A COMPARATIVE EVALUATION OF CONVENTIONAL AND PARTICLE FILTER BASED RADAR TARGET TRACKING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

BERKİN YILDIRIM

IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

NOVEMBER 2007

Approval of the thesis:

A COMPARATIVE EVALUATION OF CONVENTIONAL AND PARTICLE FILTER BASED RADAR TARGET TRACKING

submitted by BERKİN YILDIRIM in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. Mübeccel Demirekler _____
Supervisor, **Electrical and Electronics Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Kemal Leblebicioğlu _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Mübeccel Demirekler _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Mustafa Kuzuoğlu _____
Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. Afşar Saranlı _____
Electrical and Electronics Engineering Dept., METU

Aydın Bayri (MSc.) _____
ASELSAN

Date: **30.11.2007**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Berkin Yıldırım

Signature :

ABSTRACT

A COMPARATIVE EVALUATION OF CONVENTIONAL AND PARTICLE FILTER BASED RADAR TARGET TRACKING

Yıldırım, Berkin

M. Sc., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Mübeccel Demirekler

November 2007, 153 pages

In this thesis the radar target tracking problem in Bayesian estimation framework is studied. Traditionally, linear or linearized models, where the uncertainty in the system and measurement models is typically represented by Gaussian densities, are used in this area. Therefore, classical sub-optimal Bayesian methods based on linearized Kalman filters can be used. The sequential Monte Carlo methods, i.e. particle filters, make it possible to utilize the inherent non-linear state relations and non-Gaussian noise models. Given the sufficient computational power, the particle filter can provide better results than Kalman filter based methods in many cases. A survey over relevant radar tracking literature is presented including aspects as estimation and target modeling. In various target tracking related estimation applications, particle filtering algorithms are presented.

Keywords: Particle Filter, sequential Monte Carlo methods, target tracking

ÖZ

KLASİK VE PARÇACIK SÜZGECİ TABANLI RADAR HEDEF TAKİBİNİN KARŞILAŞTIRMALI DEĞERLENDİRMESİ

Yıldırım, Berkin

Yüksek Lisans, Elektrik Elektronik Mühendisligi Bölümü
Tez Yöneticisi: Prof. Dr. Mübeccel Demirekler

Kasım 2007, 153 sayfa

Bu tezde radar hedef takip problemi Bayesian tahmin yapısında çalışılmıştır. Bu alanda geleneksel olarak sistem ve ölçüm modellerindeki belirsizliğin Gaussian yoğunluklar ile temsil edildiği doğrusal veya doğrusal hale getirilmiş modeller kullanılmaktadır. Bundan dolayı, ileri Kalman süzgecine dayalı klasik yarı optimal Bayesian yöntemleri kullanılabilmektedir. Sıralı Monte Carlo yöntemleri, bir başka değişle parçacık süzgeçleri, problemin doğasında bulunan doğrusal olmayan sistem denklemlerini ve Gaussian olmayan gürültü modellerini kullanmayı olanaklı kılar. Parçacık süzgeci yeterli hesaplama gücü sağlandığında pek çok durumda Kalman süzgeci tabanlı yöntemlerden daha iyi sonuçlar verebilmektedir. İlgili radar takip literatürü üzerinde bir araştırma tahmin ve hedef modellemesini de içerecek şekilde verilmiştir. Çeşitli hedef takip ve ilgili tahmin uygulamalarında parçacık süzgeci algoritmaları sunulmuştur.

Anahtar Kelimeler: Parçacık süzgeci, sıralı Monte Carlo yöntemleri, hedef takibi

To My Family

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor Prof. Dr. Mübeccel Demirekler for his encouragements, guidance, advice, criticism and insight throughout the research.

I would like to thank ASELSAN Inc. for facilities provided for the competition of this thesis.

I would like to forward my appreciation to all my friends and colleagues who contributed to my thesis with their continuous encouragement.

I would also like to express my profound appreciation to my family for their continuous support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

APF	:	Auxiliary Particle Filter
ASIR	:	Auxiliary Sampling Importance Resampling
ATC	:	Air Traffic Control
BOT	:	Bearings-Only Tracking
CA	:	Constant Acceleration
CCW	:	Counter-Clockwise
CDF	:	Cumulative Distribution Function
CDKF	:	Central Difference Kalman Filter
CPU	:	Central Processing Unit
CT	:	Coordinated Turn
CV	:	Constant Velocity
CW	:	Clockwise
DF	:	Direction Finding
EKF	:	Extended Kalman Filter
GPS	:	Global Positioning System
HMM	:	Hidden Markov Model
IEKF	:	Iterated Extended Kalman Filter
IMM	:	Interacting Multiple Model
IMM-EKF	:	Interacting Multiple Model Extended Kalman Filter
IMM-PF	:	Interacting Multiple Model Particle Filter
INS	:	Inertial Navigation System
KF	:	Kalman Filter
MC	:	Monte Carlo
MCMC	:	Markov Chain Monte Carlo
MM-PF	:	Multiple Model Particle Filter
MMSE	:	Minimum Mean Square Error
MPF	:	Marginalized Particle Filter

MSC	:	Modified Spherical Coordinates
pdf	:	Probability Density Function
PF	:	Parricle Filter
RPEKF	:	Range Parameterized Extended Kalman Filter
RPF	:	Regularized Particle Filter
RWR	:	Radar Warning Receiver
SIR	:	Sampling Importance Resampling
SIR-PF	:	Sampling Importance Resampling Particle Filter
SIS	:	Sequential Importance Sampling
SMC	:	Sequential Monte Carlo
SPKF	:	Sigma Point Kalman Filter
TMA	:	Target Motion Analysis
UAV	:	Unmanned Air Vehicle
UKF	:	Unscented Kalman Filter
UPF	:	Unscented Particle Filter
UT	:	Unscented Transform

CHAPTER 1

INTRODUCTION

1.1 Background

In the estimation theory the state space approach is widely used while working with dynamic systems. First, the system model is constructed such that the state includes all relevant information required to describe the real system. Then, the observations that are statistically related to the state are defined by the measurement model. After the real world problems are transferred into the theoretical domain by this modeling, either the state or the model parameters are tried to be estimated. While the system identification techniques focus on characterizing the unknown system [1], state estimation techniques try to gain information about the state.

The estimation theory is used in a wide range of engineering applications. It is also an important topic in control theory and control systems engineering [2]. Followings are the examples of particular application areas [1].

- Tracking
- Computer vision
- Inertial guidance system
- Satellite navigation systems

- Speech recognition
- Economics, in particular macroeconomics, time series, and econometrics
- Chaotic Signals
- Weather Forecasting

In stochastic filtering, the main purpose is to sequentially estimate the state by using the noisy or incomplete measurements. Throughout the years many scientists and engineers worked in this area. A brief history of stochastic filtering is given in Table 1-1 [3].

Table 1-1: A history of stochastic filtering theory [3].

Author(s) (year)	Method	Solution	Comment
Kolmogorov (1941)	Innovations	Exact	Linear, stationary
Wiener (1942)	Spectral factorization	Exact	Linear, stationary, infinite memory
Levinson (1947)	Lattice filter	Approximate	Linear, stationary, finite memory
Bode & Shannon (1950)	Innovations, whitening	Exact	Linear, stationary
Zadeb & Ragazzini (1950)	Innovations, whitening	Exact	Linear, non-stationary
Kalman (1960)	Orthogonal projection	Exact	LQR, non-stationary, discrete
Kalman & Bucy (1961)	Recursive Riccati equation	Exact	LQR, non-stationary, continuous
Stratonovich (1960)	Conditional Markov Process	Exact	Nonlinear, non-stationary
Kushner (1967)	PDE	Exact	Nonlinear, non-stationary
Zakai (1969)	PDE	Exact	Nonlinear, non-stationary
Hadechin & Mayne (1969)	Monte Carlo	Approximate	Nonlinear, non-Gaussian, non-stationary
Bucy & Senne (1971)	Point-mass, Bayes	Approximate	Nonlinear, non-Gaussian, non-stationary
Kaliath (1971)	Innovations	Exact	Linear, non-Gaussian, non-stationary
Benes (1981)	Benes	Exact solution of Zakai eqn.	Nonlinear, finite-dimensional
Daum (1986)	Daum, virtual measurement	Exact solution of FPK eqn.	Nonlinear, finite-dimensional
Gordon, Salmond, & Smith (1993)	Bootstrap, sequential Monte Carlo	Approximate	Nonlinear, non-Gaussian, non-stationary
Julier & Uhimana (1997)	Unscented transformation	Approximate	Nonlinear, (non)-Gaussian, derivative-free

Kalman filtering is one of the milestones in stochastic filtering. It seems fair to say that Kalman filter have dominated the adaptive filter theory for decades [3]. In order to overcome its limitations, numerous filtering methods have been proposed and developed. Incorporating Monte Carlo methods is an example of such efforts. Monte Carlo is a sampling approach to handle complex systems that are analytically intractable [3]. Particularly, when the linearity and Gaussianity assumptions of the Kalman filtering do not hold, Monte Carlo methods become an alternative. Combining this technique with Bayesian inference, on-line estimation can also be done if the necessary computational power is available. These filtering methods are generally called sequential Monte Carlo methods, or particle filters.

As stated previously, one important application area of estimation theory is tracking [4, 5, 6]. For the radar tracking problem, the aim is to track the targets location, speed, and acceleration by using the indirect, inaccurate and uncertain measurements. Note that, there are inherent nonlinearities in the system and measurement models, especially when the maneuvering targets are considered. Besides, noise terms are not always Gaussian due to the sensor systems. Therefore, particle filters seem to be a suitable option to improve the overall performance [4].

1.2 Scope of Thesis

In this study the discrete time formulation of the single target tracking problem is considered. Tracking via radar is investigated in order to compare the filtering methods and evaluate their performances. For such a tracking problem usually standard Kalman filter is not used without any modifications since,

- Motion and/or measurement models are nonlinear,
- There are different motion models for the same target (i.e. different modes),
- Process and measurement noises are not always Gaussian.

In various target tracking related estimation applications, particle filtering algorithms are presented and compared with the classical Kalman filter based methods using different target trajectories. Effects of tracking models and their convenience to the algorithms are also investigated.

CHAPTER 2

RECURSIVE BAYESIAN ESTIMATION

2.1 General Information

Most of the systems in the real world can be studied by using the state-space models. Within the model, the state's evolution in time and statistically related measurements are defined through experience, testing and theoretical analysis. Depending on the problem one may need to estimate the state, the model, or both. In "Estimation" problem considered here, the aim is to obtain statistical distribution (usually it is expressed as an estimate and its uncertainty) for an unknown state using noisy measurements (Figure 2-1). During this process, the inevitable noise on the state itself and measurement is eliminated as much as possible, and the information hidden in the measurement is tried to be extracted.

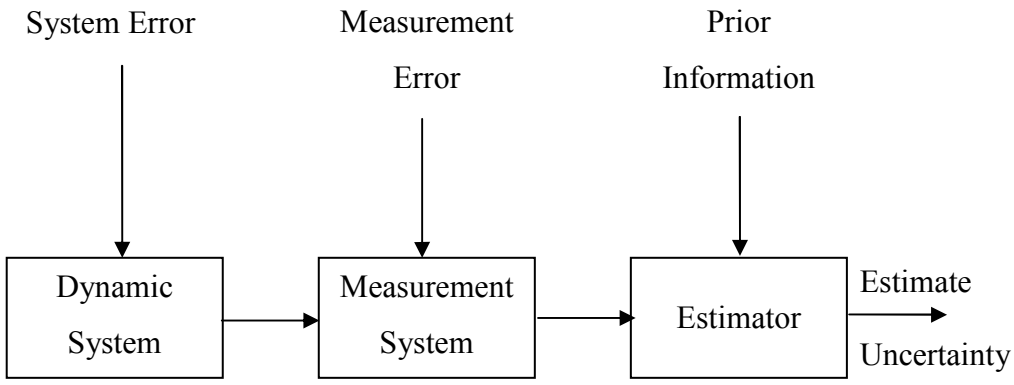

Figure 2-1: General block diagram for estimation [7].

In discrete time, state and measurement sequences are expressed as in (2-1) & (2-2) for a sampling period T.

$$\{x_{1T}, x_{2T}, ..., x_{kT}\} = \{x_1, x_2, ..., x_k\} \stackrel{\Delta}{=} x_{1:k} \tag{2-1}$$

$$\{y_{1T}, y_{2T}, ..., y_{kT}\} = \{y_1, y_2, ..., y_k\} \stackrel{\Delta}{=} y_{1:k} \tag{2-2}$$

Note that, due to the noise in the state evolution and measurement processes or uncertainty as to the exact nature of the process itself, x_k and y_k are generally regarded as random variables [5]. A general state-space model is

$$x_{k+1} = f_k(x_k, u_k, w_k) \tag{2-3}$$

$$y_k = h_k(x_k, v_k) \tag{2-4}$$

where u_k is the known input to the dynamic system, w_k and v_k are the process noise and measurement noise, respectively. On the other hand, since x_k and y_k are

treated as random variables, it is also possible to represent the same system by the following conditional probability density functions.

$$p_{x_{k+1}|x_k,u_k,w_k}\left(x_{k+1}|x_k,u_k,w_k\right) \overset{\Delta}{=} p\left(x_{k+1}|x_k,u_k,w_k\right) \tag{2-5}$$

$$p_{y_k|x_k,v_k}\left(y_k|x_k,v_k\right) \overset{\Delta}{=} p\left(y_k|x_k,v_k\right) \tag{2-6}$$

From that point on u_k in equation (2-5) is omitted during the derivations since it represents the known input.

In estimation, the main purpose is to estimate the evolving state x_{k+1} by using all the measurements collected up to step k ($y_{1:k+1}$). Having an estimate is usually not enough since one should also need to know the quality of it (i.e. a measure of uncertainty). Hence, the posterior density $p\left(x_{k+1}|y_{1:k+1}\right)$ is tried to be reached instead of point estimates. *Bayesian* filtering tries to obtain that posterior density as a function [5]. Note that, if the *posterior* probability density function (pdf) is available, a single representative (i.e. estimate), such as mean or median, can be obtained. On the contrary, there is no *prior* pdf associated with the state in non-Bayesian approaches. Hence there is no posterior distribution either. In that case, the estimation is performed using the *likelihood function* $p\left(y_{1:k+1}|x_{k+1}\right)$ which serves as a measure of *evidence from data* [1].

In Bayesian filtering, the posterior pdf can be constructed by the following two steps, namely prediction and update (Figure 2-2). "The *time update* projects the current state estimate ahead in time. The *measurement update* adjusts the projected estimate by an actual measurement at that time." [8]

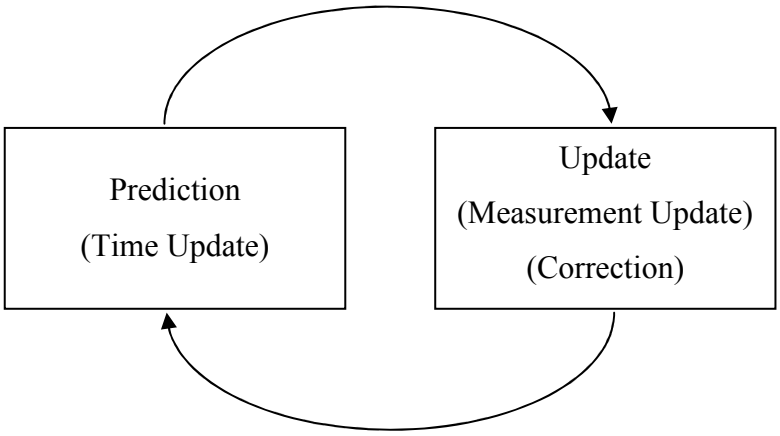

Figure 2-2: Recursive Bayesian filtering cycle.

Prediction: $$p(x_{k+1}|y_{1:k}) = \int p(x_{k+1}|x_k, y_{1:k}).p(x_k|y_{1:k}).dx_k \quad (2\text{-}7)$$

Update: $$p(x_{k+1}|y_{1:k+1}) = \frac{p(y_{k+1}|x_{k+1}, y_{1:k}).p(x_{k+1}|y_{1:k})}{p(y_{k+1}|y_{1:k})} \quad (2\text{-}8)$$

where the numerator in the update step (2-8) can be written as

$$p(y_{k+1}|y_{1:k}) = \int p(y_{k+1}|x_{k+1}, y_{1:k}).p(x_{k+1}|y_{1:k}).dx_{k+1} \quad (2\text{-}9)$$

Throughout the prediction and update steps, the new estimate ($p(x_{k+1}|y_{1:k+1})$) is obtained by using the old one ($p(x_k|y_{1:k})$) and the last measurement (y_{k+1}). However, in order to perform the algorithm recursively, one must store and use all the previous measurements ($y_{1:k}$). After the following assumptions, the previous state estimate will contain all the information about the system up to that point, and therefore there will be no need to store previous measurements.

Assumption 1: State evolution is a Markov Process.

$$p(x_{k+1}|x_{0:k}) = p(x_{k+1}|x_k) \tag{2-10}$$

In other words, the past state trajectory does not contain any valuable information as long as the present state is known.

Assumption 2: "Measurements are taken from a channel without memory. Conditioned on x_k, y_k is assumed to be independent of the rest of the state sequence and all other measurements." [5]

$$p(y_{1:k}|x_{1:k}) = \prod_{i=1}^{k} p(y_i|x_i) \tag{2-11}$$

Considering the state-space models in equations (2-3) & (2-4) or (2-5) & (2-6), these assumptions can be restated in terms of the initial state and noise distributions [1, 5].

Assumption A: Process noise $\{w_k\}$ is an independent sequence (i.e. white).

Assumption B: Measurement noise $\{v_k\}$ is an independent sequence (i.e. white).

Assumption C: Process noise $\{w_k\}$, measurement noise $\{v_k\}$ and the initial state x_0 are mutually independent.

Consequently, the prediction and update steps are shown below for recursive Bayesian filtering, which is estimating the posterior density $p(x_{k+1}|y_{k+1})$ when the previous estimate $p(x_k|y_k)$ and the last measurement y_{k+1} are given. Derivations of (2-12) and (2-13) can be found in [5]

Prediction: $$p(x_{k+1}|y_{1:k}) = \int p(x_{k+1}|x_k).p(x_k|y_{1:k}).dx_k \tag{2-12}$$

Update: $$p(x_{k+1}|y_{1:k+1}) = \frac{p(y_{k+1}|x_{k+1}).p(x_{k+1}|y_{1:k})}{p(y_{k+1}|y_{1:k})} \tag{2-13}$$

In summary, **"**Recursive Bayesian estimation is a general probabilistic approach for estimating an unknown probability density function recursively over time using incoming measurements and a mathematical process and measurement model" [9]. The above derivations are done for *filtering*, which aims to estimate the current state given the measurement. If a state that at some time beyond the data interval is estimated, then it is called *prediction*. Similarly, estimating a state at time k within the data interval is called *smoothing*.

2.2 Optimal Algorithms

Under some assumptions, optimal solutions for recursive Bayesian estimation can be obtained by finite-dimensional algorithms [4]. Namely,

- Kalman Filter: For Linear-Gaussian systems
- Grid Based Methods: For systems having discrete-valued and finite number of states
- Benes and Daum Filters: For certain subclasses of nonlinear systems (not presented here, the detailed information is given in [10, 11, 4])

2.2.1 Grid Based Method

For some particular applications like speech processing [12, 5], the state space is discrete and consists of a finite number of states. In that case, the *a priori* distribution can be expressed in terms of conditional probabilities (2-14).

$$p\left(x_k \middle| y_{1:k}\right) = \sum_{i}^{N_s} P\left\{x_k = x_k^i \middle| y_{1:k}\right\} \delta\left(x_k - x_k^i\right) = \sum_{i}^{N_s} w_{k|k}^i . \delta\left(x_k - x_k^i\right) \qquad (2\text{-}14)$$

Here N_s is the total number of possible states, $w_{k|k}^i$ is the weight, and δ is the Dirac-delta function. Then the filtering equations (2-12) and (2-13) become [4, 13]

Prediction: $$p\left(x_{k+1}\middle|y_{1:k}\right)=\sum_{i=1}^{N_s} w_{k+1|k}^{i}.\delta\left(x_{k+1}-x_{k+1}^{i}\right) \tag{2-15}$$

Update: $$p\left(x_{k+1}\middle|y_{1:k+1}\right)=\sum_{i=1}^{N_s} w_{k+1|k+1}^{i}.\delta\left(x_{k+1}-x_{k+1}^{i}\right) \tag{2-16}$$

where the weights are defined as

$$w_{k+1|k}^{i}\overset{\Delta}{=}\sum_{j=1}^{N_s} w_{k|k}^{j}.p\left(x_{k+1}^{i}\middle|x_{k}^{j}\right) \tag{2-17}$$

$$w_{k+1|k+1}^{i}\overset{\Delta}{=}\frac{w_{k+1|k}^{i}.p\left(y_{k+1}\middle|x_{k+1}^{i}\right)}{\sum_{j=1}^{N_s} w_{k+1|k}^{j}.p\left(y_{k+1}\middle|x_{k+1}^{j}\right)} \tag{2-18}$$

While the grid based method (i.e. the HMM filter) provide optimal solutions, usually it is not used in airborne target tracking since the assumptions does not hold [5].

2.2.2 Kalman Filter

In this study the discrete-time Kalman filter, where the measurements occur and the state is estimated at discrete points in time, is used. The well known Kalman filter (KF) is optimal (in the sense that it minimizes the estimated error covariance) if the assumptions below hold in addition to the ones given in Section 2.1 [14, 8, 2, 15, 4, 16, 17].

- System (i.e. equations (2-3) & (2-4)) is linear.

$$x_{k+1}=F_k.x_k+G_k.u_k+w_k \tag{2-19}$$

$$y_k=H_k.x_k+v_k \tag{2-20}$$

- Process noise $\{w_k\}$, measurement noise $\{v_k\}$ and the initial state x_0 have Gaussian distributions ((2-21), (2-22) & (2-23)) [8, 1]. If the noise sequence has non-zero mean, its mean can be incorporated in the equation as a known constant term.

$$p(w_k) \sim N(0, Q_k) \tag{2-21}$$

$$p(v_k) \sim N(0, R_k) \tag{2-22}$$

$$p(x_{0|0}) \stackrel{\Delta}{=} p(x_0) \sim N(\hat{x}_0, P_0) \tag{2-23}$$

Model underlying the Kalman Filter is shown in Figure 2-3 where the circles are vectors, squares are matrices, and stars represent Gaussian noise with the associated covariance matrix at the lower right [2].

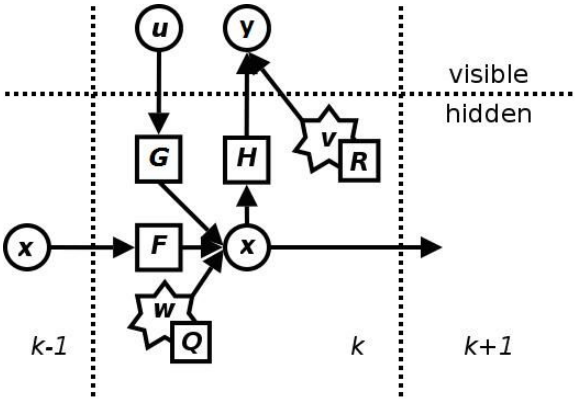

Figure 2-3: Model underlying the Kalman Filter [2].

The Bayesian filtering concepts described in section 2.1 forms the basis of the Kalman filter. Due to the assumptions above, *prior* and *posterior* state distributions are Gaussian. Therefore, in Kalman filter algorithm, the mean (i.e. the estimate) and

covariance (i.e. the error covariance) of the *prior* state distribution is processed rather than the whole pdf, in order to obtain the same parameters of the *posterior* distribution. In other words, the analytical solution of the Bayesian filtering problem is derived and from that point on algorithm runs recursively based on the defining parameters (mean and covariance) of the pdf's.

The prediction (i.e. time update) and update (i.e. measurement update) steps in Figure 2-2 and Equations (2-12) & (2-13) are the two main steps of Kalman filter too. "The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the *a priori* estimates for the next time step. "The measurement update equations are responsible for the feedback—i.e. for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate." [8]. As the common notation: $p(x_k|y_k) \sim N(\hat{x}_{k|k}, P_{k|k})$ and $p(x_{k+1}|y_k) \sim N(\hat{x}_{k+1|k}, P_{k+1|k})$.

