n!$ association hypotheses:

$$\mathbf{z}_{d,k}^{\alpha} = \begin{bmatrix} \mathbf{h}_k'^{-1}(\mathbf{z}_k^{(H_\alpha(1))}) \\ \vdots \\ \mathbf{h}_k'^{-1}(\mathbf{z}_k^{(H_\alpha(n))}) \end{bmatrix} \tag{3.62}$$

where $\mathbf{h}_k'^{-1}(\mathbf{z}_k^{(H_\alpha(\mu))}) = [r_k^{(\mu)} \cdot \cos\theta_k^{(\mu)} \;\; r_k^{(\mu)} \cdot \sin\theta_k^{(\mu)}]^T$ for $\mu = 1, \dots, n$, $H_\alpha(i)$ is the $i\text{-}th$ element of the $\alpha\text{-}th$ column of the hypothesis matrix $\mathbf{H}$ and $\mathbf{z}_k^{(i)}$ is the $i\text{-}th$ sub-measurement

of the raw un-associated measurement vector:

$$\mathbf{z}_k^\star = \begin{bmatrix} \mathbf{z}_k^{(1)} \\ \vdots \\ \mathbf{z}_k^{(n)} \end{bmatrix} \tag{3.63}$$

The associated measurement vector is constructed using the assignment hypothesis H_α which results in the minimum distance $d_k(\alpha)$:

$$\mathbf{z}_k = \left. \begin{bmatrix} \mathbf{z}_k^{(H_\alpha(1))} \\ \vdots \\ \mathbf{z}_k^{(H_\alpha(n))} \end{bmatrix} \right|_{\alpha=\min_\alpha\{d_k(\alpha)\}} \tag{3.64}$$

We continue by calculating the normalised particle weights w_k^i according to the particles' likelihood, in the normal SIR fashion:

$$\tilde{w}_k^i = p(\mathbf{z}_k|\mathbf{x}_k^i) \tag{3.65}$$

which are normalised to sum to one:

$$w_k^i = \frac{\tilde{w}_k^i}{\sum_{j=1}^{N_s} \tilde{w}_k^j} \tag{3.66}$$

The posterior state estimate $\hat{\mathbf{x}}_k$ is given then by the weighted sum of the particles:

$$\hat{\mathbf{x}}_k = \sum_{i=1}^{N_s} w_k^i \mathbf{x}_k^i \tag{3.67}$$

Finally, we apply the SR algorithm from section 3.2 and resample with replacement the particles.

Figure 3.7 illustrates graphically the data association principle of the MSIR. For clarity we use an example with two targets which move in the x dimension. The augmented state-space vector is:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_k^{(1)} \\ \mathbf{x}_k^{(2)} \end{bmatrix} \tag{3.68}$$

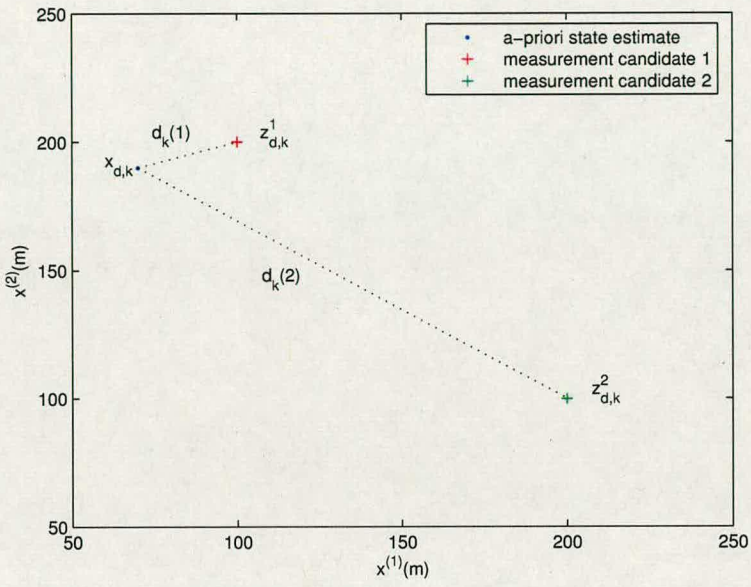


Figure 3.7: *An MSIR association example for two one-dimensional targets.*

The a-priori state estimate when transformed to the Cartesian position plane becomes:

$$\mathbf{x}_{d,k} = \begin{bmatrix} \bar{x}_k^{(1)} \\ \bar{x}_k^{(2)} \end{bmatrix} \tag{3.69}$$

The un-associated measurement:

$$\mathbf{z}_k^\star = \begin{bmatrix} \mathbf{z}_k^{(1)} \\ \mathbf{z}_k^{(2)} \end{bmatrix} \tag{3.70}$$

result in the following measurement candidates:

$$\mathbf{z}_k^1 = \begin{bmatrix} \mathbf{z}_k^{(1)} \\ \mathbf{z}_k^{(2)} \end{bmatrix}, \quad \mathbf{z}_k^2 = \begin{bmatrix} \mathbf{z}_k^{(2)} \\ \mathbf{z}_k^{(1)} \end{bmatrix} \tag{3.71}$$

which when transformed to the Cartesian position plane become:

$$\mathbf{z}_{d,k}^1 = \begin{bmatrix} \mathbf{h}_k'^{-1}(\mathbf{z}_k^{(1)}) \\ \mathbf{h}_k'^{-1}(\mathbf{z}_k^{(2)}) \end{bmatrix}, \quad \mathbf{z}_{d,k}^2 = \begin{bmatrix} \mathbf{h}_k'^{-1}(\mathbf{z}_k^{(2)}) \\ \mathbf{h}_k'^{-1}(\mathbf{z}_k^{(1)}) \end{bmatrix} \tag{3.72}$$

where functions $\mathbf{h}_k'$ and $\mathbf{h}_k'^{-1}$ are modified accordingly for our one-dimensional example. The

distances between $\bar{\mathbf{x}}_k$ and $\mathbf{z}_{d,k}^1$, $\mathbf{z}_{d,k}^2$ finally are given by:

$$d_k(1) = \left|\mathbf{x}_{d,k} - \mathbf{z}_{d,k}^1\right|_2, \quad d_k(2) = \left|\mathbf{x}_{d,k} - \mathbf{z}_{d,k}^2\right|_2 \tag{3.73}$$

3.7.2 Multitarget LLPF

The multitarget LLPF (MLLPF) uses the MSIR structure modified accordingly to account for a local linearization prediction mechanism. The association within the MLLPF is performed simultaneously with the particle prediction, since the latter is measurement-dependent. MLLPF is based again on the multitarget formulation from 3.6.2 and thus considers n targets and measurements and employs N_s $4n$-dimensional particles $\{\mathbf{x}_k^i\}_{i=1}^{N_s}$.

The single-target LLPF filter as we have seen uses the following importance density function:

$$q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{z}_k) = \mathcal{N}(\hat{\mathbf{x}}_k^i, \hat{\mathbf{P}}_k^i) \tag{3.74}$$

where $\hat{\mathbf{x}}_k^i$ and $\hat{\mathbf{P}}_k^i$ are computed using a nonlinear state estimator like the EKF. From (3.74) it is apparent that for the prediction we need the *associated* measurement vector $\mathbf{z}_k$. Since at this point we do not know the correct association, we need to predict the particles using all different measurement assignment hypotheses. We construct thus $n!$ particle sets $\{\{\mathbf{x}_k^{\alpha,i}\}_{i=1}^{N_s}\}_{\alpha=1}^{n!}$ using:

$$\mathbf{x}_k^{\alpha,i} \sim q(\mathbf{x}_k^{\alpha,i}|\mathbf{x}_{k-1}^i, \mathbf{z}_k^\alpha) = \mathcal{N}(\hat{\mathbf{x}}_k^{\alpha,i}, \hat{\mathbf{P}}_k^{\alpha,i}) \tag{3.75}$$

in which $\hat{\mathbf{x}}_k^{\alpha,i}$ and $\hat{\mathbf{P}}_k^{\alpha,i}$ are computed using an EKF:

$$[\{\hat{\mathbf{x}}_k^{\alpha,i}, \hat{\mathbf{P}}_k^{\alpha,i}\}_{i=1}^{N_s}] = \mathrm{EKF}[\{\mathbf{x}_{k-1}^i, \mathbf{P}_{k-1}^i\}_{i=1}^{N_s}, \mathbf{z}_k^\alpha] \tag{3.76}$$

The term $\mathbf{z}_k^\alpha$ is the measurement vector of the α-th association hypothesis:

$$\mathbf{z}_k^\alpha = \begin{bmatrix} \mathbf{z}_k^{(H_\alpha(1))} \\ \vdots \\ \mathbf{z}_k^{(H_\alpha(n))} \end{bmatrix} \tag{3.77}$$

where $\mathbf{z}_k^{(i)}$ is the i-th un-associated sub-measurement from (3.63) and $H_\alpha(i)$ is the i-th element of the α-th column of the hypothesis matrix $\mathbf{H}$ as described in the previous section.

We continue by calculating the a-priori estimates for the different particle sets:

$$\bar{\mathbf{x}}_k^{\alpha} = \begin{bmatrix} \bar{\mathbf{x}}_k^{\alpha,(1)} \\ \vdots \\ \bar{\mathbf{x}}_k^{\alpha,(n)} \end{bmatrix} = \frac{1}{N_s} \sum_{i=1}^{N_s} \mathbf{x}_k^{\alpha,i} \tag{3.78}$$

where $\bar{\mathbf{x}}_k^{\alpha,(\tau)} = [\bar{x}_k^{\alpha,(\tau)} \;\; \bar{y}_k^{\alpha,(\tau)} \;\; \bar{\dot{x}}_k^{\alpha,(\tau)} \;\; \bar{\dot{y}}_k^{\alpha,(\tau)}]^T$ for $\tau = 1, \ldots, n$.

The data association phase follows. The distance function to be minimised now becomes:

$$d_k(\alpha) = \left| \mathbf{x}_{d,k}^{\alpha} - \mathbf{z}_{d,k}^{\alpha} \right|_2 \tag{3.79}$$

where $\alpha = 1, \ldots, n!$. The first term $\mathbf{x}_{d,k}^{\alpha}$ is the Cartesian position of the a-priori state estimates:

$$\mathbf{x}_{d,k}^{\alpha} = \begin{bmatrix} \mathbf{x}_{d,k}^{\alpha,(1)} \\ \vdots \\ \mathbf{x}_{d,k}^{\alpha,(n)} \end{bmatrix} \tag{3.80}$$

where $\mathbf{x}_{d,k}^{\alpha,(\tau)} = [\bar{x}_k^{\alpha,(\tau)} \;\; \bar{y}_k^{\alpha,(\tau)}]^T$ for $\tau = 1, \ldots, n$. The second term $\mathbf{z}_{d,k}^{\alpha}$ is calculated as in the MSIR using (3.62). We associate the measurements using relation (3.64) constructing thus $\mathbf{z}_k$ and we set:

$$\mathbf{x}_k^i = \mathbf{x}_k^{\alpha,i} \Big|_{\alpha = \min_{\alpha}\{d_k(\alpha)\}} \tag{3.81}$$

The update phase follows, in which we compute the normalised particle weights w_k^i as in the single-target case using:

$$\tilde{w}_k^i = \frac{p(\mathbf{z}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{z}_k)} \tag{3.82}$$

and:

$$w_k^i = \frac{\tilde{w}_k^i}{\sum_{j=1}^{N_s} \tilde{w}_k^j} \tag{3.83}$$

The multitarget state estimate for the current scan k is given using relation (3.67). As always, finally the SR algorithm is used for resampling, assigning to each resampled particle its associated covariance

Figure 3.92 presents graphically the MLLPF association step using the one dimensional example from the previous sub-section. The state vector remains the same but this time we have two

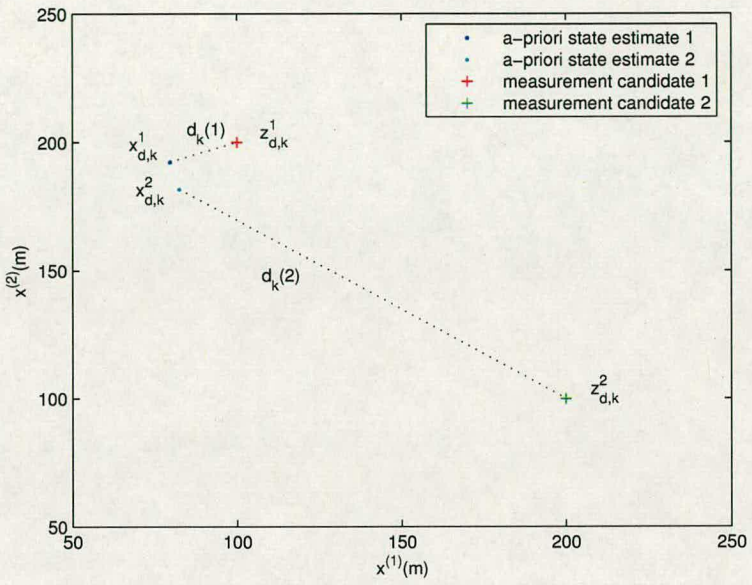


Figure 3.8: *An MLLPF association example for two one-dimensional targets.*

candidate prior state estimates:

$$\bar{\mathbf{x}}^1_k = \begin{bmatrix} \bar{\mathbf{x}}^{(1)}_k \\ \bar{\mathbf{x}}^{(2)}_k \end{bmatrix}, \quad \bar{\mathbf{x}}^2_k = \begin{bmatrix} \bar{\mathbf{x}}^{(2)}_k \\ \bar{\mathbf{x}}^{(1)}_k \end{bmatrix} \tag{3.84}$$

Relations (3.70-3.72) also hold but the distances to be minimised now become:

$$d_k(1) = \left|\mathbf{x}^1_{d,k} - \mathbf{z}^1_{d,k}\right|_2, \quad d_k(2) = \left|\mathbf{x}^2_{d,k} - \mathbf{z}^2_{d,k}\right|_2 \tag{3.85}$$

3.7.3 Adaptive multitarget LLPF

The adaptive MLLPF (A-MLLPF) is an efficient variation of the MLLPF in which the number of the particles used in the association phase varies. The A-MMLPF rather than predicting at every scan k all its N_s particles for all $n!$ measurement hypothesis, uses an association-difficulty metric and according to its value varies the particle number. The number $N^p_{s,k}$ of the predicted particles can be the same for all hypotheses or can be hypothesis-specific. For simplicity in our analysis we apply a single $N^p_{s,k}$ for all hypotheses and use the average separation s_k of the measurement hypotheses as the association-difficulty metric.

Consider having n targets and measurements at scan k and a set of $n!$ different measurement association hypotheses $\{\mathbf{z}_k^{\alpha}\}_{\alpha=1}^{n!}$ from (3.77). Before computing the separation metric, we first need to construct a $2 \times {}_{n!}\mathrm{C}_2$ *distance-indices* matrix $\mathbf{\Delta}$, where ${}_{n!}\mathrm{C}_2$ is the number of the two-element unordered combinations from the $n!$ hypotheses given by definition by:

$$ {}_{n!}\mathrm{C}_2 \equiv \frac{(n!)!}{2!(n!-2)!} \tag{3.86} $$

The matrix $\mathbf{\Delta}$ is formed from ${}_{n!}\mathrm{C}_2$ columns, each denoted Δ_β where $\beta = 1, \ldots, {}_{n!}\mathrm{C}_2$, each of which consists of one of the ${}_{n!}\mathrm{C}_2$ different two-element unordered combinations of the set $\{1, \ldots, n!\}$. For example for $n = 3$ we get ${}_{3!}\mathrm{C}_2 = 15$ and we can use the following $\mathbf{\Delta}$:

$$ \begin{array}{c} \begin{array}{cccccccccccccc} \Delta_1 & \Delta_2 & \ldots & & & & & & & & & & & \Delta_{15} \\ \downarrow & \downarrow & & & & & & & & & & & & \downarrow \end{array} \\ \mathbf{\Delta} = \left[\begin{array}{cccccccccccccc} 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 4 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 & 3 & 4 & 5 & 6 & 4 & 5 & 6 & 5 & 6 & 6 \end{array} \right] \end{array} \tag{3.87} $$

The average separation s_k of the measurement hypotheses is given by:

$$ s_k = \frac{1}{{}_{n!}\mathrm{C}_2} \sum_{\beta=1}^{{}_{n!}\mathrm{C}_2} \delta_k^{\beta} \tag{3.88} $$

in which δ_k^{β} is the separation between $\mathbf{z}_{d,k}^{\Delta_\beta(1)}$ and $\mathbf{z}_{d,k}^{\Delta_\beta(2)}$, given from (3.62) where $\Delta_\beta(i)$ is the $i\text{-}th$ element of the $\beta\text{-}th$ measurement combination. The δ_k^{β} is computed with the following:

$$ \delta_k^{\beta} = \left| \mathbf{z}_{d,k}^{\Delta_\beta(1)} - \mathbf{z}_{d,k}^{\Delta_\beta(2)} \right|_2 \tag{3.89} $$

For varying the number of the predicted particles according to the separation s_k we introduce the function $g(s_k)$. We want to increase the particle number in inverse relation with the separation, therefore we can choose:

$$ g(s_k) = \begin{cases} N_s^{p,max}, & \text{if } s_k < s^{low} \\ \dfrac{(N_s^{p,min} - N_s^{p,max})(s_k - s^{low})}{s^{hi} - s^{low}} + N_s^{hi}, & \text{if } s^{low} \leq s_k \leq s^{hi} \\ N_s^{p,min}, & \text{if } s_k > s^{hi} \end{cases} \tag{3.90} $$

where $N_s^{p,min}$, $N_s^{p,max}$, s^{low} and s^{hi} are respectively the minimum and maximum number of

the particles and the low and high limits of the separation (figure 3.10 shows an example of $g(s_k)$). Using $g(s_k)$ we finally set:

$$N_{s,k}^{p} = g(s_k) \tag{3.91}$$

Except from using an adaptive number of particles for prediction and association, the structure of the A-MLLPF algorithm remains as in the MLLPF: after computing $N_{s,k}^{p}$ with (3.91) the particles $\{\mathbf{x}_{k-1}^{i}\}_{i=1}^{N_{s,k}^{p}}$ are predicted using (3.75), the data association assigns the measurements to the targets by minimising (3.79), the remaining $N_s - N_{s,k}^{p}$ particles from the chosen hypotheses are predicted, the normalised weights $\{w_k^i\}_{i=1}^{N_s}$ (3.82-3.83) and the state estimate $\hat{\mathbf{x}}_k$ (3.67) are computed and finally the particles are resampled with the SR algorithm, assigning to each resampled particle its associated covariance.

3.8 Performance comparison

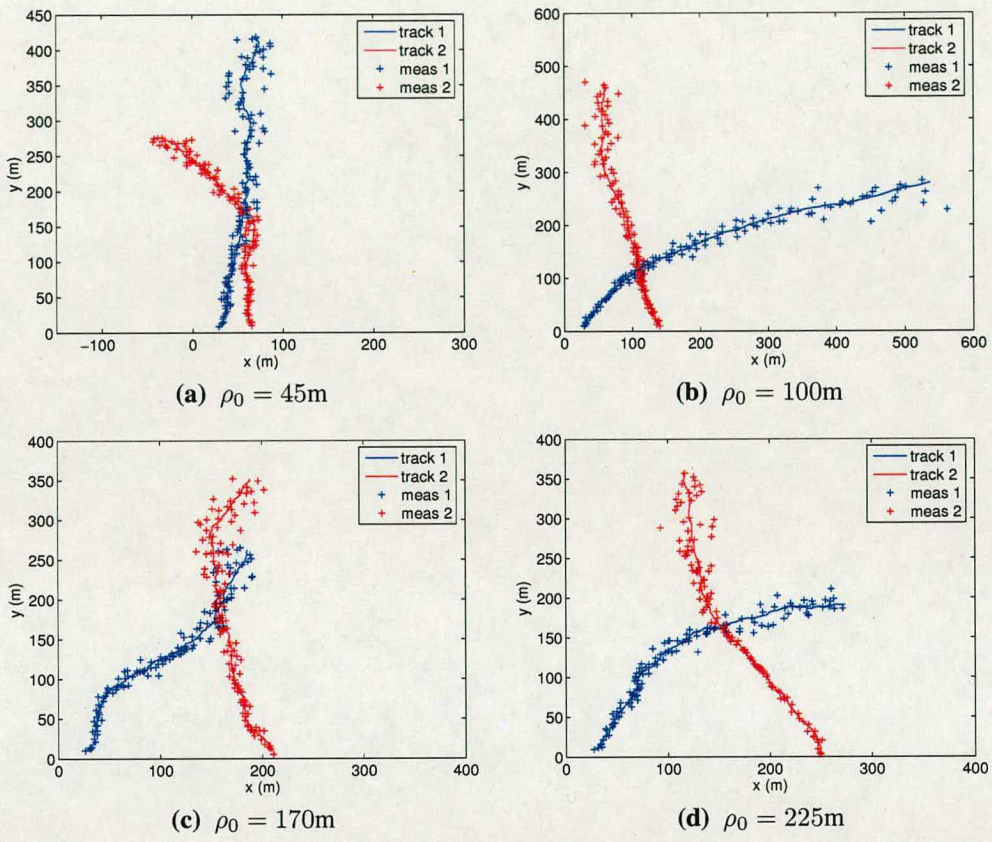
In this section we apply the MSIR, MLLPF and A-MLLPF in a multitarget problem and compare their performance in terms of both measurement association and positioning accuracy.

3.8.1 Simulation study

For comparison we use the environment presented in 3.6.2 in which two constant-velocity targets move across the x-y plane while a radar measures their azimuth angle and range. The targets are perturbed with random accelerations with standard deviation 1m/s^2 in both x and y moving directions. The radar lies at the origin of the plane at point (0,0) and has an update rate of 1sec. The standard deviation of the measurement noise is 4° for the angle and 1.2m for the range.

For the analysis we create 800 different crossing tracks, initialising the targets with the following states:

$$\mathbf{x}_0^{(1)} = \begin{bmatrix} x_0^{(1)} \\ y_0^{(1)} \\ \sigma_0 \cos\left(\arccos\left(\frac{\rho_0}{t_s\sigma_0}\right)\right) \\ \sigma_0 \sin\left(\arccos\left(\frac{\rho_0}{t_s\sigma_0}\right)\right) \end{bmatrix}, \quad \mathbf{x}_0^{(2)} = \begin{bmatrix} x_0^{(1)} + \rho_0 \\ y_0^{(1)} \\ -\sigma_0 \cos\left(\arccos\left(\frac{\rho_0}{t_s\sigma_0}\right)\right) \\ \sigma_0 \sin\left(\arccos\left(\frac{\rho_0}{t_s\sigma_0}\right)\right) \end{bmatrix} \tag{3.92}$$



(a) $\rho_0 = 45$m **(b)** $\rho_0 = 100$m

(c) $\rho_0 = 170$m **(d)** $\rho_0 = 225$m

Figure 3.9: *Representative examples of crossing tracks for different initial separations from the studied scenario.*

where σ_o is their initial speed (velocity magnitude), ρ_0 their initial separation and t_s the total simulation time. The separation ρ_0 varies uniformly between 10m and 250m throughout the 800 track sets (figure 3.9). In the absence of process noise, the initialisation with (3.92) would result the tracks to intersect at $t_s/2$, halfway through each run. Since in our system we consider process disturbances the intersection points vary randomly around $t_s/2$. We set $x_0^{(1)} = 30$m, $y_0^{(1)} = 10$m, $\sigma_0 = 3.16$m/s and $t_s = 120$s. We compare the number of disassociations[6], the track swaps and the RMS position error. Every particle filter employs 100 particles. For the $g(s_k)$ function of the A-MLLPF we also set $s^{low} = 20$m, $s^{hi} = 40$m, $N_s^{p,min} = 5$ and $N_s^{p,max} = 100$ (figure 3.10). The results below are obtained after performing 50MC runs for each of the 800 sets of tracks.

[6]The disassociations within runs in which the tracks are eventually swapped are not counted.

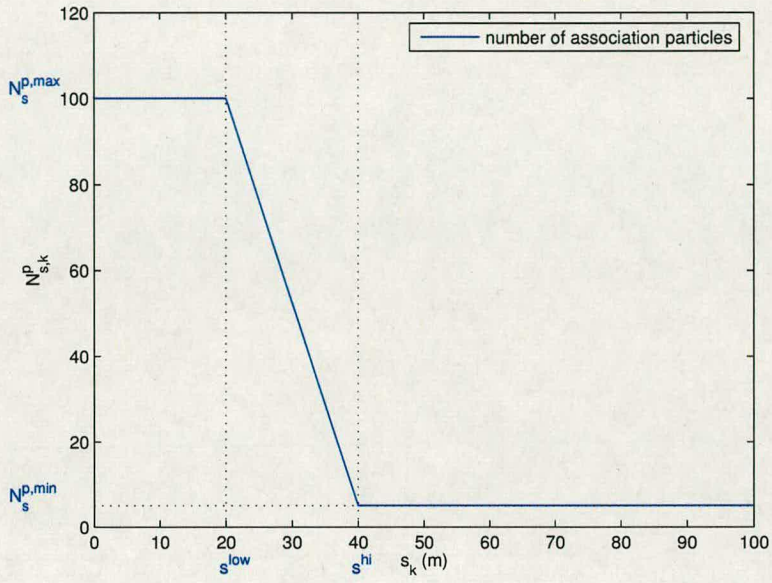


Figure 3.10: *The function $g(s_k)$ defining the number of association particles over the measurement separation from the studied scenario.*

Table 3.2 shows the performance of the algorithms. We witness that both MLLPF and A-MLLPF result in about 34% fewer disassociations than the MSIR and 11%-16% fewer track swaps. Regarding their tracking accuracy, they exhibit about 40% smaller position error. Comparing the A-MLLPF to the MLLPF we see that the performance degradation due to its varying particle mechanism is small, especially considering that the A-MLLPF used on average about 113 particles compared to the 200 of the standard MLLPF. Figure 3.11 shows how the number of the A-MLLPF particles and EKFs[7] varied according to the average s_k of every run. As expected in runs in which the average separation of the measurement was larger, A-MLLPF used less particles and EKFs.

3.8.2 Conclusions

This work presented a novel multitarget particle filter which uses a joint association and prediction phase and employs measurement-coupled local linearization techniques for predicting

[7] In general LLPF-based algorithms use EKFs just for the particles that have different states and covariance, which due to resampling are always less than N_s.

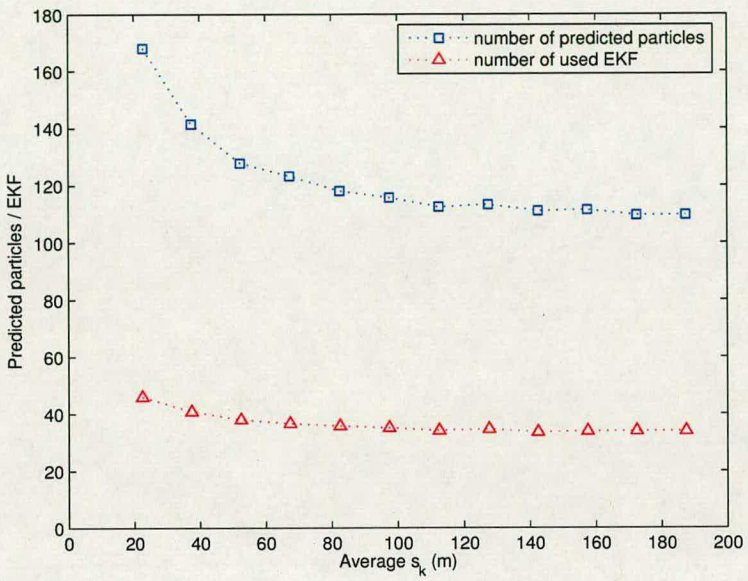


Figure 3.11: *Number of predicted particles and EKFs over the average s_k per run from the simulation study.*

its particles. Simulation results demonstrated its performance superiority in terms of association and tracking accuracy, compared with the equivalent algorithm which uses for prediction just the state prior pdf. An adaptive suboptimal variation was also introduced which varied the number of its predicted particles, and consequently its computational load, according to the difficulty of the measurements' association. The observed performance degradation of the adaptive variation was shown to be unimportant considering the computational savings resulting from the lighter particle usage.

	Disassociations (per run)	Track swaps	Position RMSE (m)	Predicted particles
MSIR	0.8060	5.9%	42.1578	100.0
MLLPF	0.5232	4.9%	25.4899	200.0
A-MLLPF	0.5456	5.2%	25.6264	113.4

Table 3.2: *Simulation results after 800 tracks × 50 MC runs.*

3.9 Chapter summary

Chapter 3 introduced particle filtering, presented several of its popular implementations and demonstrated experimentally its performance superiority compared to the extended Kalman filter in a difficult estimation problem. The auxiliary local linearization particle filter was furthermore introduced as a simple method to deal with dynamic changes in the target kinematics behaviour. Simulation studies demonstrated its robustness for maintaining the tracks of a dynamically manoeuvring target but also illustrated its weakness to filter efficiently the measurement noise when the target kinematics behaviour remained unvarying. Finally, the multitarget particle filtering problem was described and the novel adaptive multitarget local linearization particle filter was presented. Simulation results showed the improvement both in terms of data association and tracking when using the proposed approach in a scenario with two crossing targets.

Chapter 4
Vehicle tracking using road maps and particle filtering

This chapter is concentrated on vehicle tracking focusing on algorithms which exploit a-priori information about the road structure for constraining accordingly the motion of the vehicles. After a short introduction on ground tracking, we describe the single-target variable structure multiple model particle filter and present a mechanism which allows the number of its particles to vary according to the position of the vehicle on the road map, using specifically fewer particles for the constrained road parts. A simulation study demonstrates that the slight performance degradation can be considered negligible considering the significantly lighter particle usage. The remaining chapter incorporates into the algorithm gating and joint probabilistic data association features, to enable it to reject clutter and track simultaneously multiple vehicles. We analyse experimentally its tracking performance and we show that in difficult association environments both the estimation accuracy and association capability are improved compared with the equivalent un-constrained sequential importance resampling filter which employs the same association logic.

4.1 Introduction

Vehicle tracking recently has drawn considerable attention from the scientific community, which studied it extensively in a wide range of applications including highway tracking, traffic control, navigation, accident avoidance and joint classification and tracking [71–75]. This increasing interest was not only due to the growing importance of the problem itself but also due to its difficulty and complexity which make it ideal for comparing and benchmarking tracking techniques. Vehicle tracking is demanding since one often encounters physical constraints and obstructions, terrain-coupled vehicle motion, intense clutter returns and false alarm rates and closely separated slow targets that can execute abrupt turns and can even stop.

Throughout the literature many different sensors have been used for the specific problem such

as electro-optical and video [75, 76], infrared [77], GPS [78], high-range resolution radar [79], space-time adaptive processing radar [80] and ground moving target indicator (GMTI) radar [81, 82]. In our work we use two-dimensional measurements from static radars which measure the azimuth angle and the range of the ground targets. We are interested in applications in which the vehicles can move freely on and off the road. For tracking we use particle filters which employ multiple modes of operation, accounting for the different tracking subspaces and their associated dynamics. Road map information, in the form of motion constraints, is exploited for improving the estimation accuracy.

Our work in this chapter is based on the single-target variable-structure multiple model particle filter (VSMMPF) [46, 82] vehicle tracker. The VSMMPF incorporated into particle filtering the *variable-structure* approach of the variable-structure interacting multiple model (VSIMM) algorithm [83, 84]. The VSIMM aimed to address a weakness of the interacting multiple model (IMM) filter [85, 86] which in certain applications exhibited a degraded performance due to the excessive "competition" among its models [87]. The VSIMM therefore proposed to use a *varying* number of active models according to the vehicle positioning on the road map; approach which indeed enhanced the tracking accuracy and moreover reduced the computational load. The VSMMPF demonstrated an even greater performance compared to the VSIMM since the more "open" particle filtering architecture enabled it to cope better and more efficiently with the intense nonlinearity and non-Gaussianity inherent in the vehicle tracking problem.

In what follows we propose a mechanism to vary the number of the VSMMPF particles used in the prediction phased, depending on their position on the road map for reducing the computational demands. We then enhance the VSMMPF with a gating and joint probabilistic data association (JPDA) logic to generalise its use for clutter rejection and multiple-vehicle tracking.

4.2 Road-constrained vehicle modelling

For tracking we use the analysis from sections 2.5.1 and 2.5.3 modified accordingly for the two dimensional case. For convenience, we present briefly the state-space representation of the problem[1]. Considering the states as the position and velocity on the x-y plane, $\mathbf{x}_k = [x_k \;\; y_k \;\; \dot{x}_k \;\; \dot{y}_k]^T$, the measurements as the azimuth angle and range, $\mathbf{z}_k = [\theta_k \;\; r_k]^T$, and by dropping subscript k from $\mathbf{F}_k, \mathbf{G}_k$ and $\mathbf{h}_k$ for notational simplicity, we obtain the following

[1] Chapters 2 and 3 provide details for the notation used.

state-transition and measurement equations:

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{G}\mathbf{u}_{k-1} \tag{4.1}$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k \tag{4.2}$$

where:

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} T^2/2 & 0 \\ 0 & T^2/2 \\ T & 0 \\ 0 & T \end{bmatrix}, \quad \mathbf{h}(\mathbf{x}_k) = \begin{bmatrix} \arctan(y_k/x_k) \\ \sqrt{x_k^2 + y_k^2} \end{bmatrix} \tag{4.3}$$

and $\mathbf{u}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$ and $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$ are the process and measurement noise vectors, with $\mathbf{Q}_k$ and $\mathbf{R}_k$ the noise covariance matrices.

Generally in ground tracking we assume that some features on the ground scene of interest force locally the vehicle to move under specific patterns. Some of the features (like bridges and lakes [88]) impose hard constraints on the vehicle movement, whereas other (roads in our study) impose soft constraints. The objective in this class of problems is to incorporate *efficiently* a-priori information from these features into the tracking algorithm.

In this work we assume that a vehicle travels on a terrain with a known road structure, having the ability to move on and off the road. The roads impose probabilistic constraints on the movement of the vehicle, which implies that when the vehicle is on-road the state uncertainty is larger along the road than orthogonal to it. We model this by setting the variance $\sigma\{u_{\alpha,k}\}^2$ of the process noise along the road larger than the variance $\sigma\{u_{o,k}\}^2$ orthogonal to it. The direction of the on-road noise depends on the direction of the road, therefore the associated process noise covariance $\mathbf{Q}_k$ is rotated using the following relation:

$$\mathbf{Q}_{on,k}(\psi) = \mathbf{\Omega}_\psi \begin{bmatrix} \sigma\{u_{o,k}\}^2 & 0 \\ 0 & \sigma\{u_{\alpha,k}\}^2 \end{bmatrix} \mathbf{\Omega}_\psi^T \tag{4.4}$$

where $\mathbf{\Omega}_\psi$ is the rotational transformation matrix and ψ is the angle of the road measured clockwise from the y-axis:

$$\mathbf{\Omega}_\psi = \begin{bmatrix} -\cos\psi & \sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \tag{4.5}$$

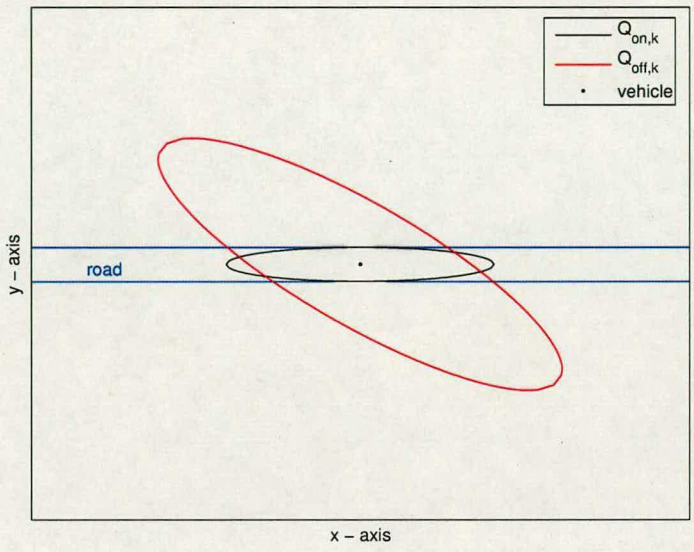


Figure 4.1: *The road is exploited to constrain probabilistically the state uncertainty.*

For off-road motion since the vehicle travels unconstrained, we use the same process noise variances for both x-y axes, $\sigma\{u_{x,k}\}^2 = \sigma\{u_{y,k}\}^2$; the covariance thus becomes:

$$\mathbf{Q}_{off,k} = \begin{bmatrix} \sigma\{u_{x,k}\}^2 & 0 \\ 0 & \sigma\{u_{y,k}\}^2 \end{bmatrix} \tag{4.6}$$

Figure 4.1 demonstrates graphically the "directional process noise" principle for an on-road vehicle. We present a vertical section of the Gaussian distribution of the vehicle process noise when constrained (ellipse $\mathbf{Q}_{on,k}$) and when not (ellipse $\mathbf{Q}_{off,k}$). From the figure it is evident that by using the road map to tighten $\mathbf{Q}_{on,k}$, we can improve consequently the estimation accuracy.

4.3 Variable-structure multiple model particle filter

For notational purposes we define $\mathcal{R}_s$ as the set of the roads r on the ground scene of interest (e.g. figure 4.2) and use the convention $r = 0$ for off-road motion. Consider the VSMMPF

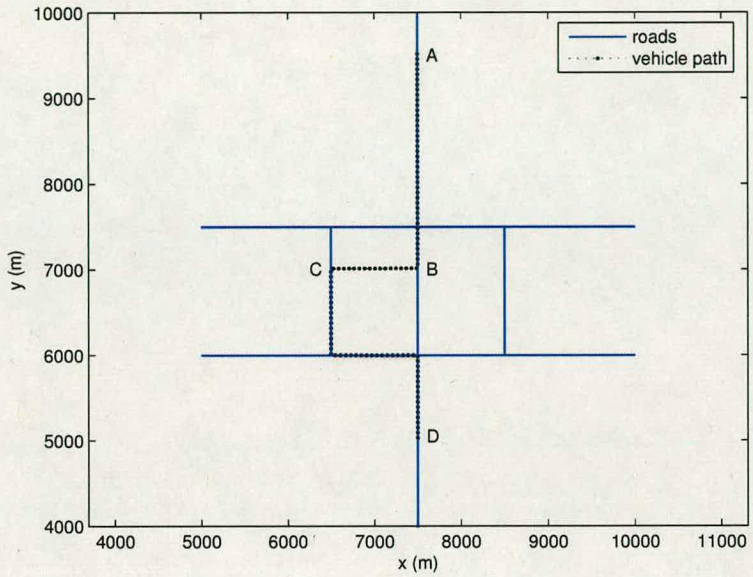


Figure 4.2: *The road map and the vehicle path from the study in 4.5.1. The vehicle travels on the road on segment A-B, it continues off the road during B-C and returns on the road for the final C-D part of its motion.*

using N_s particles $\{\mathbf{x}_k^i\}_{i=1}^{N_s}$, each associated with a mode M_k^i according to the following:

$$M_k^i = \begin{cases} r, & \text{if particle } \mathbf{x}_k^i \text{ is on the road } r, \text{ where } r \in \mathcal{R}_s \\ 0, & \text{if particle } \mathbf{x}_k^i \text{ is off-road} \end{cases} \tag{4.7}$$

The algorithm[2] consists of a time prediction and a measurement update step. For the prediction step consider a particle $\mathbf{x}_{k-1}^i$ with mode M_{k-1}^i at time $k-1$. Initially we perform a preliminary prediction of the state for time k using:

$$\mathbf{x}_k^{i-} = \mathbf{F}\mathbf{x}_{k-1}^i \tag{4.8}$$

We continue according to the mode M_{k-1}^i of the particle $\mathbf{x}_{k-1}^i$. The first case is when the mode

[2]The original VSMMPF considers as well vehicle motion through bridges and tunnels, features that for the sake of simplicity and clarity we do not consider in our analysis. For including these, we simply impose hard constraints when the vehicle travels on a bridge or tunnel. Specifically for the latter since no measurements are available we randomly predict the particles just according to the vehicle dynamics [82].

corresponds to off-road motion (i.e. $M_{k-1}^i = 0$). We introduce the following binary function:

$$c(\mathbf{x}_{k-1}^i, r) = \begin{cases} 1, & \text{if } \mathbf{x}_{k-1}^i \rightarrow \mathbf{x}_k^{i-} \text{ crosses road } r \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.9}$$

The transition probability, denoted as $p_{M_{k-1}^i \rightarrow M_k^i}$, for this case is given by:

$$p_{0 \rightarrow r}(\mathbf{x}_{k-1}^i) = \begin{cases} p^\star, & \text{if } c(\mathbf{x}_{k-1}^i, r) = 1 \\ 0, & \text{if } c(\mathbf{x}_{k-1}^i, r) = 0,\ d(\mathbf{x}_k^{i-}, r) > \tau \\ \dfrac{p^\star(\tau - d(\mathbf{x}_k^{i-}, r))}{\tau}, & \text{otherwise} \end{cases} \tag{4.10}$$

where $p^\star$ is the user-defined probability that the vehicle enters a road when crossing it, $d(\mathbf{x}_k^{i-}, r)$ is the shortest distance from particle $\mathbf{x}_k^{i-}$ to the road r, and τ is a user defined threshold according to the acceleration capabilities of the vehicle. The probability that the particle will remain off-road is:

$$p_{0 \rightarrow 0}(\mathbf{x}_{k-1}^i) = 1 - \sum_{r=1}^{\mathcal{R}_s} p_{0 \rightarrow r}(\mathbf{x}_{k-1}^i) \tag{4.11}$$

The mode $M_k^{i\star}$ is randomly drawn according to its associated transition probabilities:

$$P\{M_k^{i\star} = r\} = p_{M_{k-1}^i \rightarrow r}\Big|_{r \in \{0, R_s\}} \tag{4.12}$$

If $M_k^{i\star} = 0$ the mode implies that the particle stays off the road and therefore we propagate it simply by using the state transition equation:

$$\mathbf{x}_k^{i\star} = \mathbf{F}\mathbf{x}_{k-1}^i + \mathbf{G}\mathbf{u}_{k-1}^i \tag{4.13}$$

If $M_k^{i\star} \neq 0$ the particle is positioned on-road and its velocity is rotated using the rotation matrix (4.5) randomly towards one road direction. As we saw in the previous section, the process noise vector for both cases is given by:

$$\mathbf{u}_{k-1}^i \sim \begin{cases} \mathcal{N}(\mathbf{0}, \mathbf{Q}_{on,k}(\psi)), & \text{if } M_k^{i\star} \in \mathcal{R}_s \\ \mathcal{N}(\mathbf{0}, \mathbf{Q}_{off,k}), & \text{if } M_k^{i\star} = 0 \end{cases} \tag{4.14}$$

The second case is when M_{k-1}^i indicates on-road motion. We define $\mathcal{J}_s$ as the set of the

junctions j of the road map. We introduce the following binary function:

$$\bar{c}(\mathbf{x}_{k-1}^i, j) = \begin{cases} 1, & \text{if } \mathbf{x}_{k-1}^i \rightarrow \mathbf{x}_k^{i-} \text{ crosses junction } j \\ 0, & \text{otherwise} \end{cases} \tag{4.15}$$

The mode transition probability that the particle will continue on the same road r is:

$$p_{r \rightarrow r}(\mathbf{x}_{k-1}^i) = \begin{cases} \bar{p}, & \text{if } \bar{c}(\mathbf{x}_{k-1}^i, j) = 0, \forall j \in \mathcal{J}_s \\ 0, & \text{if } \bar{c}(\mathbf{x}_{k-1}^i, j) = 1, \text{ for some } j \in \mathcal{J}_s \end{cases} \tag{4.16}$$

where $\bar{p}$ is the user defined probability that the vehicle will remain on-road. The probability to change from road r to r' after crossing a junction is:

$$p_{r \rightarrow r'}(\mathbf{x}_{k-1}^i) = \begin{cases} \bar{p}/n_j, & \text{if } \bar{c}(\mathbf{x}_{k-1}^i, j) = 1, \text{ for some } j \in \mathcal{J} \\ 0, & \text{if } \bar{c}(\mathbf{x}_{k-1}^i, j) = 0, \forall j \in \mathcal{J} \end{cases} \tag{4.17}$$

where n_j is the number of road segments after junction j. Finally, the probability that the particle will go off-road is given by:

$$p_{r \rightarrow 0}(\mathbf{x}_{k-1}^i) = 1 - \bar{p} \tag{4.18}$$

As in the off-road case we draw randomly the mode $M_k^{i\star}$ according to the transition probabilities calculated above. There are three possibilities now for the particle $\mathbf{x}_{k-1}^i$: going off-road, remaining on the same road or crossing a junction. If it goes off-road we first rotate its velocity to be perpendicular to the road (with equal probabilities for left or right turn), and then apply equations (4.13-4.14). If the particle continues on the same road we just use (4.13-4.14) to propagate it. If the particle crosses a junction and moves to road r', we first apply (4.13-4.14) and denote the resulting particle as $\mathbf{x}_k^{i+}$. Then we use the following equation to rotate it and position it on the the new road segment:

$$\mathbf{x}_k^{i\star} = \mathbf{\Omega}_\phi^j [\mathbf{x}_k^{i+} - \mathbf{x}^j] + \mathbf{x}^j \tag{4.19}$$

where $\mathbf{\Omega}_\phi^j$ is the rotational matrix that rotates the state vector towards the road that the particle will follow, and $\mathbf{x}^j = [x^j \; y^j \; 0 \; 0]'$ where x^j and y^j are the Cartesian coordinates of junction j.

After predicting all the particles, the measurement update phase starts by assigning a weight

$\tilde{w}_k^i$ to each particle $\mathbf{x}_k^{i\star}$ according to its likelihood:

$$\tilde{w}_k^i = p(\mathbf{z}_k|\mathbf{x}_k^{i\star}) = \mathcal{N}(\mathbf{z}_k; \mathbf{h}(\mathbf{x}_k^{i\star}), \mathbf{R}_k) \tag{4.20}$$

The weights then are normalised:

$$w_k^i = \frac{\tilde{w}_k^i}{\sum_{j=1}^{N_s} \tilde{w}_k^j} \tag{4.21}$$

and the state estimate is computed from the weighted sum of the particles:

$$\hat{\mathbf{x}}_k = \sum_{i=1}^{N_s} w_k^i \mathbf{x}_k^{i\star} \tag{4.22}$$

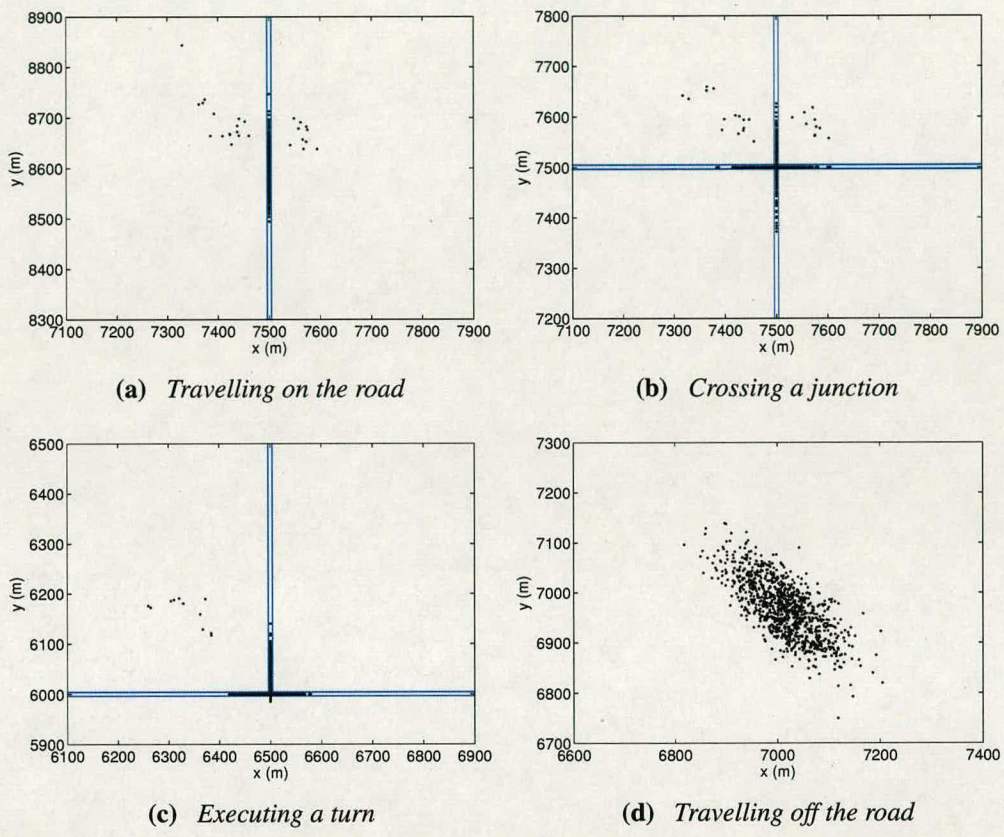
The final step is to resample N_s times with replacement from the weighted set $\{\mathbf{x}_k^{i\star}, M_k^{i\star}\}_{i=1}^{N_f}$ to eliminate particles with small weights using the SR algorithm from chapter 3.

Figure 4.3 shows the particle cloud when the vehicle travels across different areas in the tracking space. We see how the soft constraints imposed from the road structure reduce the variance of the particles and decrease consequently the state uncertainty. However, the extremely dense particle cloud that is formed on the road suggests particle *redundancy*; a problem which we address with the varying-particle mechanism introduced next.

4.4 Varying particle VSMMPF

The varying particle VSMMPF (VP-VSMMPF) is a variation of the VSMMPF in which the number of the particles within its prediction phase varies. The key idea is to use fewer particles in "easy" state subspaces so as to reduce the computational cost. Vehicle road tracking is an ideal problem to demonstrate this approach, since the tracking space includes constrained areas (roads) with smaller uncertainty and thus easier (than the unconstrained off-road) dynamics. In other words, we vary the number of the particles in relation with the breadth of the posterior pdf so as the resolution of the latter in specific state-subspaces not to exceed a certain limit, after which the estimation performance asymptotically converges and the additional particles become redundant. In what follows we present the varying-particle mechanism which we introduced in [89] in 2005.

Consider that the VP-VSMMPF employs nominally N_s off-road and N_{on} on-road particles.