Time Update Step:

1. Project the state ahead

$$\hat{x}_{k+1|k} = F_k . \hat{x}_{k|k} + B_k . u_k \tag{2-24}$$

2. Project the error covariance (i.e. state prediction covariance) ahead

$$P_{k+1|k} = F_k . P_{k|k} . F_k^{\ T} + Q_k \tag{2-25}$$

Measurement Update Step:

1. Compute the innovation covariance

$$S_{k+1} = H_{k+1} . P_{k+1|k} . H_{k+1}^{\ T} + R_{k+1} \tag{2-26}$$

2. Compute the Kalman gain

$$K_{k+1} = P_{k+1|k}.H_{k+1}{}^{T} S_{k+1}{}^{-1} \tag{2-27}$$

3. Update the state estimate with the last measurement y_k

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}.\left(y_{k+1} - H_{k+1}.\hat{x}_{k+1|k}\right) \tag{2-28}$$

4. Update the error covariance

$$P_{k+1|k+1} = \left(I - K_{k+1}.H_{k+1}\right).P_{k+1|k} \tag{2-29}$$

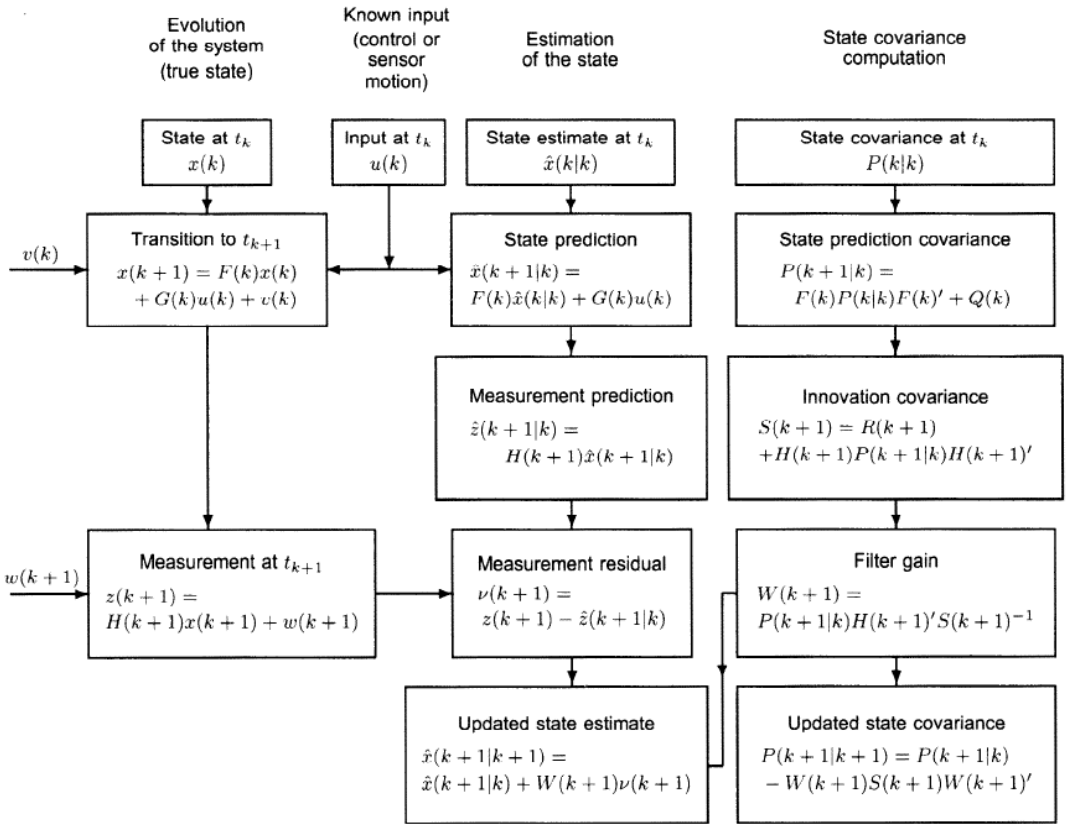

Figure 2-4: One cycle in the KF [1]. Note that the equation for the updated state covariance is equivalent to Equation (2-29).

If the first two moments are available but the noises are not Gaussian, then the Kalman filter is the best linear minimum mean square error (MMSE) estimator [15, 16, 17].

In the steady state of the above time-varying KF, the Kalman gain (K) and the covariance can be assumed to converge to some constant values. In tracking problem, result corresponds to the particular case called α-β filter for constant-velocity model or α-β-γ filter for constant-acceleration model. The symbols α, β and γ refer to the gain coefficients of the position, velocity and acceleration components respectively. These gains in the measurement update step are constant but optimized according to a design.

2.2.3 Information Filter

Information filter (of inverse covariance filter) is a variant of Kalman filter where the inverses of the covariance matrices ($P_{k+1|k}^{-1}$, $P_{k+1|k+1}^{-1}$) are recursively calculated throughout the prediction and update [1]. For the same system model in Equations (2-3) & (2-4), following relations are used instead of corresponding equations in the Kalman Filter [1]:

Noiseless state prediction information matrix:

$$A_k = \left(F_k^{-1}\right)^T . P_{k|k}^{-1} . F_k^{-1} \tag{2-30}$$

State prediction information matrix:

$$P_{k+1|k}^{-1} = A_k - A_k . \left(A_k + Q_k^{-1}\right)^{-1} . A_k \tag{2-31}$$

Kalman Gain:

$$K_{k+1} = P_{k|k} . H_{k+1}{}^T . R_{k+1}^{-1} = \left(P_{k+1|k}^{-1} + H_{k+1}{}^T . R_{k+1}^{-1} . H_{k+1}\right)^{-1} . H_{k+1}{}^T . R_{k+1}^{-1} \tag{2-32}$$

Updated state information matrix:

$$P_{k+1|k+1}^{-1} = P_{k+1|k}^{-1} + H_{k+1}{}^{T}.R_{k+1}^{-1}.H_{k+1} \tag{2-33}$$

The equations for state estimate $\hat{x}$ can remain the same. However, as a different formulation the estimated state may be replaced by $P_{k|k}^{-1}.\hat{x}_{k|k}$. The new state, which is called the information filter state, and corresponding relations are as follows,

$$\hat{s}_{k|k} = P_{k|k}^{-1}.\hat{x}_{k|k} \tag{2-34}$$

$$\hat{s}_{k+1|k} = P_{k+1|k}^{-1}.\hat{x}_{k+1|k} \tag{2-35}$$

$$\hat{s}_{k+1|k+1} = \hat{s}_{k+1|k} + H_{k+1}{}^{T}.R_{k+1}^{-1}.y_k \tag{2-36}$$

Therefore, if there is no initial estimate, the filter can be initialized with a zero information matrix ($P_{0|0}^{-1} = 0$). In that case no initial estimate of the system state is needed [1]. Later on, the actual state estimate can be recovered by simply multiplying by (2-37).

$$\hat{x}_{k+1|k+1} = P_{k+1|k+1}.\hat{s}_{k+1|k+1} \tag{2-37}$$

One other advantage of this variant of the Kalman filter is the ability to use multiple measurements at the same time step [2].

2.3 Sub-optimal Algorithms Based on Single Model

2.3.1 Approximate Grid Based Methods

Although the grid based methods (i.e. numerical methods) are not used in this study, they are presented here due to the similarity of ideas behind the particle filters and them. As stated in section 2.2.1 if the state-space is discrete and consists of a finite

number of states, optimum result is possible. However, same kind of approach can be used for non-linear and non-Gaussian cases in order to obtain approximate solutions. The idea behind is to divide the state-space into cells (Figure 2-5) and represent the prior and posterior pdf's by impulses and their weights [4, 13, 18].

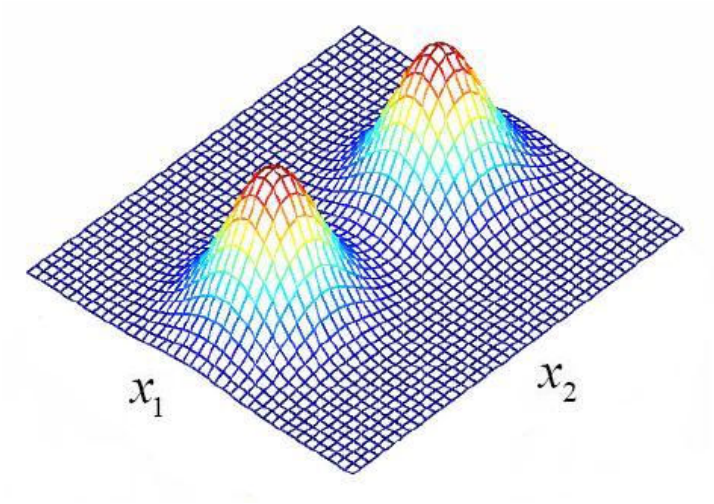

Figure 2-5: Representing the pdf with the points on the grid and weights considering a two-dimensional (x_1 & x_2) state-space.

In fact, these methods are some numerical methods where the recursive Bayesian estimation' i.e. Equations (2-12) and (2-13), are calculated by approximating the integrals in these formulas. They can be called direct numerical integration methods. Equations are the same as ones in Section 2.2.1 [13], but this time

- x_k^i 's are the predetermined points on the uniform deterministic grid,
- $p\left(x_{k+1}^i \middle| x_k^j\right)$ and $p\left(y_{k+1} \middle| x_{k+1}^i\right)$ are the evaluated (or approximated) values assuming that both distributions (i.e. process and measurement noise pdf's) are known. There is no need for them to be Gaussian (Figure 2-5).

Obviously, the grid must be sufficiently dense and large to obtain a good estimate everywhere in the state-space. A dense and large grid for a high dimensional state-space would result in a serious computational cost. Although there are some applications in speech processing [4, 12], it is not practical for most real world applications like tracking [18].

2.3.2 Extended Kalman Filter

Kalman filter is the optimal solution under the assumption that the underlying dynamics are linear. Unfortunately, the real life systems are usually, by nature, nonlinear. An idea is, first to approximate the system with a set of equations such that the assumptions hold, then run the standard KF algorithm, and it is called the Extended Kalman Filter (EKF). The key point here is that the approximation is done recursively based on the current state estimate.

EKF has its own variants according to different approaches used while approximating the real system. Depending on the order of the Taylor series expansion, the filter is called either first-order or second-order EKF. The choice of the reference point at which the linearization is done also yields to different algorithms like iterated EKF [19, 1, 4].

Derivation of the first-order EKF for the nonlinear system defined by (2-38) and (2-39) is given below. Note that, for simplicity, the noises are assumed to be additive even if the system is nonlinear, and the known input is omitted. So system equations are as follows.

$$x_{k+1} = f_k(x_k) + w_k \tag{2-38}$$

$$y_k = h_k(x_k) + v_k \tag{2-39}$$

The Taylor series expansion around the current estimate $\hat{x}_{k|k}$ is

$$\begin{aligned} x_{k+1} &= f_k\left(\hat{x}_{k|k}\right) + F_k.\left(x_k - \hat{x}_{k|k}\right) + w_k \\ &= F_k.x_k + f_k\left(\hat{x}_{k|k}\right) - F_k.\hat{x}_{k|k} + w_k \\ &= F_k.x_k + \overline{u}_k + w_k \end{aligned} \tag{2-40}$$

where the known terms are denoted as $\overline{u}_k$ since they are constant (Section 4.3.1 [20]), and F_k is the Jacobian:

$$F_k = \left(\nabla_{x_k} f_k^T\left(x_k\right)\right)^T \Big|_{x_k = \hat{x}_{k|k}} \tag{2-41}$$

Similarly, the measurement relation is linearized around the predicted state $\hat{x}_{k+1|k}$:

$$y_{k+1} = h_{k+1}\left(\hat{x}_{k+1|k}\right) + H_{k+1}.\left(x_{k+1} - \hat{x}_{k+1|k}\right) + v_{k+1} \tag{2-42}$$

$$H_{k+1} = \left(\nabla_{x_{k+1}} h_{k+1}^T\left(x_{k+1}\right)\right)^T \Big|_{x_{k+1} = \hat{x}_{k+1|k}} \tag{2-43}$$

Then, the filter equations are

Time Update Step:

$$\hat{x}_{k+1|k} = f_k\left(\hat{x}_{k|k}\right) \tag{2-44}$$

$$P_{k+1|k} = F_k.P_{k|k}.F_k^{\ T} + Q_k \tag{2-45}$$

Measurement Update Step:

$$S_{k+1} = H_{k+1}.P_{k+1|k}.H_{k+1}^{\ T} + R_{k+1} \tag{2-46}$$

$$K_{k+1} = P_{k+1|k}.H_{k+1}^{\ T} S_{k+1}^{\ -1} \tag{2-47}$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}.\left(y_{k+1} - h_{k+1}\left(\hat{x}_{k+1|k}\right)\right) \tag{2-48}$$

$$P_{k+1|k+1} = \left(I - K_{k+1}.H_{k+1}\right).P_{k+1|k} \tag{2-49}$$

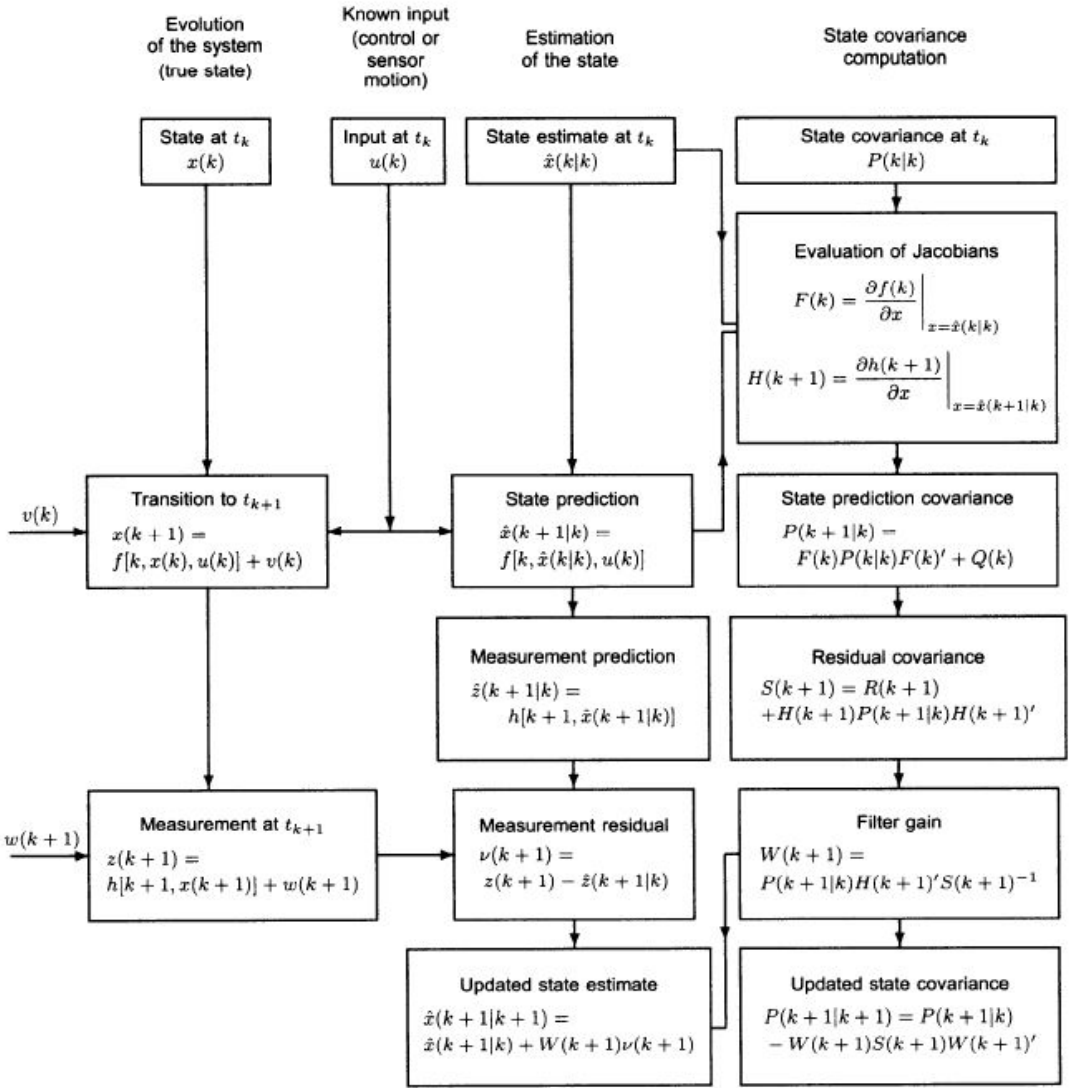

Figure 2-6: One cycle of the EKF [1]. Note that the equation for the updated state covariance is equivalent to Equation (2-49).

Although the inherent approximations may lead to divergence of the filter; in practice, it performs well if the nonlinearities are mild, initial error and noises are not too large [1, 4].

A filter is consistent provided that the estimation error has zero mean and its covariance matches with the filter calculated covariance. According to [Ref 9] for the bearings-only tracking application first-order and second-order EKF's gives inconsistent results. The reason is claimed to be the linearization that does not work well with large measurement noise.

2.3.3 Iterated Extended Kalman Filter

The EKF's performance is degraded by the linearization errors. Like the second-order EKF, the iterated EKF (IEKF) is one of the methods proposed to reduce the effect of these errors. The key idea is linearizing the measurement model around the updated state $\hat{x}_{k+1|k+1}$ rather than the predicted state $\hat{x}_{k|k}$ [1, 4, 21]. Hence the prediction step is the same as EKF. Filter equations are the followings:

Relinearized measurement equation:

$$H_{k+1}^{i} = \left(\nabla_{x_{k+1}} h_{k+1}^{T}\left(x_{k+1}\right)\right)^{T}\Big|_{x_{k+1}=\hat{x}_{k+1|k+1}^{i}} \tag{2-50}$$

Note that for the first iteration ($i = 0$)

$$\hat{x}_{k+1|k+1}^{0} \overset{\Delta}{=} \hat{x}_{k+1|k}^{0} \tag{2-51}$$

State estimate and state covariance update equations:

$$\begin{aligned}\hat{x}_{k+1|k+1}^{i+1} = \hat{x}_{k+1|k+1}^{i} + P_{k+1|k+1}^{i}.\left(H_{k+1}^{i}\right)^{T}.R_{k+1}^{-1}.\left(y_{k+1} - h_{k+1}\left(\hat{x}_{k+1|k+1}^{i}\right)\right) \\ - P_{k+1|k+1}^{i}.P_{k+1|k}^{-1}.\left(\hat{x}_{k+1|k+1}^{i} - \hat{x}_{k+1|k}\right)\end{aligned} \tag{2-52}$$

$$P_{k+1|k+1}^{i+1} = P_{k+1|k} - P_{k+1|k}.\left(H_{k+1}^{i}\right)^{T}.\left(H_{k+1}^{i}.P_{k+1|k}.\left(H_{k+1}^{i}\right)^{T} + R_{k+1}\right)^{-1}.H_{k+1}^{i}.P_{k+1|k} \tag{2-53}$$

According to [21, 4], the IEKF gives accurate results only when the measurement model fully observes the state.

2.3.4 Unscented Kalman Filter

The unscented Kalman filter (UKF), also known as sigma point Kalman filter (SPKF), is another sub-optimal algorithm for nonlinear estimation [4, 2, 18, 21, 22, 23]. In EKF the nonlinear functions are approximated (linearized) locally at each step. However, in UKF there is no such analytical approximation. Instead, the state distribution is assumed to be Gaussian and represented by a small set of support points known as sigma points (Figure 2-7) [22]. These deterministically chosen points are then propagated through the nonlinear state & measurement equations for filtering. This class of nonlinear filters is referred as linear regression Kalman filters (LRKF) in [21] since they are all based on statistical linearization rather that analytical linearization in the EKF.

The sigma points are chosen in accordance with the unscented transform (UT), which is a method for calculating the statistics of a random variable that undergoes a nonlinear transform. These points completely capture the mean and covariance of a Gaussian random variable [4]. If they are propagated through a nonlinear system, then the resultant points capture first two moments of the posterior distribution accurately to the second-order (third-order if the input is Gaussian) Taylor series expansion [22, 23].

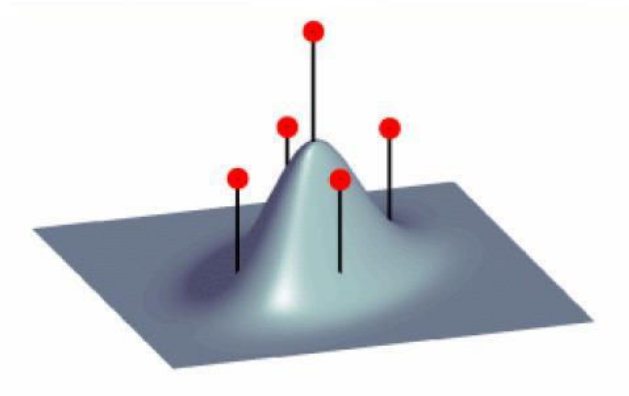

Figure 2-7: A representative picture of the sigma points for a two-dimensional distribution [23].

In order to tune the UKF according to a particular nonlinear system, some scaling parameters are defined in the UT. These parameters control the spread and relative weighting of the sigma points. Resultant modified algorithm is called the *scaled unscented transform* [22]. Considering the general nonlinear system defined by Equations (2-3) & (2-4), derivation of the UKF using that scaled transform is shown below [24].

1. Augmenting the previous estimate of the state with the process noise (w_k) and measurement noise (v_k):

$$\hat{x}^a_{k|k} \overset{\Delta}{=} \begin{bmatrix} \hat{x}^T_{k|k} & w^T_k & v^T_k \end{bmatrix}^T \tag{2-54}$$

$$P^a_{k|k} \overset{\Delta}{=} \begin{bmatrix} P_{k|k} & 0 & 0 \\ 0 & Q_k & 0 \\ 0 & 0 & R_k \end{bmatrix} \tag{2-55}$$

2. Let L be the number of the augmented state $\hat{x}^a_{k|k}$. Calculating the sigma points (columns of $X^a_{k|k}$):

$$\begin{aligned} X^a_{k|k} &\overset{\Delta}{=} \left[\hat{x}^a_{k|k} \quad \hat{x}^a_{k|k}.Ones(L)+\gamma.\sqrt{P^a_{k|k}} \quad \hat{x}^a_{k|k}.Ones(L)-\gamma.\sqrt{P^a_{k|k}}\right] \\ &\overset{\Delta}{=} \left[\left(X^x_{k|k}\right)^T \quad \left(X^w_{k|k}\right)^T \quad \left(X^v_{k|k}\right)^T\right] \end{aligned} \tag{2-56}$$

such that $Ones(L)$ is the L-dimensional row vector full of ones, and

$$\gamma = \sqrt{L+\lambda} \tag{2-57}$$

$$\lambda = \alpha^2.(L+\kappa)-L \tag{2-58}$$

3. Time Update

   a. Propagating the relevant sigma points through the state equation:

$$X^x_{k+1|k} = f_k\left(X^x_{k|k}, u_k, X^w_{k|k}\right) \tag{2-59}$$

   b. Obtaining the predicted state and its covariance:

$$\hat{x}_{k+1|k} = \sum_{i=0}^{2L} w_i^m.\left(X^x_{k+1|k}\right)_i \tag{2-60}$$

$$P_{k+1|k} = \sum_{i=0}^{2L} w_i^c.\left(\left(X^x_{k+1|k}\right)_i - \hat{x}_{k+1|k}\right).\left(\left(X^x_{k+1|k}\right)_i - \hat{x}_{k+1|k}\right)^T \tag{2-61}$$

   where $(\ldots)_i$ represents the i[th] column of the matrix (i.e. the i[th] sigma point) and w_i^m (mean), w_i^c (covariance) are the scalar weights such that

$$w_0^m = \frac{\lambda}{L+\lambda} \tag{2-62}$$

$$w_0^c = \frac{\lambda}{L+\lambda} + \left(1-\alpha^2+\beta\right) \tag{2-63}$$

$$w_i^m = w_i^c = \frac{1}{2.(L+\lambda)} \qquad for\ i = 1,...,2L \tag{2-64}$$

4. Measurement Update

   a. Propagating the relevant sigma points through the measurement equation:

$$Y_{k+1|k} = h_{k+1}\left(X^x_{k+1|k}, X^v_{k+1|k}\right) \tag{2-65}$$

   b. Obtaining the predicted measurement, innovation covariance and cross covariance:

$$\hat{y}_{k+1|k} = \sum_{i=0}^{2L} w_i^m .\left(Y_{k+1|k}\right)_i \tag{2-66}$$

$$P_{\hat{y}_{k+1}\hat{y}_{k+1}} = \sum_{i=0}^{2L} w_i^c .\left(\left(Y_{k+1|k}\right)_i - \hat{y}_{k+1|k}\right).\left(\left(Y_{k+1|k}\right)_i - \hat{y}_{k+1|k}\right)^T \tag{2-67}$$

$$P_{\hat{x}_{k+1}\hat{y}_{k+1}} = \sum_{i=0}^{2L} w_i^c .\left(\left(X^x_{k+1|k}\right)_i - \hat{x}_{k+1|k}\right).\left(\left(Y_{k+1|k}\right)_i - \hat{y}_{k+1|k}\right)^T \tag{2-68}$$

   c. Finding the Kalman gain, updated state estimate and covariance

$$K_{k+1} = P_{\hat{x}_{k+1}\hat{y}_{k+1}} .P^{-1}_{\hat{y}_{k+1}\hat{y}_{k+1}} \tag{2-69}$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}.\left(y_{k+1} - \hat{y}_{k+1|k}\right) \tag{2-70}$$

$$P_{k+1|k+1} = P_{k+1|k} - K_{k+1}.P_{\hat{y}_{k+1}\hat{y}_{k+1}}.K^T_{k+1} \tag{2-71}$$

Note that $2L+1$ symmetric (with respect to the mean) sigma points are chosen as convention [22, 23, 24]. The scalar parameters α, β and κ are for adjusting the filter. α controls how much the sigma points spread around the mean, and typical range is $10^{-3} < \alpha \leq 1$. β adjusts the relative weighting of the zeroth sigma point

(the one that corresponds to the mean itself), and for Gaussian *prior*s it is set equal to 2. Lastly, κ is usually chosen as 0 [22, 23, 24].

Similar to EKF, the predicted state and its covariance can be obtained separately from the final estimate. This provides the opportunity to use only the prediction or the update steps according to the considered nonlinear system. One such marginalization can be found in [22], in which usual KF steps are used for the time update because the state relation is linear in bearing only tracking. Further simplifications are possible if the noises are additive and Gaussian (like in (2-38) & (2-39)) [4]. In that case, there is no need for the augmentation.

It is stated in [25] that "the UKF consistently outperforms the EKF in terms of prediction and estimation error, at an equal computational complexity for general state-space problems". Besides, since no explicit calculation of the Jacobian's is necessary, the discontinuity in nonlinear relations can also be tolerated [4]. Nevertheless as the *posterior* become non-Gaussian, performance of the UKF can degrade because the sigma points cannot fully characterize the posterior [4, 22]. The detailed information about the cousins of UKF can be found in [21].

2.4 Suboptimal Algorithms Based on Multiple Models

The above mentioned filtering techniques inherently assume that the state-space model is known for each time step. However it may not be always true in real world. The state and measurement relations can be time varying and partially unknown. This introduces an additional uncertainty to the problem other than the usual process and measurement noises [1]. Consequently, these issues have to be considered while constructing the estimator. A natural idea is to estimate the state by an adaptive algorithm that takes the model induced uncertainties into account.

The variation of the model with time can be caused by the unknown inputs. In this approach, the input is usually viewed as a parameter to be estimated along with the

state. Either in the case of unknown input and system model, the interacting multiple model (IMM) filters are widely used [1, 26].

The model can also change depending on parameters other than the input. One such case occurs when approximating a nonlinear model by a set of linear ones. This time, model changes with respect to the region where the linearization takes place [27].

For the radar tracking problem, both approaches are used extensively throughout the years. That is because, the state relations are usually nonlinear, and the model changes due to the maneuvers of the target.