(a) *Travelling on the road*

(b) *Crossing a junction*

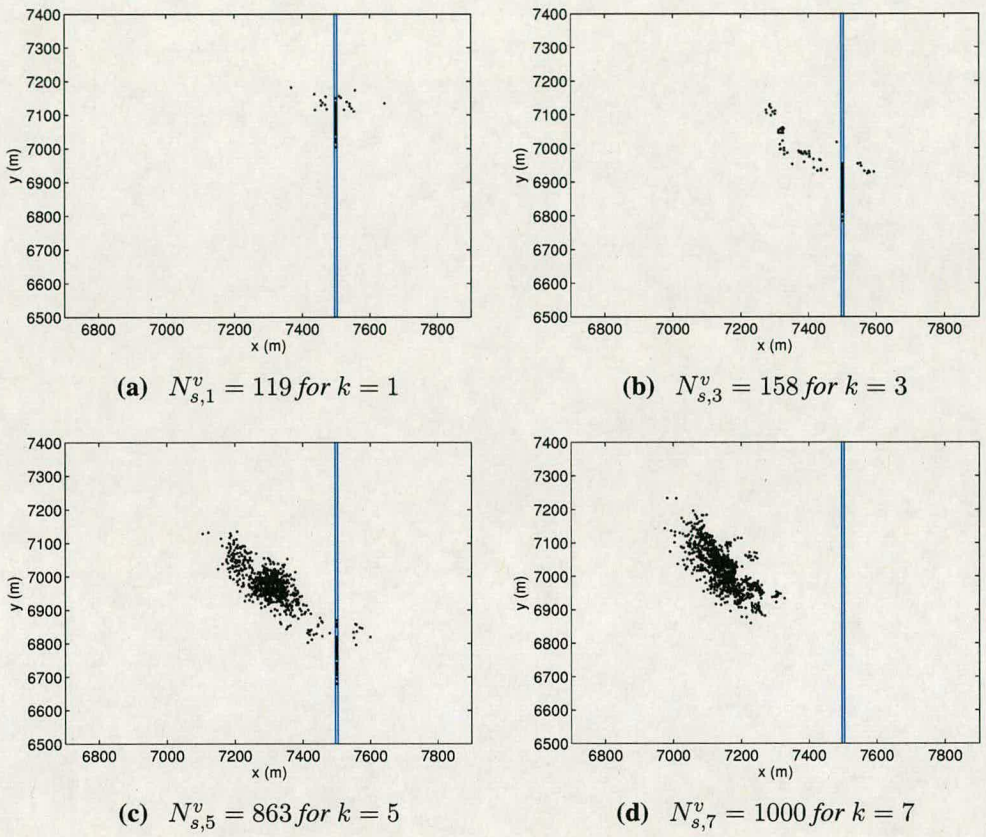
(c) *Executing a turn*

(d) *Travelling off the road*

Figure 4.3: *The particle cloud of the VSMMPF in different modes of operation* ($N_s = 1000$).

The algorithm is based on the particle propagation methodology of the VSMMPF, with the main difference that it varies the number of its particles so as to use always N_{on} particles on the road. A minor difference also is that its on-road particles lie always in the middle of the road with velocities parallel to the road direction. This results in an easier implementation and furthermore non-trivial computational savings, since now the tracking space is reduced just along the line in the middle of the road. With this approach we do not expect significant (if any) performance degradation, since the information from modelling the movement orthogonal to the road has negligible importance since the targets are usually distant.

Algorithm 7 presents an outline of the structure of the VP-VSMMPF. At every time index k, the algorithm starts with N_s particles. In the first *downsampling* step it checks whether more than N_{on} particles lie on the road and if there are it randomly keeps N_{on} of them. It predicts



(a) $N^v_{s,1} = 119$ *for* $k = 1$

(b) $N^v_{s,3} = 158$ *for* $k = 3$

(c) $N^v_{s,5} = 863$ *for* $k = 5$

(d) $N^v_{s,7} = 1000$ *for* $k = 7$

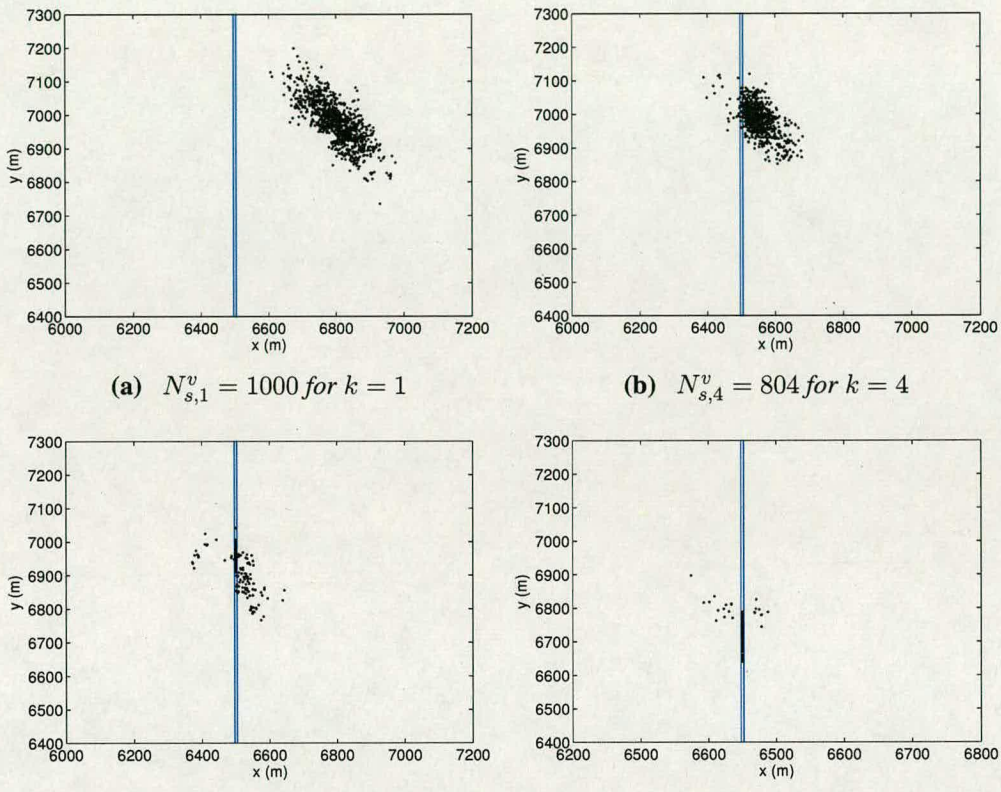
Figure 4.4: *The number of VP-VSMMPF particles* $N^v_{s,k}$ *increases in off-road transitions* ($N_s = 1000$, $N_{on} = 100$).

the remaining $N^v_{s,k}$ particles using the VSMMPF method and then, if appropriate, generates in the *upsampling* phase randomly $N_s - N^v_{s,k}$ particles from the ones that were predicted on-road in the downsampling step. It then resamples the N_s particles with the systematic resampling algorithm[3] and finally computes the state estimate.

The downsampling and upsampling pseudo-codes are given as algorithms 8 and 9. In the beginning of the downsamping method, we propagate all particles using:

$$\tilde{\mathbf{x}}^{i-}_k = \mathbf{F}\mathbf{x}^i_{k-1} \tag{4.23}$$

[3]Rather than using upsampling and systematic resampling one could "upscale" the on-road particles and use a resampling algorithm which could vary its sample size so as to result always with N_s particles (like in chapter 5).



(a) $N^v_{s,1} = 1000$ *for* $k = 1$

(b) $N^v_{s,4} = 804$ *for* $k = 4$

(c) $N^v_{s,6} = 202$ *for* $k = 6$

(d) $N^v_{s,8} = 120$ *for* $k = 8$

Figure 4.5: *The number of VP-VSMMPF particles* $N^v_{s,k}$ *decreases in on-road transitions* ($N_s = 1000$, $N_{on} = 100$).

and we find their preliminary modes according to (4.7). We then compute the number of the on-road particles $N^{on}_{s,k}$, and if they are more than N_{on}, we calculate the percentage $p_{N^{on}_{s,k}}$:

$$p_{N^{on}_{s,k}} = \frac{N_{on} + N^{on}_{s,k}(1 - \bar{p})}{N^{on}_{s,k}} \tag{4.24}$$

of the ones that we randomly keep. After downsamping, we calculate the resulting number of particles $N^v_{s,k}$ and denote as $\{\mathbf{x}^{i-}_k\}_{i=1}^{N^v_{s,k}}$ the particles of this phase. We apply then the VSMMPF

Algorithm 7 VP-VSMMPF

$[\{\mathbf{x}_k^i, M_k^i\}_{i=1}^{N_s}] = \text{VP-VSMMPF}[\{\mathbf{x}_{k-1}^i, M_{k-1}^i\}_{i=1}^{N_s}]$

1: $[\{\mathbf{x}_k^{i-}, M_k^{i-}\}_{i=1}^{N_{s,k}^v}, N_{s,k}^{on}, N_{s,k}^v] = \text{Downsampling}[\{\mathbf{x}_{k-1}^i\}_{i=1}^{N_s}]$
2: $[\{\mathbf{x}_k'^{i\star}, w_k'^{i\star}, M_k'^{i\star}\}_{i=1}^{N_{s,k}^v}] = \text{Prediction}[\{\mathbf{x}_{k-1}^i, M_{k-1}^i\}_{i=1}^{N_{s,k}^v}]$
3: $[\{\mathbf{x}_k^{i\star}, w_k^{i\star}, M_k^{i\star}\}_{i=1}^{N_s}] = \text{Upsampling}[\{\mathbf{x}_k'^{i\star}, w_k'^{i\star}, M_k'^{i\star}, M_k^{i-}\}_{i=1}^{N_{s,k}^v}, N_{s,k}^{on}, N_{s,k}^v]$
4: $[\{\mathbf{x}_k^i, w_k^i, j^i\}_{i=1}^{N_s}] = \text{SR}[\{\mathbf{x}_k^{i\star}, w_k^{i\star}\}_{i=1}^{N_s}]$
5: Assign M_k^i to each resampled particle $\mathbf{x}_k^i$ using j^i
6: $\hat{\mathbf{x}}_k = \sum_{i=1}^{N_s} w_k^i \mathbf{x}_k^i$

Algorithm 8 Downsampling

$[\{\mathbf{x}_k^{i-}, M_k^{i-}\}_{i=1}^{N_{s,k}^v}, N_{s,k}^{on}, N_{s,k}^v] = \text{Downsampling}[\{\mathbf{x}_{k-1}^i\}_{i=1}^{N_s}]$

for $i = 1 : N_s$ **do**
  Propagate particle: $\tilde{\mathbf{x}}_k^{i-} = \mathbf{F}\mathbf{x}_{k-1}^i$
  Find preliminary particle mode $\tilde{M}_k^{i-}$ using (4.7)
end for
Calculate the number of the on-road particles $N_{s,k}^{on}$ (when $\tilde{M}_k^{i-} \neq 0,\ \forall i$)
if $N_{s,k}^{on} > N_{on}$ **then**
  Calculate: $p_{N_{s,k}^{on}} = (N_{on} + N_{s,k}^{on}(1 - \bar{p}))/N_{s,k}^{on}$
  Initialise index: $j = 1$
  for $i = 1 : N_s$ **do**
    if $\tilde{M}_k^{i-} \neq 0$ **then**
      Draw: $u \sim \mathcal{U}[0, 1]$
      if $u < p_{N_{s,k}^{on}}$ **then**
        Set: $\mathbf{x}_k^{j-} = \tilde{\mathbf{x}}_k^{i-}$
        Set: $M_k^{j-} = \tilde{M}_k^{i-}$
        Increase index: $j = j + 1$
      end if
    else
      Set: $\mathbf{x}_k^{j-} = \tilde{\mathbf{x}}_k^{i-}$
      Set: $M_k^{j-} = \tilde{M}_k^{i-}$
      Increase index: $j = j + 1$
    end if
  end for
  Set number of particles $N_{s,k}^v = j - 1$
else
  Set: $\{\mathbf{x}_k^{i-}, M_k^{i-}\}_{i=1}^{N_s} = \{\tilde{\mathbf{x}}_k^{i-}, \tilde{M}_k^{i-}\}_{i=1}^{N_s}$
  Set number of particles: $N_{s,k}^v = N_s$
end if

Algorithm 9 Upsampling

$[\{\mathbf{x}_k^{i\star}, w_k^{i\star}, M_k^{i\star}\}_{i=1}^{N_s}] = \text{Upsampling}[\{\mathbf{x}_k^{\prime i\star}, w_k^{\prime i\star}, M_k^{\prime i\star}, M_k^{i-}\}_{i=1}^{N_{s,k}^v}, N_{s,k}^{on}, N_{s,k}^v]$

1: **if** $N_{s,k}^{on} > N_{on}$ **then**
2: Construct $\{\mathbf{x}_k^{i\star on}, w_k^{i\star on}, M_k^{i\star on}\}_{i=1}^{N_{s,k}^{on}}$ from $\{\mathbf{x}_k^{\prime i\star}, w_k^{\prime i\star}, M_k^{\prime i\star}\}_{i=1}^{N_{s,k}^v}$ with $M_k^{i-}, M_k^{\prime i\star} \neq 0$
3: **for** $j = 1 : N_s - N_{s,k}^v$ **do**
4: Draw: $i \sim \mathcal{U}[1, N_{s,k}^{on}]$
5: Set: $\mathbf{x}_k^{\prime\prime j\star} = \mathbf{x}_k^{i\star on}$
6: Set: $w_k^{\prime\prime j\star} = w_k^{i\star on}$
7: Set: $M_k^{\prime\prime j\star} = M_k^{i\star on}$
8: **end for**
9: **end if**
10: Construct: $\{\mathbf{x}_k^{i\star}, w_k^{i\star}, M_k^{i\star}\}_{i=1}^{N_s} = \left\{\{\mathbf{x}_k^{\prime i\star}, w_k^{\prime i\star}, M_k^{\prime i\star}\}_{i=1}^{N_{s,k}^v}, \{\mathbf{x}_k^{\prime\prime i\star}, w_k^{\prime\prime i\star}, M_k^{\prime\prime i\star}\}_{i=1}^{N_s - N_{s,k}^v}\right\}$

equations (4.9-4.21)[4] for prediction and weighting and obtain $\{\mathbf{x}_k^{\prime i\star}, w_k^{\prime i\star}, M_k^{\prime i\star}\}_{i=1}^{N_{s,k}^v}$, where $\mathbf{x}_k^{\prime i\star}$ stands for the predicted particle, $w_k^{\prime i\star}$ its weight and $M_k^{\prime i\star}$ its mode. The upsampling follows, in which we replicate randomly $N_s - N_{s,k}^v$ particles from the predicted particles which were lying at $k-1$ and k on the road (i.e. whose $M_k^{i-}, M_k^{\prime i\star} \neq 0$). The resulting particle set is denoted as $\{\mathbf{x}_k^{\prime\prime i\star}, w_k^{\prime\prime i\star}, M_k^{\prime\prime i\star}\}_{i=1}^{N_s - N_{s,k}^v}$. The final augmented set becomes:

$$\{\mathbf{x}_k^{i\star}, w_k^{i\star}, M_k^{i\star}\}_{i=1}^{N_s} = \left\{\{\mathbf{x}_k^{\prime i\star}, w_k^{\prime i\star}, M_k^{\prime i\star}\}_{i=1}^{N_{s,k}^v}, \{\mathbf{x}_k^{\prime\prime i\star}, w_k^{\prime\prime i\star}, M_k^{\prime\prime i\star}\}_{i=1}^{N_s - N_{s,k}^v}\right\} \tag{4.26}$$

The systematic resampling algorithm follows, after which we assign the modes to the resampled particles using the $\{j^i\}_{i=1}^{N_s}$ indices. The state estimate is finally obtained as the weighted sum of the particles:

$$\hat{\mathbf{x}}_k = \sum_{i=1}^{N_s} w_k^i \mathbf{x}_k^i \tag{4.27}$$

Figures 4.4 and 4.5 show two representative examples of the particle cloud during off-road and on-road transitions. We see how the algorithm just in few time steps varied adaptively the number of its particles between $N_{on} = 100$ and $N_s = 1000$ depending on the vehicle position. We present next a simulation study testing the performance of the proposed mechanism.

[4] When for a k it holds that $N_{s,k}^{on} > N_{on}$, we use $\bar{p}_{on}$ as the probability that the vehicle remains on the road:

$$\bar{p}_{on} = 1 - \frac{N_s(1-\bar{p})}{N_{on} + N_s(1-\bar{p})} \tag{4.25}$$

so as to propagate off-road the same number of on-road particles as if $N_{s,k}^v = N_s$.

4.5 Performance comparison

In this section we compare the tracking performance of the VP-VSMMPF with the standard VSMMPF. We use the road structure and the vehicle path shown in figure 4.2. For tracking we use a VP-VSMMPF switching between 100 and 1000 particles and two VSMMPF one employing 100 and the other 1000 particles[5].

4.5.1 Simulation study

From figure 4.2 we see that the vehicle travels on the road on segment A-B, it continues off he road during B-C and returns on the road for the final C-D part of its motion. Its velocity is constant 12m/s. The width of the roads is 8m. For sensing we use a static radar lying at the origin of the plane at point (0,0), measuring the azimuth angle and the range of the vehicle. Its angular accuracy is 0.5^o, its range resolution 20m and its measurement update rate $T = 5$s. For a fair comparison we use the same parameters as in the original VSMMPF paper [53] and therefore $\sigma\{u_{x,k}\} = \sigma\{u_{y,k}\} = \sigma\{u_{\alpha,k}\} = 0.6\text{m/s}^2$, $\sigma\{u_{o,k}\} = 0.0001\text{m/s}^2$, $\bar{p} = p^\star = 0.98$, $\tau = 18.75$m and $N_{on} = 100$. All algorithms were initialised with the true target states. The results were obtained after 100 Monte Carlo runs, each consisting of 109 time scans.

Figure 4.6 presents the RMS position error of the three trackers, from which we observe that the tracking performance of the VP-VSMMPF and the VSMMPF-1000 is similar. In fact from table 4.1 we obtain that the VP-VSMMPF results in just a 2% on-road and 0.7% off-road error increase, but while using on average respectively just 187.79 and 807.38 particles. Compared to the VSMMPF-100, the VP-VSMMPF error is smaller 13% on-road and 40% off-road.

[5]For convenience we call these implementations VSMMPF-100 and VSMMPF-1000

	Average particles on-road	Average particles off-road	RMS position error on-road (m)	RMS position error off-road (m)
VP-VSMMPF	187.79	807.38	27.77	63.76
VSMMPF-1000	1000.00	1000.00	27.13	63.30
VSMMPF-100	100.00	100.00	31.82	105.16

Table 4.1: *The average RMS position error of the trackers during the on-road and off-road parts of the vehicle path.*

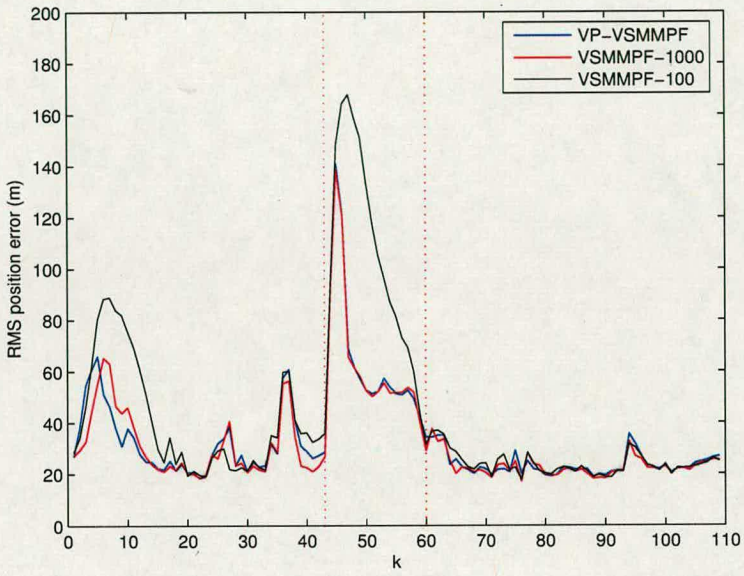


Figure 4.6: *The RMS position error of the three trackers over the scan number k. The area between the red dotted lines indicates off-road motion.*

We note here that the difference in the on-road performance between the VSMMPF-100 and the VSMMPF-1000/VP-VSMMPF in terms of the RMS position error, is the result of the worse VSMMPF-100 initialisation phase (up to $k = 14$). If we exclude this phase from our calculations for the on-road error we see that all algorithms perform similarly (VSMMPF-100: 25.94m, VP-VSMMPF: 25.72m, VSMMPF-1000: 24.75m). This justifies our main thesis for the particle redundancy of the on-road propagation mechanism of the original VSMMPF.

Figure 4.7 illustrates how the VP-VSMMPF varies the number of its active particles in the studied scenario. The algorithm is initialised with $N_s = 1000$ particles and after the first scans when identifies on-road motion it reduces them to about 150. A peak formed around $k = 27$ is the result of the junction crossing. Soon after the vehicle goes off-road at $k = 43$, the particles rapidly increase to 1000. Their number decreases again to about 150 at $k = 60$ when the vehicle enters the road and remains roughly at that value, except at $k = 77$ where a peak is formed due to the abrupt 90° left turn.

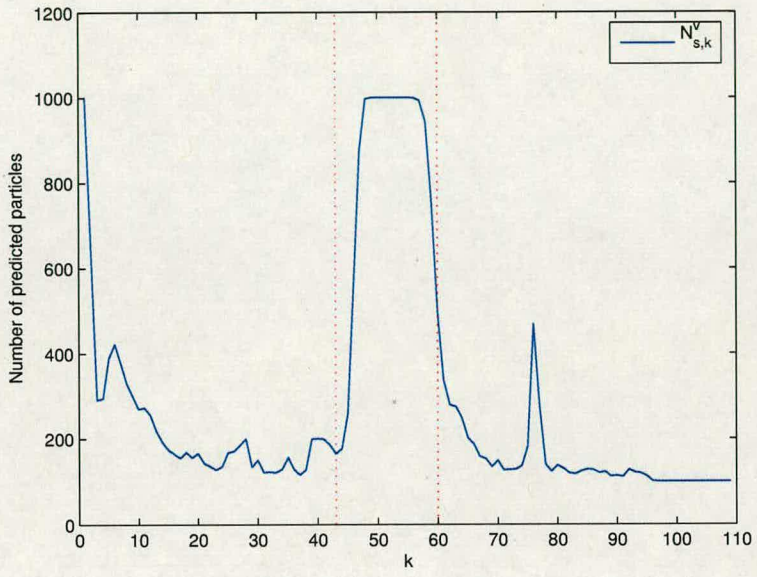


Figure 4.7: *The number of the particles of the VP-VSMMPF ($N^v_{s,k}$) over the scan number k. The area between the red dotted lines indicate off-road motion.*

4.5.2 Conclusions

This work studied the road constrained vehicle tracking problem and proposed a modification of the VSMMPF in which the number of its active particles was allowed to vary. The central idea was to exploit the fact that due to the road constraints the state uncertainty is smaller when the vehicle travels on the road than when off the road. A fact which implies that since the on-road state probability distribution is tighter, it becomes redundant to use more that a certain number of on-road particles. Simulation results demonstrated the on-road efficiency of the proposed algorithm since while using significantly fewer particles than the standard VSMMPF, it managed to attain a very similar tracking performance.

Although the VP-VSMMPF does decrease the computational requirements by using fewer on-road particles within its prediction mechanism, still it has to preliminary propagate and resample at every time step all nominal N_s particles. A better approach to the problem is described in the fifth chapter in which we introduce the variable mass particle filter, which adopts as well a varying particle philosophy but approaches the problem from a different way.

4.6 Tracking multiple vehicles with the VSMMPF

In the second part of the chapter we incorporate gating and data association features into the VSMMPF to enable it to track multiple vehicles. For gating we use varying-volume ellipses around the measurements and for associating the ambiguous measurement-to-track pairs we utilise the JPDA algorithm. A simulation study that we present analyses experimentally the gating and association functions. We described first this work in [90] in 2005.

For our analysis we assume that at every scan k we have a fixed number of vehicles n_v and a time-varying number of measurements $n_{m,k}$. We set the probability of detection to be one, $P_D = 1$, which implies that at every scan we get at least n_v measurements originating from the vehicles. The remaining measurements are false alarms whose number at every k is Poisson distributed with mean value λ.

The algorithm employs a total of $n_v \times N_s$ particles, where N_s is the number of particles for each vehicle. The augmented particle set that we use becomes thus:

$$\mathbf{X}_k = \left\{ \mathbf{X}_k^{(1)}, \ldots, \mathbf{X}_k^{(n_v)} \right\} \tag{4.28}$$

where $\mathbf{X}_k^{(\beta)} = \{\mathbf{x}_k^{i(\beta)}\}_{i=1}^{N_s}$ is the particle set of the β-th vehicle, for $\beta = 1, \ldots, n_v$. The states are the x-y position and velocity of each vehicle: $\mathbf{x}_k^{i(\beta)} = [x_k^{i(\beta)} \;\; y_k^{i(\beta)} \;\; \dot{x}_k^{i(\beta)} \;\; \dot{y}_k^{i(\beta)}]^T$. In the same fashion, the set of the particles' modes is augmented as well:

$$\mathbf{M}_k = \left\{ \mathbf{M}_k^{(1)}, \ldots, \mathbf{M}_k^{(n_v)} \right\} \tag{4.29}$$

where $\mathbf{M}_k^{(\beta)} = \{M_k^{i(\beta)}\}_{i=1}^{N_s}$ are the modes associated with the particles of the β-th vehicle. Finally, the set with the $n_{m,k}$ "raw" measurements that we obtain from the radar is:

$$\mathbf{Z}_k^0 = \left\{ \mathbf{z}_k^{(1)}, \ldots, \mathbf{z}_k^{(n_{m,k})} \right\} \tag{4.30}$$

in which $\mathbf{z}_k^{(\mu)} = [\theta_k^{(\mu)} \;\; r_k^{(\mu)}]^T$ is the μ-th measurement for $\mu = 1, \ldots, n_{m,k}$, where the radar provides measurements of the azimuth angle and range of the possible targets.

The first step of the proposed multitarget ground tracking particle filter (MGTPF) is to predict its particles one step ahead. This is done by propagating through the prediction phase of the

VSMMPF the set $\{\mathbf{X}_{k-1}, \mathbf{M}_{k-1}\}_{i=1}^{n_t \times N_s}$. For this step we use the VSMMPF as in the single-vehicle case shown in section 4.3, but now using $n_t \times N_s$ particles, and use relations (4.8-4.19) to predict the particles and their modes. The augmented sets of the predicted particles and their associated modes for all vehicles become:

$$\mathbf{X}_k^\star = \left\{\mathbf{X}_k^{\star(1)}, \ldots, \mathbf{X}_k^{\star(n_v)}\right\} \tag{4.31}$$

$$\mathbf{M}_k^\star = \left\{\mathbf{M}_k^{\star(1)}, \ldots, \mathbf{M}_k^{\star(n_v)}\right\} \tag{4.32}$$

where according to the previous $\mathbf{X}_k^{\star(\beta)} = \{\mathbf{x}_k^{i\star(\beta)}\}_{i=1}^{N_s}$ and $\mathbf{M}_k^{\star(\beta)} = \{M_k^{i\star(\beta)}\}_{i=1}^{N_s}$, $\forall\beta$.

We continue with the gating phase in which we construct groups of vehicles with candidate update measurements. We first transform the predicted particles $\{\mathbf{x}_k^{i\star(\beta)}\}_{i=1}^{N_s}$ to the polar plane using:

$$\mathbf{x}_{p,k}^{i\star(\beta)} = \begin{bmatrix} \arctan(y_k^{i\star(\beta)}/x_k^{i\star(\beta)}) \\ \sqrt{\left(x_k^{i\star(\beta)}\right)^2 + \left(y_k^{i\star(\beta)}\right)^2} \end{bmatrix} := \begin{bmatrix} \theta_{p,k}^{i\star(\beta)} \\ r_{p,k}^{i\star(\beta)} \end{bmatrix} \tag{4.33}$$

for all targets $\beta = 1, \ldots, n_v$. We then construct on the $\theta - r$ plane ellipsoidal gates around every measurement $\mathbf{z}_k^{(\mu)}$ using the following equation:

$$\frac{(\theta - \theta_k^{(\mu)})^2}{(\delta_{\sigma,k} \cdot \sigma\{v_{\theta,k}\}/2)^2} + \frac{(r - r_k^{(\mu)})^2}{(\delta_{\sigma,k} \cdot \sigma\{v_{r,k}\}/2)^2} = 1 \tag{4.34}$$

for $\mu = 1, \ldots, n_{m,k}$,where $\sigma\{v_{\theta,k}\}$ and $\sigma\{v_{r,k}\}$ are respectively the standard deviations of the measurement noise of the azimuth and range. The parameter $\delta_{\sigma,k}$ at every k is initially set to a user-defined value but under certain conditions it varies for increasing the gate volume (see later).

The physical meaning of $\delta_{\sigma,k}$ is that it defines at how many standard deviations from the mean (i.e. the measurement) the ellipses will be formed, setting thus the confidence limits of the gates. By setting for example $\delta_{\sigma,k} = 3$, we form gates which 99.73% of the times enclose the *true* vehicle positions when transformed to the polar plane. In practice we do not know the exact vehicle positioning, therefore the state uncertainty should be taken into consideration for computing the gates' volume. We do that indirectly by gating the particles (and not just the predicted vehicle position) whose spread implies the state uncertainty. With this approach we account moreover for the multimodel nature of the states. Normally, we set $\delta_{\sigma,k} = 4$ or even 5

for tolerating modelling errors and tracking inaccuracies[6]. If in our analysis we were considering a non unitary probability of detection, $P_D < 1$, then it would be more efficient to form gates about the predicted particle position accounting additionally for the state mulitmodality as in reference [35].

For every vehicle β we create then a candidate measurement set $\mathbf{Z}_k^{\star(\beta)}$ which consists from the measurements which have at least one of the vehicle's particles $\mathbf{x}_{p,k}^{i\star(\beta)}$ within their gates (the vehicle is said to satisfy these gates):

$$\mathbf{Z}_k^{\star(\beta)} = \left\{ \mathbf{z}_k^{\left(\mathbf{g}_k^{\star(\beta)}(1)\right)}, \ldots, \mathbf{z}_k^{\left(\mathbf{g}_k^{\star(\beta)}(\nu_k^{\beta})\right)} \right\} \tag{4.35}$$

where $\mathbf{g}_k^{\star(\beta)}(j)$ stands for the $j\text{-}th$ element of the ν_k^{β}-element set $\mathbf{g}_k^{\star(\beta)}$ which consists from the indices of the above measurements in the initial $\mathbf{Z}_k^0$ set[7], and ν_k^{β} denotes the number of the gates in which the particles of the $\beta\text{-}th$ vehicle lie. For example if $\mathbf{Z}_k^0 = \{\mathbf{z}_k^{(1)}, \mathbf{z}_k^{(2)}, \mathbf{z}_k^{(3)}\}$ and the $\beta\text{-}th$ vehicle satisfies the gates of $\mathbf{z}_k^{(1)}$ and $\mathbf{z}_k^{(3)}$, then $\nu_k^{\beta} = 2$ and $\mathbf{g}_k^{\star(\beta)} = \{1, 3\}$.

If for a vehicle β we get that $\nu_k^{\beta} = 0$ (i.e. it does not satisfy any gate), we increase accordingly the volume of the gates by increasing the variable $\delta_{\sigma,k}$ from (4.34) and we repeat until $\nu_k^{\beta} \neq 0$, $\forall\beta$. We then check if for any vehicle β it holds that $\nu_k^{\beta} = 1$ (i.e. it satisfies the gate of just one measurement). If this is true, we first *hard*-associate that measurement $\mathbf{z}_k^{\left(\mathbf{g}_k^{\star(\beta)}(1)\right)}$ with the vehicle and then remove it from the other candidate measurement sets by changing their respective index sets:

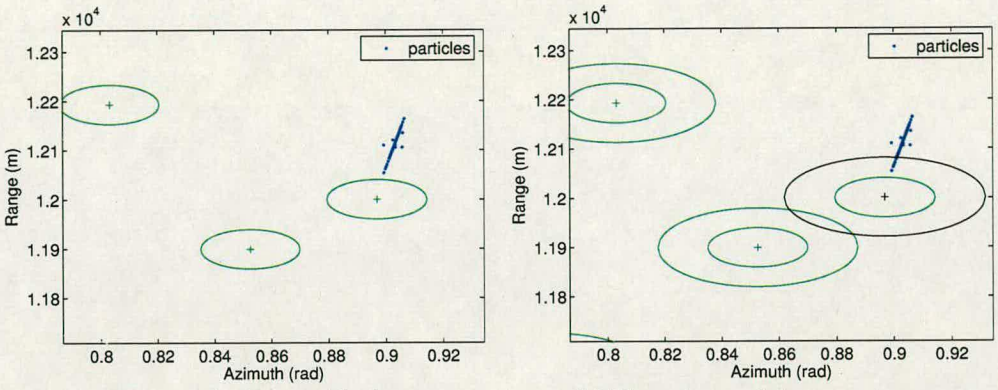
$$\mathbf{g}_k^{\star(j)} = \mathbf{g}_k^{\star(j)} \setminus \{\mathbf{g}_k^{\star(\beta)}(1)\}\Big|_{\forall j \in \{1,\ldots,n_v\} \neq \beta} \tag{4.36}$$

where notation $\setminus\{\}$ stands for the set difference. If (4.36) results in other vehicles satisfying just one gate, we repeat the final step until no other hard-associations can be performed. If during the process a vehicle becomes un-gated, we increase the gate volume as described before and we re-iterate.

We construct then the groups with the ambiguous measurement-to-vehicle assignments. Each group can consist either of a single vehicle and multiple measurements or of multiple vehicles and multiple measurements. For the latter case, the group is formed with vehicles that share at

[6] As we will see next, at certain cases we increase the value $\delta_{\sigma,k}$ to account for more severe positioning inaccuracies.

[7] A mathematical structure as $\mathbf{Z}_k^{\star(\beta)}$ can also be called more formally an *indexed family of elements*.



(a) *All particles lie outside the gates* **(b)** *The volume of the gates increases*

Figure 4.8: *The gates' volume increases if none of a targets' particles lies within a gate.*

least one common measurement. These groups are then passed through the JPDA algorithm (as described in section 2.9) and using equations (2.86-2.92)[8] we select the maximum joint probability assignments. Denoting as $\mathbf{g}_k$ the n_v-element index set that describes these assignments, the final associated measurement set becomes:

$$\mathbf{Z}_k = \left\{ \mathbf{z}_k^{(\mathbf{g}_k(1))}, \ldots, \mathbf{z}_k^{(\mathbf{g}_k(n_v))} \right\} \tag{4.38}$$

where once more $\mathbf{g}_k(j)$ stands for the j-th element of the set $\mathbf{g}_k$.

The particles obtain then a normalised weight according to their likelihood:

$$\tilde{w}_k^{i(\beta)} = p(\mathbf{z}_k^{(\mathbf{g}_k(\beta))} | \mathbf{x}_k^{i\star(\beta)}) \tag{4.39}$$

and:

$$w_k^{i(\beta)} = \frac{\tilde{w}_k^{i(\beta)}}{\sum_{j=1}^{N_s} \tilde{w}_k^{j(\beta)}} \tag{4.40}$$

where $i = 1, \ldots, N_s$ and $\beta = 1, \ldots, N_v$. The calculation of the state estimate of each vehicle β is given next.

[8] For the specific equations, for obtaining the prior state of each vehicle β we average their particles:

$$\mathbf{x}_k^{(\beta)-} = \frac{1}{N_s} \sum_{i=1}^{N_s} \mathbf{x}_k^{i\star(\beta)} \tag{4.37}$$

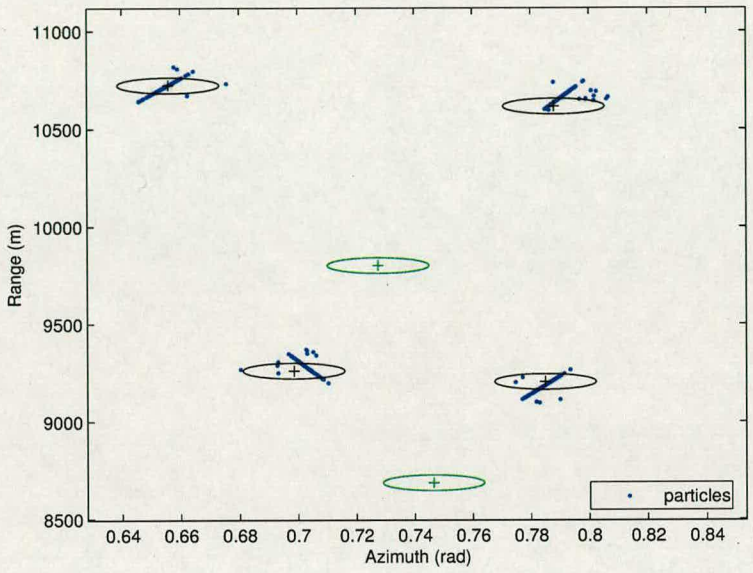


Figure 4.9: *The gated measurements and particles on the polar plane. For this example the associations are obtained after the gating function.*

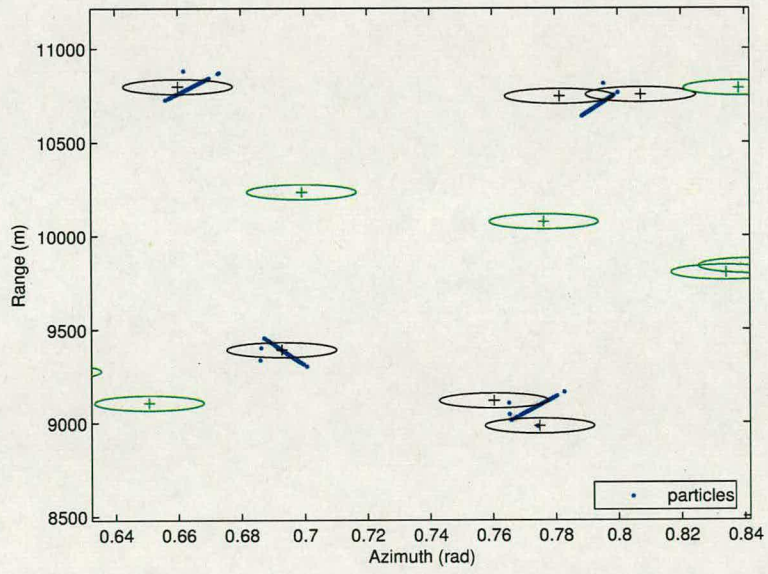


Figure 4.10: *The gated measurements and particles on the polar plane. Now two associations after gating are still ambiguous and the JPDA should be used for them.*

$$\hat{\mathbf{x}}_k^{(\beta)} = \sum_{i=1}^{N_s} w_k^{i(\beta)} \mathbf{x}_k^{i\star(\beta)} \tag{4.41}$$

As always, we resample using the SR algorithm to finally obtain:

$$\mathbf{X}_k = \left\{ \mathbf{X}_k^{(1)}, \ldots, \mathbf{X}_k^{(n_v)} \right\} \tag{4.42}$$

$$\mathbf{M}_k = \left\{ \mathbf{M}_k^{(1)}, \ldots, \mathbf{M}_k^{(n_v)} \right\} \tag{4.43}$$

Figures 4.8.a and 4.8.b illustrate graphically the varying-size gates. In the example shown due to tracking ill-performance the particles of the target in the first figure lie outside the gates of the measurements. The gates thus increase their volume until enclosing at least one particle as shown in the second figure. For the specific example the parameter $\delta_{\sigma,k}$ was 4 and the volume was set to increase by a factor of 2. Regarding the measurement assignment, figure 4.9 presents an example in which the gating function results in unambiguous measurement-vehicle pairs, whereas figure 4.10 shows a case in which the JPDA algorithm should be additionally used to resolve the association ambiguity.

4.7 Simulation analysis

In the final part of the chapter we apply the MGTPF to a simulated multitarget environment and we analyse its performance focusing particularly on its data association function.

4.7.1 Simulation study

We first use the MGTPF in a scenario in which four separated vehicles move across a known road structure assuming that the probability of false alarm is 0 (i.e. no measurement clutter). For comparison we also use a variation of the algorithm which employs the same gating and data association logic but utilises a standard SIR particle filter for estimation, which we call multitarget SIR (MSIR[9]). By contrasting these two approaches we aim to show that in an easy (association-wise) tracking environment the road constraints that the MGTPF exploits benefit mostly its positioning accuracy and not its association capability.

The road map and the paths of the vehicles are shown in figure 4.11. For consistency, we keep

[9]Not to be confused with the MSIR from section 3.7.1 which employs a nearest neighbour association method.

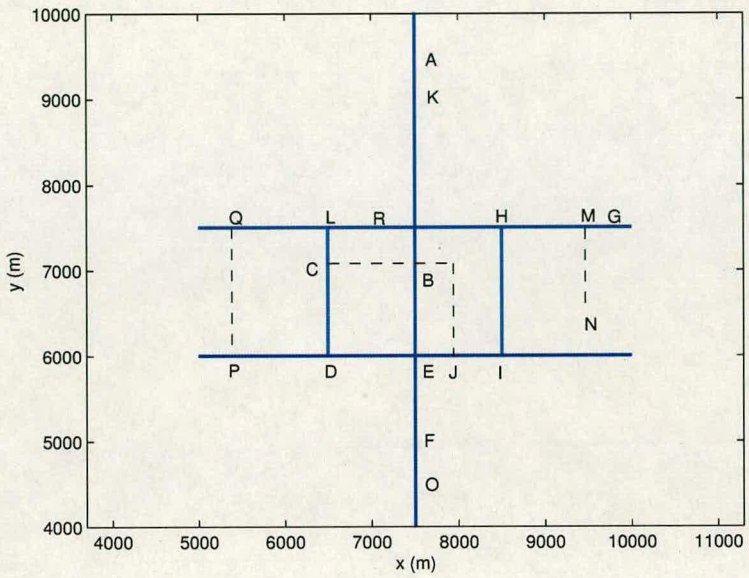


Figure 4.11: *The road map for scenario 1. The vehicle paths are: 1-ABCDEF, 2-GHIJBK, 3-JDLMN, 4-OEPQR. The dashed lines indicate off-road motion.*

the parameters of the simulation as in the single-vehicle case described in section 4.5. Therefore the radar lies at the origin (at point 0,0) with angular accuracy 0.5^o, range resolution 20m and update rate $T = 5$m. We set the road width to 8m, $\tau_r = 18.75$m, $\sigma\{u_{x,k}\} = \sigma\{u_{y,k}\} = \sigma\{u_{\alpha,k}\} = 0.6\text{m/s}^2$, $\sigma\{u_{o,k}\} = 0.0001\text{m/s}^2$ and $\bar{p} = p^\star = 0.98$. The parameter $\delta_{\sigma,k}$ is set initially to 4 and when required it is increased by a factor of 1.5. Both filters employ $N_s = 1000$ particles for each vehicle. The results presented are obtained after 1000 Monte Carlo runs. At the initialisation step we assume that we know the states of the vehicles and thus we seed accordingly the particles randomly around their position. Every simulation last 109 time scans.

Figure 4.12 shows the RMS position error of the MGTPF. As expected similar patterns as in figure 4.6 emerge, which verify once more that when the vehicles are travelling on-road the tracking error is reduced due to the road constrains. For the specific scenario, the on-road RMSE of the MGTPF is on average 55.87% smaller than the MSIR. The error peaks that appear in the figure are due to vehicle turns or junction crossings. Off the road the performance of both algorithms is the same, since there the SIR and the VSMMPF (and thus also the MGTPF) share a very similar estimation logic. Table 4.2 presents the average position error of each vehicle for

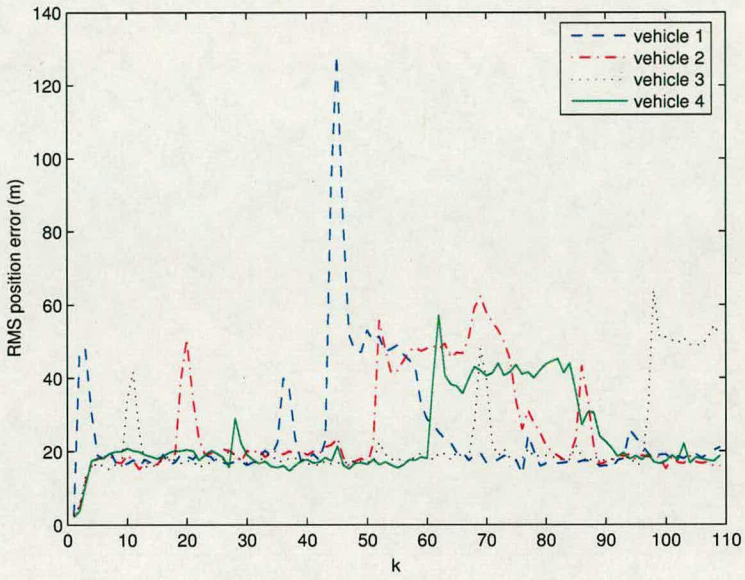


Figure 4.12: *The RMS position error of the MGTPF over the scan number k.*

both algorithms. For this scenario the gating and assignment functions of both trackers perform equally well and all the measurements are associated correctly.

We test then the measurement assignment capability of the MGTPF in a more difficult scenario. We increase therefore by three times the measurement noise and we use two vehicles whose paths are closely separated as depicted in figure 4.13. The first moves off the road and the second travels along the road. We expect the most disassociations and track swaps to occur when the on-road vehicle executes its turn, since then the vehicles' separation is minimum. For this scenario we perform 1000 MC runs and we count the number of the track swaps. Both filters use 1000 particles and are initialised with their true vehicles' states, so as not to bias the results with initialisation artifacts.

Table 4.2 shows that the MGTPF exhibits an enhanced association performance, resulting on average in 34.30% fewer track swaps. This is because the MGTPF particles of the on-road vehicle are mostly concentrated along the road, a fact which limits significantly the particle overlapping of the two vehicles and decreases the association uncertainty/difficulty. Figure 4.14 presents a representative example in which a track swap occurs when using the MSIR.

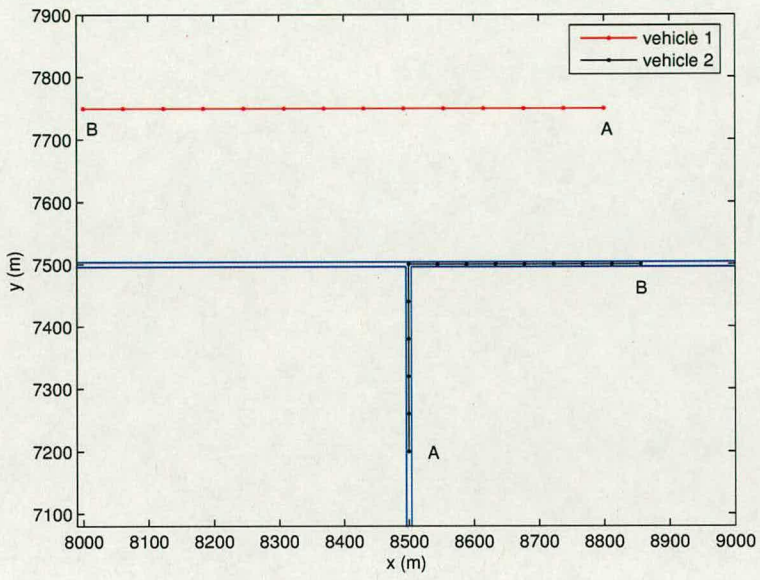


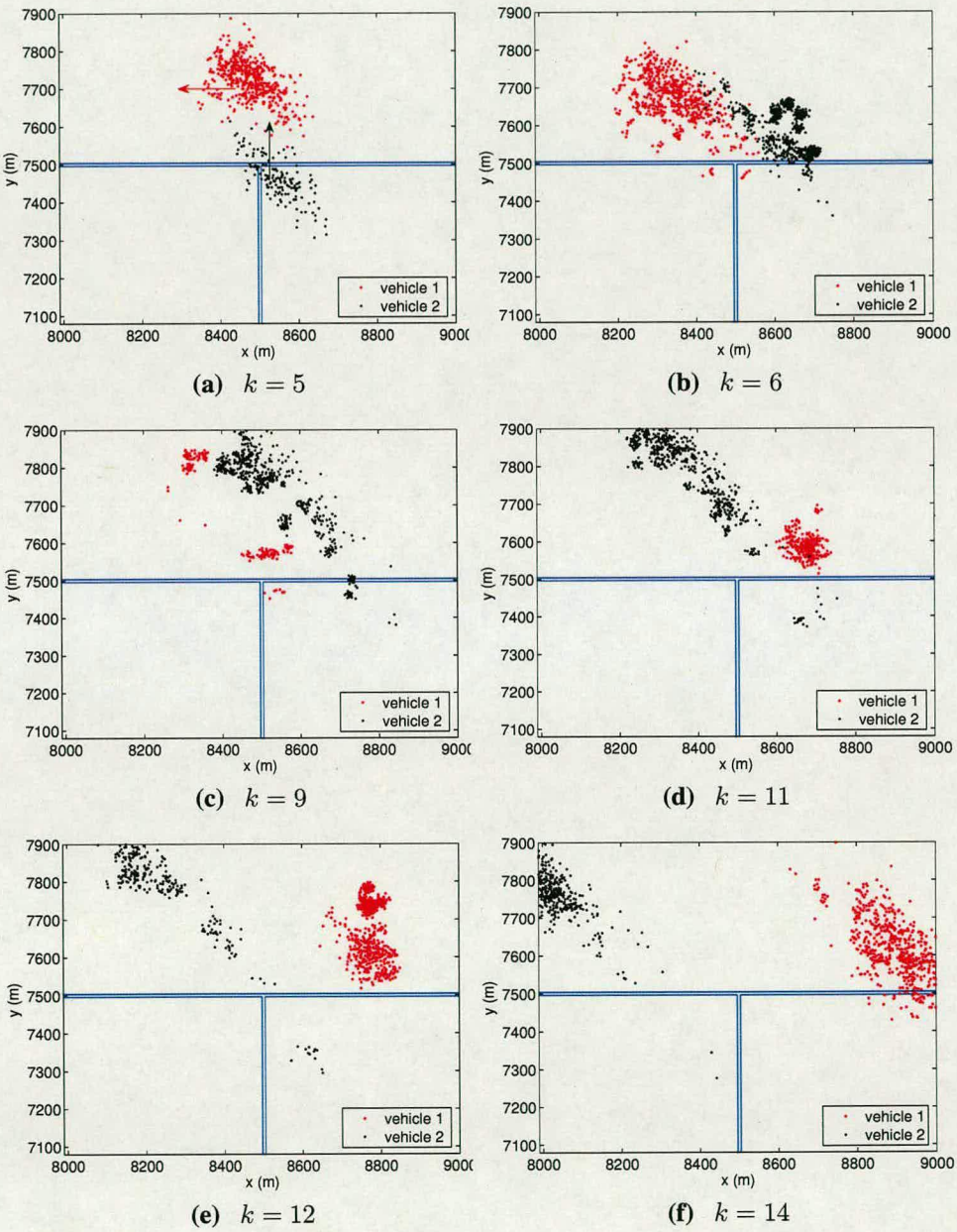
Figure 4.13: *The road map for scenario 2. Both vehicles travel from points A to B.*

After the vehicle's turn at $k = 9$ a measurement outlier results in a disassociation, which finally leads to a track swap due to the more spread (unconstrained) particles of the on-road vehicle. In general, for the chosen scenario the MSIR can not cope efficiently with the abrupt 90^{o} turn of the second vehicle and the consequent severe particle overlapping finally degrades its association performance.