2.4.1 Interacting Multiple Model Filter

In the interacting multiple model (IMM) approach a finite number of model matched filters are used. At each step the previous estimates of the individual filters are combined according to the mixing probabilities [1, 15]. IMM is widely used in target tracking applications, since the target maneuverability changes the dynamic model dramatically. The usual way is to define different dynamic models for different maneuvers and run them in IMM framework.

The mixing of the models is characterized by the Markov transition probabilities. These probability values are assigned according to the properties of the dynamic system (for instance the maneuverability of the target). The IMM filter can be expressed by the following four fundamental steps [15, 1] (Figure 2-8):

- **Interaction/Mixing:** In order to initialize each filter, the mode-conditioned state estimates and covariance are combined by using the mixing probabilities.

- **Mode-matched filtering:** The mode-conditioned state estimates and covariance are obtained by running the filter bank. Corresponding likelihood functions are also calculated for the next step.

- **Mode probability update:** Mixing and updated mode probabilities are calculated with respect to the likelihoods.

- **Overall state estimate and covariance:** The mode-conditioned state estimates and covariance are combined to obtain a joint estimate and covariance for output.

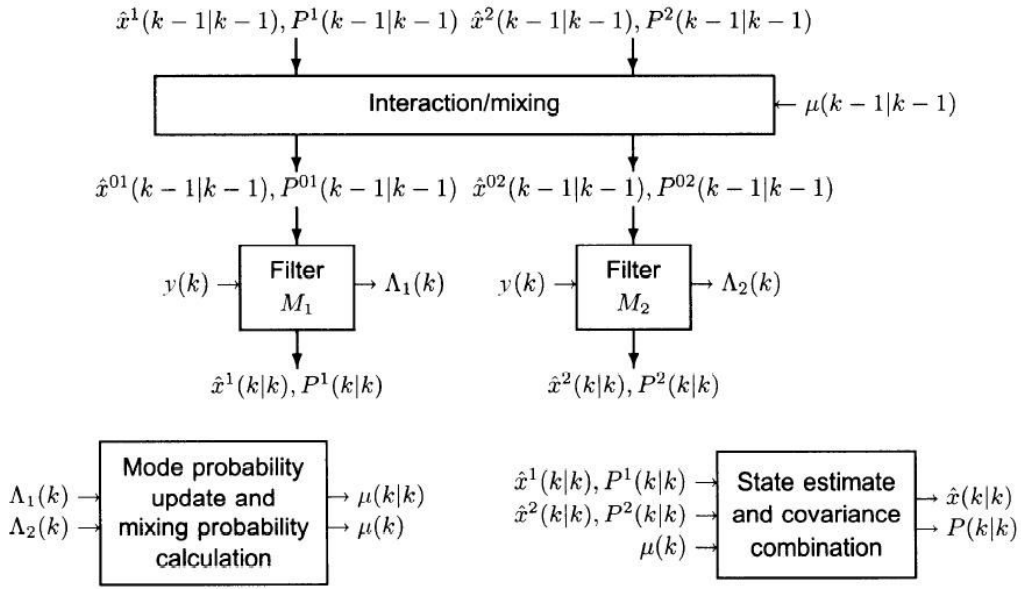

Figure 2-8: One cycle of the IMM Estimator [1].

From the tracking point of view, applications with the IMM filter are very common [1, 15, 5, 26, 28, 29, 30]. Note that IMM is a general filtering approach. Not only Kalman filter but other filtering techniques are also applicable to the framework. For instance, in [31] the UKF is utilized and in [32, 33, 34] the particle filters are used.

2.4.2 Range Parameterized Extended Kalman Filter

The range parameterized extended Kalman filter (RPEKF) is an example of using multiple models to handle the nonlinearity. It is specific to the radar target tracking problem unlike the other general algorithms mentioned in CHAPTER 2. In RPEKF approach, there is a bank of independent EKFs, each tuned to a certain range [27, 35, 36]. At each step filters are weighted for their consistency with the measurement. Consequently, after a number of steps some filters are removed due to their low likelihoods. Eventually the correct filter (i.e. the model) dominates and the overall performance is improved without much computational complexity. Implementation details of RPEKF for Cartesian and polar coordinates can be found in [27, 35, 36].

CHAPTER 3

SEQUENTIAL MONTE CARLO METHODS

3.1 General Information

Monte Carlo methods employ statistical sampling and estimation techniques to solve mathematical problems [3]. It is a powerful tool for handling numerical integration problems, one of which occurs during recursive Bayesian estimation (Equation (2-12) & (2-9)). Kalman filter based methods essentially solve these integrals analytically, and do the calculations only by using the defining parameters (mean & covariance) of the distributions. However the linearity and Gaussianity assumptions do not always hold, resulting in analytically intractable systems. As mentioned in Section 2.3, there are various alternatives to overcome this difficulty. If the problem includes severe nonlinearities or the noises are non-Gaussian, then the distributions cannot be expressed by their mean and covariance. In that case a straightforward idea is to use actual representative data points chosen from the state-space. In particular, the approximate gird based methods (Section 2.3.1) and the UKF (Section 2.3.4) rely on deterministic sampling of prior and posterior distributions. A similar idea is to utilize statistical sampling techniques to represent these distributions. It is achieved by combining the Monte Carlo sampling methods with Bayesian inference [3]. The resultant approach is called sequential Monte Carlo (SMC) method or *particle filter* (PF).

Throughout the development of the approach, particle filter has been called different names such as bootstrap filter, condensation algorithm, interacting particle approximation, Monte Carlo filter, sequential imputation, survival of the fittest, and likelihood weighting algorithm [4, 13, 3].

The derivation of the particle filter is done in the following sections. First Monte Carlo sampling methods are described. Then the idea is applied to recursive Bayesian estimation to obtain the particle filter algorithm. Some different versions of the algorithm are presented after explaining some critical points.

3.2 Monte Carlo Sampling

The main purpose here is to approximate the expectation below

$$E_p\{f(x)\} = \int f(x).p(x).dx \tag{3-1}$$

For most of the practical problems it may not be possible to find an analytical solution to this problem. A close value can be obtained by approximating the pdf $p(x)$ of the random variable $\tilde{x}$ by its samples (i.e. particles) as given in Equation (3-2)

$$p(x) \approx \sum_{i=1}^{N} w^i .\delta(x - x^i) \tag{3-2}$$

In this formulation N is the total number of particles, w^i is the weight of the ith particle, x^i is the location of the particle, and δ is the Dirac function. As a result, the approximate value of the integral becomes

$$\hat{f} = \sum_{i=1}^{N} w^i .f(x^i) \tag{3-3}$$

So, there are two different issues in the computation of the required expected value. First one is to draw samples from a probability distribution $p(x)$, and second one is

to approximate the integral [3]. First issue can be addressed by following techniques.

3.2.1 Importance Sampling

For some particular cases it is possible to take samples directly from the distribution in hand. Gaussian and uniform densities are examples of such distributions. Besides, the inversion sampling, if applicable, can also give the opportunity to create exact samples [50]. However, sampling directly from a known density may be difficult if it does not fit into standard forms. In handling such cases, importance sampling is a powerful tool where a secondary distribution called *importance density* or *proposal distribution*, denoted by $q(x)$, is introduced. First samples are chosen from this secondary distribution, and then they are evaluated, i.e. weighted, according to the convenience to the true density.

The expectation in Equation (3-1) can be rewritten as follows.

$$E_p\{f(x)\} = \int f(x).p(x).dx = \int f(x).\frac{p(x)}{q(x)}.q(x).dx \tag{3-4}$$

Then N independent samples are taken from the importance density $q(x)$, and it is approximated as

$$q(x) \approx \sum_{i=1}^{N} \delta(x - x^i) \tag{3-5}$$

Then the integral can be approximated by

$$\hat{f} = \sum_{i=1}^{N} \tilde{w}(x^i).f(x^i) \tag{3-6}$$

where w^i is the normalized version of $\tilde{w}^i$ defined below.

$$\tilde{w}\left(x^i\right) \stackrel{\Delta}{=} \tilde{w}^i = \frac{p\left(x^i\right)}{q\left(x^i\right)} \tag{3-7}$$

The $\tilde{w}^i$ is called the *importance weight*. Normalization is needed since the area below the pdf is 1.

$$\hat{f} = \frac{\sum_{i=1}^{N} \tilde{w}^i . f\left(x^i\right)}{\sum_{i=1}^{N} \tilde{w}^i} = \sum_{i=1}^{N} w^i . f\left(x^i\right) \tag{3-8}$$

The w^i is called the *normalized importance weight*. Note that if $p\left(x^i\right) < q\left(x^i\right)$, then there are more samples than there should be with small weights. On the contrary if $p\left(x^i\right) > q\left(x^i\right)$, then there are fewer samples, and the corresponding weights increase to compensate that [50].

Taking $f(x)$ as unity gives the sampling just for $p(x)$. Generally, for effective sampling the importance density should be as close as possible to the true one. The choice of $q(x) \approx |f(x)| p(x)$ results in faster convergence than the $q(x) \approx p(x)$. As another example, consider the computation of some statistics of a rare event. To get the required statistics, approximating the true pdf, i. e. $p(x)$, by its samples may not be possible since it would be almost impossible to draw samples that correspond to such a rare event. To avoid this, a different importance density should be chosen to cover that region. Detailed information can be found in [3, 49]. Choosing a suitable importance density significantly increase the estimation performance. It is further discussed in Section 3.3.5.

In approximate grid based methods (Section 2.3.1) samples (grids) are distributed to the whole state-space. Therefore these methods are not computationally feasible especially for high dimensional state spaces, where the distribution to be estimated usually covers a relatively small area in the whole state space. Importance sampling can be viewed as changing the grid at each step adaptively. In other words, state-

space is partitioned unevenly to give greater resolution in high probability density regions. During sampling, these regions have a higher level of importance. Thus more samples are drawn from that region, making it possible to represent the distribution with fewer samples (particles).

An important property of importance sampling is that it is a fixed period algorithm where the quality of the approximation depends on the number of samples used and the chosen proposal density. Due to this reason it is widely used in particle filters.

3.2.2 Rejection Sampling

Rejection sampling, in other words accept-reject methods, generates an approximation of $p(x)$ again by drawing samples from $q(x)$. This method differs from the previous one that a sample may be rejected after its generation. Rejection sampling requires an upper bound for the true distribution. Assuming that there exists a finite constant C such that

$$p(x) < C.q(x) \text{ for every } x, \tag{3-9}$$

the procedure is the following [3]. Algorithm is shown graphically in Figure 3-1.

Algorithm 1: Rejection Sampling

- FOR i=1:N
  - Draw a sample x^i from $q(x)$
  - Draw a sample u from the uniform distribution $U_{[0,1]}$
  - If $u \le \frac{p(x^i)}{C.q(x^i)}$ then accept the sample, otherwise go to the first step
- END FOR

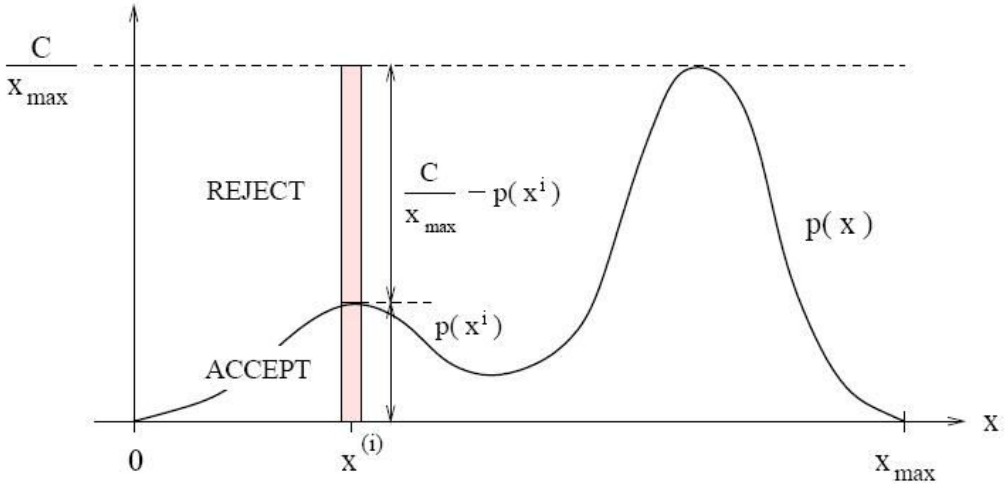

Figure 3-1: Rejection sampling. The upper bound is chosen as uniform distribution [5].

The choice of the upper bound is substantial for the performance of the algorithm. The expected value of each sample's acceptance probability in Algorithm 1 is inversely proportional to the C as shown in Equation (3-10) [5].

$$\int \frac{p(x)}{C.q(x)}.q(x).dx = \int \frac{p(x)}{C}dx = \frac{1}{C} \tag{3-10}$$

If C is too small then the samples are not reliable, if it is too large then the low acceptance rate makes the algorithm computationally inefficient [3]. The possible improvement in the sampling is illustrated in Figure 3-2 in the expense of computational power.

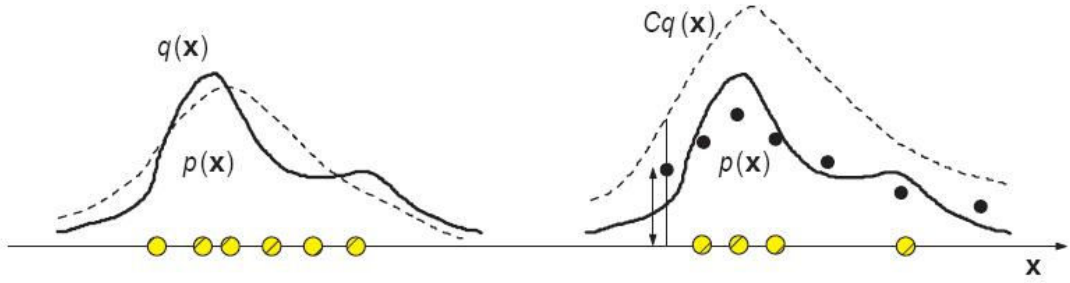

Figure 3-2: Importance sampling (left) and rejection sampling (right) [3].

The rejection sampling is not a fixed period algorithm. It provides an answer (not guaranteeing any convergence) at any time but the quality of the approximation also depends on the time spent. Due to this drawback, it is not widely used in particle filter approximations unless there are no real time fixed period constraints.

3.3 Generic Particle Filter

3.3.1 Sequential Importance Sampling (SIS) Algorithm

As stated in the above sections, the fundamental idea of the particle filer is to use a number of independent random variables, i.e. samples or particles, to represent the posterior probability. The locations of the particles in the state-space, and their relative weights are updated recursively according to the Bayesian rule [13, 3]. The basic form of generic particle filter algorithm is called sequential importance sampling (SIS) [4, 13, 3, 6, 5].

Using the importance sampling, the posterior distribution for the general case is represented by particles. At this stage, for the sake of generalization the posterior distribution includes the whole state history.

$$p\left(x_{1:k+1}|y_{1:k+1}\right) \approx \sum_{i=1}^{N} w_{k+1}^{i}.\delta\left(x_{1:k+1} - x_{1:k+1}^{i}\right) \tag{3-11}$$

where the normalized importance weights are

$$w_{k+1}^{i} \propto \frac{p\left(x_{1:k+1}^{i} \middle| y_{1:k+1}\right)}{q\left(x_{1:k+1}^{i} \middle| y_{1:k+1}\right)} \tag{3-12}$$

The sequential algorithm is constructed such that, at each iteration a particle set $\left\{x_{1:k}^{i}, w_{k}^{i}\right\}_{i=1}^{N}$ for $p\left(x_{1:k} \middle| y_{1:k}\right)$ is assumed to be available, and a new set $\left\{x_{1:k+1}^{i}, w_{k+1}^{i}\right\}_{i=1}^{N}$ for $p\left(x_{1:k+1} \middle| y_{1:k+1}\right)$ is obtained by using the latest observation y_{k+1}. As the basic idea of SIS, the importance density is formed sequentially. Therefore the chosen importance density should factorize like

$$q\left(x_{1:k+1} \middle| y_{1:k+1}\right) = q\left(x_{k+1} \middle| x_{1:k}, y_{1:k+1}\right) q\left(x_{1:k} \middle| y_{1:k}\right) \tag{3-13}$$

Hence, the new sample set can be reached by augmenting the new state component ($x_{k}^{i} \sim q\left(x_{k+1} \middle| x_{1:k}, y_{1:k+1}\right)$) to the old sample set [13]. To find the weight update relation, $p\left(x_{1:k+1} \middle| y_{1:k+1}\right)$ is decomposed like,

$$\begin{aligned} p\left(x_{1:k+1} \middle| y_{1:k+1}\right) &= \frac{p\left(y_{k+1} \middle| x_{1:k+1}, y_{1:k}\right) p\left(x_{1:k+1} \middle| y_{1:k}\right)}{p\left(y_{k+1} \middle| y_{1:k}\right)} \\ &= \frac{p\left(y_{k+1} \middle| x_{1:k+1}, y_{1:k}\right) p\left(x_{k+1} \middle| x_{1:k}, y_{1:k}\right)}{p\left(y_{k+1} \middle| y_{1:k}\right)} . p\left(x_{1:k} \middle| y_{1:k}\right) \\ &= \frac{p\left(y_{k+1} \middle| x_{k+1}\right) p\left(x_{k+1} \middle| x_{k}\right)}{p\left(y_{k+1} \middle| y_{1:k}\right)} . p\left(x_{1:k} \middle| y_{1:k}\right) \end{aligned} \tag{3-14}$$

The denominator can be viewed as a normalizing constant, and in the estimation literature it is referred as the *evidence*. So the posterior density to be approximated is proportional to the equation below.

$$p\left(x_{1:k+1} \middle| y_{1:k+1}\right) \propto p\left(y_{k+1} \middle| x_{k+1}\right) p\left(x_{k+1} \middle| x_{k}\right) p\left(x_{1:k} \middle| y_{1:k}\right) \tag{3-15}$$

The importance weights in Equation (3-12) can be obtained by combining Equations (3-13) and (3-15).

$$w_{k+1}^{i} \propto \frac{p\left(y_{k+1}\middle|x_{k+1}^{i}\right)p\left(x_{k+1}^{i}\middle|x_{k}^{i}\right)p\left(x_{1:k}^{i}\middle|y_{1:k}\right)}{q\left(x_{k+1}^{i}\middle|x_{1:k}^{i}, y_{1:k+1}\right)q\left(x_{1:k}^{i}\middle|y_{1:k}\right)} \tag{3-16}$$

Furthermore, the importance density is chosen such that it satisfies

$$q\left(x_{k+1}\middle|x_{1:k}, y_{1:k+1}\right) = q\left(x_{k+1}\middle|x_{k}, y_{k+1}\right) \tag{3-17}$$

i.e. the Assumption 1 and Assumption 2 in Section 2.1 are valid for the importance density too. In that case, the weight update relation can be written as

$$w_{k+1}^{i} \propto \frac{p\left(y_{k+1}\middle|x_{k+1}^{i}\right)p\left(x_{k+1}^{i}\middle|x_{k}^{i}\right)}{q\left(x_{k+1}^{i}\middle|x_{k}^{i}, y_{k+1}\right)}w_{k}^{i} \tag{3-18}$$

Since only the filtered estimate is required at each step, the posterior density is

$$p\left(x_{k+1}\middle|y_{k+1}\right) \approx \sum_{i=1}^{N} w_{k+1}^{i}.\delta\left(x_{k+1} - x_{k+1}^{i}\right) \tag{3-19}$$

The SIS particle filter algorithm is expressed below [13].

Algorithm 2: SIS Particle Filter

$$\left\{x_{k+1}^i, w_{k+1}^i\right\}_{i=1}^N = SIS\left(\left\{x_k^i, w_k^i\right\}_{i=1}^N, y_{k+1}\right)$$

- FOR *i*=1:N
  - Draw a sample x_{k+1}^i from $q\left(x_{k+1}^i \middle| x_k^i, y_{k+1}\right)$
  - Update the particle weight: $\tilde{w}_{k+1}^i = \frac{p\left(y_{k+1}\middle|x_{k+1}^i\right)p\left(x_{k+1}^i\middle|x_k^i\right)}{q\left(x_{k+1}^i\middle|x_k^i, y_{k+1}\right)} w_k^i$
- END FOR
- FOR i=1:N
  - Normalize the particle weights: $w_{k+1}^i = \frac{\tilde{w}_{k+1}^i}{\sum_{j=1}^N \tilde{w}_{k+1}^j}$
- END FOR

Expressing the posterior by particles makes it possible to approximate the integrations in Bayesian filtering easily. This structure also brings two important advantages. First, any kind of probability distribution can be used since the distributions are represented by samples chosen from the state-space instead of some parameters like mean and covariance. Second, state and measurement equations can be non-linear since the particles are propagated by using the same equations instead of their approximated versions.

3.3.2 Degeneracy Phenomenon

The SIS algorithm updates the weights of the particles at each step according to the measurements taken. No matter how big number of particles is used, after some iteration eventually all but a few particles will have negligible weights. That is the variance of the weights always tends to increase with time [13]. While processing those small unimportant particles makes the algorithm ineffective, the overall

ability to approximate the posterior is degraded due to the reduced number of effective particles. This problem is referred as the *degeneracy phenomenon*. Needless to say, if the posterior cannot be approximated well, then algorithm can easily diverge, i.e. weights of all particles become zero.

In order to overcome the degeneracy problem a number of modifications should be done on the basic structure. First precaution is to eliminate the particles with smaller weights and focus on the regions where the particles with higher weights exist. The mentioned process is called as resampling, and further described in Section 3.3.3. A measure for degeneracy is the effective sample size N_{eff} which is defined below [3, 13].

$$N_{eff} = \frac{N}{1 + Var\left(w_k^{*i}\right)} \tag{3-20}$$

Where w_k^{*i} is referred to as the "true weight" [13].

$$w_k^{*i} = \frac{p\left(x_{k+1}^i \middle| y_{1:k+1}\right)}{q\left(x_{k+1}^i \middle| x_k^i, y_{k+1}\right)} \tag{3-21}$$

Since N_{eff} cannot be computed exactly, an estimate of it is used.

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N}\left(w_k^i\right)^2} \tag{3-22}$$

The second precaution is, of course, choosing the importance (Section 3.3.5) density as close as possible to the posterior density. By this way, particles are tried to be kept in the important regions of the state-space, which reduces the possibility to encounter particles with negligible weights.

3.3.3 Resampling

A resampling step is inserted into the particle filter algorithms in order to prevent degeneracy. As a result, some particles are deleted and some particles are duplicated according to their relative weights.

Another point of view is that the new information came from the measurement is inserted into the weights of the particles. Since the particle filter tries to carry all the information with the positions of particles, this information should be transferred back somehow. Resampling step arranges the particles such that they all have same importance, and all contribute to the approximation by their individual position in the state-space.

Although resampling helps to overcome the degeneracy phenomenon, it unfortunately introduces other problems. Resampling causes the samples that have high importance weights to be statistically selected many times, thus the algorithm suffers from the loss of diversity [3]. As the time passes particles tend to be piled up to some locations, which degrade the approximation quality of the whole particle cloud. This occurs especially in the case of small process noise since the particles at the same place cannot separate enough. It is also referred as *sample impoverishment*. It is also mentioned in [13] that using the particle filter is not appropriate when the process noise is zero.

Besides particle filter already has a serious computational cost due to the number of particles, resampling makes it even harder by limiting the opportunity to parallelize. All particles should be combined during resampling.

Note that, the posterior density is approximated by the particles, i.e. by a discrete valued distribution. Resampling step, in fact, generates a new set of particles from that approximate discrete representation of posterior density. In the following subsections some resampling algorithms are presented.

3.3.3.1 Systematic Resampling

Systematic resampling (or minimum variance sampling) has the complexity $O(N)$, and given below [13]. U is the uniform distribution over the specified interval. The parent index of each particle is calculated for the auxiliary particle filter (APF) which is described in Section 3.4.1.

Algorithm 3: Systematic Resampling

$\left\{x_k^{j*}, w_k^{j*}, i^j\right\}_{j=1}^{N} = RESAMPLE\left(\left\{x_k^i, w_k^i\right\}_{i=1}^{N}\right)$

- Construct the CDF
  - $c_1 = 0$
  - FOR *i*=1:N
    - $c_{i+1} = c_i + w_k^j$
  - END FOR
- Start from the initial point of CDF: $i = 1$
- Draw a starting point: $u_1 \sim U\left[0, N^{-1}\right]$
- FOR *j*=1:N
  - Move along the CDF: $u_j = u_1 + N^{-1}.(j-1)$
  - WHILE $u_j > c_i$
    - $i = i + 1$
  - END WHILE
  - Assign sample: $x_k^{j*} = x_k^{i-1}$
  - Assign weight : $w_k^{j*} = N^{-1}$
  - Assign parent : $i^j = i - 1$
- END FOR

Algorithm assumes that the weights are continuous random variables in the interval (0, 1) which are randomly ordered [3]. The grid points (u_j's) in the each interval $[c_i, c_{i+1})$ are counted, and particles corresponding to that interval are either duplicated by that amount or eliminated if there are no grid points.