In the last set of simulations we use the same four-vehicle environment from the first scenario but we introduce to the problem erroneous measurement clutter. We assume that the number

Scenario 1			
		MGTPF	SIR
Average	vehicle 1	26.11	59.04
RMSE	vehicle 2	27.12	62.54
on-road	vehicle 3	25.22	57.60
(m)	vehicle 4	24.12	53.40
Scenario 2			
Track swaps		35.5	54.00

Table 4.2: *Simulation results after 1000 MC runs.*



(a) $k = 5$ **(b)** $k = 6$

(c) $k = 9$ **(d)** $k = 11$

(e) $k = 12$ **(f)** $k = 14$

Figure 4.14: *An example of the MSIR particle cloud in scenario 2. A track swap occurs following a disassociation at* $k = 9$.

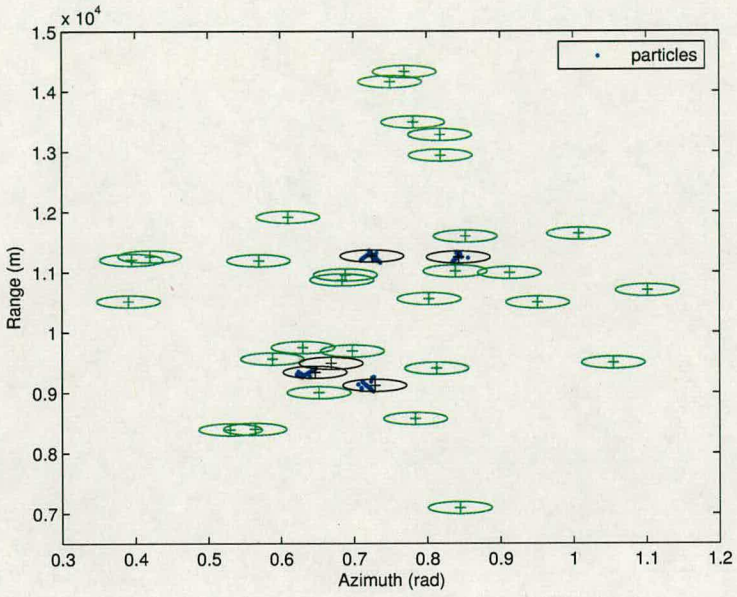


Figure 4.15: *A snapshot of the gated measurements and particles on the polar plane when* $\delta_{\sigma,k} = 10$ *and* $\lambda = 30$.

of clutter returns at every scan is random and drawn from the Poisson distribution with mean-value λ. We use the MGTPF and we count the disassociations for two different gate volumes and four clutter densities: $\lambda = 10, 20, 30$ and 40. We count moreover the times the algorithm calls the JPDA function and the ambiguous vehicles at every call. For this study the tracker employs just $N_s = 200$ particles for each target, so as to force the association functions to make more difficult decisions (since due to the relatively small number of particles the tracking performance is expected to degrade).

Table 4.3 presents the average results[10] after 100 MC runs for every different case. We observe that for this scenario the number of disassociations increases approximately linearly with the clutter density, ranging from as low as 4.78 ($\delta_{\sigma,k} = 10, \lambda = 10$) to 28.17 ($\delta_{\sigma,k} = 20, \lambda = 40$) per run. Considering that at every run 436 association decisions are made, the association performance is indeed satisfactory, since even in the worst (unrealistically difficult) case the disassociations are still just about the 6% from the total associations. Figure 4.15 illustrates an

[10]The results include just the non-divergent runs, which for $N_s = 200$ were about 96-97% of the total.

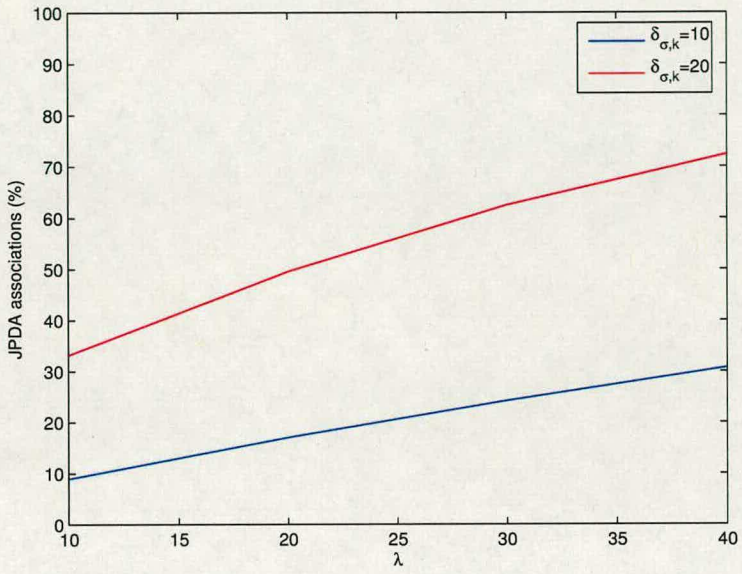


Figure 4.16: *The percentage of associations using the JPDA at every run over the clutter density λ for $\delta_{\sigma,k} = 10$ and 20.*

	$\delta_{\sigma,k} = 10$				$\delta_{\sigma,k} = 20$			
clutter λ	10	20	30	40	10	20	30	40
disassociations	4.78	14.00	17.26	23.13	11.16	15.20	18.30	28.17
1 x JPDA (1 vehicle)	28.15	41.47	44.77	44.93	42.14	41.88	33.75	25.19
2 x JPDA (1 vehicle)	3.98	11.95	20.69	27.81	15.00	29.64	35.58	37.05
3 x JPDA (1 vehicle)	0.19	1.61	4.23	7.75	2.42	11.00	20.58	26.14
4 x JPDA (1 vehicle)	0.00	0.06	0.38	1.01	0.18	1.64	5.08	10.81
1 x JPDA (2 vehicles)	0.02	0.02	0.02	0.09	1.60	2.94	4.14	4.97
1 x JPDA (3 vehicles)	0.53	1.00	1.39	1.27	11.19	12.07	13.29	14.25
1 x JPDA (4 vehicles)	0.13	0.19	0.23	0.45	6.59	7.32	7.80	8.94
2 x JPDA (2 vehicles)	0.01	0.00	0.01	0.04	0.33	0.90	1.42	1.50
JPDA associations (%)	8.91	17.02	24.20	30.70	33.16	49.47	62.39	72.34
gate associations (%)	91.08	82.98	75.80	69.30	66.84	50.53	37.61	27.65

Table 4.3: *The average disassociations, number of JPDA calls and the percentage of the associations obtained using the JPDA or gating functions per run. At every run 436 associations decisions are taken.*

example of the dense clutter environment in which the simulations take place. In table 4.3 we also present the percentage of the associations that are performed using just the gating function or require additionally the JPDA algorithm. Figure 4.16 depicts the linear relation between the associations with the JPDA and the clutter density. For the smaller gate volume ($\delta_{\sigma,k} = 10$), only 8.91-30.7% of the associations called for the JPDA, fact which emphasises the importance of a gating function within a data association system regarding the computational efficiency.

4.7.2 Summary

This work introduced a multiple vehicle structure for the VSMMPF algorithm and studied experimentally its association performance. The proposed MGTPF propagates its particle set, which is augmented with the particles of all vehicles, using the prediction mechanism of the standard VSMMPF. It then uses ellipsoidal gates about each measurement to assign to the vehicles candidate measurement updates. If a vehicle is assigned with more than one candidate measurement, the JPDA algorithm is used to resolve the association ambiguity. After the association phase, the particles from each vehicle are treated separately and are weighted and resampled in the normal PF fashion.

The simulation analysis demonstrated the suitability of the MGTPF within the multiple vehicle environment. It showed that the road constraints that it exploits, result in an improved association capability compared to the equivalent multiple-vehicle unconstrained SIR. This was especially evident when the vehicles travelled closely, in which case the tighter and more precise MGTPF posterior state distribution resulted not only in the improvement of the estimation accuracy but also in the reduction of disassociations and track swaps. Regarding the efficiency of the gating scheme for associating the raw measurements, we saw that in our environment depending on the clutter density, up to 91.08% of the associations could be obtained just after the gating function.

4.8 Chapter summary

The fourth chapter studied the vehicle tracking problem. Based on the variable structure multiple model particle filter, it introduced a varying-particle variation which allowed the number

of the particles to vary. Aiming in improving the particle (i.e. computational) efficiency, fewer particles were used while the vehicle was travelling on-road, since the road constraints were making the estimation easier. Simulation results demonstrated that the degradation in the RMS position error while using the varying-particle algorithm was negligible when compared to the lighter particle usage and the resulting computational gains. We furthermore presented an enhancement of the standard vehicle tracker with a gating and data association capability, for rejecting measurement clutter and tracking simultaneously multiple vehicles. For measurement-to-track assignment we exploited the joint probabilistic data association algorithm. A simulation study analysed the proposed tracker in various scenarios, assessed its suitability in different association environments and showed that under difficult association conditions both the estimation accuracy and association capability could be improved when compared with the equivalent multitarget sequential importance resampling filter.

Chapter 5
Variable-mass particle filtering for vehicle tracking

In the fifth chapter we continue with the vehicle tracking problem presenting a different approach to deal with its inherent multi-modality. We propose a mechanism which allows the number of the particles within each mode to vary adaptively over time, depending not only on the prior mode probabilities but according to the mode likelihood and difficulty. Moreover, we introduce an on-road architecture which incorporates Kalman filtering elements and uses just one on-road particle. Simulation results demonstrate that the proposed variable mass particle filter can achieve better performance, while using fewer particles and less computational power, when contrasted with the variable structure multiple model particle filter.

5.1 Introduction

As we saw in the previous chapters, when the target switches between two or more motion dynamics, *multiple-mode* estimators should be used. Depending on the tracking technique, the estimates are obtained using a mechanism that combines the outputs of the possible operating modes. Historically, the interacting multiple model filter (IMM) [91] from the 80's was the first powerful multiple-mode estimator. It was based on the Kalman filtering framework [8] and was used extensively throughout the years. In 2000, reference [84] introduced the variable-structure IMM (VSIMM), essentially an IMM whose number of the active modes was allowed to vary. The VSIMM improved the performance of the IMM when they were both tested in the terrain-aided vehicle tracking problem. In 2002, the variable-structure multiple model particle filter (VSMMPF) [82] made its appearance, incorporating the varying modes mechanism into the powerful particle filtering technique. The VSMMPF compared to the VSIMM on the same vehicle tracking problem, exhibited an even greater improvement on the estimation accuracy.

The work in this chapter, presented first in [92], is an attempt to improve the particle efficiency of the VSMMPF with its key contribution being the use of particles with variable masses.

Whereas in the VSMMPF the number of the particles allocated to its modes is proportional to *fixed* mode probabilities, in the proposed variable mass particle filter (VMPF) that number is allowed to vary according to arbitrary user-defined criteria. The VMPF compensates for the statistical irregular particle patterns that emerge by rescaling the particles of every mode using appropriate masses.

The vehicle tracker that we introduce in this chapter, adopting the variable-mass approach, is allowed to exploit information from the measurement and the mode difficulty for allocating the particles to its modes. The benefits thus are twofold: firstly more particles are allocated to the most probable and/or difficult modes for improving the tracking accuracy and secondly modes which are less probable and/or have easier dynamics obtain fewer particles for reducing the computational requirements. Other - more application specific - features of the proposed vehicle tracker is an on-road propagation mechanism which employs one particle and a Kalman filter (KF) for reducing further the computational demands and a technique which enables the algorithm to deal with random road departure angles (instead of just $\pm 90^{\circ}$ as the VSMMPF does).

5.2 Variable-mass technique

In this section we first summarise the VSMMPF logic for allocating the particles to the multiple modes and then introduce the notion of the particle *mass*. Consider a n_m-mode particle filter which at time instant $k-1$ has $N_{\alpha,k-1}$ particles at mode α. Let the known *a-priori* probability switching[1] from mode α to mode β be $p_{\alpha\to\beta} \in \mathbb{R}[0,1]$ where $\alpha, \beta \in \mathbb{N}[1, n_m]$. Following the logic of the VSMMPF the number of the transferred particles to a mode is proportional to the fixed prior mode probability:

$$N_{\alpha\to\beta,k} = |\{\nu_i < p_{\alpha\to\beta} : \{\nu_i : \nu_i \sim \mathcal{U}[0,1]\}_{i=1}^{N_{\alpha,k-1}}\}| \tag{5.1}$$

where $N_{\alpha\to\beta,k}$ is the number of the particles that are transferred from mode α to mode β and notation $|\{\}|$ stands for the set cardinality. For a large number of particles it holds that:

$$\lim_{N_{\alpha,k-1}\to\infty} N_{\alpha\to\beta,k}|_{N_{\alpha,k-1}} = p_{\alpha\to\beta} \cdot N_{\alpha,k-1} \tag{5.2}$$

[1] A *switch* from mode α to β, refers to a change of the particle propagation model from the one of mode α to β.

which implies that on average we get:

$$\bar{N}_{\alpha\to\beta,k} = p_{\alpha\to\beta} \cdot N_{\alpha,k-1} \tag{5.3}$$

Furthermore it holds that:

$$\sum_{\beta=1}^{n_m} p_{\alpha\to\beta} = 1, \ \forall\alpha \tag{5.4}$$

$$\sum_{\beta=1}^{n_m} N_{\alpha\to\beta,k} = N_{\alpha,k-1}, \ \forall\alpha \tag{5.5}$$

Relation (5.4) emphasises that since $\{p_{\alpha\to\beta}\}_{\beta=1}^{n_m}$ is a set of probabilities, its elements should add to one. Relation (5.5) implies that all the particles from mode α at $k-1$ should be propagated at k, i.e. the overall number of the particles of the estimator remains constant.

The variable-mass mechanism introduces another degree of freedom in the estimation procedure by employing particle triples consisting of {state,weight,mass}. Using the VMPF logic the particle allocation is not directly linked to the prior mode probabilities, but is done according to arbitrary and application-specific probabilistic metrics as a way to *indirectly exploit additional information*. The "probabilistic weight" of each mode (implied from the fixed a-priori mode probabilities) is sustained since after the particle allocation, appropriate masses are computed which scale the weights of the particles. Therefore, the particles in the heavier-populated modes obtain smaller masses whereas the ones in the lighter-populated modes are assigned with larger masses.

In particle filtering the resolution of the estimated state distribution in a certain area depends on the number of the particles that lie within it. Therefore, the variable-mass technique essentially enables us to modify that resolution, since now we directly control the number of the particles that we can seed in different state areas. According to our requirements the resolution can be locally enhanced or impoverished, for either improving the estimation accuracy or reducing the particle redundancy and lowering thus the computational complexity.

Consider again the n_m-mode particle filter defined at the beginning of the section. In the VMPF the total mass of each mode is proportional to the prior mode probability:

$$N'_{\alpha\to\beta,k} \cdot m_{\alpha\to\beta,k} = p_{\alpha\to\beta} \cdot N_{\alpha,k-1} \tag{5.6}$$

where now $N'_{\alpha\to\beta,k}$ is the number of the particles that are transferred from mode α to mode β and $m_{\alpha\to\beta,k}$ is their mass. The equation above is the equivalent of equation (5.3) of the VSMMPF, but now it is evident that by introducing $m_{\alpha\to\beta,k}$ as an extra degree of freedom, we can change arbitrary $N'_{\alpha\to\beta,k}$ and retain the quantity $p_{\alpha\to\beta} \cdot N_{\alpha,k-1}$ fixed. We can either vary $N'_{\alpha\to\beta,k}$ in proportion to a certain user defined probabilistic metric $\gamma_{\alpha\to\beta,k} \in \mathbb{R}[0,1]$:

$$N'_{\alpha\to\beta,k} = \gamma_{\alpha\to\beta,k} \cdot N_{\alpha,k-1} \tag{5.7}$$

or fix its value according to available prior knowledge or other specified requirements:

$$N'_{\alpha\to\beta,k} = const. \tag{5.8}$$

The parameter $\gamma_{\alpha\to\beta,k}$ is called *gamma* metric and can vary over time or remain fixed. For $\gamma_{\alpha\to\beta,k}$ it also holds that:

$$\sum_{\beta=1}^{n_m} \gamma_{\alpha\to\beta,k} = 1, \ \forall\alpha \tag{5.9}$$

Another property of the VMPF is that in the general case the total number of particles is allowed to vary:

$$\sum_{\beta=1}^{n_m} N'_{\alpha\to\beta,k} \neq N_{\alpha,k-1}, \ \forall\alpha \tag{5.10}$$

This implies that we can keep the particle number of certain modes constant and *at the same time* vary the particle number of other modes, without biasing the underline estimation statistics.

Finally, by rearranging (5.6) we define the mass of the particles that are transferred from mode α to β as:

$$m_{\alpha\to\beta,k} = p_{\alpha\to\beta} \cdot \frac{N_{\alpha,k-1}}{N'_{\alpha\to\beta,k}} \tag{5.11}$$

These masses are used to rescale the weights of the particles, just before computing the final state estimate.

Figures 5.1 and 5.2 illustrate graphically through a simple example the particle allocation approaches of the VSMMPF and the VMPF in a two dimensional x-y Cartesian space. We consider that the 100 particles from mode α at $k-1$ are propagated to the equiprobable modes β and

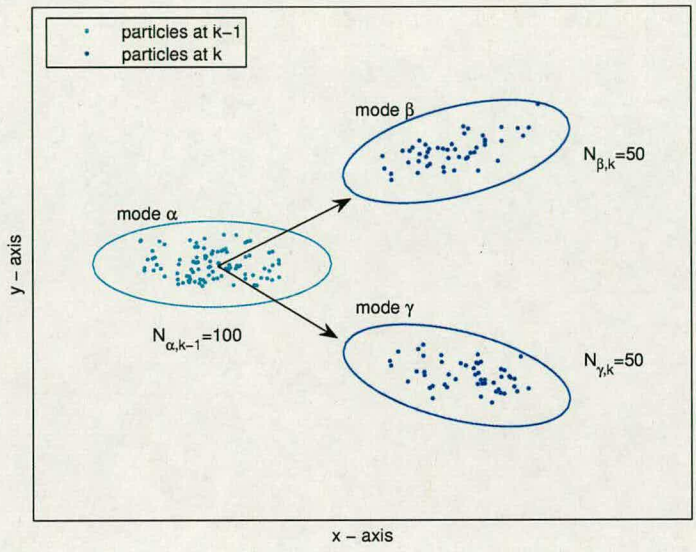


Figure 5.1: *The VSMMPF allocates the particles proportionally to the prior mode probabilities. In this example both modes are equiprobable.*

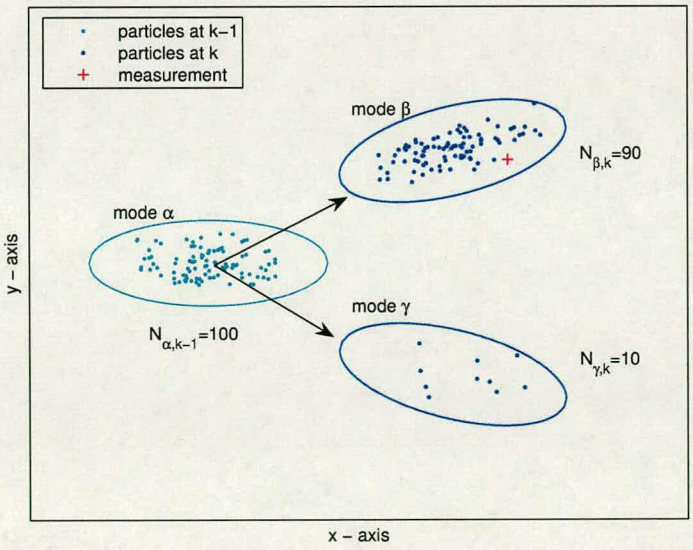


Figure 5.2: *For the same example, the VMPF allows for an uneven particle allocation to account for the measurement, which corrects later using variable particle masses.*

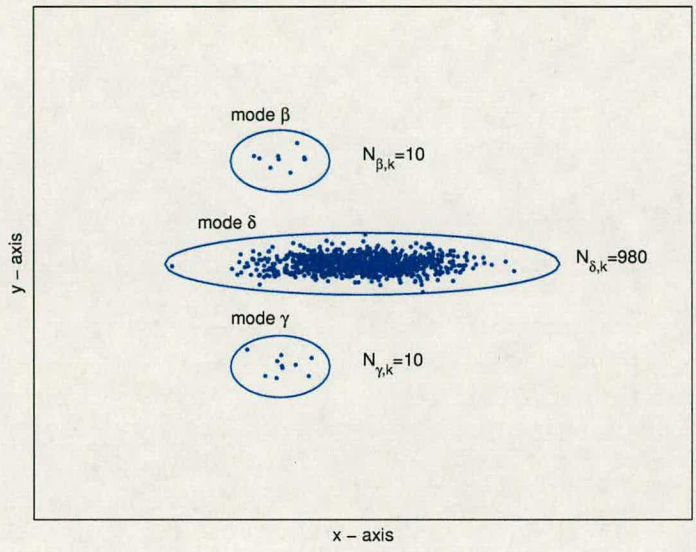


Figure 5.3: *The VSMMPF uses 1000 particles which correctly allocates to its modes in proportion with the fixed mode probabilities (*98% *for* δ *and* 1% *for each* β *and* γ*).*

γ at k. The first figure shows the VSMMPF seeding its particles according to the prior mode probabilities, allocating thus 50 particles to each mode. On the other hand, the VMPF using a gamma metric which exploits the measurement, propagates most particles, say 90, in the more likely mode β. Using (5.11) we calculate for the specific example that the mass of the particles of mode β is 0.55 and of mode γ is 5.00. Thus we expect to increase the estimation accuracy, since the resolution of the state posterior will be higher in areas with higher likelihood which contribute more in the state characterisation, and furthermore reduce the particle redundancy, since the particles on mode γ is more likely to be replaced during the resampling step.

Figures 5.3, 5.4 and 5.5 show another example in which the variable-mass technique can enhance the performance of the VSMMPF. We consider a system having three possible modes at k, with prior probability 98% for mode δ and 1% for each mode β and γ. In figures 5.3 and 5.4 we use a VSMMPF which employs respectively 1000 and 30 particles. In the first case correctly 980 particles are allocated to the central mode and 10 particles to the modes on the sides. In the second case however, the small 1% mode probabilities in conjunction with the small number of particles result in all 30 particles populating *just* the central mode. On the

other hand, the equivalent 30-particle VMPF in figure 5.5 can allocate to each lower probability mode a number of particles, say 5 for this example, and thus exploit the true multimodel nature of the system[2]. Again using (5.11) we compute that the 20 particles of mode δ obtain a mass of 1.47 while the remaining in modes β and γ are assigned with masses of 0.06. We expect once more the performance to improve since now the system model is used more accurately and the possibly essential information from all modes is utilized.

5.3 Variable mass particle filter

We begin this section by outlining the novel features of the vehicle-tracking VMPF and then we describe in detail how the algorithm works.

5.3.1 Novel features of the vehicle tracker

The VMPF employs the varying mass technique for propagating its on-road particles on and off the road. Specifically for these particles, the tracker uses as the gamma metric an approximation of the posterior mode probabilities, obtained by fusing the *fixed* prior mode probabilities with the *varying* modes' likelihoods conditioned on the current measurement. As described before, the varying masses that the algorithm uses, compensate for the resulting over- or under-population of its modes. The fact that in contrast to the VSMMPF, the VMPF is not "blind" to the measurements when allocating its on-road particles to their corresponding modes, results in a more efficient particle use, which translates consequently to a performance improvement. For the off-road particles both algorithms use a similar propagation mechanisms.

Another feature of the new vehicle tracker is that it employs just one particle on the road. This is because the on-road dynamics are easier to estimate due to the soft constraints that the roads themselves impose [89]. Following the varying-mass logic, the mass of that on-road particle is proportional to the posterior probability of the on-road mode. Compared to the VSMMPF, the fact that the variable mass approach allows the tracker to use just one particle for this mode, results in significant computational gains when the vehicle travels on the road.

[2]Reference [93] addresses the specific problem by exploiting a bank of particle filters, one for each mode, which are weighted appropriately before calculating jointly the estimate. The drawback is that compared to the *variable-structure* filters (VSIMM, VSMMPF, VMPF) their modes always employ a fixed number of particles (more computational demands) which cannot be directly transferred from one mode to another (less flexible structure).

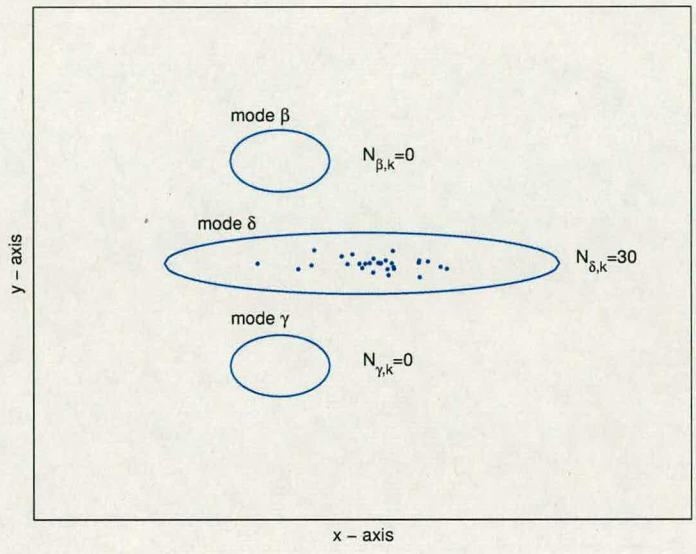


Figure 5.4: *For the same example when the VSMMPF uses just 30 particles, it cannot seed them to all modes and only the highest-probability mode δ is populated.*

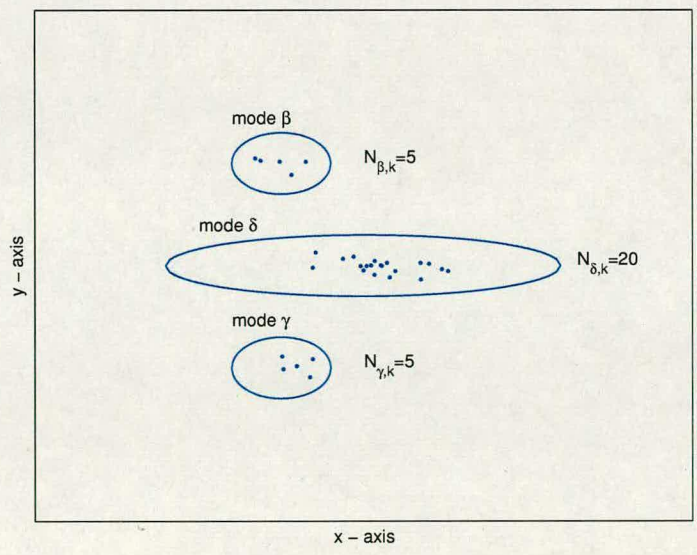


Figure 5.5: *Even with 30 particles, the VMPF is allowed to propagates them to all modes, by assigning them later appropriate masses to correct the estimate.*

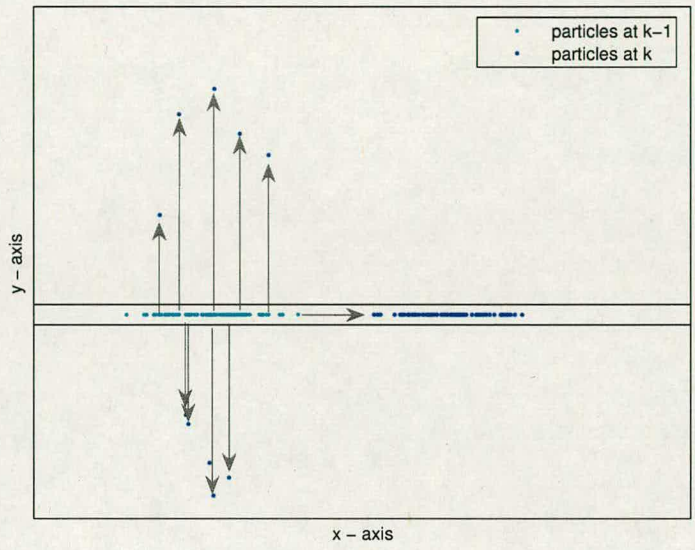


Figure 5.6: *In the road-constrained vehicle tracking problem, the VSMMPF employs a large number of on-road particles, which according to the fixed mode probabilities are propagated on or off the road.*

For the prediction of the on-road particle the VMPF employs a Kalman filter. For running the KF, it converts the 2-D polar radar measurements to 1-D Cartesian pseudo-measurements (approximated as Gaussian) that lie in the middle of the road. The KF operates in a reduced-dimension 2-D state-space along the middle of the road and feeds sequentially the tracker with an estimate of the mean and covariance of the on-road states. The estimate of the mean is transformed and placed into the original 4-D tracking state-space to finally form the on-road particle. The estimated on-road probability distribution is used in the prediction step to *draw* randomly from it particles and propagate them off the road. The number of these departing particles is determined from the posterior road-exit mode probabilities.

Figures 5.6, 5.7 and 5.8 contrast the on-road propagation mechanisms of the VSMMPF and VMPF. In particular the last two figures illustrate graphically the 'interfacing" component of the hybrid structure of the proposed tracker. We see there that rather than employing a large number of on-road particles like the VSMMPF, the VMPF uses just one particle (propagated with the KF) from which the off-road particles are spawned.

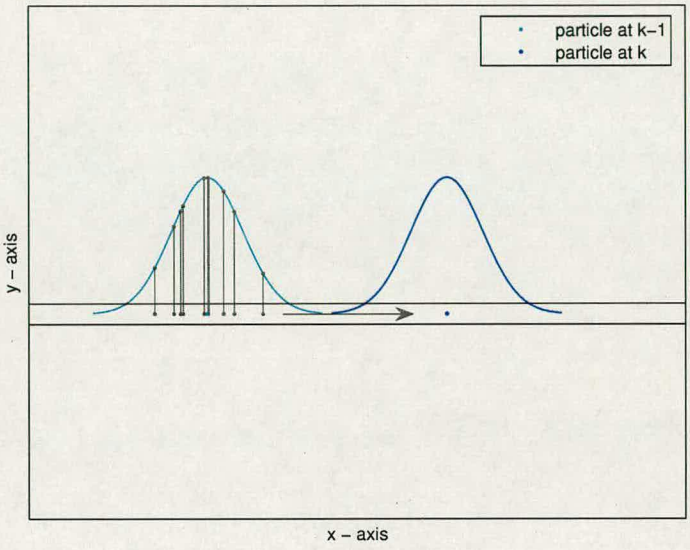


Figure 5.7: *The proposed varying-mass vehicle tracker uses one on-road particle propagated on-road with a KF. Its state distribution at $k-1$ is sampled accordingly to generate particles to be predicted off-road.*

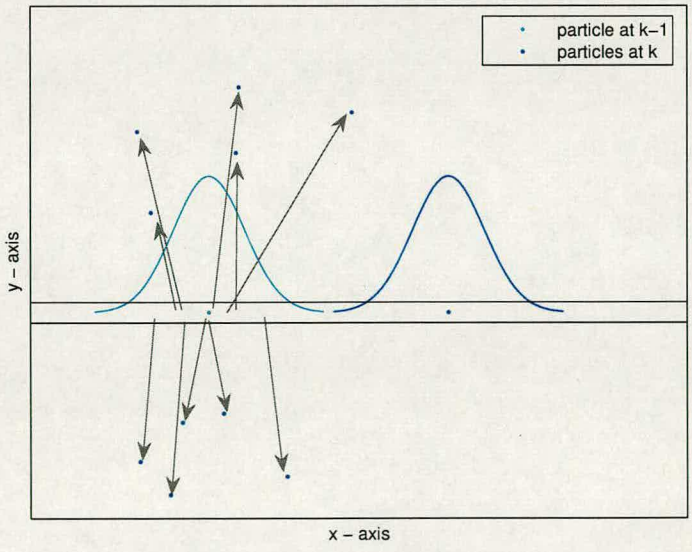


Figure 5.8: *The final particle cloud at k after predicting off-road the generated particles.*

5.3.2 The algorithm

For the sake of clarity we do not consider a junction-crossing prediction model and focus just on an environment with a vehicle travelling on and off non-intersecting roads. The VMPF consists of a prediction, an update and a resampling step, which we describe next.

A Prediction step

In the prediction step the algorithm predicts the particles one step ahead according to their mode dynamics. First we describe the prediction phase for the road particles and then for the off-road particles.

A.i Prediction of the on-road particles

This phase consist of the prediction of the on-road particles which either continue on the road or depart from it. We employ one particle for modelling the on-road motion. For the on-road prediction we first generate an on-road pseudo-measurement $\hat{z}_{on,k}$ with its associated variance and then apply a KF. We consider figure 5.9 assuming that line AB lies in the middle of the road. For clarity and simplicity in our analysis the roads are set parallel to the x-axis.

At time instant k we receive a radar measurement $\mathbf{z}_k = [\theta_k \; r_k]^T$ which we transform to the Cartesian space to obtain $\mathbf{z}_k^c$:

$$\mathbf{z}_k^c = h^{-1}(\mathbf{z}_k) = \begin{bmatrix} r_k \cdot \cos\theta_k \\ r_k \cdot \sin\theta_k \end{bmatrix} \tag{5.12}$$

The skewed ellipse around $\mathbf{z}_k^c$ at figure 5.9, is the n_σ-th standard deviation ($\hat{\sigma}_{z,k}$) confidence interval of the measurement noise, after being transformed to the Cartesian plane using function $\mathbf{h}^{-1}(\cdot)$ from (5.12). $\mathrm{C}_1 = (x_{C1}, y_{C1})$ and $\mathrm{C}_2 = (x_{C2}, y_{C2})$ are the cross section points of the interval and the middle of the road. The value of n_σ is chosen arbitrary (usually 3-4) since later on equation (5.13) cancels it out.

The assumption of VMPF is that the cross section of line AB and the 2-D skewed-Gaussian measurement noise pdf can be approximated as a 1-D *Gaussian* pdf along AB. Therefore, since we are also using a *linear* constant velocity vehicle model, we track on-road on a reduced state space (along AB) with a 2-D Kalman filter. The tracking space of the KF consists of the

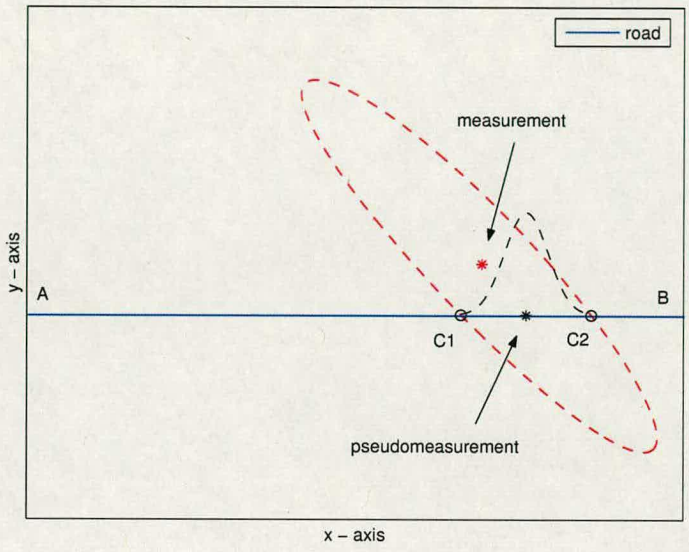


Figure 5.9: *The pseudo-measurement is set on the mode of the distribution resulting from the cross section of line AB (the middle of the road) with the measurement distribution. The skewed ellipse (red dashed line) around the measurement is a vertical section of the measurement distribution. We fit on the pseudo-measurement a one dimensional Gaussian pdf (black dashed line rotated 90° for illustration).*

vehicle's position $x_{on,k}$ and velocity $\dot{x}_{on,k}$ *just* along the middle of the road. This is because any attempt to track any possible on-road movement orthogonal to the road, will have negligible significance; especially since the roads seem to have zero width when the radar is far.

For computing the pseudo-measurement $\hat{z}_{on,k}$ on AB we find the point within the segment C_1C_2 which maximises the measurement likelihood (i.e. the statistical *mode*) and fit to it a Gaussian pdf. The standard deviation of the pdf can be approximated numerically as:

$$\hat{\sigma}_{z,on,k} = \frac{|x_{C1} - x_{C2}|}{n_\sigma} \tag{5.13}$$

Using $\hat{z}_{on,k}$, we predict the on-road particle $\mathbf{x}^{2D}_{on,k-1}$ one step ahead with the following set of KF equations.

$$\mathbf{x}_{on,k}^{2D-} = \mathbf{F}_{on} \cdot \mathbf{x}_{on,k-1}^{2D} \tag{5.14}$$

$$\mathbf{P}_{on,k}^{-} = \mathbf{F}_{on} \cdot \mathbf{P}_{on,k-1} \cdot \mathbf{F}_{on}^{T} + \mathbf{G}_{on} \cdot Q_{on} \cdot \mathbf{G}_{on}^{T} \tag{5.15}$$

$$\mathbf{K}_{k} = \mathbf{P}_{on,k}^{-} \cdot \mathbf{H}_{on} \cdot [\mathbf{H}_{on} \cdot \mathbf{P}_{on,k}^{-} \cdot \mathbf{H}_{on}^{T} + \hat{R}_{on,k}]^{-1} \tag{5.16}$$

$$\mathbf{x}_{on,k}^{2D} = \mathbf{x}_{on,k}^{2D-} + \mathbf{K}_{k} \cdot [\hat{z}_{on,k} - \mathbf{H}_{on} \cdot \mathbf{x}_{on,k}^{2D-}] \tag{5.17}$$

$$\mathbf{P}_{on,k} = [\mathbf{I} - \mathbf{K}_{k} \cdot \mathbf{H}_{on}] \cdot \mathbf{P}_{on,k}^{-} \tag{5.18}$$

where:

$$\mathbf{F}_{on} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} , \ \mathbf{G}_{on} = \begin{bmatrix} T^2/2 \\ T \end{bmatrix} , \ Q_{on} = \sigma\{u_{\alpha,k}\}^2 , \ \mathbf{H}_{on} = [0 \ 1] \tag{5.19}$$

$\hat{R}_{on,k} = (\hat{\sigma}_{z,on,k})^2$ is the variance of $\hat{z}_{on,k}$ and $\mathbf{x}_{on,k}^{2D} = [x_{on,k} \ \dot{x}_{on,k}]^T$ is the truncated 2-D version of the on-road particle. We augment then the $\mathbf{x}_{on,k}^{2D}$ and place it into the original 4-D state-space:

$$\mathbf{x}_{on,k}^{\star} = \begin{bmatrix} x_{on,k} \\ y_{on,k} \\ \dot{x}_{on,k} \\ 0 \end{bmatrix} \tag{5.20}$$

where $y_{on,k}$ is the y-axis value of the middle of the road.

Next we compute the likelihood of the vehicle continuing on the road or departing from it. For that, we employ n_ϕ modes $M_{\phi,k}^{j}$, for the following set of propagation angles:

$$\{\phi^j\}_{j=1}^{n_\phi} = \{\phi^1, \ldots, \phi^{n_\phi}\} \tag{5.21}$$

where ϕ^j is measured anti-clockwise from the road. As a convention we always set $\phi^1 = 0^\circ$ accounting for the on-road propagation. The nominal positions $\mathbf{x}_{\phi,k}^{j-}$ of the road-prediction modes $M_{\phi,k}^{j}$ are given by the following relation:

$$\mathbf{x}_{\phi,k}^{j-} = \begin{bmatrix} x_{on,k-1} + (x_{on,k} - x_{on,k-1}) \cdot \cos\phi^j \\ y_{on,k-1} + (x_{on,k} - x_{on,k-1}) \cdot \sin\phi^j \\ \dot{x}_{on,k-1} \cdot \cos\phi^j \\ \dot{x}_{on,k-1} \cdot \sin\phi^j \end{bmatrix} \tag{5.22}$$

where $j \in \{1...n_\phi\}$. The probability of each mode j is then computed:

$$\tilde{p}^j_{\phi,k} = p(M^j_{\phi,k}|\mathbf{z}_k) = \mathcal{N}(\mathbf{z}_k; \mathbf{h}(\mathbf{x}^{j-}_{\phi,k}), \mathbf{R}_k) \tag{5.23}$$

where $\mathbf{h}(\cdot)$ is defined in (4.3). The normalised probabilities are:

$$p^j_{\phi,k} = \frac{\tilde{p}^j_{\phi,k}}{\sum_{\zeta=1}^{n_\phi} \tilde{p}^\zeta_{\phi,k}} \tag{5.24}$$

We then use a weighted sum of the *varying* $p^j_{\phi,k}$ and the *fixed* prior probability $\bar{p}$:

$$\tilde{p}^j_k = \begin{cases} w_p \cdot \bar{p} + (1 - w_p) \cdot p^j_{\phi,k}, & j = 1 \text{ (on-road)} \\ w_p \cdot (1 - \bar{p})/(n_\phi - 1) + (1 - w_p) \cdot p^j_{\phi,k}, & j \neq 1 \text{ (off-road)} \end{cases} \tag{5.25}$$

where $0 \leq w_p \leq 1$ is a user defined parameter. A value of w_p closer to 1 weights more the prior $\bar{p}$ whereas closer to 0 more the measurement-dependent $p^j_{\phi,k}$. The final normalised mode probability is given by:

$$p^j_k = \frac{\tilde{p}^j_k}{\sum_{\zeta=1}^{n_\phi} \tilde{p}^\zeta_k} \tag{5.26}$$

We use p^j_k as the *gamma* metric from (5.7) to calculate the number of the particles $N^j_{\phi,k}$ that we will allocate to each mode $M^j_{\phi,k}$:

$$N^j_{\phi,k} = \begin{cases} 1, & j = 1 \text{ (on-road)} \\ p^j_k \cdot N_{on,k-1}, & j \neq 1 \text{ (off-road)} \end{cases} \tag{5.27}$$

where $N_{on,k-1}$ is the *nominal* number of the on-road particles at $k-1$ (as we will see later the resampling step spawns temporarily $N_{on,k}$ on-road particles, which are finally discarded). As described before, for the on-road submode ($j = 1$) we always use one particle.

Next, according to p^j_k, we predict a number of particles off the road. First, we generate the particles required by sampling the on-road state pdf ($\mathbf{P}_{on,k-1}$), derived from the KF at the previous time instant:

$$\{\mathbf{x}^i_{off+,k}\}_{i=1}^{N_{off+,k}} = \{[x^i_{off+,k}\ \dot{x}^i_{off+,k}]^T\}_{i=1}^{N_{off+,k}} \sim \mathcal{N}(\mathbf{x}_{on,k-1}, \mathbf{P}_{on,k-1}) \tag{5.28}$$

where $N_{off+,k} = \sum_{j=2}^{n_\phi} N^j_{\phi,k}$. The new-born particles $\{\mathbf{x}^i_{off+,k}\}_{i=1}^{N_{off+,k}}$ which initially lie on the

road, are propagated off the road according to the mode departure angles $\{\phi^j\}_{j=1}^{n_\phi}$, using the relation below:

$$\mathbf{x}_{off+,k}^{ij} = \begin{bmatrix} (x_{off+,k}^i \cdot \tan\phi^j - x_{on,k-1} \cdot \tan(\phi^j/2))/(\tan\phi^j - \tan(\phi^j/2)) \\ \frac{\tan\phi^j \cdot (x_{off+,k}^i \cdot \tan\phi^j - x_{on,k-1} \cdot \tan(\phi^j/2))}{(\tan\phi^j - \tan(\phi^j/2))} - x_{off+,k}^i \cdot \tan\phi^j + y_{on,k-1} \\ \dot{x}_{off+,k}^i \cdot \cos\phi^j \\ \dot{x}_{off+,k}^i \cdot \sin\phi^j \end{bmatrix} \tag{5.29}$$

The resulting particle set is:

$$\{\mathbf{x}_{off+,k}^{i\star}\}_{i=1}^{N_{off+,k}} = \{\{\mathbf{x}_{off+,k}^{ij}\}_{i=1}^{N_{\phi,k}^j}\}_{j=2}^{n_\phi} \tag{5.30}$$

A.ii Prediction of the off-road particles

We continue with the second phase and we predict the particles which were off-road at $k-1$. Consider that we have $N_{off,k}$ such particles. These are propagated using the off-road prediction scheme of the VSMMPF. Using equation (4.8) we preliminary propagate them obtaining $\{\mathbf{x}_{off,k}^{i-}\}_{i=1}^{N_{off,k}}$. We apply then equations (4.10-4.11) and from the resulting mode probabilities we infer if the particles will continue off-road or will enter the road. If a particle stays off-road we use (4.13-4.14); if not, we set its position at the shortest point on the road, rotating its velocity randomly left or right. The predicted particles from this phase are denoted as $\{\mathbf{x}_{off,k}^{i\star}\}_{i=1}^{N_{off,k}}$.

The resulting set of the particles from the prediction step finally becomes:

$$\{\mathbf{x}_k^{i\star}\}_{i=1}^{N_{v,k}} = \{\mathbf{x}_{on,k}^{\star}, \{\mathbf{x}_{off,k}^{\zeta\star,R}\}_{\zeta=1}^{N_{off,k}^R}, \{\mathbf{x}_{off,k}^{\zeta\star,L}\}_{\zeta=1}^{N_{off,k}^L}, \{\mathbf{x}_{off,k}^{\zeta\star}\}_{\zeta=1}^{N_{off,k}}\} \tag{5.31}$$

after partitioning the set of the off-road particles $\{\mathbf{x}_{off+,k}^{i\star}\}_{i=1}^{N_{off+,k}}$ that originated from the road, into two subsets with particles that lie (clockwise) right and (anti-clockwise) left from the road:

$$\{\mathbf{x}_{off+,k}^{i\star}\}_{i=1}^{N_{off+,k}} = \{\{\mathbf{x}_{off,k}^{\zeta\star,R}\}_{\zeta=1}^{N_{off,k}^R}, \{\mathbf{x}_{off,k}^{\zeta\star,L}\}_{\zeta=1}^{N_{off,k}^L}\} \tag{5.32}$$

where it holds $N_{off+,k} = N_{off,k}^R + N_{off,k}^L$. In relation (5.31), $N_{v,k}$ stands for the total number of particles that the VMPF uses at the specific time instant k:

$$N_{v,k} = 1 + N_{off+,k} + N_{off,k} \tag{5.33}$$

B Update step

At the beginning of the update step we compute the particle weights. For simplicity we compute approximately the weights using the likelihood in the normal VSMMPF fashion[3]:

$$\tilde{w}_k^i = p(\mathbf{z}_k|\mathbf{x}_k^{i\star}) = \mathcal{N}(\mathbf{z}_k; \mathbf{h}(\mathbf{x}_k^{i\star}), \mathbf{R}_k) \tag{5.34}$$

which we normalise:

$$w_k^i = \frac{\tilde{w}_k^i}{\sum_{j=1}^{N_{v,k}} \tilde{w}_k^j} \tag{5.35}$$

where in analogy with (5.32) we obtain:

$$\{w_k^i\}_{i=1}^{N_{v,k}} = \{w_{on,k},\ \{w_{off,k}^{\zeta,R}\}_{\zeta=1}^{N_{off,k}^R},\ \{w_{off,k}^{\zeta,L}\}_{\zeta=1}^{N_{off,k}^L},\ \{w_{off,k}^{\zeta}\}_{\zeta=1}^{N_{off,k}}\} \tag{5.36}$$

At this point we calculate the masses. Just for illustration we present once more the relation (5.11) which we use to compute the masses:

$$m_{\alpha\to\beta,k} = p_{\alpha\to\beta} \cdot \frac{N_{\alpha,k-1}}{N'_{\alpha\to\beta,k}} \tag{5.11}$$

The particles obtain a mass according to the subset in which they belong. The mass of the on-road particle is:

$$m_{on,k} = \bar{p} \cdot \frac{N_{on,k-1}}{1} \tag{5.37}$$

since at $k-1$ we *nominally* had $N_{on,k-1}$ particles on road (see later), $\bar{p}$ was the probability for the particles to remain on-road and at k the current mode employs one particle.

The masses of the particles that were predicted departing from the road are:

$$m_{off,k}^R = \frac{1-\bar{p}}{2} \cdot \frac{N_{on,k-1}}{N_{off,k}^R} \tag{5.38}$$

$$m_{off,k}^L = \frac{1-\bar{p}}{2} \cdot \frac{N_{on,k-1}}{N_{off,k}^L} \tag{5.39}$$

using the same logic as before, at $k-1$ we had $N_{on,k-1}$ particles on the road, $(1-\bar{p})/2$ was the probability for the particles to exit either right of left the road and at k we have respectively

[3]The specific use of the Kalman filter for predicting the on-road particles (on/off the road), makes a full analytical derivation for the weights intractable. The likelihood *approximation* is justified here, since the main aim of this part of the chapter is the illustration itself of the use of the variable-mass approach, through a vehicle tracking application.

$N^R_{off,k}$ and $N^L_{off,k}$ particles.