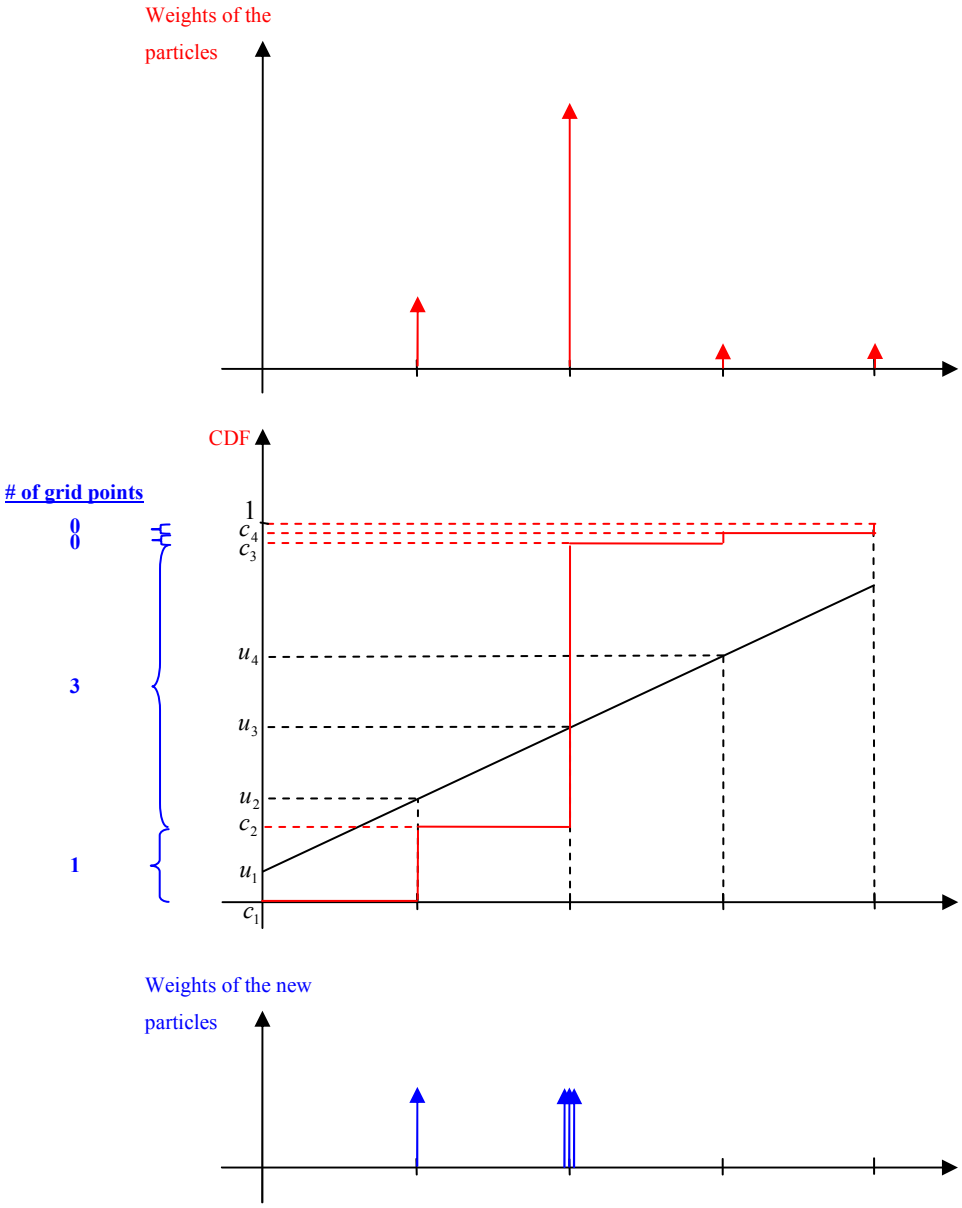

Figure 3-3: Illustration of the systematic resampling for 4 particles. All three particles are in the same place in the last graph.

Systematic resampling is easy to implement and it minimizes the Monte Carlo (MC) variation $Var\left(w_k^i\right)$.

3.3.3.2 Other Resampling Methods

There are also many other resampling schemes trying to reduce the negative effects mentioned in Section 3.3.3. Some of them are briefly explained below.

Multinomial Resampling [3]: Algorithm is similar to the systematic resampling:

- Construct the CDF for the weights
- For each particle
    - Draw a sample from the uniform distribution *U (0, 1)*
    - Find the i[th] particle such that $c_{i-1} \le u < c_i$
    - Duplicate x_k^i and assign its weight to N^{-1}

Residual Resampling [3]: It is a partially deterministic approach.

- For each particle
    - Create $k_i = \lfloor N.w_k^i \rfloor$ copies of x_k^i
- Create $N_r = N - \sum_{i=1}^{N} k_i$ i.i.d draws from the particle set with probabilities proportional to $N.w_k^i - k_i$
- Assign all weights to N^{-1}

There are also more sophisticated resampling methods like *stratified resampling* and *Markov Chain Monte Carlo (MCMC) resampling* (Metropolis-Hastings Algorithm, Gibbs Sampling). Detailed information can be found in [3, 49, 50].

3.3.4 Sequential Importance Resampling (SIR) Algorithm

For the reasons explained above, particle filter needs a modification called the resampling step. According to when and how does the resampling is done various particle filter algorithms are developed. The most common two algorithms are presented below.

First approach is to apply resampling when there is a certain degree of degeneracy. The effective particle size in Section 3.3.2 can be used for that purpose.

Algorithm 4: SIS Particle Filter with Resampling

$\left\{x_{k+1}^i, w_{k+1}^i\right\}_{i=1}^N = SIS\_R\left(\left\{x_k^i, w_k^i\right\}_{i=1}^N, y_{k+1}\right)$

- FOR *i*=1:N
  - Draw a sample x_{k+1}^i from $q\left(x_{k+1}^i \middle| x_k^i, y_{k+1}\right)$
  - Update the particle weight: $\tilde{w}_{k+1}^i = \dfrac{p\left(y_{k+1}\middle|x_{k+1}^i\right)p\left(x_{k+1}^i\middle|x_k^i\right)}{q\left(x_{k+1}^i\middle|x_k^i, y_{k+1}\right)} w_k^i$
- END FOR
- FOR *i*=1:N
  - Normalize the particle weights: $w_{k+1}^i = \dfrac{\tilde{w}_{k+1}^i}{\sum_{j=1}^N \tilde{w}_{k+1}^j}$
- END FOR
- Calculate the effective sample size: $\hat{N}_{eff} = \dfrac{1}{\sum_{i=1}^N \left(w_k^i\right)^2}$
- IF $\hat{N}_{eff} < N_{Threshold}$ THEN resample using Algorithm 3

Second widely used method is to apply the resampling at each iteration of the filter. The particular filter is named after the sequence of its steps: *Sampling Importance Resampling (SIR) Particle Filter*. Its steps are illustrated in Algorithm 5.

Algorithm 5: SIR Particle Filter

$$\left\{x_{k+1}^i, w_{k+1}^i\right\}_{i=1}^N = SIR\left(\left\{x_k^i, w_k^i\right\}_{i=1}^N, y_{k+1}\right)$$

- FOR *i*=1:N
  - Draw a sample x_{k+1}^i from $q\left(x_{k+1}^i \middle| x_k^i, y_{k+1}\right)$
  - Update the particle weight: $\tilde{w}_{k+1}^i = \frac{p\left(y_{k+1} \middle| x_{k+1}^i\right) p\left(x_{k+1}^i \middle| x_k^i\right)}{q\left(x_{k+1}^i \middle| x_k^i, y_{k+1}\right)} w_k^i$
- END FOR
- FOR *i*=1:N
  - Normalize the particle weights: $w_{k+1}^i = \frac{\tilde{w}_{k+1}^i}{\sum_{j=1}^N \tilde{w}_{k+1}^j}$
- END FOR
- Resample using Algorithm 3

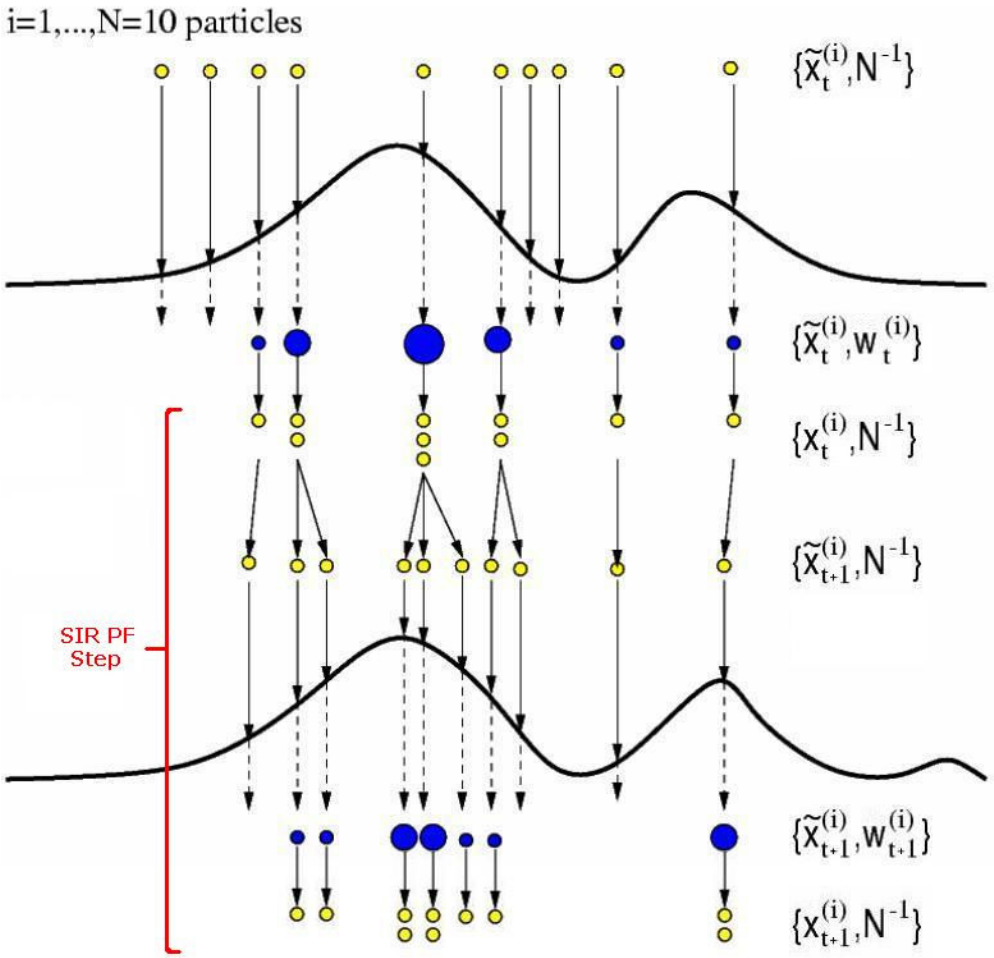

Figure 3-4: Illustration of the SIR Particle Filter steps [23].

3.3.5 Choice of Importance Density

As explained in Section 3.2.1 the choice of the importance density plays a crucial role in the filter performance. While the resampling tries to solve an existing degeneracy, the quality of the importance density affects the severity of degeneracy at the next iteration. With a good choice the particles usually stay in the high probability regions and as a consequence, degeneracy would be less severe after the weight update.

Posterior Distribution (Optimal Important Density):

Naturally the optimal importance density is the posterior distribution itself [13]. Considering the Equation (3-13)I it can be expressed like below.

$$q_{opt}\left(x_{k+1}\middle|x_k^i, y_{k+1}\right) \overset{\Delta}{=} p\left(x_{k+1}\middle|x_k^i, y_{k+1}\right) = \frac{p\left(y_{k+1}\middle|x_{k+1}\right).p\left(x_{k+1}\middle|x_k^i\right)}{p\left(y_{k+1}\middle|x_k^i\right)} \tag{3-23}$$

Inserting it into the weight update equation (Equation (3-18)) of the particle filter:

$$w_{k+1}^i \propto p\left(y_{k+1}\middle|x_k^i\right)w_k^i = w_k^i.\int p\left(y_{k+1}\middle|\tau\right).p\left(\tau\middle|x_k^i\right)d\tau \tag{3-24}$$

However there is some practical difficulties of using the optimal importance density. First drawing samples directly from the posterior may not be possible, second evaluating the integral in Equation (3-24) may not be easy.

The above formulation can easily be used when the state is a member of a finite set. In that case it is possible to take samples from the posterior and the integral in the weight update equation turns into summation. Using the optimal importance density for the discrete modal states of a Jump-Markov system is an example [4].

The problems caused by the optimal importance density can be overcome for the special case where the posterior density is Gaussian. The system described by the Equations (3-25) and (3-26) is an example of this case. Detailed formulation can be found in [4].

$$x_{k+1} = f_k\left(x_k, u_k\right) + w_k \tag{3-25}$$

$$y_k = H_k.x_k + v_k \tag{3-26}$$

Prior Distribution:

The prior distribution is probably the most common choice for the importance density due to its intuitiveness and simplicity. The resultant weight update equation

can be found by the same way. In this case, particles are propagated by using the state transition model (i.e. prior) and weighted by using the measurement model (i.e. likelihood).

$$q\left(x_{k+1}\middle|x_k^i, y_{k+1}\right) \stackrel{\Delta}{=} p\left(x_{k+1}\middle|x_k^i\right) \tag{3-27}$$

$$w_{k+1}^i \propto p\left(y_{k+1}\middle|x_{k+1}^i\right)w_k^i \tag{3-28}$$

Obviously it is not always a very good approximation of the posterior since the current observation is neglected.

Likelihood Distribution:

As described in Equation (3-18) the posterior distribution contains the total information which is present in the prior and likelihood separately. Normal formulation of the particle filter is based on drawing particles according to the prior and weighting them according to their likelihood. However if the likelihood is tighter than the prior, i.e. closer in shape to the posterior, it may be better to change the order. Sampling from the tighter (dominant) distribution makes the particles to stay important regions of the state-space, and then the minor corrections due to the broader distribution can easily be done by updating the weights.

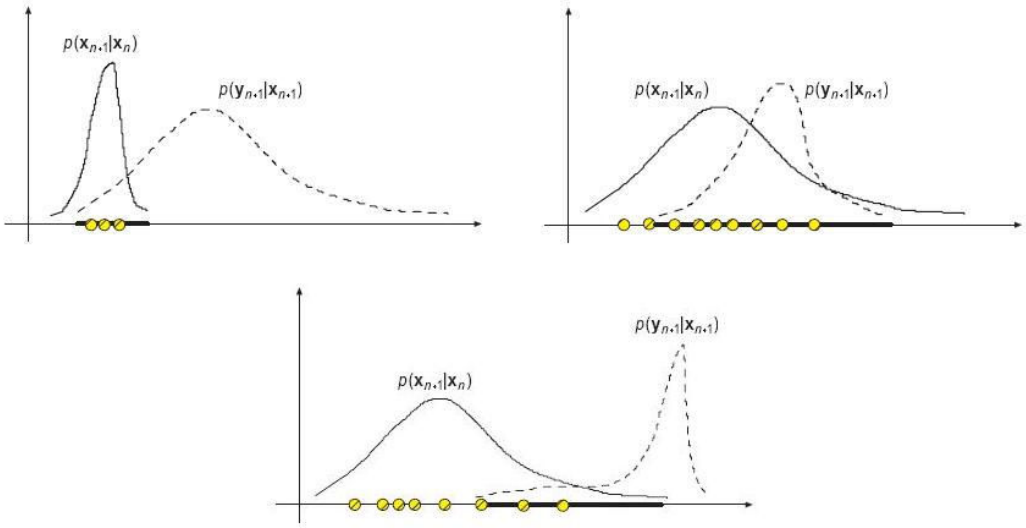

Figure 3-5: Illustration of the relative shape of prior and likelihood [3].

First two graphs in Figure 3-5 illustrates the cases where using the prior as importance density is appropriate. But, as seen in the third graph if the prior is much broader, then the generated particles have difficulty to incorporate the new information supplied by the likelihood.

The straightforward idea is to use the likelihood as the importance density and then assign the weights proportional to the state transition density (Equations (3-29) and (3-30)). The resultant algorithm is called the *Likelihood Particle Filter* [3]. Idea behind is similar to the Auxiliary Particle Filter (Section 3.4.1), since during the particle generation they both try to select the ones having high likelihoods.

$$q\left(x_{k+1}\middle|x_k^i, y_{k+1}\right) \stackrel{\Delta}{=} p\left(y_{k+1}\middle|x_{k+1}^i\right) \tag{3-29}$$

$$w_{k+1}^i \propto p\left(x_{k+1}^i\middle|x_k^i\right)w_k^i \tag{3-30}$$

Unfortunately, using the likelihood is not so straightforward. Taking samples may not be possible since the mapping (Equation (2-4)) is usually not one-to-one.

Therefore for given measurement a unique (or a finite number of) state distribution $p(x_{k+1}|y_{k+1})$ may not be found.

Other Suboptimal Distributions:

If the optimal importance density is not available, then its approximations may be used. These approximations are located between the prior and the posterior in terms of the information they contain. One way is to assume a Gaussian distribution for the posterior and obtain it by local linearization techniques. Utilization of the EKF is a frequently encountered example of that method. Estimation of the importance density by UKF (or SPKF) is also possible (UPF) [23, 24, 25].

In some cases the peaked likelihood can be factorized into several broader distributions since each distribution is a function of a different part of the state. Hence samples can be generated from each importance density separately and combined later. The method is referred as *partitioned sampling* and useful especially when the measurement components are independent and have different individual likelihood models [3].

3.4 Other Particle Filter Algorithms

According to how the importance density is chosen and resampling is done different particle filters have been proposed. In Section 3.3.5 different forms of particle filters (ex: Likelihood PF, EKF-PF, UPF) are briefly mentioned. They are also referred as Local Linearization Particle Filters. Similarly applied resampling (Section 3.3.3) method also changes the whole algorithm (ex: MCMC-PF).

Particle filters basic intension is to overcome the problems introduced by non-linear and non-Gaussian states. In many cases, not all the states are defined by non-linear relations or not all the noise components are non-Gaussian. Therefore a straightforward idea is to use particle filtering only for those state components, and the rest can be handled by optimal methods (i.e. Kalman filter). Resultant class of

algorithms is referred as Marginalized Particle Filter (MPF) or Rao-Blackwellized Particle Filter.

Three important particle filter variants are described in the following subsections.

3.4.1 Auxiliary Particle Filter (APF)

In the APF (also called Auxiliary SIR Filter - ASIR) the usual order of SIS/SIR particle filter steps are modified to alleviate degeneracy problem. The underlying idea is to select the particles that will be propagated in the subsequent updated estimate. A brief comparison of SIR-PF and APF in terms of algorithm steps is given in Table 3-1

Table 3-1: Algorithm steps of APF compared to SIR PF.

SIR PF Steps
• Draw samples from the importance density (i.e. shift the particles) • Update the weights (i.e. evaluate the particles) • Resample
APF Steps
• Weight the particles based on some characterization of $p\left(x_{k+1}\middle\|x_k^i\right)$ (i.e. do a pre-evaluation based on the prior information). • Resample • Draw samples from the importance density (i.e. shift the particles) • Re-evaluate the particle weights in order to take the measurement into account

By applying resampling before, more promising particles are tried to be chosen for the shifting step. To decide which particle is more likely to survive at the end of the current cycle, an auxiliary state variable i^j is introduced [4, 13, 45]. It denotes the index of the j^{th} particle at time k. Then the derivation of the posterior density is

$$\begin{aligned} p\left(x_{k+1}, i^j \middle| y_{1:k+1}\right) &= \frac{p\left(y_{k+1}\middle|x_{k+1}\right).p\left(x_{k+1}, i^j \middle| y_{1:k}\right)}{p\left(y_{k+1}\middle|y_{1:k}\right)} \\ &\propto p\left(y_{k+1}\middle|x_{k+1}\right).p\left(x_{k+1}, i^j \middle| y_{1:k}\right) \\ &= p\left(y_{k+1}\middle|x_{k+1}\right).p\left(x_{k+1}\middle|i^j, y_{1:k}\right).p\left(i^j \middle| y_{1:k}\right) \\ &= p\left(y_{k+1}\middle|x_{k+1}\right).p\left(x_{k+1}\middle|x_k^i\right)w_k^i \end{aligned} \tag{3-31}$$

Note that if the predictive likelihood $p\left(y_{k+1}\middle|x_k^i\right)$ (which will be large for the "good" particles) can be computed analytically, then the optimal importance density itself is used (Equation (3-23)). If not, then an approximation should be done [3].

Although an auxiliary variable is augmented to the state it is ignored during the generation of samples. That is to say, taking the sample from the joint importance density $q\left(x_{k+1}, i^j \middle| y_{1:k+1}\right)$, then omitting i^j. The importance density is

$$q\left(x_{k+1}, i^j \middle| y_{1:k+1}\right) \propto p\left(y_{k+1}\middle|\mu_{k+1}^i\right).p\left(x_{k+1}\middle|x_k^i\right)w_k^i \tag{3-32}$$

where μ_{k+1}^i is some characterization of x_{k+1} when x_k^i is given [13]. It can be the expected value or only a sample.

$$\begin{aligned} \mu_{k+1}^i &= E\left\{x_{k+1}\middle|x_k^i\right\} \\ \mu_{k+1}^i &\sim p\left(x_{k+1}\middle|x_k^i\right) \end{aligned} \tag{3-33}$$

The importance density is also factorized like:

$$q\left(x_{k+1}, i^j \middle| y_{1:k+1}\right) = q\left(i^j \middle| y_{1:k+1}\right)q\left(x_{k+1}\middle|i^j, y_{1:k+1}\right) \tag{3-34}$$

Combining with the Equation (3-32):

$$q\left(x_{k+1}\middle|i^{j},y_{1:k+1}\right)\overset{\Delta}{=}p\left(x_{k+1}\middle|x_{k}^{i}\right) \tag{3-35}$$

$$q\left(i^{j}\middle|y_{1:k+1}\right)\propto p\left(y_{k+1}\middle|\mu_{k+1}^{i}\right)w_{k}^{i} \tag{3-36}$$

After the initial weight update according to the Equation (3-36) and resampling, the final weight correction is done by Equation (3-37) (incorporating the new information due to the measurement). The index of the new set of particles after resampling is denoted as j ($\left\{x_{k}^{j},w_{k}^{j}\right\}_{j=1}^{N}$) instead of i. Algorithm for APF is also given below [4].

$$w_{k+1}^{j}\propto\frac{p\left(y_{k+1}\middle|x_{k+1}^{j}\right)p\left(x_{k+1}^{j}\middle|x_{k}^{i^{j}}\right)}{q\left(x_{k+1}^{j},i^{j}\middle|y_{1:k+1}\right)}w_{k}^{j}=\frac{p\left(y_{k+1}\middle|x_{k+1}^{j}\right)}{p\left(y_{k+1}\middle|\mu_{k+1}^{i^{j}}\right)} \tag{3-37}$$

Algorithm 6: Auxiliary Particle Filter

$$\left\{x_{k+1}^{j}, w_{k+1}^{j}\right\}_{j=1}^{N} = APF\left(\left\{x_{k}^{i}, w_{k}^{i}\right\}_{i=1}^{N}, y_{k+1}\right)$$

- FOR i=1:N
  - Calculate μ_{k+1}^{i}
  - Calculate $\tilde{w}_{k+1}^{i} = q\left(i^{j} \middle| y_{1:k+1}\right) \propto p\left(y_{k+1} \middle| \mu_{k+1}^{i}\right) w_{k}^{i}$
- END FOR
- FOR i=1:N
  - Normalize the particle weights: $w_{k+1}^{i} = \dfrac{\tilde{w}_{k+1}^{i}}{\sum_{j=1}^{N} \tilde{w}_{k+1}^{j}}$
- END FOR
- Resample using Algorithm 3
- FOR j=1:N
  - Draw a sample x_{k+1}^{j} from $q\left(x_{k+1} \middle| i^{j}, y_{1:k+1}\right) = p\left(x_{k+1} \middle| x_{k}^{i^{j}}\right)$
  - Update the particle weight: $w_{k+1}^{j} = \dfrac{p\left(y_{k+1} \middle| x_{k+1}^{j}\right)}{p\left(y_{k+1} \middle| \mu_{k+1}^{i^{j}}\right)}$
- END FOR

The resampling step between two weight updates selects the good particles according to the quality of the pre-evaluation. In other words the performance of APF depends on how well μ_{k+1}^{i} characterizes the $p\left(x_{k+1} \middle| x_{k}^{i}\right)$. However, if the process noise is large, then a point estimate does not give sufficient information about $p\left(x_{k+1} \middle| x_{k}^{i}\right)$. As a result APF may be worse than normal SIR-PF.

3.4.2 Regularized Particle Filter (RPF)

During resampling a new set of particles are selected from the discrete approximation of posterior density instead of a continuous one. Hence after this step duplicated particles occupy the same place. If they cannot be spread out then all particles may collapse to the same point in the state-space, which is a severe case of sample impoverishment [4, 13]. Regularized Particle Filter (RPF) tries to overcome this problem by introducing a continuous approximation of the posterior.

Instead of Equation (3-11), the below relation is used.

$$p\left(x_{1:k+1}\middle|y_{1:k+1}\right)\approx\sum_{i=1}^{N}w_{k+1}^{i}.K_h\left(x_{1:k+1}-x_{1:k+1}^{i}\right) \tag{3-38}$$

$$K_h\left(x\right)=\frac{1}{h^{n_x}}.K\left(\frac{x}{h}\right) \tag{3-39}$$

where $K\left(x\right)$ is the Kernel density, $h>0$ is the Kernel bandwidth, n_x is the dimension of the state-space [4]. Kernel density function and its bandwidth determine the quality of the approximation in Equation (3-38). For particles having equal weights and Gaussian posterior density the optimal choices are the followings:

Epanechnikov kernel: $$K_{opt}\left(x\right)=\begin{cases}\dfrac{n_x+2}{2\,c_{n_x}}\left(1-\left\|x\right\|^2\right), & if\ \left\|x\right\|<1\\ 0, & otherwise\end{cases} \tag{3-40}$$

$$h_{opt}=A.N^{1/\left(n_x+4\right)} \tag{3-41}$$

$$A=\left[8\,c_{n_x}^{-1}\left(n_x+4\right)\left(2\sqrt{\pi}\right)^{n_x}\right]^{1/\left(n_x+4\right)} \tag{3-42}$$

where c_{n_x} is the volume of the unit hypersphere in R^{n_x}.

The usage of Kernel density is illustrated in Figure 3-6. Note that, the Gaussian distribution itself can also be used as a Kernel density.