For the particles that were off-road at $k-1$, using a varying-mass analogy, we argue that their prediction was within a single mode and consequently they are set with unitary masses:

$$m_{off,k} = 1 \cdot \frac{N_{off,k-1}}{N_{off,k}} = 1 \tag{5.40}$$

We derive then the *scaled* weights of the particles by multiply them with their corresponding masses:

$$\tilde{w}'_{on,k} = m_{on,k} \cdot w_{on,k} \tag{5.41}$$

$$\{\tilde{w}'^{i,R}_{off,k}\}_{i=1}^{N^R_{off,k}} = m^R_{off,k} \cdot \{w^{i,R}_{off,k}\}_{i=1}^{N^R_{off,k}} \tag{5.42}$$

$$\{\tilde{w}'^{i,L}_{off,k}\}_{i=1}^{N^L_{off,k}} = m^L_{off,k} \cdot \{w^{i,L}_{off,k}\}_{i=1}^{N^L_{off,k}} \tag{5.43}$$

$$\{\tilde{w}'^{i}_{off,k}\}_{i=1}^{N_{off,k}} = m_{off,k} \cdot \{w^{i}_{off,k}\}_{i=1}^{N_{off,k}} \tag{5.44}$$

which are subsequently normalised to sum to 1:

$$w'^i_k = \frac{\tilde{w}'^i_k}{\sum_{j=1}^{N_{v,k}} \tilde{w}'^j_k} \tag{5.45}$$

where:

$$\{\tilde{w}'^i_k\}_{i=1}^{N_{v,k}} = \{\tilde{w}'_{on,k},\ \{\tilde{w}'^{\zeta,R}_{off,k}\}_{\zeta=1}^{N^R_{off,k}},\ \{\tilde{w}'^{\zeta,L}_{off,k}\}_{\zeta=1}^{N^L_{off,k}},\ \{\tilde{w}'^{\zeta}_{off,k}\}_{\zeta=1}^{N_{off,k}}\} \tag{5.46}$$

The state estimate at k is finally given by the weighted sum of the particles:

$$\hat{\mathbf{x}}_k = \sum_{i=1}^{N_{v,k}} w'^i_k \mathbf{x}^{i\star}_k \tag{5.47}$$

C Resampling step

The next step is to resample the weighted particle set to discard particles with small weights. The order of the particles and their weights should remain unaltered as in (5.32) and (5.36). We use the systematic resampling algorithm, modified accordingly for the VMPF (see next for the pseudo-code). Its characteristic now is that it treats the on-road particle as the *parent* of multiple particles with the same states, with multiplicity proportional to the on-road mass $m_{on,k}$. For this reason, we use the unscaled versions of the weights as computed in (5.35). After resampling,

Algorithm 10 VMPF Resampling

Set nominal number of on-road particles: $N_{on,k}^{res} = N_f - N_{v,k} + 1$
Initialise the cumulative density function (cdf) of the weights: $c_1 = w_k^1$
for $i = 2 : N_{on,k}^{res}$ **do**
  Construct cdf: $c_i = c_{i-1} + c_1$
end for
for $i = (N_{on,k}^{res} + 1) : N_f$ **do**
  Construct cdf: $c_i = c_{i-1} + w_k^{(i-N_{on,k}^{res}+1)}$
end for
Start at the bottom of the cdf: $i = 1$
Draw a starting point: $u_1 \sim \mathcal{U}[0, c_{N_f}/N_f]$
for $j = 1 : N_f$ **do**
  Move along the cdf: $u_j = u_1 + (c_{N_f}/N_f) \cdot (j-1)$
  while $u_j > c_i$ **do**
    $i = i + 1$
  end while
  if $i < N_{on,k}^{res} + 1$ **then**
    Assign sample: $x_k^j = x_k^{i\star}$
  else
    Assign sample: $x_k^j = x_k^{(i-N_{on,k}^{res}+1)\star}$
  end if
end for

the size of the resulting resampled particle set $\{\mathbf{x}_k^i\}_{i=1}^{N_f}$ is increased from $N_{v,k}$ to N_f and all particles obtain equal weights and masses.

The final step of VMPF is to re-estimate the states of the on-road particle, accounting for particles that might have entered the road. Let us assume that after resampling $N_{on,k}$ particles lie on the road $\{\mathbf{x}_{on,k}^{i,r}\}_{i=1}^{N_{on,k}}$. The corrected states of the on-road particle will be:

$$\mathbf{x}_{on,k} = \frac{1}{N_{on,k}} \cdot \sum_{i=1}^{N_{on,k}} \mathbf{x}_{on,k}^{i,r} \tag{5.48}$$

Since we are using one on-road particle, we just forward the $\mathbf{x}_{on,k}$ to the next time step $k+1$ and discard the remaining $N_{on,k}$ on-road particles.

5.4 Simulation results

In this section we study the performance of the tracking algorithms using the road structure of figure 5.10. For a fair comparison we use the same parameters as in [82, 84]. The vehicle is

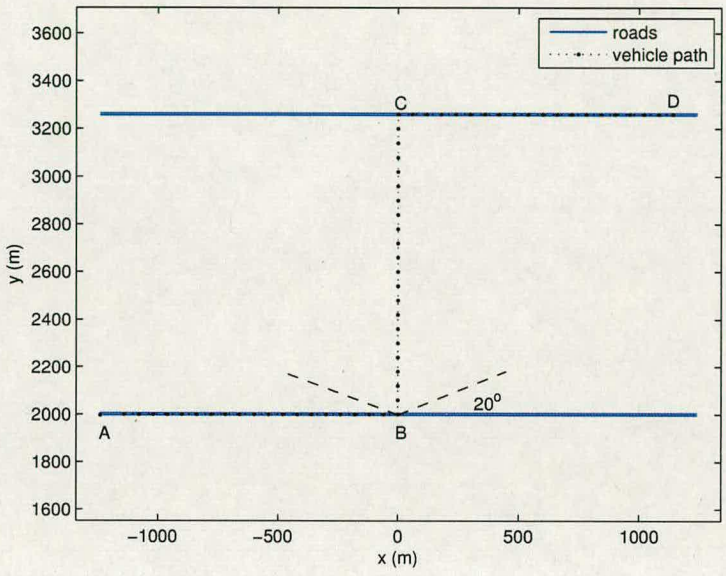


Figure 5.10: *The road map of the simulation scenario. Although the figure presents a constant velocity ABCD path and a 90° departure angle, for the simulation runs the velocity is perturbed with random accelerations and the departure angle varies randomly between 20°-160°.*

moving along points A, B, C and D, on-road along segments AB and CD and off-road along BC. In the MC runs that we perform, we vary the angle of departure φ randomly uniformly between $20^\circ < \varphi < 160^\circ$. The total simulation steps are 60 (20 for each segment) and the radar update rate is $T = 5$s. The width of the road is 8m.

The nominal velocity of the vehicle is 12m/s which on-road is perturbed along its direction by random accelerations with standard deviation $\sigma\{u_{\alpha,k}\} = 0.6\text{m/s}^2$. The radar has angular accuracy 0.5° and range resolution 20m. The standard deviation of the process noise is set $\sigma\{u_{x,k}\} = \sigma\{u_{y,k}\} = 0.6\text{m/s}^2$ (off-road) and $\sigma\{u_{o,k}\} = 0.0001\text{m/s}^2$ (orthogonal to the road). We set the mode probabilities $\bar{p} = p^\star = 0.98$ and the threshold $\tau = 18.75$. For the VMPF we set $w_p = 0.5$ from relation (5.25) to, weighting thus equally the prior and the measurement-dependent mode probabilities. A smaller w_p value would improve the transition from on- to off-road and worsen the on-road performance; for a larger value the opposite would hold.

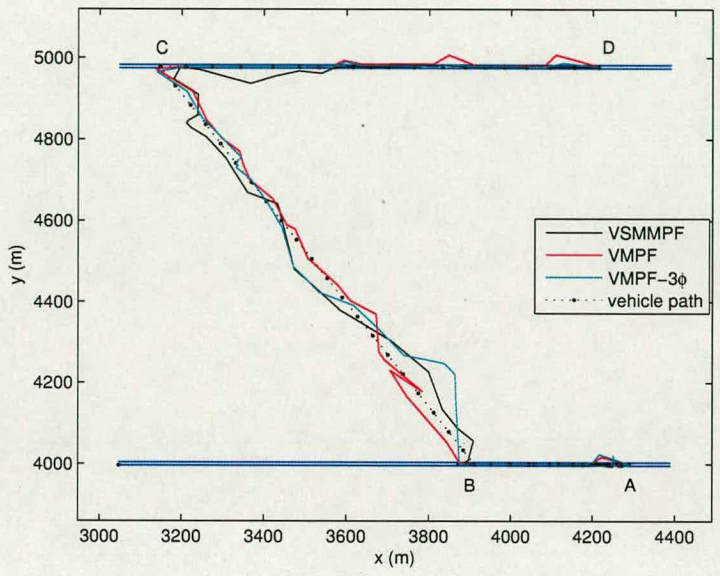


Figure 5.11: *The estimated and true vehicle track for a representative example in which the angle of departure was 128° and* $N_f = 50$.

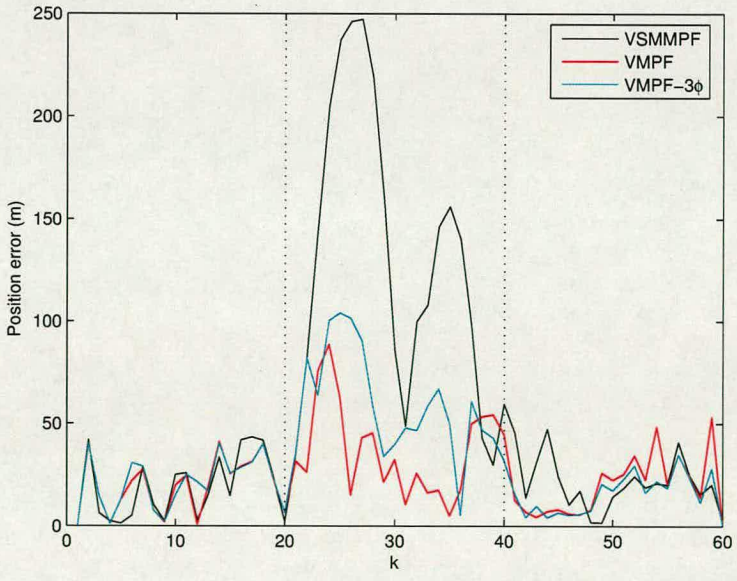


Figure 5.12: *Comparison of the position error of the algorithms for the example above. The dotted lines indicate the off-road interval.*

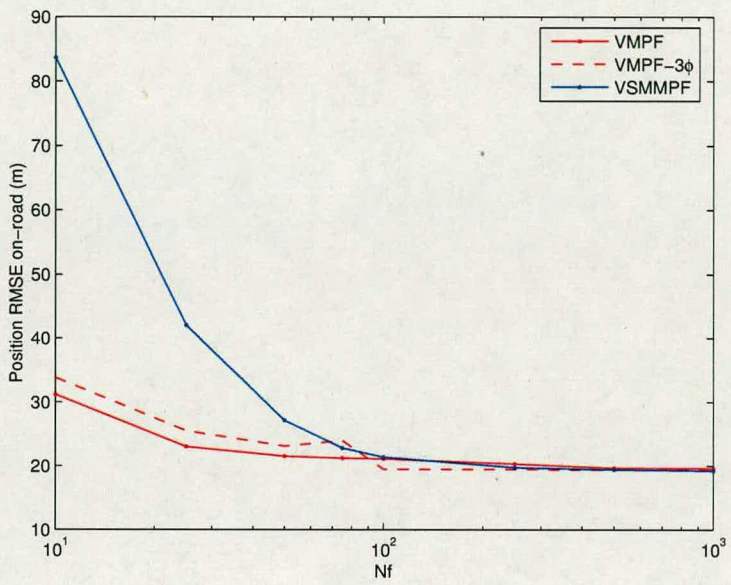


Figure 5.13: *Comparison of the RMS position error when the vehicle is on-road, over the nominal number of particles N_f.*

We use a VSMMPF, a VMPF with $n_\phi = 3$ (which we call VMPF$_{3\phi}$) and a VMPF with $n_\phi = 7$:

$$\{\phi^j\}_{j=1}^{3} = \{0^\circ, 90^\circ, 270^\circ\}, \quad \{\phi^j\}_{j=1}^{7} = \{0^\circ, 45^\circ, 90^\circ, 125^\circ, 225^\circ, 270^\circ, 315^\circ\} \tag{5.49}$$

The performance gains of the VMPF$_{3\phi}$ are solely due to its varying-mass structure and of the VMPF come as well from the more departure angles it considers. For our analysis we vary the nominal number of the particles of the trackers: $N_f = 10, 25, 50, 75, 100, 250, 500, 1000$. For every N_f we perform 3000 MC runs and we measure the on- and off-road RMS position error, the maximum value of the position error during the error overshoot when the vehicle departs from the road, the number of the particles that VMPF uses and the on-road CPU time. All algorithms were initialised by randomly seeding particles about the true states.

Figures 5.11 and 5.12 present respectively the vehicle tracks and the RMS position error of the three trackers, in a representative example in which $N_f = 50$ and $\varphi = 128^\circ$. For the particular run, when the vehicle was on the road, both VMPF and VMPF$_{3\phi}$ employed about half of the particles that the VSMMPF used. From the figures we observe that although all algorithms

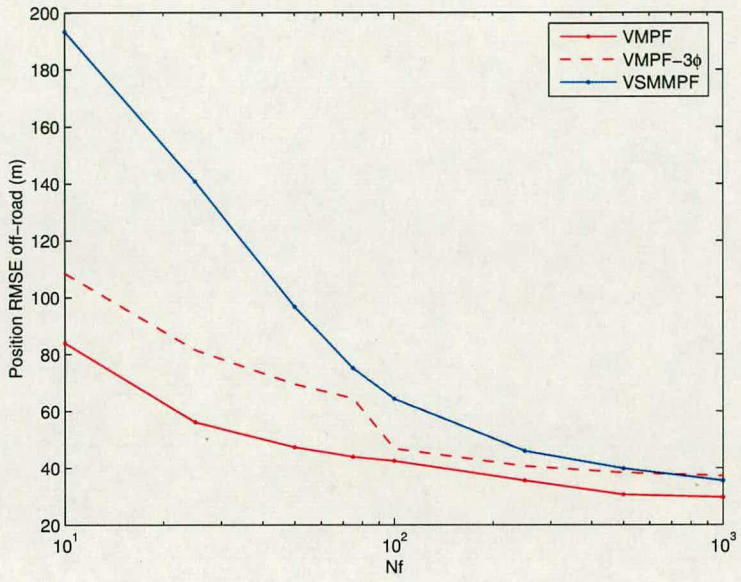


Figure 5.14: *Comparison of the RMS position error when the vehicle is off-road, over the nominal number of particles N_f*

attained a similar performance on-road, when the vehicle departed from the road the transient response of the VSMMPF was considerably slower and less accurate.

Figure 5.13 shows the overall MC results for the on-road RMS position error over the nominal number of the particles N_f. The VMPF demonstrates better performance than the VSMMPF for $N_f < 138$, while for larger values it converges to a slightly sub-optimal (1.1% for $N_f = 1000$) RMSE. Compared to the VMPF$_{3\phi}$, the VMPF has smaller RMSE for $N_f < 90$ because it uses more road-exit sub-modes and thus more particles. For $N_f > 90$ the on-road VMPF$_{3\phi}$ performance is better, because the fact that it considers just $\pm 90^o$ road-exit turns, as N_f increases, make it more robust to measurement noise. The VMPF$_{3\phi}$ improvement of the performance over the VSMMPF for $N_f > 83$, is due to the on-road Kalman filtering propagation mechanism.

From figures 5.14 and 5.15 we witness that the off-road transient response of the VMPF during road segment BC is overall superior. We remind here that when the vehicle is off-road, the estimation schemes for both VMPF and VSMMPF converge to the same unconstrained sequential importance resampling particle filter. The difference in performance that we observe is the result of the different mechanisms for propagating off the road the on-road vehicle. From

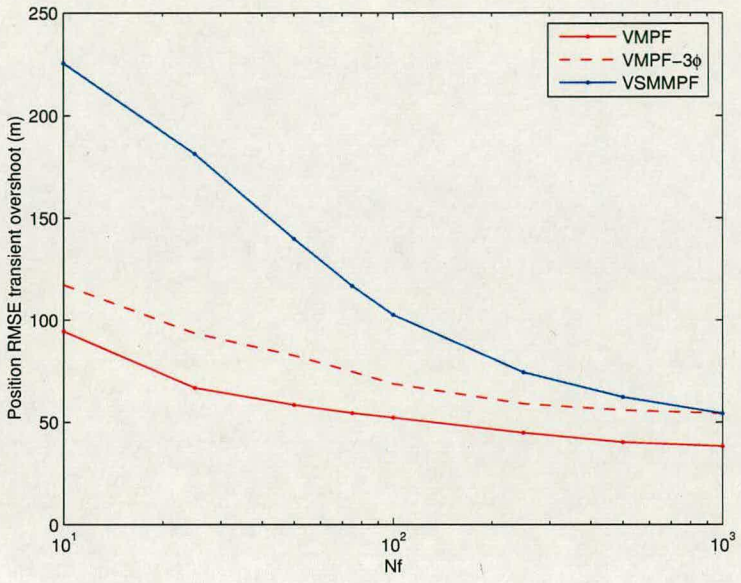


Figure 5.15: *Comparison of the RMS position error overshoot when the vehicle departs from the road, over the nominal number of particles N_f.*

figure 5.15 we see that even when $N_f = 1000$, the VMPF has 36% smaller overshoot than the VSMMPF. Once more, the VMPF$_{3\phi}$ response demonstrates which amount of performance improvement comes just from the varying-mass particles technique.

Figure 5.16 shows the percentage of the particles that the VMPF and VMPF$_{3\phi}$ use over the nominal number of particles N_f. When the vehicle is on-road the algorithms use respectively about 33%-41% and 19%-29% of the N_f. When the vehicle exits the road they rapidly increase their number of particles until reaching N_f. For continuing our analysis, we define the particle efficiency f of VMPF over VSMMPF as *the ratio of the number of the VSMMPF particles to the VMPF particles for a given performance*. For example $f(20) = 2$ for on-road RMSE, indicates that the VSMMPF employs 2 times more particles than the VMPF, when both attain a 20m on-road RMSE. Using figures 5.13, 5.14, 5.15 and 5.16, we calculate f for the various performance metrics. The results are presented in table 5.1 and clearly demonstrate the efficiency of the proposed algorithm. In the studied scenario, the VSMMPF uses up to 14.69 times more particles than the VMPF for achieving the same performance, in the RMSE ranges within which f could be calculated.

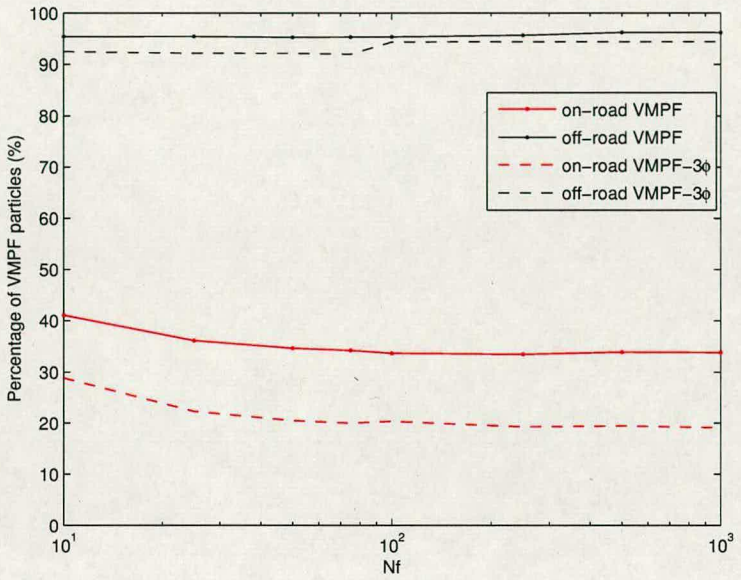


Figure 5.16: *The percentage of the particles of the VMPF and VMPF$_{3\phi}$ to the particles of the VSMMPF, when the vehicle is on- and off-road, over the nominal number of particles N_f.*

Figure 5.17 compares the on-road CPU time of the algorithms (run on a Linux platform with an Intel Xeon 3GHz processor and a 1GB DDR2 memory). For $N_f < 40$ the VMPF trades off its on-road performance superiority compared to the VSMMPF with computing power. For larger values of N_f the VMPF is computationally cheaper and has a CPU time linearly related to the N_f. On the road, depending on the N_f, VMPF$_{3\phi}$ requires 6%-23% less CPU time than the VMPF, while using on average almost half of the particles (figure 5.16). Off the road all algorithms have the same computational demands. Finally, on the robustness of the algorithms, we observe poor performance of the VSMMPF for $N_f = 10$ and 25, where it resulted respectively in 40.5% and 9.1% diverged[4] runs (respectively 8.1 and 3.7 times more than the VMPF). Nevertheless, for bigger -and more realistic- values of N_f, all algorithms did demonstrate a robust performance. All the simulation results presented in this section, were calculated just from the converged runs.

[4] An algorithm was considered to be diverged if at any point its position error exceeded 600m.

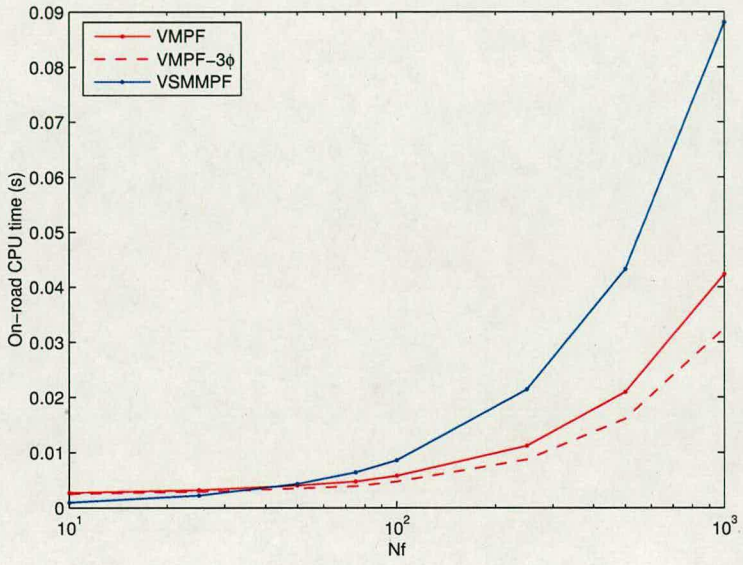


Figure 5.17: *Comparison of the CPU time when the vehicle is on-road, over the nominal number of particles N_f.*

RMSE on-road				
RMSE (m)	19.58	23.43	27.29	31.14
VSMMPF number of particles - N_f	339.77	70.62	49.62	41.46
VMPF average number of particles	337.51	8.62	5.26	4.10
Particle efficiency f	1.01	8.19	9.43	10.11
RMSE off-road				
RMSE (m)	35.55	51.66	67.77	83.88
VSMMPF number of particles - N_f	1000.00	188.19	91.18	63.62
VMPF average number of particles	240.93	33.72	16.26	9.54
Particle efficiency f	4.15	5.58	5.61	6.67
RMSE transient overshoot				
RMSE (m)	54.20	67.61	81.01	94.42
VSMMPF number of particles - N_f	1000.00	369.23	201.82	130.30
VMPF average number of particles	72.64	25.13	14.86	9.54
Particle efficiency f	13.77	14.69	13.58	13.66

Table 5.1: *Particle efficiency: the ratio of the number of the VSMMPF particles to the VMPF particles for a given performance. We focus on the RMS position error when the vehicle is on-road and off-road, and on the RMS transient overshoot when the vehicle departs from the road.*

5.5 Chapter summary

This work introduced the variable mass particle filter and used the terrain-aided tracking problem for comparing it with the variable structure multimodel particle filter. Both algorithms exploit generic multi-model particle filtering structures which differ in their mode-switching and particle allocation mechanisms. For switching between its modes, the VSMMPF uses a fixed prior mode probability, while the VMPF employs an adaptive scheme involving varying posterior measurement-dependent mode probabilities and variable mass particles. For the specific vehicle tracking problem that we study, the VMPF uses furthermore a reduced-dimension Kalman filter for its on-road mode and considers more angles for road departure.

Simulation results demonstrated the improved efficiency of the VMPF, since generally required fewer particles than the VSMMPF for achieving a better estimation accuracy. The variable-mass architecture enabled the vehicle tracker to incorporate efficiently the measurement information within the particle allocation mechanism, which in turn resulted in a better transitional response when the vehicle was departing from the road. Moreover, the Kalman-based single-particle technique for on-road tracking reduced the on-road computational demands of the algorithm. Based on our simulation results, we can argue that the variable-mass approach can be proved a useful feature of any multi-mode particle filter, allowing for a direct exploitation of available information within the particle allocation mechanism and resulting consequently in a better characterisation of the posterior state distribution.

$N_f = 10$					
	RMSE on-road (m)	RMSE off-road (m)	Transient overshoot (m)	Particles vehicle on-road	Particles vehicle off-road
VSMMPF	83.63	193.22	225.40	10.00	10.00
VMPF	31.14	83.88	94.42	4.10	9.54
$N_f = 25$					
	RMSE on-road (m)	RMSE off-road (m)	Transient overshoot (m)	Particles vehicle on-road	Particles vehicle off-road
VSMMPF	41.98	140.70	181.21	25.00	25.00
VMPF	22.96	56.08	66.67	9.02	23.86
$N_f = 50$					
	RMSE on-road (m)	RMSE off-road (m)	Transient overshoot (m)	Particles vehicle on-road	Particles vehicle off-road
VSMMPF	27.12	96.63	139.71	50.00	50.00
VMPF	21.47	47.25	58.38	17.29	47.64
$N_f = 75$					
	RMSE on-road (m)	RMSE off-road (m)	Transient overshoot (m)	Particles vehicle on-road	Particles vehicle off-road
VSMMPF	22.79	75.06	116.65	75.00	75.00
VMPF	21.16	43.98	54.32	25.61	71.50
$N_f = 100$					
	RMSE on-road (m)	RMSE off-road (m)	Transient overshoot (m)	Particles vehicle on-road	Particles vehicle off-road
VSMMPF	21.36	64.32	102.53	100.00	100.00
VMPF	21.06	42.47	52.10	33.62	95.36
$N_f = 250$					
	RMSE on-road (m)	RMSE off-road (m)	Transient overshoot (m)	Particles vehicle on-road	Particles vehicle off-road
VSMMPF	19.72	45.97	74.45	250.00	250.00
VMPF	20.28	35.60	44.71	83.56	239.22
$N_f = 500$					
	RMSE on-road (m)	RMSE off-road (m)	Transient overshoot (m)	Particles vehicle on-road	Particles vehicle off-road
VSMMPF	19.39	39.82	62.30	500.00	500.00
VMPF	19.63	30.64	40.06	169.29	481.11
$N_f = 1000$					
	RMSE on-road (m)	RMSE off-road (m)	Transient overshoot (m)	Particles vehicle on-road	Particles vehicle off-road
VSMMPF	19.21	35.55	54.19	1000.00	1000.00
VMPF	19.57	29.67	38.17	337.51	962.03

Table 5.2: *Performance comparison between the VSMMPF and the VMPF for various N_f. The results were averaged after 3000 Monte Carlo runs for every N_f.*

Chapter 6
Conclusions

Chapter 6 summarises the work presented in the thesis and highlights the main research achievements. It discusses about the limitations of our work and proposes some possible areas for further research.

6.1 Summary and achievements of work

Throughout the thesis we concentrated on particle filtering focusing on the track estimation problem. Our basic research orientation had been on designing tracking algorithms which could exhibit an improved particle efficiency. For achieving that we proposed techniques that either varied appropriately the number of the particles or predicted the particles in state space areas which had greater significance in characterising the posterior state distribution. Although we dealt primarily with the single target problem, we proposed also two algorithms for the multiple target case.

In chapter 2 and in the first part of chapter 3 we presented background material and a literature review on Bayesian estimation, target tracking and particle filtering. We described conventional Kalman filtering estimation methods, basic target kinematics and radar models, measurement-to-track assignment techniques and standard particle filtering algorithms. We included several short simulation studies for contrasting the presented methods.

Next we briefly studied the problem of tracking a target which could dynamically change its kinematic behaviour by executing unpredictable abrupt turns, a problem common to the vehicle tracking literature. Rather than using a multimodel algorithm we examined the suitability of the ALLPF which adopted the auxiliary mechanism of the ASIR and used the local linearization predicting logic of the LLPF. Simulation studies quantified the performance gains and drawbacks of the ALLPF when compared to the SIR and the ASIR. The experimental results suggested that the ALLPF could be used in non-demanding tracking systems which are more concerned on the track-maintenance capability than purely the estimation accuracy.

We then examined an new approach which utilised the measurement within the particle prediction and association mechanisms of a multitarget tracker. The specific technique employed the local linearization method for propagating the particles for all different association hypotheses before the actual association phase. The association function adopted a nearest neighbour logic to assign the measurements to the targets. The problem that we had to overcome was mainly computational, since for every target we had to predict multiple times its particles before the association. We addressed that by using a mechanism which allowed the algorithm to vary the number of the predicted particles according to the association difficulty. We compared the proposed A-MLLPF with the equivalent SIR-based algorithm, the MSIR, in a simulation scenario with two crossing targets. The A-MLLPF exhibited and improved performance both in terms of estimation accuracy and association capability. Additionally, we showed that its varying-particle approach decreased significantly the particle requirements with a minimum performance degradation.

Chapters 4 and 5 were entirely dedicated to the vehicle tracking problem. We studied there multiple-model particle filters which exploited road information, basing our work on the VS-MMPF. In the first part of chapter 4 we presented a novel variation of the VSMMPF which enabled the algorithm to vary its particles according to the tracking difficulty. The key idea was to use fewer particles in "easy" state subspaces so as to reduce the computational cost. Specifically for our vehicle tracking application, the proposed VP-VSMMPF employed a smaller number of particles when the vehicle was travelling along the roads on the monitoring ground scene. Simulation results demonstrated the on-road efficiency of the proposed method, since even though the tracker was using significantly fewer particles than the standard VSMMPF, it managed to attain a very similar tracking performance. These results revealed a certain inefficiency of the VSMMPF to cope with the system multimodality and made us investigate further the problem and finally propose the variable mass approach described in chapter 5.

In the remaining of chapter 4 we suggested a multi-target structure for the VSMMPF. We enhanced the originally single-target VSMMPF with a measurement-to-track logic, enabling it to track simultaneously multiple vehicles within a measurement cluttered environment. As customary to multi-target tracking, for minimising the workload of the data association function we exploited a gating technique. For that, we formed varying-volume ellipsoidal gates around the measurements, which would discard highly unlikely assignment pairings. Our gating approach was based on the vehicle's particles to allow the function to account for the multimodality of

the vehicle's motion dynamics. For measurements-to-track assignment we utilised the JPDA method. A simulation analysis demonstrated the suitability of the proposed MGTPF approach to the multiple vehicle environment, the efficiency of the gating scheme and quantified the improved performance both in terms of estimation and association accuracy over the equivalent unconstrained particle filter which exploited the same gating and association features.

In chapter 5 we presented a different approach to deal with the inherent multi-modality of the vehicle tracking problem in an attempt to improve the particle efficiency. The key contribution of the work there was the use of particles with variable masses. Whereas in the VSMMPF the number of the particles allocated to its modes was proportional to *fixed* mode probabilities, in the proposed VMPF that number was allowed to *vary* according to arbitrary user-defined criteria. The VMPF compensated for the statistical irregular particle patterns by rescaling appropriately the mode particles using the masses.

The variable-mass approach allowed the proposed vehicle tracker to exploit information from the measurement and the mode difficulty for allocating its particles to the modes. Thus more particles were allocated to the most probable and/or difficult modes for improving the tracking accuracy and furthermore modes which were less probable and/or had easier dynamics obtained fewer particles for reducing the computational requirements. Other features of the proposed vehicle tracker were an on-road propagation mechanism which used just one particle and a Kalman filter (KF) for reducing the computational demands and a technique which enabled the algorithm to deal with random road departure angles.

Simulation results demonstrated the improved efficiency of the VMPF, since in general required fewer particles than the VSMMPF for achieving a better estimation accuracy. The variable-mass architecture enabled the vehicle tracker to incorporate efficiently the measurement information within the particle allocation mechanism which in turn resulted in a better transitional response when the vehicle was departing from the road. Moreover, the on-road propagation mechanism reduced the on-road computational requirements of the algorithm. Based on our study we can suggest that the variable-mass approach can be proved a useful component for any multi-mode particle filter, allowing for a *direct exploitation* of any available information within the particle allocation mechanism and thus for a better characterisation of the posterior state distribution.

6.2 Limitations of work and scope for further research

In this last section we examine the main limitations of our work and present several areas for future research. Our first comment is on the tracking studies presented throughout the thesis. Because of the complex nature of the problem it is almost intractable to simulate a realistic tracking device. A real-world system, in addition to the actual track estimation logic, is enhanced with features accounting for sensor de-biassing and calibration, sensor fusion, track initiation, track deletion, out-of-sequence measurement processing, gating, clutter rejection, data association, target classification and risk analysis and assessment. In our work we concentrated mainly on the track estimation function and just the A-MLLPF from chapter 3 and the MGTPF from chapter 4 had basic data association capabilities. It would be thus interesting to enhance the algorithms introduced in this work with extra tracking features and test for further benefits or drawbacks.

We continue by addressing each proposed algorithm separately. We start with the ALLPF from chapter 2, for which it would be useful to develop variations with more sophisticated weighting schemes and contrast them in various model-mismatch simulation scenarios, for gaining more insight in their performance efficiency. A direct comparison with a multimodel tracker would be also interesting, for analysing the computational requirements and performance differences between the single- and multiple-model approach.

Regarding the multitarget A-MLLPF from chapter 3, we first note that due to its increased association complexity if it is to be tested in a realistic scenario, it is essential to employ a gating scheme before the association function. But still, and even with a perfect clutter rejection capability, the tracker will probably require substantial computational power in an environment with a number of very closely-separated multiple targets. An attractive solution thus would be to include a logic which depending on the association complexity would switch the local-linearization prediction technique on or off. When off, an SIR-like propagation mechanism could be used instead. Another point concerns the choice of the number of the association particles. In our analysis we varied their number according to the measurements distance using for simplicity a static profile parameterised according to our test scenario. Although this was justified for the purposes of our comparison, it would be attractive to research on a general method accounting for arbitrary target positions and system parameters.

Continuing with the VP-VSMMPF, in our simulation study for varying its particles we chose

arbitrary to use 100 particles on-road and 1000 off-road. It would be interesting to conduct further analysis which could propose a more "optimal" ratio between the on- and off-road particles for lowering more the particle redundancy. This in a way would also quantify the particle demands of the different propagation models, which could offer valuable insight for future studies.

Concerning the multi-vehicle architecture that we proposed for the VSMMPF in chapter 4, it would be interesting to apply it as well to the VP-VSMMPF and the VMPF. We could assess thus the effect of the different particle propagation mechanisms to the association capability of the tracker. Although we would not expect to witness any noteworthy differences using the VP-VSMMPF, the VMPF with its advanced tracking accuracy would most possibly improve the association performance. Regarding the proposed gating function, it would worth investigating ways to integrate the road constraints within the gates. This additional information would improve the effectiveness of the gating method and therefore would lower further the computation load from the data association phase.

For the VMPF vehicle tracker we note first that in our analysis we just considered roads that lied parallel to the x-axis. For completeness it would be useful deriving a generalisation for roads at an arbitrary angle. This would be quite straightforward mathematically since it would essentially just involve geometrical rotations of the propagation models of the particles. For the purpose of clarity as well, since we aimed mainly to examine the variable mass technique itself, we did not consider junction or bridge crossing capabilities for our tracker. The incorporation of these, especially of the first, would be necessary for dealing with a more realistic road structure scenario.

A more attractive problem would be to propose a more refined on-road propagation mechanism for the VMPF, particularly regarding the pseudo-measurement generation and the KF use. A further analysis studying the Gaussianity approximation for the measurement noise along the on-road cross section, could also assess the validity of our approach. Moreover, the comparison of the proposed tracker with a variation which instead of pseudomeasurements and the KF would use just an EKF would be interesting. We could thus examine when the linearity approximation of the EKF could result in a better performance than our Gaussianity assumption. The incorporation of the variable mass technique to other multiple model estimation applications and a performance analysis in a range of different scenarios could also be proved useful for assessing the strengths and weaknesses of the method.

References

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

[2] S. Kay, *Fundamentals of Statistical Signal Processing, Estimation Theory*. New Jersey: Prentice Hall, 1993.

[3] M. Srinath and P. Rajasekaran, *An introduction to statistical signal processing with applications*. New York: John Wiley and Sons, 1979.

[4] A. Papoulis, *Probability, random variables, and stochastic processes*. New York, NY: McGraw-Hill, 1984.

[5] E. Jaynes and G.Bretthorst, eds., *Probability Theory: The Logic of Science*. Cambridge: Cambridge University Press, 2003.

[6] A. Gelman, J. Carlin, H. Stern, and D. Rubin, *Bayesian Data Analysis*. Boca Raton: Champan and Hall/CRC, 2004.

[7] R. Kalman, "A new approach to linear filtering and prediction problem," *ASME Journal of Basic Engineering*, vol. Series D, 82, pp. 34–45, 1960.

[8] M. Grewal and A. Andrews, *Kalman Filtering Theory and Practise*. New Jersey: Prentice Hall, 1993.

[9] R. Bucy and P. Joseph, *Filtering for Stochastic Processes, with Applications to Guidance*. New York: Wiley, 1968.

[10] P. Dyer and S. McReynolds, "Extension of square-root filtering to include process noise," *Journal of Optimization Theory and Applications*, vol. 3, pp. 444–458, 1969.

[11] G. Schmidt, ed., *Practical Aspects of Kalman Filtering Implementation*, vol. AGARD-LS-82. London: Nato Advisory Group for Aerospace Research and Development, 1976.

[12] P. Maybeck, *Stochastic Models Estimation and Control*. San Diego, CA: Academic Press, 1982.

[13] H. Sorenson, ed., *Kalman filtering: theory and application*. New Jersey: IEEE Press, 1985.

[14] M. Abramowitz and I. Stegun, eds., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. New York: Dover, 1972.

[15] D. Lerro and Y. Bar-Shalom, "Tracking with debiased consistent converted measurements vs. EKF," *IEEE Transactions on Aerospace and Electronics Systems*, vol. AES-29, pp. 1015–1022, July 1993.

[16] F. Schweppe, *Uncertain Dynamic Systems*. New Jersey: Prentice Hall, 1973.

[17] S. Julier and J. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *Proc. AeroSence: The 11th International Symposium on Aerospace/Defence Sensing, Simulation and Controls*, 1997.

[18] D. Harville, *Matrix algebra from a statistician perspective*. New York: Springer-Verlag, 1971.

[19] S. Julier, J. Uhlmann, and H. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Transactions on Automatic Control*, vol. 5, pp. 477–482, March 2000.

[20] S. Julier and J. K. Uhlmann, "The scaled unscented transformation," in *Proc. of the IEEE American Control Conference*, (Anchorage AK, USA), pp. 4555–4559, May 2002.

[21] R. van der Merwe and E. Wan, "The square-root unscented Kalman filter for state and parameter-estimation," in *International Conference on Acoustics, Speech and Signal Processing*, (Utah), May 2001.

[22] E. Wan and R. van der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proc. of Symposium 2000 on Adaptive Systems for Signal Processing, Communication and Control (AS-SPCC)*, (Alberta, Canada), IEEE, October 2000.

[23] B. Stenger, P. Mendonca, and R. Cipolla, "Model-based hand tracking using an unscented Kalman filter," *Proc. of British Machine Vision Conference*, vol. 1, pp. 63–72, September 2001.

[24] Y. Chen, T. Huang, and Y. Rui, "Parametric contour tracking using unscented Kalman filter," in *Proc. of International Conference on Image Processing*, (Rochester, New York), September 2002.

[25] W. Li, H. Leung, and Y. Zhou, "Space time registration of radar and ESM using unscented Kalman filter," *IEEE Transactions on Aerospace and Electronics Systems*, vol. 40, no. 2, pp. 824–836, 2004.

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

[27] M. Skolnik, ed., *Radar Handbook*. McGraw-Hill, 1970.

[28] N. Levanon, *Radar Principles*. New York: John Wiley and Sons, 1988.

[29] *IEEE Standard Definitions of Terms for Antennas, IEEE Std 145-1993*. New York: IEEE, 1993.

[30] P. Maybeck, W. Worsley, and P. M. Flynn, "Investigation of constant turn rate dynamics for airborne vehicle tracking," in *Proc. IEEE NAECON*, pp. 896–903, May 1982.

[31] S. Blackman, *Multiple Target Tracking with Radar Applications*. Norwood: Artech House, 1986.

[32] Y. Bar-Shalom and E. Tse, "Tracking in a cluttered environment with probabilistic data association," *Automatica*, vol. II, pp. 451–460, September 1975.

[33] T. Fortmann, Y. Bar-Shalom, and M. Scheffeet, "Sonar tracking of multiple targets using joint probabilistic data association," *IEEE Journal of Oceanic Engineering*, vol. OE-8, pp. 173–184, July 1983.

[34] D. Reid, "An algorithm for tracking multiple targets," *IEEE Transactions on Automatic Control*, vol. AC-24, pp. 843–854, December 1979.

[35] X. Wang, S. Challa, and R. Evans, "Gating techniques for manoeuvring target tracking in clutter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, pp. 1087–1097, July 2002.

[36] D. Musicki and M. Morelande, "Gate volume estimation for target tracking," in *Proc. Fusion 2004*, (Stockholm), 2004.

[37] D. Bertsekas, "The auction algorithm: a distributed relaxation method for the assignment problem," *Annals of Operational Research*, vol. 14, pp. 105–123, 1988.

[38] D. Bertsekas, "The auction algorithm for assignment and other netwok flow problems: A tutorial," *Interfaces*, vol. 20, pp. 133–149, 1990.

[39] M. Padberg, *Linear Optimisation and Extensions*. Berlin: Springer-Verlag, 1999.

[40] L. Chin, "Application of neural networks aided target tracking," in *Proc. IEEE ICNN*, (Washington), Jun 1996.

[41] R. Karlsson and F. Gustafsson, "Monte Carlo data association for multiple target tracking," *IEE Workshop on Target Tracking: Algorithms and applications, Netherlands*, 2001.

[42] H. Blom and E. Bloem, "Joint IMMPDA particle filter," in *Proc. 6th International Conference in Information Fusion*, July 2003.

[43] D. Musicki, R. Evans, and S. Stankovic, "Integrated probabilistic data association," *IEEE Transactions on Automatic Control*, vol. 39, pp. 1237–1241, June 1994.

[44] A. Doucet, "On sequential simulation-based methods for Bayesian filtering," Tech. Rep. CUED/F-INFENG TR310, University of Cambridge, 1998.

[45] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proc.-F*, vol. 140, no. 2, 1993.

[46] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman filter: particle filters for tracking applications*. Norwood: Artech House, 2004.

[47] J. Liu and R. Chen, "Sequential Monte Carlo methods for dynamic systems," *Journal of the American Statistical Association*, 1998.

[48] P. Djuric, "Monte Carlo methods for signal processing: recent advances," in *Proc. EUSIPCO 2004*, (Vienna), 2004.

[49] A. Doucet and W. Wang, "Monte Carlo methods for signal processing," *IEEE Signal Processing Magazine*, pp. 152–170, November 2005.

[50] 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.

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

[52] R. Merwe, A. Doucet, N. de Freitas, and E. Wan, "The unscented particle filter," Tech. Rep. CUED/F-INFENG TR380, University of Cambridge, 2000.

[53] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing, Special Issue on Monte Carlo Methods*, vol. 50, pp. 174–188, February 2002.

[54] A. Kong, J. Liu, and W. Wong, "Sequential imputations and Bayesian missing data problems," *Journal of the American Statistical Association*, vol. 89, no. 425, pp. 278–288, 1994.

[55] M. Pitt and N. Shephard, "Filtering via simulation: auxiliary particle filters," *Journal of the American Statistical Association*, vol. 94, pp. 590–599, 1999.

[56] A. Doucet, N. de Freitas, and N. Gordon, eds., *Sequential Monte Carlo Methods in Practise*. New York: Springer-Verlag, May 2001.

[57] M. Klaas, N. de Freitas, and A. Doucet, "Toward practical N^2 Monte Carlo: the marginal particle filter," in *Proc. UAI 2005*, 2005.

[58] G. Kravaritis and B. Mulgrew, "Particle filters: sampling density for nonlinear target tracking," in *Proc. 6th IMA Conference in Mathematics in Signal Processing*, (Cirencester), December 2004.

[59] S. Godsill and J. Vermaak, "Variable rate particle filters for tracking applications," in *Proc. IEEE Workshop on Statistical Signal Processing*, (Bordeaux), July 2005.

[60] G. Kravaritis and B. Mulgrew, "A study on multitarget tracking with adaptive local linearisation particle filters," in *Proc. IEEE Workshop on Statistical Signal Processing*, (Bordeaux), July 2005.

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

[62] C. Kreucher, K. Kastella, and A. Hero, "Multitarget tracking using the joint multitarget probability density," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, pp. 1396–1414, October 2005.

[63] R. Mahler, "Multitarget Bayes filtering via first-order multitarget moments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, pp. 1152–1178, October 2003.

[64] B.-N. Vo, S. Singh, and A. Doucet, "Sequential Monte Carlo implementation of the PHD filter for multi-target tracking," in *Proc. 6th International Conference in Information Fusion*, (Australia), July 2003.

[65] D. Clark and J. Bell, "Convergence results for the particle PHD filter," *IEEE Transactions on Signal Processing*, vol. 54, July 2006.

[66] K. Panta, B.-N. Vo, S. Singh, and A. Doucet, "Probability hypothesis density filter versus multiple hypothesis tracking," in *Proc. SPIE Conference 2004*, (Florida), April 2004.

[67] K. Giholm, S. Godsill, S. Maskell, and D. Salmond, "Poisson models for extened target and group tracking," in *Signal and Data Processing of Small Targets, Proc. of SPIE*, (August), San Diego 2005.

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

[69] S. Sarkka, A. Vehtari, and J. Lampinen, "Rao-Blackwellized Monte Carlo data association for multiple target tracking," in *Proc. 7th International Conference on Information Fusion*, (Stockholm), June 2004.

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

[71] F. Gustafsson, F. Gunnarsson, N. Bergaman, U. Forssell, J. Jansson, R. Karlsson, and P. Nordlund, "Particle filters for positioning, navigation and tracking," *IEEE Transactions on Signal Processing, Special Issue on Monte Carlo Methods*, vol. 50, pp. 425–437, February 2002.

[72] D. Pham, K. Dabia, and C. Musso, "A Kalman-particle kernel filter and its application to terrain navigation," in *Proc. 6th International Conference of Information Fusion*, (Cairns), July 2003.

[73] S. Gattein and P. Vannoorenberghe, "A comparative analysis of two approaches using the road network for tracking ground target," in *Proc. 7th International Conference on Information Fusion*, (Stockholm), June 2004.

[74] L. Mihaylova and R. Boel, "A particle filter for freeway traffic estimation," in *Proc. IEEE Conf. on Decision. and Control*, (Atlantis), December 2004.

[75] R. Rad and M. Jamzad, "Real time classification and tracking of multiple vehicles in highways," *Pattern recognition letters*, vol. 26, pp. 1597–1607, 2005.

[76] D. Magee, "Tracking multiple vehicles using foreground, background and motion models," *Image and Vision Computing*, vol. 22, pp. 143–155, 2004.

[77] A. Lauberts, M. Karlsson, F. Nasstrom, and R. Aljasmi, "Ground target classification using combined radar and IR with simulated data," in *Proc. 7th International Conference on Information Fusion*, (Stockholm), June 2004.

[78] J. Kim and J. Oh, "A land vehicle tracking algorithm using stand-alone GPS," *Control Engineering Practise*, vol. 8, pp. 1189–1196, 2000.

[79] E. Blasch and C. Yang, "Ten methods to fuse GMTI and HRRR measurements for joint tracking and identification," in *Proc. 7th International Conference on Information Fusion*, (Stockholm), June 2004.

[80] W. Koch and R. Klemm, "Ground target tracking with STAP radar," *IEE Proc. on Radar, Sonar and Navigation*, vol. 148, pp. 173–185, June 2001.

[81] R. Sullivan, *Microwave Radar: imaging and advanced concepts*. Boston: Artech House, 2000.

[82] S. Arulampalam, N. Gordon, M. Orton, and B. Ristic, "A variable structure multiple model particle filter for GMTI tracking," in *Proc. 5th International Conference in Information Fusion*, (Annapolis), July 2002.

[83] T. Kirubarajan, Y. Bar-Shalom, K. Pattipati, and I. Kadar, "Ground target tracking with topography-based variable structure IMM estimator," in *SPIE Signal and Data Processing of Small Targets*, vol. 3373, (Orlando), pp. 222–233, April 1998.

[84] T. Kirubarajan, Y. Bar-Shalom, K. Pattipati, and I. Kadar, "Ground target tracking with variable structure IMM estimator," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, pp. 26–46, January 2000.

[85] H. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with Markovian switching coefficients," *IEEE Transactions on Automatic Control*, vol. 33, no. 8, 1988.

[86] Y. Bar-Shalom and X. Li, *Estimation and tracking: principles, techniques and software*. Norwood: Artech House, 1993.

[87] X. Li and Y. Bar-Shalom, "Multiple-model estimation with variable structure," *IEEE Transactions on Automatic Control*, vol. 41, pp. 478–493, April 1996.

[88] M. Mallick, S. Maskell, T. Kirubarajan, and N. Gordon, "Littoral tracking using particle filter," in *Proc. 5th International Conference on Information Fusion*, (Annapolis), July 2002.

[89] G. Kravaritis and B. Mulgrew, "Ground tracking using a variable structure multiple model particle filter with varying number of particles," in *Proc. IEEE International Radar Conference*, (Arlington), May 2005.

[90] G. Kravaritis and B. Mulgrew, "Multitarget ground tracking with road maps and particle filters.," in *Proc. IEEE International Symposium in Signal Processing and Information Theory*, (Athens), December 2005.

[91] H. Blom, "An efficient filter for abruptly changing systems," in *Proc. 23rd Conference on Decision and Control*, (Las Vegas), pp. 656–658, December 1984.

[92] G. Kravaritis and B. Mulgrew, "Hybrid variable mass particle filter for road-constrained vehicle tracking," *Journal of Applied Signal Processing*, Submitted.

[93] J.Vermaak, A. Doucet, and P. Perez, "Maintaining multimodality through mixture tracking," in *Proc. IEEE International Conference on Computer Vision*, (Nice), October 2003.