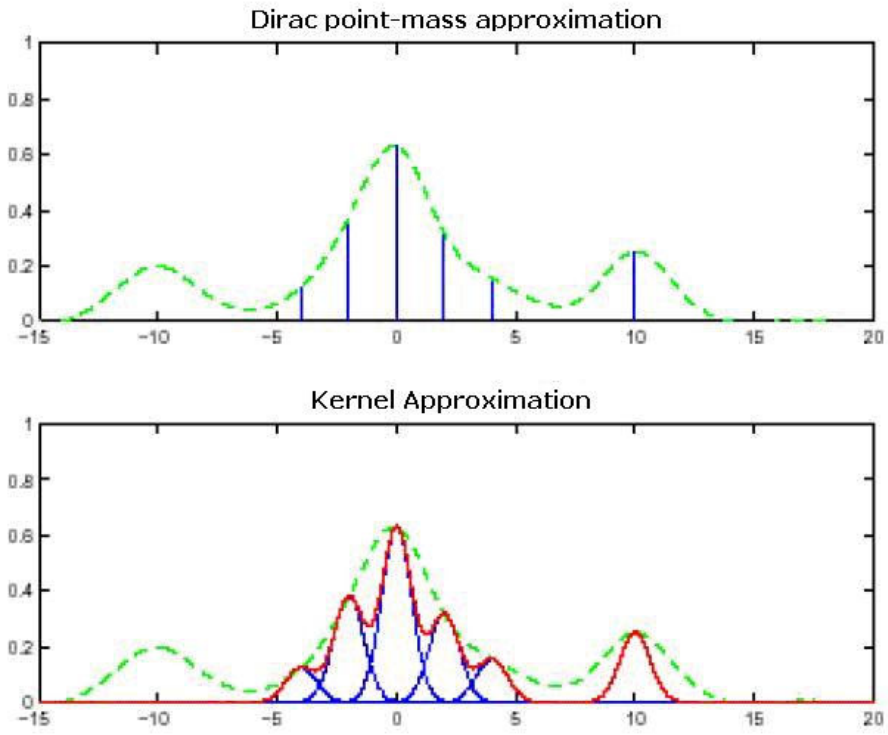

Figure 3-6: Kernel approximation.

Algorithm 7: Regularized Particle Filter [4]

$$\left\{x_{k+1}^j, w_{k+1}^j\right\}_{j=1}^N = RPF\left(\left\{x_k^i, w_k^i\right\}_{i=1}^N, y_{k+1}\right)$$

- FOR *i*=1:N
  - Draw a sample x_{k+1}^i from $q\left(x_{k+1}^i \middle| x_k^i, y_{k+1}\right)$
  - Update the particle weight: $\tilde{w}_{k+1}^i = \frac{p\left(y_{k+1}\middle|x_{k+1}^i\right)p\left(x_{k+1}^i\middle|x_k^i\right)}{q\left(x_{k+1}^i\middle|x_k^i, y_{k+1}\right)} w_k^i$
- END FOR
- FOR *i*=1:N
  - Normalize the particle weights: $w_{k+1}^i = \frac{\tilde{w}_{k+1}^i}{\sum_{j=1}^N \tilde{w}_{k+1}^j}$
- END FOR
- Calculate the effective sample size: $\hat{N}_{eff} = \frac{1}{\sum_{i=1}^N \left(w_k^i\right)^2}$
- IF $\hat{N}_{eff} < N_{Threshold}$
  - Calculate the empirical covariance matrix S_k of $\left\{x_k^i, w_k^i\right\}_{i=1}^N$
  - Compute D_k such that $D_k D_k^T = S_k$
  - Resample using Algorithm 3
  - FOR *i*=1:N
    - Draw $\varepsilon^i \sim K$ from the Epanechnikov Kernel
    - Jitter the resampled particles:

      $$x_{k+1}^i = x_{k+1}^i + h_{opt} D_k \varepsilon^i$$
  - END FOR
- END IF

3.4.3 Multiple Model Particle Filter (MM-PF)

Like in Kalman filter based methods, multiple state transition models can be utilized in particle filter algorithms to represent the dynamic system more precisely. PF can be used as a block in other frameworks like IMM [32, 51, 52, 53, 54]. For more sophisticated applications, the internal structure of PF is modified in order to exploit its characteristic properties.

In [4] the MM-PF is constructed based on a regime conditioned SIS step. First the regime variable for each particle is drawn based on the transition probabilities. Then each particle is propagated through the importance density assigned to its regime.

The MM-PF structure presented here is taken from [6, 45]. The difference from SIR-PF is that, at each step every particle is propagated through all system models. So the intermediate particle count is the particle count (N) times the number of models (M). The drawn particles are weighted according to the common measurement and the corresponding Markov transition probabilities. During resampling total number is reduced to "N" again. In Algorithm 8 $T\left(\omega_{k+1}^{i} = j \middle| \omega_{k}^{i}\right)$ denotes the transition probability, i.e. probability of the system being in j^{th} mode given that its previous mode was ω_{k}^{i}.

Algorithm 8: MM-PF

$$\left\{x_{k+1}^{i}, w_{k+1}^{i}\right\}_{i=1}^{N} = MM\_PF\left(\left\{x_{k}^{i}, w_{k}^{i}\right\}_{i=1}^{N}, y_{k+1}\right)$$

- FOR *i*=1:N
  - FOR *j*=1:M
    - Draw a sample $x_{k+1}^{i,j}$ from $q^{j}\left(x_{k+1}^{i,j}\middle|x_{k}^{i}, y_{k+1}\right)$
    - Update the particle weight:

$$\tilde{w}_{k+1}^{i,j} = \frac{p\left(y_{k+1}\middle|x_{k+1}^{i,j}\right)p^{j}\left(x_{k+1}^{i}\middle|x_{k}^{i}\right)}{q^{j}\left(x_{k+1}^{i,j}\middle|x_{k}^{i}, y_{k+1}\right)}T\left(\omega_{k+1}^{i} = j\middle|\omega_{k}^{i}\right)w_{k}^{i}$$

  - END FOR
- END FOR
- FOR *i*=1:N
  - Normalize the particle weights: $w_{k+1}^{i,j} = \dfrac{\tilde{w}_{k+1}^{i,j}}{\sum_{j=1}^{M}\sum_{i=1}^{N}\tilde{w}_{k+1}^{i,j}}$
- END FOR
- Resample using a modified version of Algorithm 3
  - $\left\{x_{k}^{i}, w_{k}^{i}\right\}_{i=1}^{N} = RESAMPLE\left(\left\{\left\{x_{k}^{i,j}, w_{k}^{i,j}\right\}_{i=1}^{N}\right\}_{j=1}^{M}\right)$

CHAPTER 4

SINGLE TARGET TRACKING

4.1 General Information

The Radar (radio detection and ranging) is probably the mostly used sensor for airborne target tracking applications. Radar's basic operating principle is to send a signal and detect the echo from the target. By analyzing the returned signal, a great deal of information about the target can be obtained. Extensive information on radars and radar signal processing can be found in [37, 38]. It can be used as an active or passive sensor. Active radar, which is the most widely used one, measures the relative angle (azimuth & elevation) and range. For Doppler-radars, range rate is also available as a measurement. On the other hand, passive radar does not emit energy and can only measure the angle. These kinds of sensors are often referred to as RWR's (Radar Warning Receiver) since they search for targets emitting radar signals. Systems can obtain different kinds of information about target by advanced signal processing methods. However, often parameters like SNR or received power is not used in tracking filters, since they are not always reliable.

There can be various subsystems measuring the same parameter by using different methods, which will yield different properties like noise, availability, false alarm rate, etc. For example while passive direction finding (DF) methods based on amplitude comparison provide faster and relatively continuous measurements,

methods using phase comparison give more accurate measurements which are not available all the time [39].

The scenarios can be divided into three groups as shown in Table 4-1. Although the whole characteristics vary, basically, the aim is to track the motions of targets relative to the observer.

Table 4-1: Tracking scenarios.

Observer	**Target**	**Example**
Stationary	Moving	Radar on ground tracking flying platforms, ground-to-air engagements
Moving	Stationary	RWR on an aircraft trying to locate a radar, air-to-ground engagements
Moving	Moving	Radar on a ship tracking an aircraft, a torpedo tracking a ship, air-to-air, sea-to-air, sea-to-sea engagements

The basic properties of interest are the target's position and velocity. The set of equations defining the evolution of the motion of platform forms the state model in Equation (4-3). According to the scenarios given in Table 4-1, there can be situations that both observer and target are moving. In that case, first their motions are considered separately with respect to a fixed point in the earth frame.

$$x_{k+1}^{tg} = f^{tg}\left(x_k^{tg}\right) + G^{tg}.w_k \tag{4-1}$$

$$x_{k+1}^{o} = f^{o}\left(x_{k}^{o}\right) - G^{o}.u_{k} \tag{4-2}$$

The unknown target input signal (usually the target acceleration) is defined as process noise w_k, and the known observer input signal (usually the observer acceleration) is denoted as $-u_k$. Then the two equations are subtracted in order to obtain the relative motion of the target (Equations (4-3) and (4-4)). Here usual assumption is that the observer platform knows both its own states and the input affecting them. This assumption is logical due to the widespread usage of the on-board inertial navigation systems (INS) aided by a global positioning system (GPS).

$$x_{k+1} = x_{k+1}^{tg} - x_{k+1}^{o} = f_{k}\left(x_{k}\right) + G^{tg}.w_{k} + G^{o}.u_{k} \tag{4-3}$$

$$f_{k}\left(x_{k}\right) = f^{tg}\left(x_{k}^{tg}\right) - f^{o}\left(x_{k}^{o}\right) \tag{4-4}$$

Therefore, if the tracking platform and the target are not described by the same dynamics or the dynamics is not linear, then the resultant state relation will be nonlinear. In that case, $f\left(x_k\right)$ is calculated under the assumption that x_k^o is known.

Set of equations defining the sensor outputs in terms of target states forms the measurement relation (4-5). Generally, there are three parameters available; the azimuth angle, elevation angle, and the range. According to the system model different combinations are considered. For two dimensional system representation only azimuth and range are taken into account, since the height is ignored. For angle-only applications azimuth and elevation angles, for bearings-only applications only azimuth angle is used.

$$y_{k} = h\left(x_{k}\right) + v_{k} \tag{4-5}$$

Under the previous assumption that the observer position, velocity and heading are known, angles are measured on a fixed reference frame on earth. To be more specific, the azimuth angle is with respect to true north and elevation angle with respect to ground frame.

During this study both two (omitting the elevation angle) and three dimensional models are used. Tracking via radar, by nature, results in nonlinear state-space models. The below mentioned coordinate systems are used to investigate the effect of different ways of representing the nonlinearity in the equations.

4.2 System Models in Cartesian Coordinate System

Basic states of the target are its relative position and velocity which are both written in the Cartesian coordinates (Equations (4-6) and (4-7), Figure 4-1, Figure 4-2). Observer's initial position is assumed to be the origin of the reference frame. "x-axis" is assigned to the observer's initial heading and x-y plane is horizontal.

$$\text{For 2D system } x = \left[d_x, d_y, v_x, v_y\right]^T \tag{4-6}$$

$$\text{For 3D system } x = \left[d_x, d_y, d_z, v_x, v_y, v_z\right]^T \tag{4-7}$$

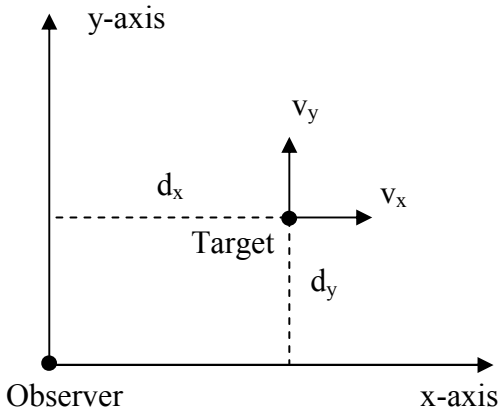

Figure 4-1: Two-dimensional representation in the Cartesian coordinate system.

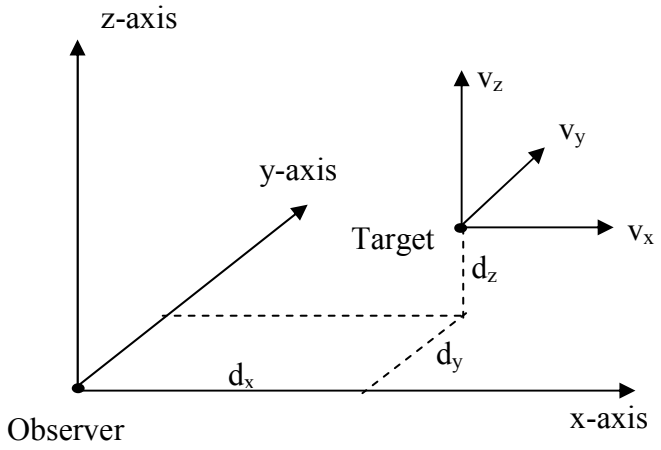

Figure 4-2: Three -dimensional representation in the Cartesian coordinate system.

4.2.1 Constant Velocity (CV) Model

The constant velocity (CV) motion model intends to represent the dynamics of a non-maneuvering platform. The noise term, which is zero-mean white Gaussian, corresponds to the accelerations. The continuous time model for two-dimensional motion is shown in Equation (4-8), where a_x and a_y are the accelerations along x & y axes respectively. Similarly, u_x and u_y correspond to the unknown acceleration components along x & y axes.

$$\dot{x} = \begin{bmatrix} \dot{d}_x \\ \dot{d}_y \\ \dot{v}_x \\ \dot{v}_y \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ a_x \\ a_y \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} . \begin{bmatrix} d_x \\ d_y \\ v_x \\ v_y \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} . \begin{bmatrix} u_x \\ u_y \end{bmatrix} = A.x + B.u \qquad (4\text{-}8)$$

The continuous time state equation (4-8) has the following solution.

$$x_{t+T} = F^{CV}(t+T,t).x_k + \int_t^{T+t} F^{CV}(T+t,\tau).B.u.d\tau \qquad (4\text{-}9)$$

where $F^{CV}(t+T,t)$ is the state transition matrix. Assuming t=0, for a time-invariant system like this, one has

$$F^{CV}(T) = F^{CV}(T,0) = e^{AT} \tag{4-10}$$

Writing the exponential in terms of infinite series, one can get

$$e^{AT} = \sum_{i=0}^{\infty} \frac{(AT)^i}{i!} = I + AT + \frac{(AT)^2}{2!} + \ldots \tag{4-11}$$

Solution is simple since only the first two terms are nonzero. Similarly by evaluating the integral one can found the following discrete time counterpart.

$$x_{k+1} = F^{CV}.x_k + G^{CV}.u_k = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.x_k + \begin{bmatrix} T^2/2 & 0 \\ 0 & T^2/2 \\ T & 0 \\ 0 & T \end{bmatrix}.u_k \tag{4-12}$$

From this point on equations are expressed in terms of steps (k) instead of time (t),. The constant time difference between the steps is denoted as "T". For three-dimensional motion the state equation is very similar.

$$x_{k+1} = \begin{bmatrix} I_3 & T.I_3 \\ 0 & I_3 \end{bmatrix}.x_k + \begin{bmatrix} T^2/2.I_3 \\ T.I_3 \end{bmatrix}.u_k = F^{CV}.x_k + G^{CV}.u_k \tag{4-13}$$

If (4-12) or (4-13) is used as a motion model for observer, u_k represents the known acceleration input; if it is used for target, then u_k represents the unknown acceleration and called w_k, process noise.

Since the acceleration is assumed to be an independent (i.e. white noise) process, this relation is also called *white-noise acceleration model*. Note that, noise term used for the acceleration model has a much higher intensity than the noise in CV model in order to represent target maneuver. However, one should keep in mind

that a maneuver, in fact, aims accomplishing a certain task, and thus rarely is independent of time.

4.2.2 Constant Acceleration (CA) Model

In this model, the acceleration is assumed to be a process with independent increments. According to [5], CA model is intended to represent the substantial, but transient, accelerations that are present at the beginning and the end of the maneuvers (e.g. the transition from constant velocity flight to a coordinated turn). In this model, the accelerations along each axis also take place as state variables. New state vectors for 2D and 3D motions are given respectively, as follows:

$$x = \left[d_x, d_y, v_x, v_y, a_x, a_y\right]^T \tag{4-14}$$

$$x = \left[d_x, d_y, d_z, v_x, v_y, v_z, a_x, a_y, a_z\right]^T \tag{4-15}$$

The model has two main versions. Discrete time state equation for the Wiener-sequence acceleration model, where the acceleration increment is assumed to be a white noise process, is shown below. Replacing I_2 in Equation (4-17) by I_3 one could get the 3D motion model.

$$x_{k+1} = F^{CA}.x_k + G^{CA}.w_k \tag{4-16}$$

$$F^{CA} = \begin{bmatrix} I_2 & T.I_2 & T^2/2.I_2 \\ 0 & I_2 & T.I_2 \\ 0 & 0 & I_2 \end{bmatrix} \qquad G^{CA} = \begin{bmatrix} T^2/2.I_2 \\ T.I_2 \\ I_2 \end{bmatrix} \tag{4-17}$$

Note that the acceleration increment is the integral of the acceleration derivative in the interval between adjacent steps. In another version of the model, referred as white-noise jerk model, is that the acceleration derivative (i.e. jerk) is an independent process. It is more convenient to use a random jerk process to model the maneuvers of agile targets.

As in CV model, one should keep in mind that actual maneuvers rarely have constant accelerations that are uncoupled across coordinate directions [40].

4.2.3 Singer Acceleration Model

The target maneuver, in this case the acceleration, can be assumed to be correlated in time. This is due to the assumption that if a target is accelerating at step k, it is more likely that it will continue accelerating at step k+1, where T is a small time increment between the steps [41, 6, 42]. Therefore, as in CA model the accelerations along each axis take place in the state ((4-14) & (4-15)). In the model target acceleration is a zero-mean stationary first-order Markov process with autocorrelation

$$R_a(\tau) = E\{a(t+\tau).a(t)\} = \sigma^2 e^{-\alpha|\tau|} \tag{4-18}$$

where $a(t)$ represents the accelerations along each axis, σ^2 is the variance of the target acceleration and $\alpha = 1/\tau_m$ is the reciprocal of the maneuver time constant τ_m. It can be expressed by the following linear time-invariant system:

$$\dot{a}(t) = -\alpha a(t) + w(t) \tag{4-19}$$

whose discrete-time equivalent is:

$$a_{k+1} = \beta.a_k + w_k \ , \quad \beta = e^{-\alpha T} \tag{4-20}$$

where w_k is a zero-mean white noise sequence with variance $\sigma^2(1-\beta^2)$. The continuous-time state equation for 2D is given in (4-21), where $w(t)$ stands for the jerk:

$$\dot{x}(t)=\begin{bmatrix}0 & I_2 & 0\\ 0 & 0 & I_2\\ 0 & 0 & -\alpha.I_2\end{bmatrix}x(t)+\begin{bmatrix}0\\ 0\\ I_2\end{bmatrix}w(t) \tag{4-21}$$

Discretizing the system matrix yields,

$$F^{\alpha}=\begin{bmatrix}I_2 & T.I_2 & \left(\alpha.T-1+\beta\right)/\alpha^2.I_2\\ 0 & 1 & \left(1-\beta\right)/\alpha.I_2\\ 0 & 0 & \beta.I_2\end{bmatrix} \tag{4-22}$$

Instead of jerk, acceleration increment can also be used as the process noise like in CA model which gives the following equation based on Equation (4-20). Replacing I_2 by I_3 one could get the 3D motion model.

$$x_{k+1}=F^{SA}.x_k+G^{SA}.w_k=\begin{bmatrix}I_2 & T.I_2 & T^2/2.I_2\\ 0 & I_2 & T.I_2\\ 0 & 0 & \beta.I_2\end{bmatrix}x_k+\begin{bmatrix}T^2/2.I_2\\ T.I_2\\ I_2\end{bmatrix}w_k \tag{4-23}$$

When maneuver time constant increases, Singer model reduces to the CA model. Note that as τ_m increases further (i.e. $\alpha.T$ increases) $\beta\to 1$, and the corresponding noise variance, $\sigma^2\left(1-\beta^2\right)$, gets smaller. On the other hand, as τ_m decreases (i.e. $\alpha.T$ increases) $\beta\to 0$, thus accelerations are only represented by the noise term. In that case, Singer acceleration model reduces to the CV model and the corresponding noise variance approaches to σ^2.

In [41, 40, 42], for an aircraft it is mentioned that $\tau_m\approx 60\,\text{s}$ for a slow turn, and 10-20s for an evasive maneuver. A distribution for σ^2, referred as ternary-uniform mixture, is also given in [41, 40, 42].

4.2.4 Coordinated Turn (CT) Model with Known Turn Rate

The CV and CA models both assume that the motion along each coordinate is uncoupled from that of the other coordinates. This leads to block diagonal matrices in the state equation (F^{CV}, F^{CA}, F^{SA} become block diagonal if the order of states is changed). While this is a convenient assumption, actual target maneuvers produce motion that is highly correlated across the coordinate directions [5]. The most common model to represent this correlation is the coordinated turn (i.e. *constant-speed circular turn*) model [43, 4, 44, 1]. In this model, target is assumed to move in a plane with constant speed (magnitude of the velocity vector) and turn with a constant angular velocity. While the maneuver does occur in a plane, it is not necessary for that plane to be horizontal (i.e., parallel to the x-y plane). Therefore, if the maneuver plane is vertical, the CT model can actually be used to represent climbs and descents as well [5]. Derivation of the CT equation in discrete-time is given below. Note that the position and velocity parameters are measured with respect to a fixed point, not with respect to observer. Therefore, the resultant relation should be used as either f^{tg} or f^{o} in Equations (4-1) and (4-2).

Let $\vec{v}_k$ be the velocity of the platform at step k, and $\vec{v}_{k+1}$ the velocity at step k+1. Corresponding vector decompositions over x & y axes are shown in Figure 4-3. Ω represents the angular turning rate at step k and T is the time difference between the simulation steps. Ω > 0 for counter-clockwise turn.

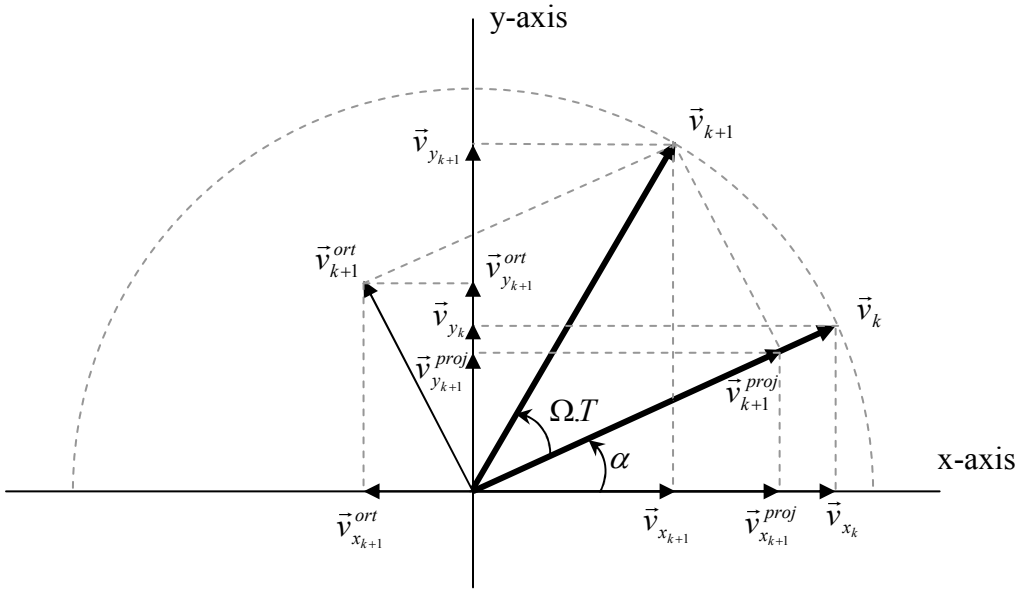

Figure 4-3: Vector diagram for CT.

$$\vec{v}_k = \vec{v}_{x_k} + \vec{v}_{y_k} = |v_k|.\cos\alpha.\hat{i} + |v_k|.\sin\alpha.\hat{j} \tag{4-24}$$

$$\vec{v}_{k+1} = \vec{v}_{x_{k+1}} + \vec{v}_{y_{k+1}} \tag{4-25}$$

where

$$\begin{aligned}
\vec{v}_{x_{k+1}} &= \vec{v}_{x_{k+1}}^{proj} + \vec{v}_{x_{k+1}}^{ort} \\
&= \left|\vec{v}_{k+1}^{proj}\right|.\cos\alpha.\hat{i} - \left|\vec{v}_{k+1}^{ort}\right|.\sin\alpha.\hat{i} \\
&= |\vec{v}_{k+1}|.\cos(\Omega.T).\cos\alpha.\hat{i} - |\vec{v}_{k+1}|.\sin(\Omega.T).\sin\alpha.\hat{i} \\
&= |\vec{v}_k|.\cos(\Omega.T).\cos\alpha.\hat{i} - |\vec{v}_k|.\sin(\Omega.T).\sin\alpha.\hat{i} \\
&= \cos(\Omega.T).\vec{v}_{x_k} - \sin(\Omega.T).\vec{v}_{y_k}
\end{aligned} \tag{4-26}$$

Similarly, for the y-axis component of $\vec{v}_{k+1}$

$$
\begin{aligned}
\vec{v}_{y_{k+1}} &= \vec{v}^{\,proj}_{y_{k+1}} + \vec{v}^{\,ort}_{y_{k+1}} \\
&= \left|\vec{v}^{\,proj}_{k+1}\right|.\sin\alpha.\hat{j} + \left|\vec{v}^{\,ort}_{k+1}\right|.\cos\alpha.\hat{j} \\
&= \left|\vec{v}_{k+1}\right|.\cos(\Omega.T).\sin\alpha.\hat{j} + \left|\vec{v}_{k+1}\right|.\sin(\Omega.T).\cos\alpha.\hat{j} \\
&= \left|\vec{v}_{k}\right|.\cos(\Omega.T).\sin\alpha.\hat{j} + \left|\vec{v}_{k}\right|.\sin(\Omega.T).\cos\alpha.\hat{j} \\
&= \sin(\Omega.T).\vec{v}_{x_k} + \cos(\Omega.T).\vec{v}_{y_k}
\end{aligned}
\tag{4-27}
$$

For position components of the state vector, let d_{x_k} be position of the target at step k, and $d_{x_{k+1}}$ position at step k+1. The $\dot{d}_{x_k}$ (or $\vec{v}_{x_k}$) is the known initial velocity.

$$
\begin{aligned}
d_{x_{k+1}} &= d_{x_k} + \dot{d}_{x_k}T + \int_0^T \ddot{d}_{x_k}.t.dt = d_{x_k} + \vec{v}_{x_k}T + \int_0^T \frac{\vec{v}_{x_{k+1}} - \vec{v}_{x_k}}{t}.t.dt \\
&= d_{x_k} + \int_0^T \vec{v}_{x_{k+1}}.dt = d_{x_k} + \int_0^T \left(\cos(\Omega.t).\vec{v}_{x_k} - \sin(\Omega.t).\vec{v}_{y_k}\right).dt \\
&= d_{x_k} + \frac{\sin(\Omega.t)}{\Omega}.\vec{v}_{x_k} + \frac{\left(\cos(\Omega.t)-1\right)}{\Omega}.\vec{v}_{y_k}
\end{aligned}
\tag{4-28}
$$

Similarly, for the y-axis component of the position,

$$
\begin{aligned}
d_{y_{k+1}} &= d_{y_k} + \dot{d}_{y_k}T + \int_0^T \ddot{d}_{y_k}.t.dt = d_{y_k} + \vec{v}_{y_k}T + \int_0^T \frac{\vec{v}_{y_{k+1}} - \vec{v}_{y_k}}{t}.t.dt \\
&= d_{y_k} + \int_0^T \vec{v}_{y_{k+1}}.dt = d_{y_k} + \int_0^T \left(\sin(\Omega.t).\vec{v}_{x_k} + \cos(\Omega.t).\vec{v}_{y_k}\right).dt \\
&= d_{y_k} + \frac{\left(1-\cos(\Omega.t)\right)}{\Omega}.\vec{v}_{x_k} + \frac{\sin(\Omega.t)}{\Omega}.\vec{v}_{y_k}
\end{aligned}
\tag{4-29}
$$

Converting the Equations (4-26), (4-27), (4-28) and (4-29) into matrix form, the coordinated turn transition matrix is

$$
F^{CT}(\Omega) = \begin{bmatrix}
1 & 0 & \frac{\sin(\Omega.t)}{\Omega} & -\frac{\left(1-\cos(\Omega.t)\right)}{\Omega} \\
0 & 1 & \frac{\left(1-\cos(\Omega.t)\right)}{\Omega} & \frac{\sin(\Omega.t)}{\Omega} \\
0 & 0 & \cos(\Omega.T) & -\sin(\Omega.T) \\
0 & 0 & \sin(\Omega.T) & \cos(\Omega.T)
\end{bmatrix}
\tag{4-30}
$$

The state equation is given in (4-31), where the zero-mean (Gaussian) white noise w_k represents the tangential accelerations along x & y axes.

$$x_{k+1} = F^{CT}(\Omega).x_k + G^{CV}.w_k \tag{4-31}$$

Turn rate is not shown in (4-30) for simplicity, but it can be obtained from the latest velocity estimate. Under the assumption that slower moving targets can perform sharper turns, turn rate (Ω_k) can be written as a nonlinear function of speed [4, 44, 1].

$$\Omega_k = \frac{\pm a_m}{\sqrt{\left(v_{x_k}\right)^2 + \left(v_{y_k}\right)^2}} \tag{4-32}$$

$$\begin{aligned} a_m > 0 \quad &\Rightarrow \quad counter-clockwise\ turn \\ a_m = 0 \quad &\Rightarrow \quad straight\ \ flight\ (i.e.\ same\ as\ the\ CV\ \ Model) \\ a_m < 0 \quad &\Rightarrow \quad clockwise\ turn \end{aligned}$$

In (4-32), $\pm a_m$ is the typical maneuver acceleration of the particular platform (target or observer). Different nonlinear models can be constructed for different values of a_m [44, 45]. Therefore, simple and linear set of state equations can be obtained, which is very useful for multi-model trackers like IMM.

4.2.5 CT model with Unknown Turn Rate

In the model above, the turn rate at each step (Ω_k) is either calculated by using the previous states or assumed constant from the beginning. However, it can be modeled as a Wiener process or first-order Markov process, and augmented in the state vector. The idea is to model the turn rate, and at each step use a suitable estimate to obtain an approximate linear model ($F^{CT}(\Omega)$) for target states. As that estimate Ω_k, Ω_{k+1} or their average can be used [40]. Corresponding state relation, where turn rate is a Wiener process, is given below.

$$x_{k+1} = \begin{bmatrix} F^{CT}(\Omega) & 0 \\ 0 & 1 \end{bmatrix} . x_k + \begin{bmatrix} G^{CV} & 0 \\ 0 & 1 \end{bmatrix} . w_k \tag{4-33}$$

4.2.6 CT Model with Polar Velocity and Unknown Turn Rate

Based on the idea of constant speed in the coordinated turn, velocity of the target can be expressed in polar coordinates. However, positions are still expressed in terms of Cartesian coordinates. New state vector in 2D is the following:

$$x = \left[d_x, d_y, V, \phi, \Omega \right]^T \tag{4-34}$$

The discretized system model is

$$x_{k+1} = f^{CTP}(x_k) + w_k^{CTP} = \begin{bmatrix} d_x + \frac{2}{\Omega} . V . \sin \frac{\Omega T}{2} . \cos\left(\phi + \frac{\Omega T}{2} \right) \\ d_y + \frac{2}{\Omega} . V . \sin \frac{\Omega T}{2} . \sin\left(\phi + \frac{\Omega T}{2} \right) \\ V \\ \phi + \Omega . T \\ \Omega \end{bmatrix}_k + w_k^{CTP} \tag{4-35}$$

where the corresponding process noise covariance is [40, 41]

$$Q^{CTP} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & T^2 . \sigma_V^2 & 0 & 0 \\ 0 & 0 & 0 & T^3 . \sigma_{\dot{\Omega}}^2 / 3 & T^2 . \sigma_{\dot{\Omega}}^2 / 2 \\ 0 & 0 & 0 & T^2 . \sigma_{\dot{\Omega}}^2 / 2 & T^2 . \sigma_{\dot{\Omega}}^2 \end{bmatrix} \tag{4-36}$$

4.2.7 Measurement Model

A measurement model in Cartesian coordinates is used along with the process models above. It corresponds to the tracking in mixed coordinates approach in [46].

The azimuth (i.e. bearing) angle (φ), elevation angle (θ), reciprocal of range ($\frac{1}{r}$) and reciprocal of range rate ($\frac{\dot{r}}{r}$) measurements are expressed in terms of state variables in Cartesian coordinates. Equations for 2D or equations with fewer components (like azimuth and elevation only) can be obtained by adapting (4-38).

$$y = \left[\varphi, \theta, \frac{1}{r}, \frac{\dot{r}}{r}\right]^T \tag{4-37}$$

$$y_{k+1} = \begin{bmatrix} \arctan\left(\frac{d_y}{d_x}\right) \\ \arctan\left(\frac{d_z}{\sqrt{d_x^2 + d_y^2}}\right) \\ \frac{1}{\sqrt{d_x^2 + d_y^2 + d_z^2}} \\ \frac{v_x.d_x + v_y.d_y + v_z.d_z}{d_x^2 + d_y^2 + d_z^2} \end{bmatrix} + v_k \tag{4-38}$$

Unlike the CV or CA models, measurements are not decoupled among each other. Moreover, the nonlinear relation is an unwanted situation considering the linear filter based tracking systems. Measurement noise is assumed to be uncorrelated, since each of them is taken by a separate system.

4.3 System Model in Modified Spherical Coordinates (MSC)

In the MSC, or modified polar coordinates, the relative position of the target is defined by using azimuth angle (φ), elevation angle (θ), and range (r) [6, 27]. Targets velocity is then expressed by the derivatives of them ((4-39), (4-40)). As general convention, ($\frac{1}{r}$) is used in the states instead of range itself.

$$\text{For 2D system } x = \left[\varphi, \frac{1}{r}, \dot{\varphi}, \frac{\dot{r}}{r}\right]^T \tag{4-39}$$

$$\text{For 3D system } x = \left[\varphi, \theta, \frac{1}{r}, \dot{\varphi}, \dot{\theta}, \frac{\dot{r}}{r}\right]^T \tag{4-40}$$

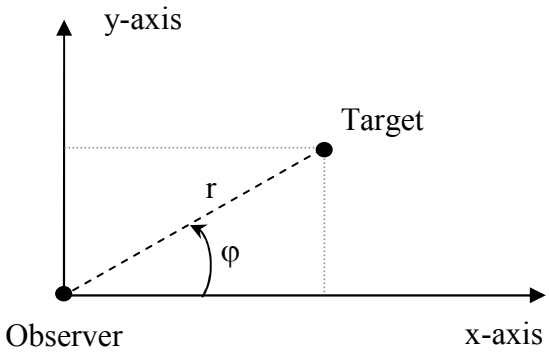

Figure 4-4: Two-dimensional representation in the modified spherical coordinates.

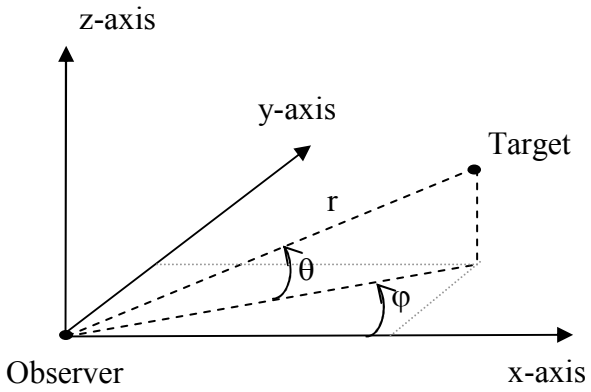

Figure 4-5: Three-dimensional representation in the modified spherical coordinates.

4.3.1 CV Model in MSC

One way to represent the state and measurement equations is to rewrite them in MSC. Doing that, the measurement relation becomes linear, since the measured quantities are the elements of state itself. Obviously, this method would introduce

high amount of nonlinearities to the state relation. However, in [47] it is claimed that the resultant performance would increase for particle filters. Relations between the state components expressed in MSC and Cartesian coordinate are given in Table 4-2.

Table 4-2: States in modified spherical coordinates. $x = g\left(x^{CC}\right)$ where x^{CC} is the state vector in Cartesian coordinates (Equations (4-6), (4-7)) [47].

(a) Two-dimensional	(b) Three-dimensional
	$x_1 = \varphi = \arctan\left(\frac{d_y}{d_x}\right)$
$x_1 = \varphi = \arctan\left(\frac{d_y}{d_x}\right)$	$x_2 = \theta = \arctan\left(\frac{d_z}{\sqrt{d_x^2 + d_y^2}}\right)$
$x_2 = \frac{1}{r} = \frac{1}{\sqrt{d_x^2 + d_y^2}}$	$x_3 = \frac{1}{r} = \frac{1}{\sqrt{d_x^2 + d_y^2 + d_z^2}}$
$x_3 = \dot{\varphi} = \frac{d_x.\dot{d}_y + \dot{d}_x.d_y}{d_x^2 + d_y^2}$	$x_4 = \dot{\varphi} = \frac{d_x.\dot{d}_y + \dot{d}_x.d_y}{d_x^2 + d_y^2}.\cos\left(\arctan\left(\frac{d_z}{\sqrt{d_x^2 + d_y^2}}\right)\right)$
$x_4 = \frac{\dot{r}}{r} = \frac{\dot{d}_x.d_x + \dot{d}_y.d_y}{d_x^2 + d_y^2}$	$x_5 = \dot{\theta} = \frac{\dot{d}_z.\left(d_x^2 + d_y^2\right) - d_z.\left(\dot{d}_x.d_x + \dot{d}_y.d_y\right)}{\left(d_x^2 + d_y^2 + d_z^2\right)\sqrt{d_x^2 + d_y^2}}$
	$x_6 = \frac{\dot{r}}{r} = \frac{\dot{d}_x.d_x + \dot{d}_y.d_y + \dot{d}_z.d_z}{d_x^2 + d_y^2 + d_z^2}$

We should also express the previously used acceleration input and noise in MSC. Here a transformation matrix (T_{CC}^{MSC}) is used by taking roll angle zero. Rows of u^{MSC} represents target accelerations along $\dot{\varphi}$, $\dot{\theta}$ and $\frac{\dot{r}}{r}$ respectively. For 2D modeling taking $\theta = 0$, $\dot{\theta} = 0$ and omitting the rows corresponding to them give:

$$u^{MSC} = T_{CC}^{MSC}.u = \begin{bmatrix} -\sin\varphi & \cos\varphi & 0 \\ -\sin\theta.\cos\varphi & \cos\theta.\sin\varphi & \cos\theta \\ \cos\theta.\cos\varphi & \cos\theta.\sin\varphi & \sin\theta \end{bmatrix}.u \tag{4-41}$$

Corresponding nonlinear state relations for continuous time is given below. Details can be found in [48, 47].

$$\dot{x} = f^{MSC}\left(x, u^{MSC}\right) + w^{MSC} = \begin{bmatrix} \frac{x_5}{\cos x_3} \\ x_6 \\ -x_1.x_4 \\ \left(-2.x_4 + x_6.\tan x_3\right).x_5 - x_1.u_1^{MSC} \\ -2.x_4.x_6 - x_5^2.\tan x_3 + x_1.u_2^{MSC} \\ x_6^2 + x_5^2 - x_4^2 - x_1.u_3^{MSC} \end{bmatrix} + w^{MSC} \tag{4-42}$$

Here w^{MSC} stands for the process noise in MSC, and assumed to be Gaussian. On the other hand, it can be incorporated in f^{MSC} if one can handle more nonlinearity. As in CV model, input in the state relation is replaced with noise (w) having a suitable covariance.

From that point on, the *linearized discretization* (i.e. first linearizing then discretizing) method in [48] is applied. After linearizing around previous state variable and discretizing, the resultant state relation is:

$$x_{k+1} = x_k + \int_0^T e^{\dot{f}^{MSC}\left(x, u^{MSC}\right).\tau} d\tau.f^{MSC}\left(x, u^{MSC}\right) + w^{MSC} \tag{4-43}$$

The initialization of covariance matrix is also discussed in [47]. First, the transformation in Table 4-2 ($g\left(x^{CC}\right)$) is approximated by first order Taylor expansion (4-44). Then the initial uncertainty of target states in MSC is expressed by (4-45). P_0 is the covariance matrix in Cartesian coordinates.

$$x = g\left(x^{CC}\right) = g\left(\hat{x}^{CC}\right) + \nabla_{x^{CC}} g\left(\hat{x}^{CC}\right).\left(x^{CC} - \hat{x}^{CC}\right) \tag{4-44}$$

$$P_o^{MSC} = \left(\nabla_{x^{CC}} g\left(\hat{x}^{CC}\right)\right).P_0.\left(\nabla_{x^{CC}} g\left(\hat{x}^{CC}\right)\right)^T \tag{4-45}$$

4.3.2 Measurement Model

The measurement model below is used along with the above process model in spherical (or polar) coordinates. It is much simpler than Equation (4-38), due to the coordinate system choice. Relations for 2D or with fewer components (like azimuth and elevation only) can be obtained by adapting (4-46). Measurement noise is assumed to be uncorrelated, since each of them is taken by a separate system:

$$y_{k+1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} . x_k + v_k \tag{4-46}$$

CHAPTER 5

SIMULATIONS AND DISCUSSION

In this chapter some of the algorithms and system models outlined in the previous chapters are applied to representative target trajectories and results are discussed.

5.1 Problem Formulation

Simulations are constructed such that the maneuvering target platform is tracked by a fixed observer (radar) on the ground. Artificial noise is added on the measurements to imitate the real sensor output. Target states (position, velocity and acceleration) are tracked via noisy measurements. While the state transition model is nonlinear for CT models, the measurement model is always nonlinear because of the polar to Cartesian conversion. Overall system model obeys the Equations (4-3) and (4-5) where u_k is taken as zero since the observer is not moving.

During the data generation process, target is assumed to move ideally with respect to the motion model. Measurements are the positions of the target in 3D (or sometimes 2D) polar coordinates, i.e. azimuth angle (in rad), elevation angle (in rad) and range (in meters). Noise on the range measurement is uniform unless it is stated otherwise. A sample trajectory is shown in Figure 5-1 together with the noisy measurements. The trajectories that used in the simulations are given in Figure 5-2 and Figure 5-3.

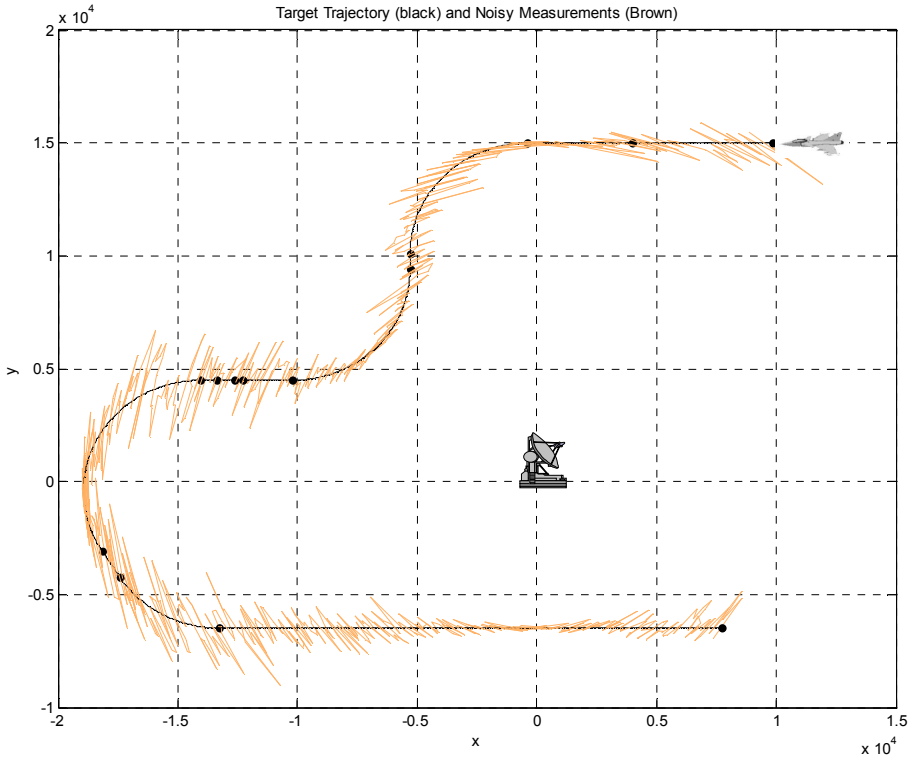

Figure 5-1: A sample target trajectory in 2D. Black dots indicate the points that the maneuver changes.

5.2 Implementation Details

The following "tracking filter – system model" combinations are implemented throughout this study.

- EKF with CV Model in 2D Cartesian coordinates
- EKF with CV Model in 3D Cartesian coordinates
- EKF with CA Model in 3D Cartesian coordinates

- IMM-EKF with 2 CV Models in 3D Cartesian coordinates
- IMM-EKF with CV & CA Models in 3D Cartesian coordinates
- IMM-EKF with CV, CA, CCW-CT (Counter-Clockwise Coordinated Turn) and CW-CT (Clockwise Coordinated Turn) Models in 3D Cartesian coordinates
- SIR-PF with CV model in 2D Cartesian coordinates
- SIR-PF with CV Model in 3D Cartesian coordinates
- SIR-PF with CA Model in 3D Cartesian coordinates
- MM-PF with CV, CA, CCW-CT and CW-CT Model in 3D Cartesian coordinates

Traditionally the tracking filters are initialized from first few measurements. However in this study the mean value of the initial state is taken as the true value to investigate the estimation performance independent of the initial transition period.

Each particle filter has two versions where the range noise is assumed to be Gaussian or uniform. For particle filter implementations prior distribution is taken as the importance density.

Measurement model is common for every filter. The range itself instead of its reciprocal (Section 4.2.7) is used as the measurement component for simplicity. During EKF's and IMM-EKF's the measurement relation is linearized at each step. Moreover, CCW-CT and CW-CT models in the IMM-EKF are also linearized. The coefficients that determine the lateral acceleration components (a_x, a_y) are calculated while linearizing the CT models even if they do not affect the position or velocity.

Both in MM-PF and IMM-EKF with four models the state is augmented with the turn rate variable (10^{th} component). There is minor difference about the meaning of turn rate in that two filter. In IMM-EKF the turn rate in the state denotes the maneuver that will be done in the following step. On the contrary, in MM-PF the turn rate state is only for information and shows maneuver of the previous step. The turn rate of the current time step is calculated explicitly according to Equation (4-32). Typical maneuver acceleration a_m is taken as 1 unless it is specified.

In the MM-PF there is a practical problem that cannot be solved by the regular systematic resampling (Algorithm 3) described in Section 3.3.3.1. In the resampling step of MM-PF the increased number of particles (particle count (N) times the number of models) are decreased back to "N". The procedure is given in Algorithm 9 where the new input parameter r ($0 < r \leq 1$) determines the reduction in the particle count.

Algorithm 9: Systematic Resampling for MM-PF

$$\left\{x_k^{j*}, w_k^{j*}, i^j\right\}_{j=1}^{N} = RESAMPLE\_MM\left(\left\{x_k^i, w_k^i\right\}_{i=1}^{N}, r\right)$$

- Do systematic resampling according to Algorithm 3:

  $$\left\{\tilde{x}_k^j, \tilde{w}_k^j, \tilde{i}^j\right\}_{j=1}^{N} = RESAMPLE\left(\left\{x_k^i, w_k^i\right\}_{i=1}^{N}\right)$$

- Construct $\left\{\bar{x}_k^j, \bar{w}_k^j, \bar{i}^j\right\}_{j=1}^{N_d}$ such that
  - N_d is the total number of different samples
  - $\left\{\bar{x}_k^j\right\}_{j=1}^{N_d}$ contains different samples (each selected sample $\tilde{x}_k^j$ appear only once)
  - $\left\{\bar{w}_k^j\right\}_{j=1}^{N_d}$ contains the sample counts for $\left\{\bar{x}_k^j\right\}_{j=1}^{N_d}$
  - $\bar{i}^j$ is the same as $\tilde{i}^j$ for samples in $\left\{\bar{x}_k^j\right\}_{j=1}^{N_d}$
- Desired sample size: $N_{new} = \lfloor N.r \rfloor$
- IF $N_{new} \geq N_d$
  - Augment dummy samples with zero weights & normalize: $\left\{\bar{x}_k^j, \bar{w}_k^j\right\}_{j=1}^{N_d} \rightarrow \left\{\bar{x}_k^i, \frac{\bar{w}_k^i}{N}\right\}_{i=1}^{N_{new}}$
  - $\left\{x_k^{j*}, w_k^{j*}, \hat{i}^j\right\}_{j=1}^{N_{new}} = RESAMPLE\_MM\left(\left\{\bar{x}_k^i, \bar{w}_k^i\right\}_{i=1}^{N_{new}}, 1\right)$
  - $\left\{i^j\right\}_{j=1}^{N_{new}} = \bar{i}^j \ \ for\ each\ j = \hat{i}^j$
- ELSE
  - Determine the new ratio: $r_{new} = \frac{N_{new}}{N_d}$
  - $\left\{x_k^{j*}, w_k^{j*}, \hat{i}^j\right\}_{j=1}^{N_{new}} = RESAMPLE\_MM\left(\left\{\bar{x}_k^i, \bar{w}_k^i\right\}_{i=1}^{N_d}, r_{new}\right)$
  - $\left\{i^j\right\}_{j=1}^{N_{new}} = \bar{i}^j \ \ for\ each\ j = \hat{i}^j$
- END IF

The two main trajectories used throughout the simulations are presented below. Artificial measurements are created from these trajectories by sampling according to the selected time step. Filters operate at the same time step. All turns are created by using the relations in Section 4.2.4.

Trajectory 1:

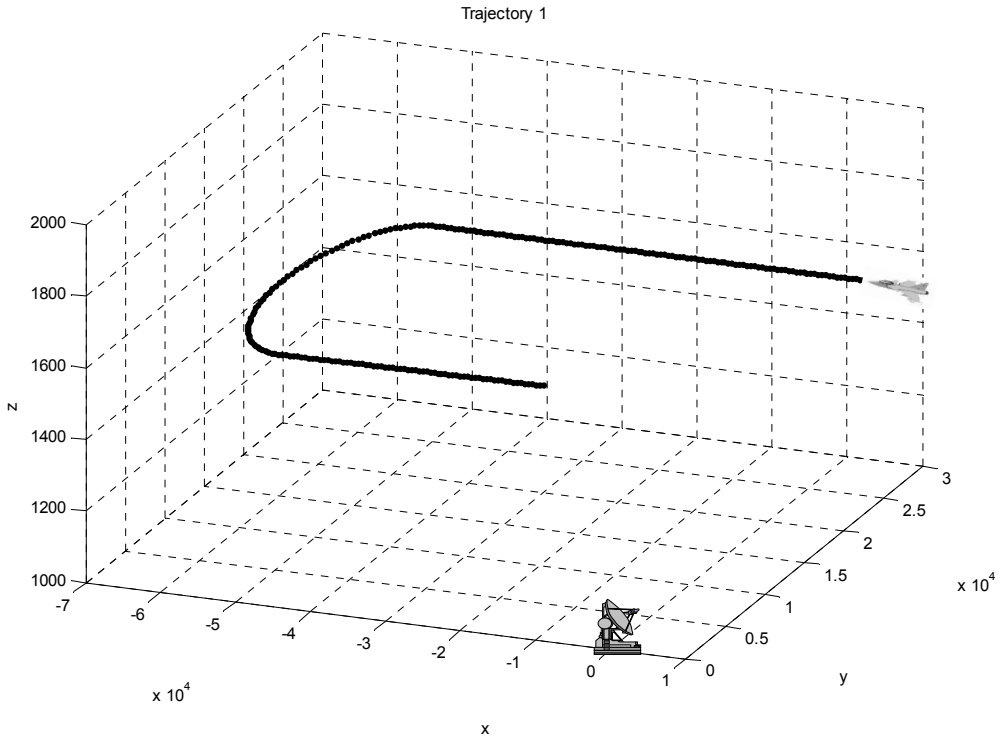

Figure 5-2: Trajectory 1 in 3D.

Table 5-1: Trajectory 1 maneuver details.

Target initial state	[5000 30000 -100 0]'
Maneuvers	Straight flight for 600s
	Coordinated Turn for 315s (~180°)
	Straight flight for 350s

Trajectory 2:

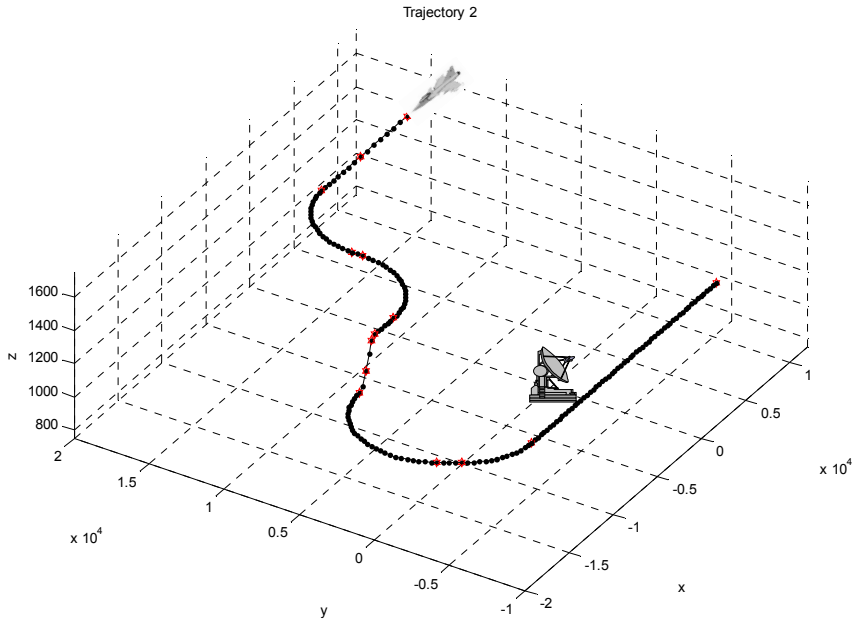

Figure 5-3: Trajectory 2 in 3D.

Table 5-2: Trajectory 2 maneuver details.

Target initial state	[10000 15000 1500 -150 0 0 0 0 0]'
Maneuvers	Straight flight for 40s
	Slowing down with 2g for 40s
	Coordinated Turn for 110s (~90°)
	Straight flight for 10s
	Coordinated Turn for 110s (~90°)
	Straight flight for 30s
	Acceleration on z-axis with -2g for 5s
	Acceleration on z-axis with -g for 10s
	Acceleration on z-axis with +3g for 10s
	Coordinated Turn for 150s (~120°)
	Straight flight for 20s
	Coordinated Turn for 70s (~60°)
	Straight flight for 300s

Trajectory 3:

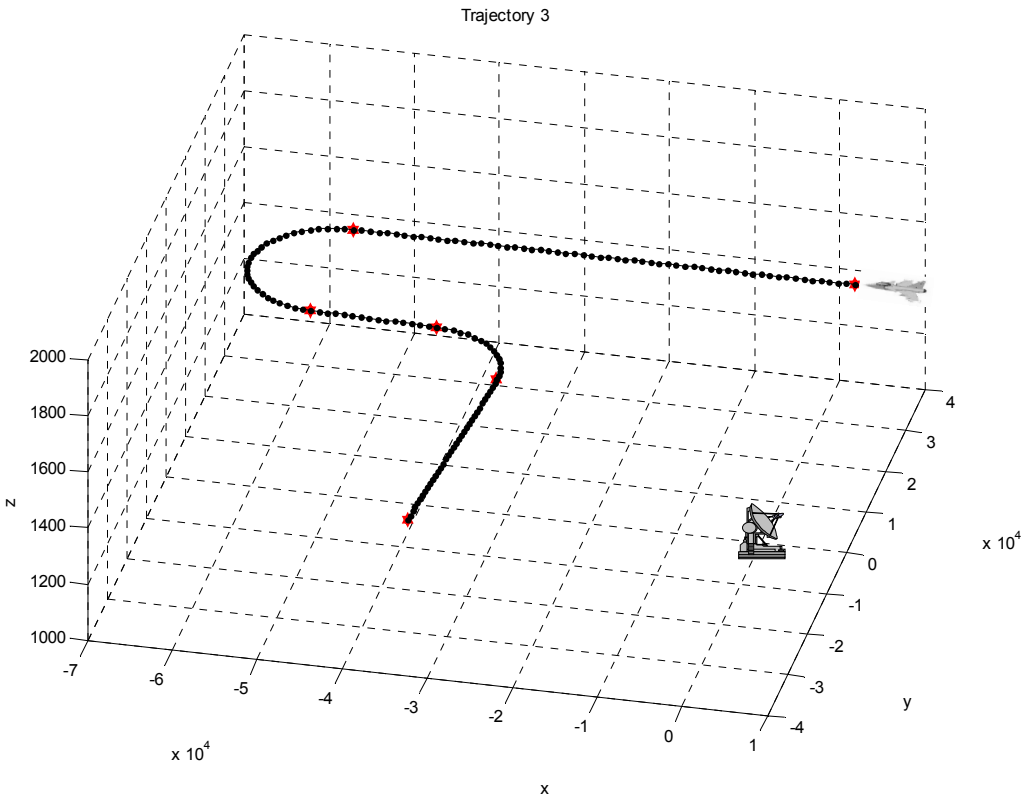

Figure 5-4: Trajectory 3 in 3D.

Table 5-3: Trajectory 3 maneuver details.

Target initial state	[5100 30000 1500 -100 0 0 0 0 0]'
Maneuvers	Straight flight for 600s
	Coordinated Turn for 310s (~180°)
	Straight flight for 150s
	Coordinated Turn for 160s (~90°)
	Straight flight for 350s

5.3 Comparison Criteria

The main parameter used for comparison in this study is the Root-Mean-Square-Error (RMSE) of the position states. It can be in 2D (positions on the x & y axes) or in 3D. RMSE is calculated at each time step (Equation (5-1)) to be able to observe the maneuver dependent characteristics of the algorithms. If there are Monte Carlo runs (N_{mc}) for a specific scenario RMSE is averaged.

$$RMSE_k = \sqrt{\frac{1}{N_{mc}} \sum_{i=1}^{N_{mc}} \left(\left(d_{x_k}^{(i)} - d_{x_k}^{true} \right)^2 + \left(d_{y_k}^{(i)} - d_{y_k}^{true} \right)^2 \right)} \tag{5-1}$$

As an additional parameter, the total RMSE is also calculated for each simulation. Here L is the number of time steps.

$$RMSE = \sqrt{\frac{1}{L} \sum_{k=1}^{L} \frac{1}{N_{mc}} \sum_{i=1}^{N_{mc}} \left(\left(d_{x_k}^{(i)} - d_{x_k}^{true} \right)^2 + \left(d_{y_k}^{(i)} - d_{y_k}^{true} \right)^2 \right)} \tag{5-2}$$

The time elapsed during each algorithm is also calculated. The execution priority of the simulation environment (MATLAB) is adjusted to “Realtime” to prevent other applications blocking the CPU. However the computation times can only be compared within a simulation since computers with different configurations are used.

5.4 Results

5.4.1 Simulation Set 1

5.4.1.1 Part A

EKF and SIR-PF are compared for the first scenario defined in 2D Cartesian coordinates. The range noise is assumed as Gaussian to eliminate its effect. The

filters have the same process and measurement noise models (covariances). Measurement noise model is also the same as the one used while creating the artificial measurement. Some important parameters are given in the following table.

Table 5-4: Simulation parameters.

Noise in the measurements	Azimuth (rad): Gaussian, σ^2=0.00052 Range (m): Gaussian, σ^2= 100
Measurement noise model (covariance) in the filters	For all filters same as the actual covariances
Process noise model (covariance) in the filters	[4 0 ; 0 4]
Time Step (s)	5
Particle Count for SIR-PF	10000

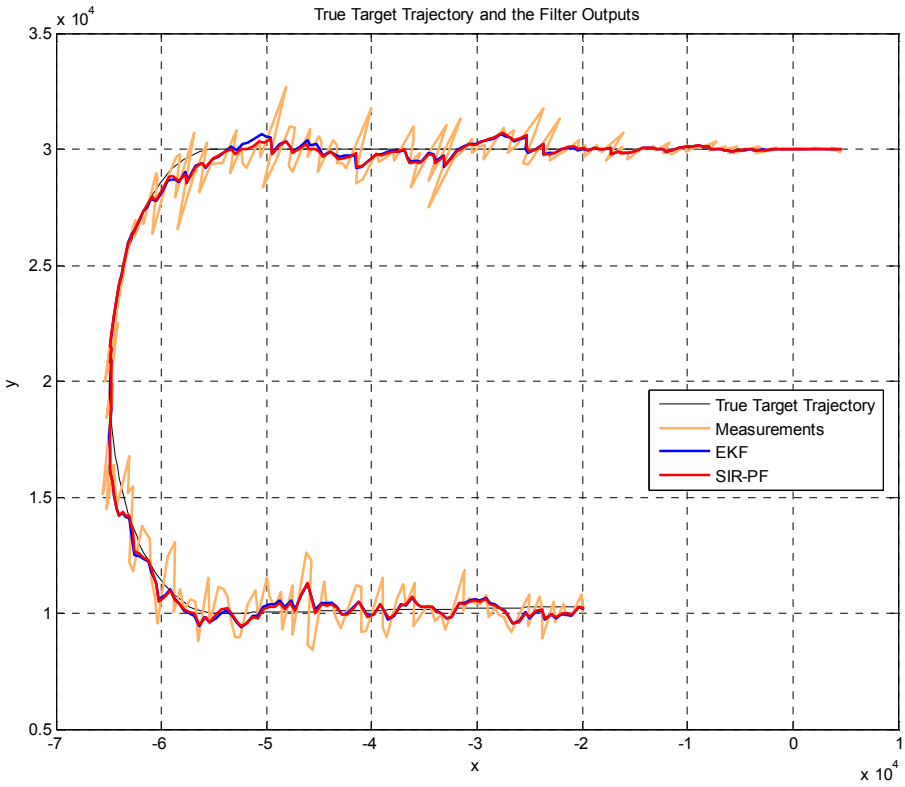

Figure 5-5: True target trajectory and the filter outputs for 10000 particles.

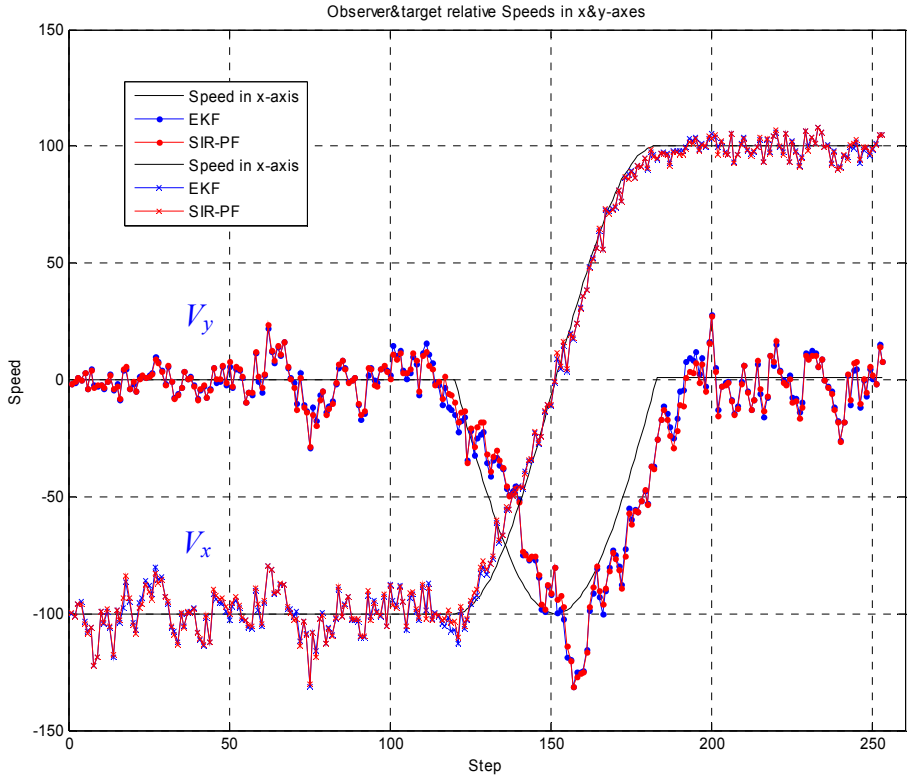

Figure 5-6: True target speeds and the filter outputs for 10000 particles.

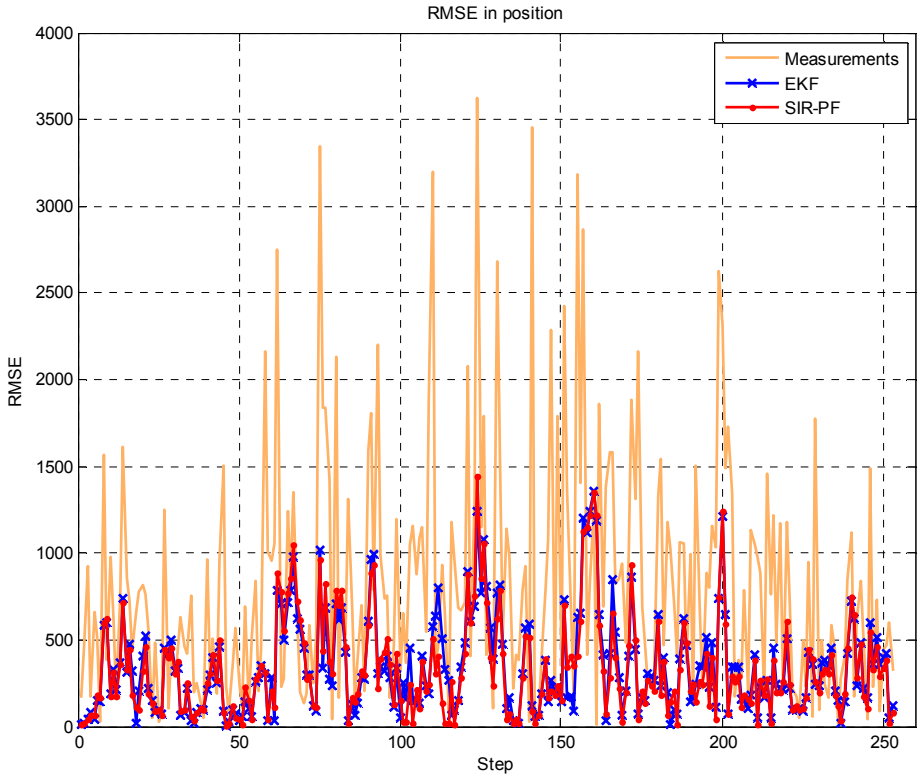

Figure 5-7: RMSE of measurements, EKF and SIR-PF for 10000 particles.

Table 5-5: Simulation result.

	Measurement	**EKF**	**SIR-PF**
Total RMSE in Position	1.0756×10^3	442.9838	434.9442
Computation Time (s)	-	0.069	1137.131

Remarks:

- For this trajectory SIR-PF and EKF results almost same for 10000 particles.
- Since both filters include CV models and same noise distributions are used, the only improvement that SIR-PF can introduce would be a correction for

the nonlinear measurement equation by presenting a better representation of likelihood distribution.

- In this case this difference cannot be observed due to the low measurement noise covariances. The likelihood is peaked, so both filters more or less follow the measurement. Higher number of particles is needed to realize the improvement done SIR-PF by representing the shape of the likelihood.

- Time spent for the SIR-PF is significantly higher.

5.4.1.2 Part B

Since the PF has no significant advantage over EKF in Part A, to observe the difference the covariance of measurement noise is increased.

Table 5-6: Simulation parameters.

Noise in the measurements	Azimuth (rad): Gaussian, σ^2=0.0052 Range (m): Gaussian, σ^2= 100
Measurement noise model (covariance) in the filters	For all filters same as the actual covariances
Process noise model (covariance) in the filters	[4 0 ; 0 4]
Time Step (s)	5
Particle Count for SIR-PF	5000

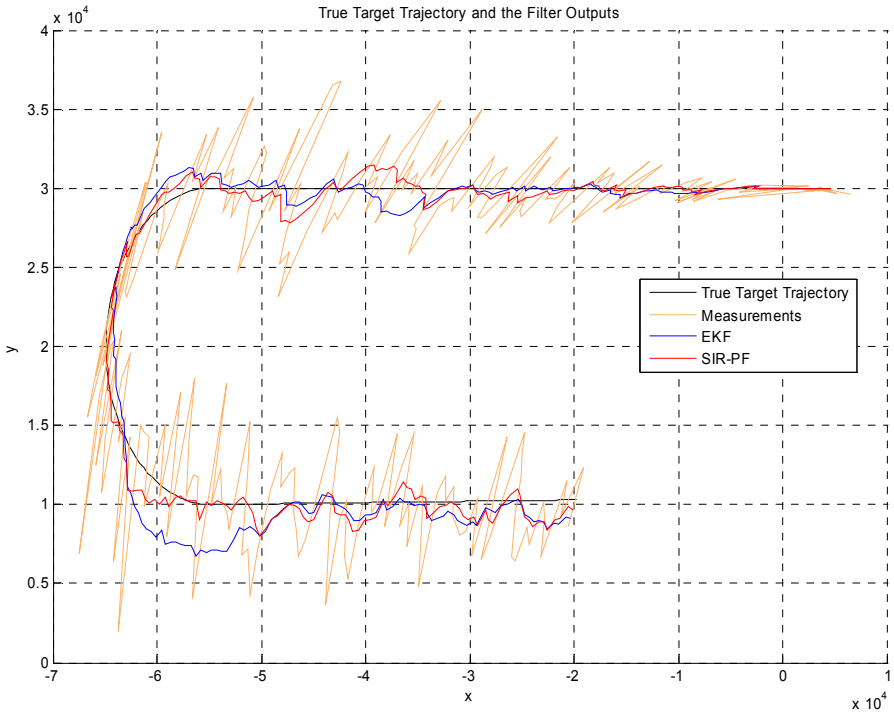

Figure 5-8: True target trajectory and the filter outputs for 5000 particles.

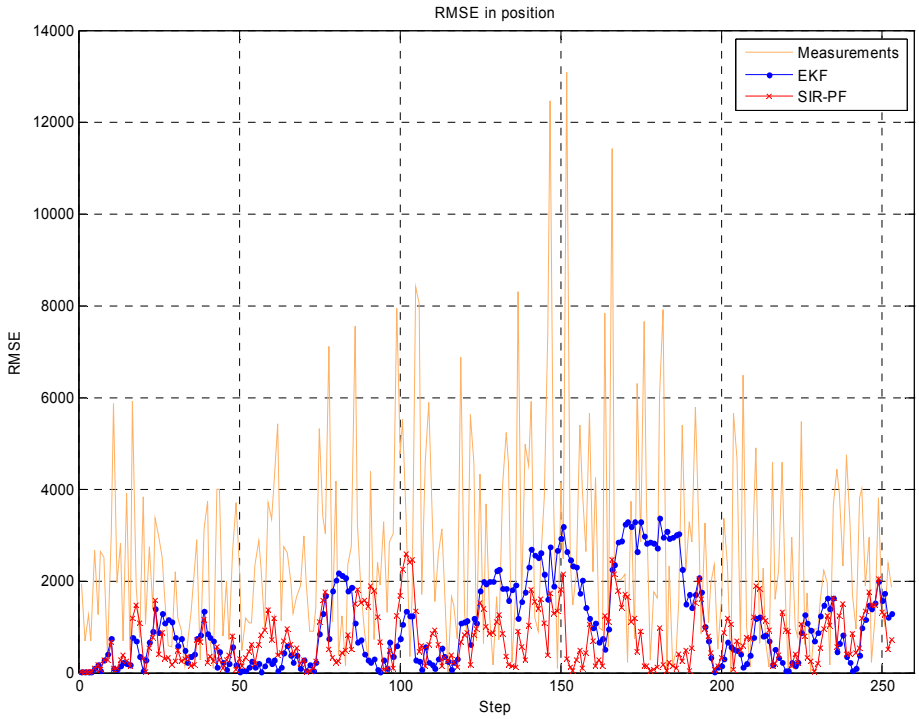

Figure 5-9: RMSE of measurements, EKF and SIR-PF for 5000 particles.

Table 5-7: Simulation result.

	Measurement	**EKF**	**SIR-PF**
Total RMSE in Position	3.4396×10^3	1.4050×10^3	943.4269
Computation Time (s)	-	0.067	406.02

Remarks:

- Compared to Part A, the improvement can be observed even for a less number of particles (5000) when the measurements become more inaccurate. The shape of the likelihood is now broader, so it is represented much better (compared to Part A) by SIR-PF, yielding a better performance.

5.4.1.3 Part C

Target maneuvers are represented by the noise terms in CV model. In this part, process noise assumptions in the filters are reduced to simulate a worse case in terms of model mismatch.

Table 5-8: Simulation parameters.

Noise in the measurements	Azimuth (rad): Gaussian, σ^2=0.0052 Range (m): Gaussian, σ^2= 100
Measurement noise model (covariance) in the filters	For all filters same as the actual covariances
Process noise model (covariance) in the filters	[1 0 ; 0 1]
Time Step (s)	5
Particle Count for SIR-PF	5000

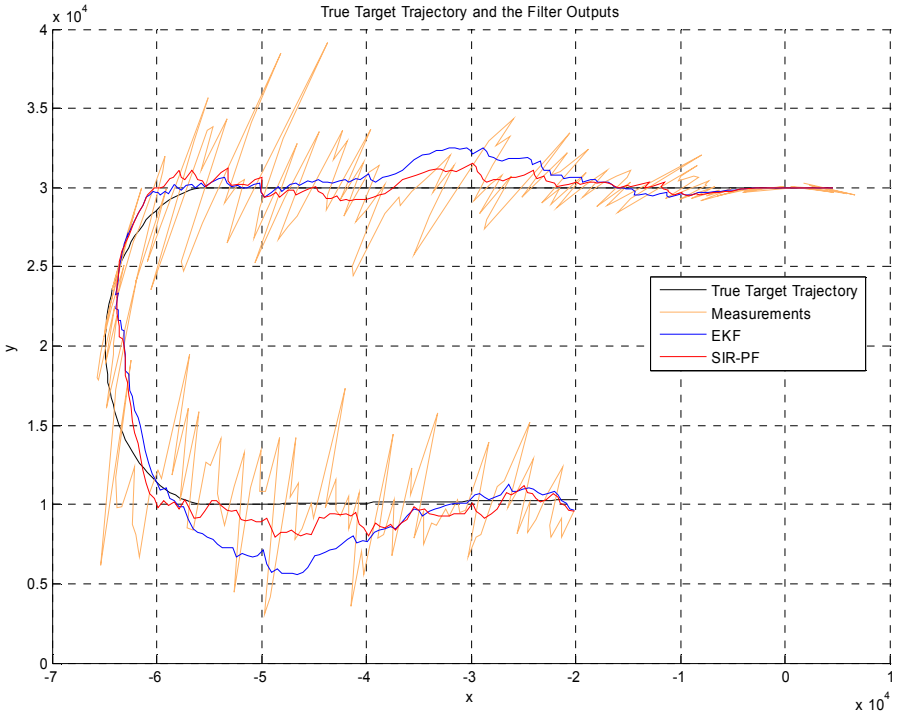

Figure 5-10: True target trajectory and the filter outputs for 5000 particles.

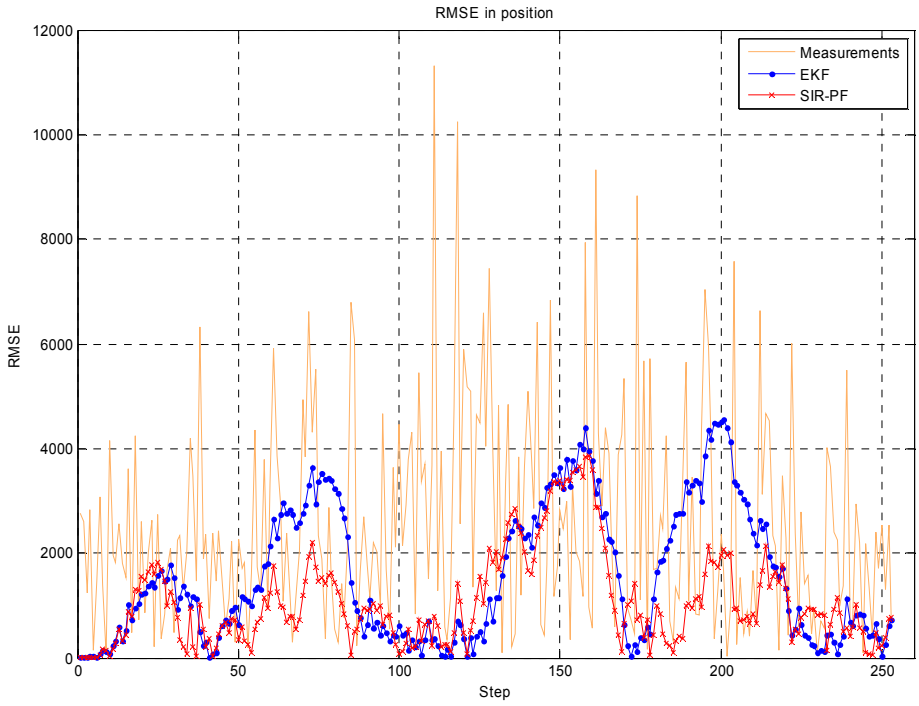

Figure 5-11: RMSE of measurements, EKF and SIR-PF for 5000 particles.

Table 5-9: Simulation result.

	Measurement	**EKF**	**SIR-PF**
Total RMSE in Position	3.2586×10^3	2.0196×10^3	1.4063×10^3
Computation Time (s)	-	0.067	403.42

Remarks:

- The overall RMSE for both filters is increased since the process noise model is much tighter than it should be to overcome the maneuver. Further reduction of the process noise may result in divergence.
- SIR-PF provides approximately the same performance improvement.

5.4.1.4 Part D

The whole scenario is divided into a less number of points, i.e. fewer measurements are taken.

Table 5-10: Simulation parameters.

Noise in the measurements	Azimuth (rad): Gaussian, σ^2=0.0052 Range (m): Gaussian, σ^2= 100
Measurement noise model (covariance) in the filters	For all filters same as the actual covariances
Process noise model (covariance) in the filters	[1 0 ; 0 1]
Time Step (s)	10
Particle Count for SIR-PF	5000

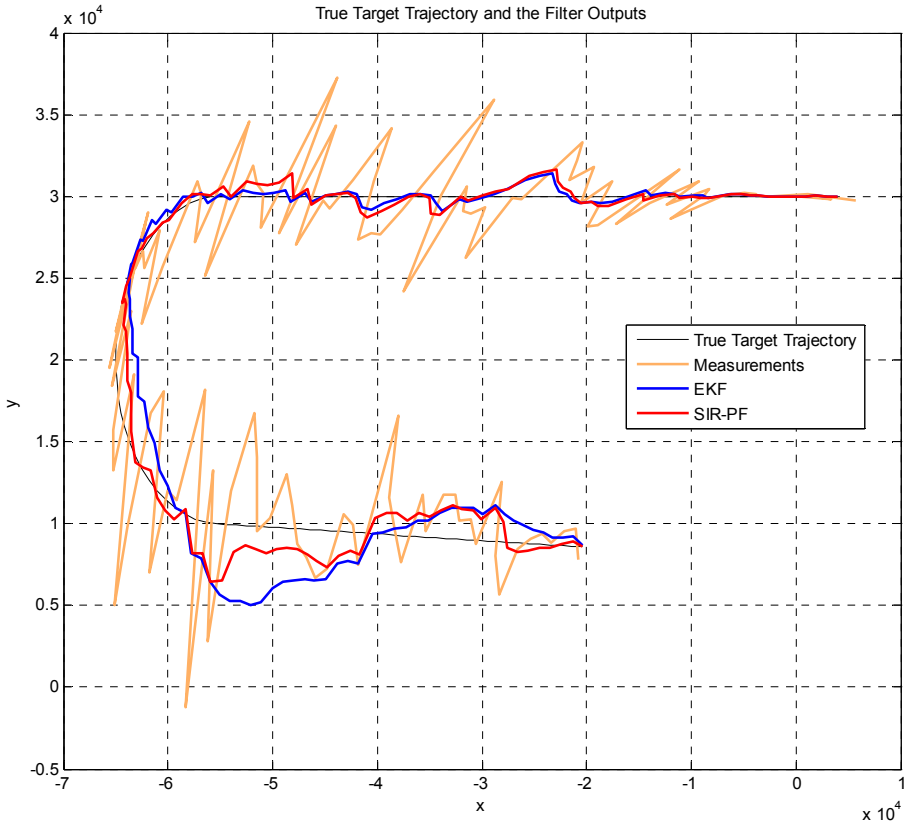

Figure 5-12: True target trajectory and the filter outputs for 5000 particles.

Figure 5-13: RMSE of measurements, EKF and SIR-PF for 5000 particles.

Table 5-11: Simulation result.

	Measurement	**EKF**	**SIR-PF**
Total RMSE in Position	3.5162×10^3	1.8641×10^3	1.2446×10^3
Computation Time (s)	-	0.041	203.44

Remarks:

- Particle filter again results in better performance.

5.4.1.5 Part E

The noise distribution of the range measurement is taken as Gaussian in the above simulations. However due to the characteristic properties of radars it is uniform. So, a second SIR-PF for uniform range noise is implemented, and the artificial measurement data is generated. Note that, the exaggeration in the noise terms aims to show the performance difference.

Table 5-12: Simulation parameters.

Noise in the measurements	Azimuth (rad): Gaussian, σ^2=0.000000052 Range (m): Uniform on [-10000, 10000]
Measurement noise model (covariance) in the filters	Azimuth: For all filters same as the actual covariance Range for EKF: Gaussian, σ^2=25.10^6 Range for 1st SIR-PF: Gaussian, σ^2=25.10^6 Range for 2nd SIR-PF: Uniform on [-10000, 10000]
Process noise model (covariance) in the filters	[4 0 ; 0 4]
Time Step (s)	5
Particle Count for SIR-PF's	2500

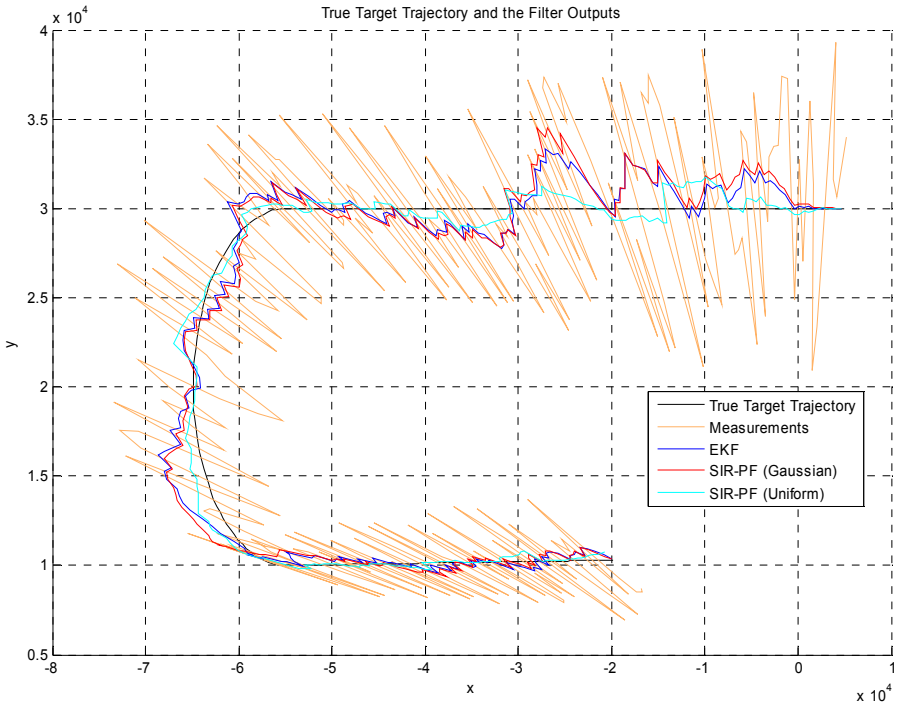

Figure 5-14: True target trajectory and the filter outputs for 2500 particles.

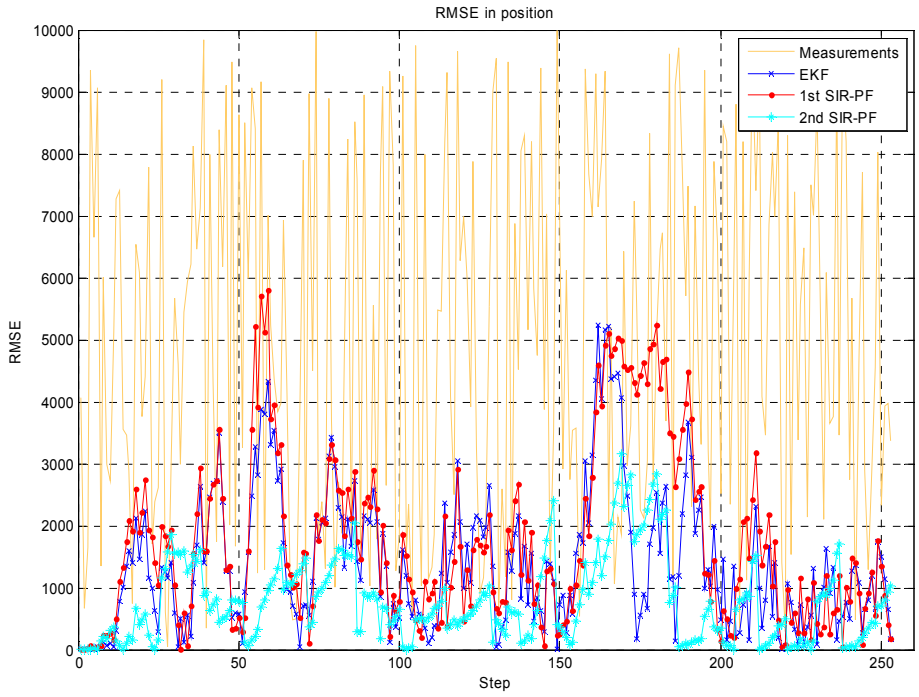

Figure 5-15: RMSE of measurements, EKF and SIR-PF's for 2500 particles.

Table 5-13: Simulation result.

	Meas.	**EKF**	**SIR-PF (Gaussian)**	**SIR-PF (Uniform)**
Total RMSE in Position	5.8183×10^3	1.7831×10^3	2.2478×10^3	1.0282×10^3
Computation Time (s)	-	0.068	162.985	147.003

Remarks:

- In theory the new particle filter should give better results since it suits the actual noise better. But there is no noticeable difference between two particle filters when the range noise comparable with the ones in Part A to Part D. Reason is that the range component being the finest measurement.

The marginal likelihood function is so peaked that modeling it as Gaussian or uniform does not matter.

- In order to demonstrate the modeling properties of the particle filter parameters are changed unrealistically such that the range noise becomes dominant. Corresponding covariances in EKF & 1st SIR-PF are increased, too.

- SIR-PF with uniform range noise model runs faster since the evaluation of the likelihood is much easier than the Gaussian case.

5.4.2 Simulation Set 2

The 3 dimensional versions of the tracking filters in Simulation Set 1 are implemented and the effects of using two CV models together in an IMM structure are investigated.

5.4.2.1 Part A

EKF, IMM and SIR-PF are compared for a scenario defined in 3D Cartesian coordinates. IMM consist of two CV models with different noise assumptions. Although there is no elevation change in the scenario, tracking is done based on 3D coordinates. Elevation noise is assumed to be much lower for not obtaining negative angles. RMSE is calculated based on the positions on x and y axes.

Table 5-14: Simulation parameters.

Noise in the measurements	Azimuth (rad): Gaussian, σ^2=0.0052 Elevation (rad): Gaussian, σ^2=0.00087 Range (m): Uniform on [-20, 20]
Measurement noise model (covariance) in the filters	Azimuth and Elevation: For all filters same as the actual covariance Range for EKF: Gaussian, σ^2=400 Range for 1st SIR-PF: Gaussian, σ^2=400 Range for 2nd SIR-PF: Uniform on [-20, 20]
Process noise model (covariance) in the filters	EKF: CV Model → [1 0 ; 0 1] IMM: CV Model 1 → [1 0 ; 0 1] CV Model 2 → [4 0 ; 0 4] SIR-PF: [4 0 ; 0 4]
IMM Transition Prob.	[0.9 0.1; 0.1 0.9]
Time Step (s)	5
Particle Count for SIR-PF's	10000

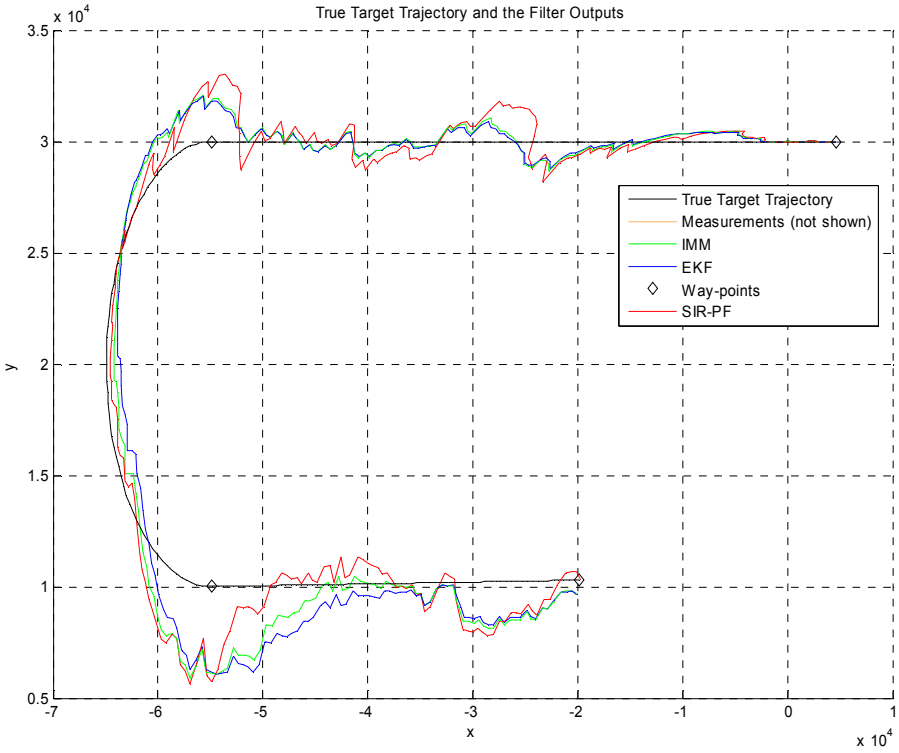

Figure 5-16: True target trajectory and the filter outputs for 10000 particles.

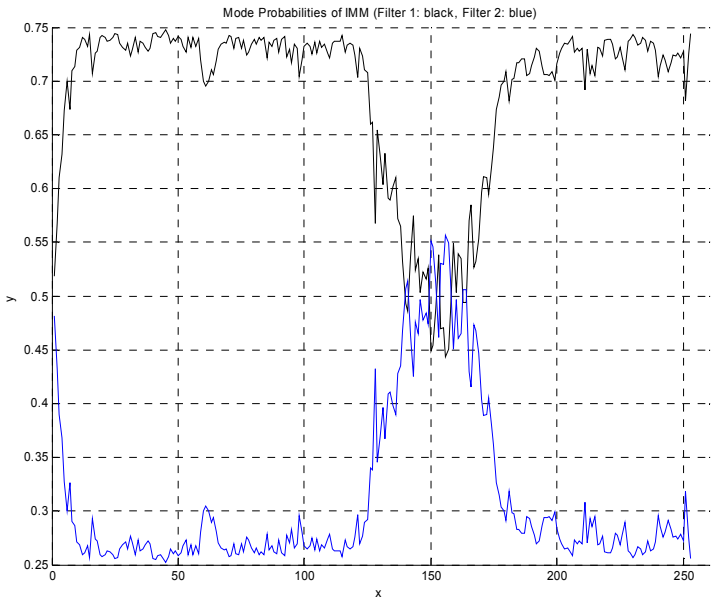

Figure 5-17: Mode probabilities of IMM-EKF.

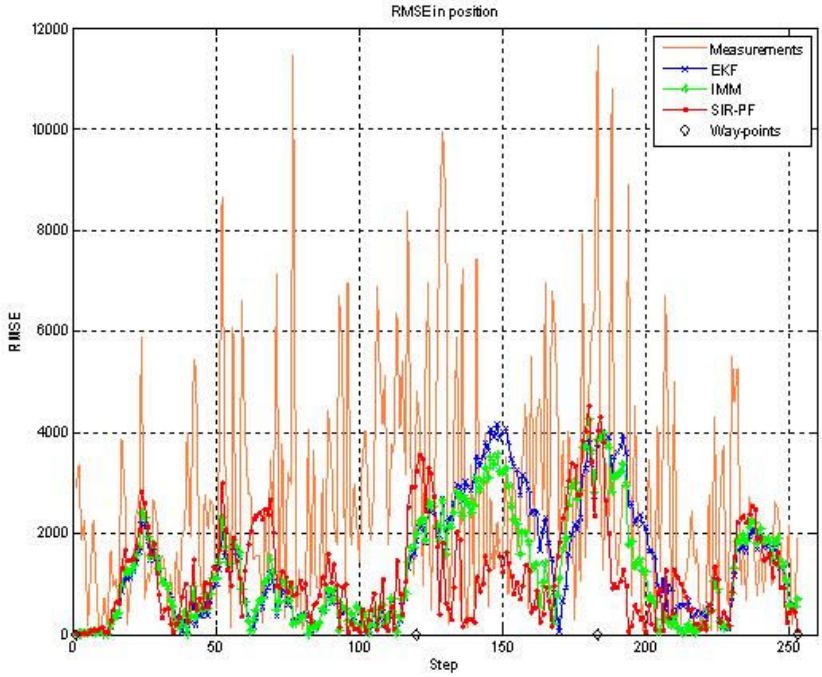

Figure 5-18: RMSE of measurements, EKF, IMM-EKF and SIR-PF for 10000 particles.

Table 5-15: Simulation result.

	Meas.	**EKF**	**IMM-EKF**	**SIR-PF (Uniform)**
Total RMSE in Position	3.5564×10^3	1.8147×10^3	1.6779×10^3	1.5286×10^3
Computation Time (s)	-	0.069	0.236	1404.420

Remarks:

- Using the same CV model with two different process noise covariances in IMM may increase the performance compared to the EKF. Main idea is to adjust one model to straight flight (low process noise) and the other one to maneuvers (high process noise).

- Designing two different filters and combining them under the IMM framework makes the whole algorithm capable to work under different conditions. However for a certain maneuver (for example straight flight), the adjusted EKF alone performs better than IMM even if the IMM includes that optimized filter. Reason is that the other filters designed for different situations interfere and change the IMM output. In the simulation above EKF has an arbitrary process noise covariance ([1 0 ; 0 1]). On the other hand IMM includes that EKF and another EKF having a optimized process noise covariance ([4 0 ; 0 4]).

- For increased number of particles, again SIR-PF works well.

5.4.2.2 Part B

The filters in Part A are applied to a different 3D scenario. Particle count is 5000 for these simulations.

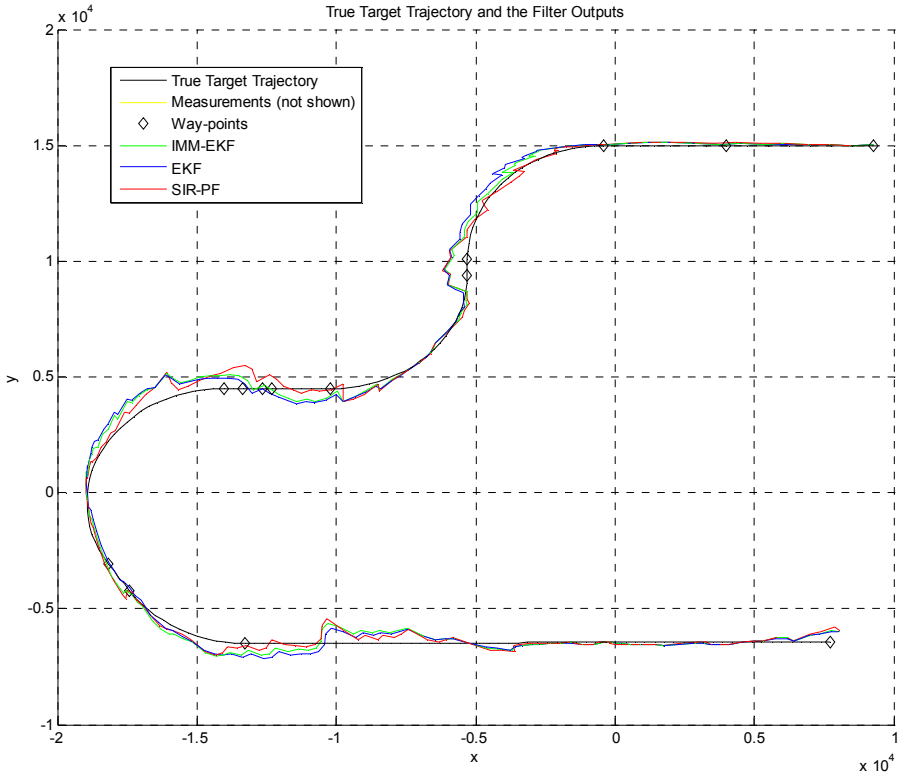

Figure 5-19: True target trajectory and the filter outputs for 5000 particles.

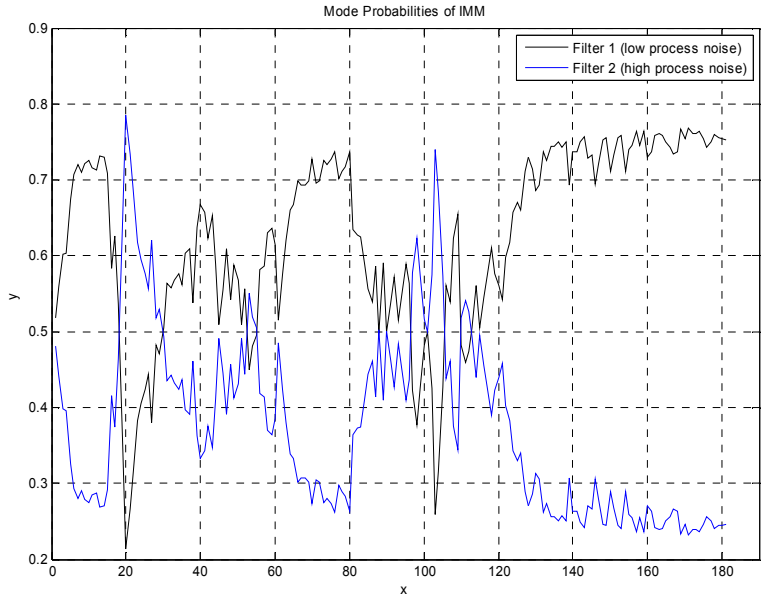

Figure 5-20: Mode probabilities of IMM-EKF.

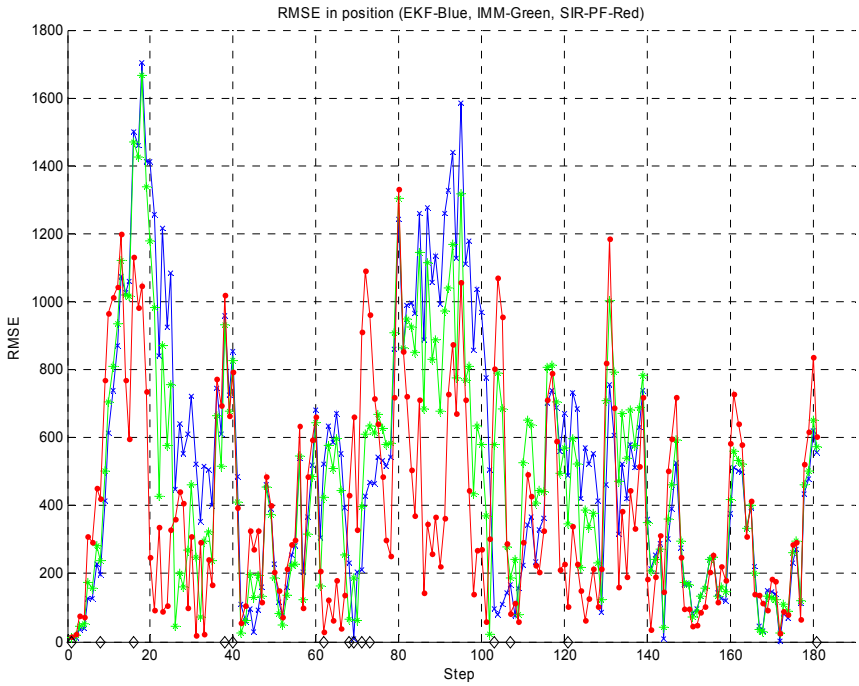

Figure 5-21: RMSE of measurements, EKF, IMM-EKF and SIR-PF for 5000 particles.

Table 5-16: Simulation result.

	Meas.	EKF	IMM-EKF	SIR-PF (Uniform)
Total RMSE in Position	1.0701×10^3	646.6873	590.9882	504.6776
Computation Time (s)	-	0.052	0.177	327.697

5.4.3 Simulation Set 3

5.4.3.1 Part A

Using the 3D scenario in Simulation Set 1, EKF with CA model and IMM-EKF with CV and CA models are simulated. I_3 is the three dimensional identity matrix.

Table 5-17: Simulation parameters.

Noise in the measurements	Azimuth (rad): Gaussian, σ^2=0.00052 Elevation (rad): Gaussian, σ^2=0.00087 Range (m): Gaussian, σ^2=400
Measurement noise model (covariance) in the filters	For all filters same as the actual covariance
Process noise model (covariance) in the filters	EKF: CA Model → 0.1 * I_3 IMM: CV Model → 0.01 * I_3 CA Model →0.1 * I_3 SIR-PF: [4 0 ; 0 4]
IMM Transition Prob.	[0.9 0.1; 0.1 0.9]
Time Step (s)	10
Particle Count for SIR-PF	5000

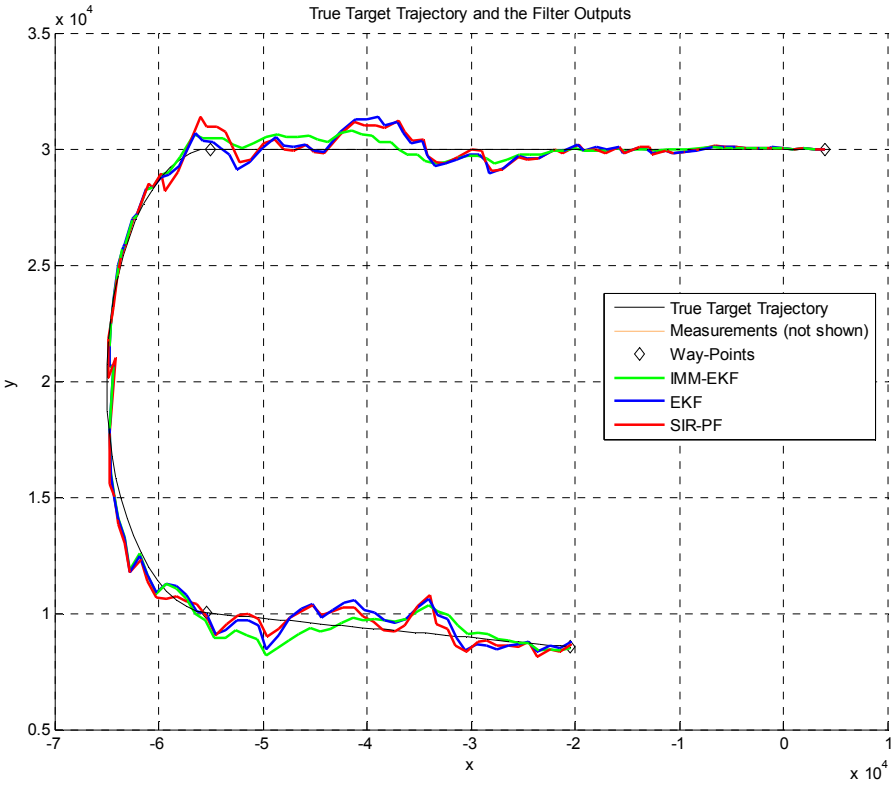

Figure 5-22: True target trajectory and the filter outputs shown in 2D for 5000 particles.

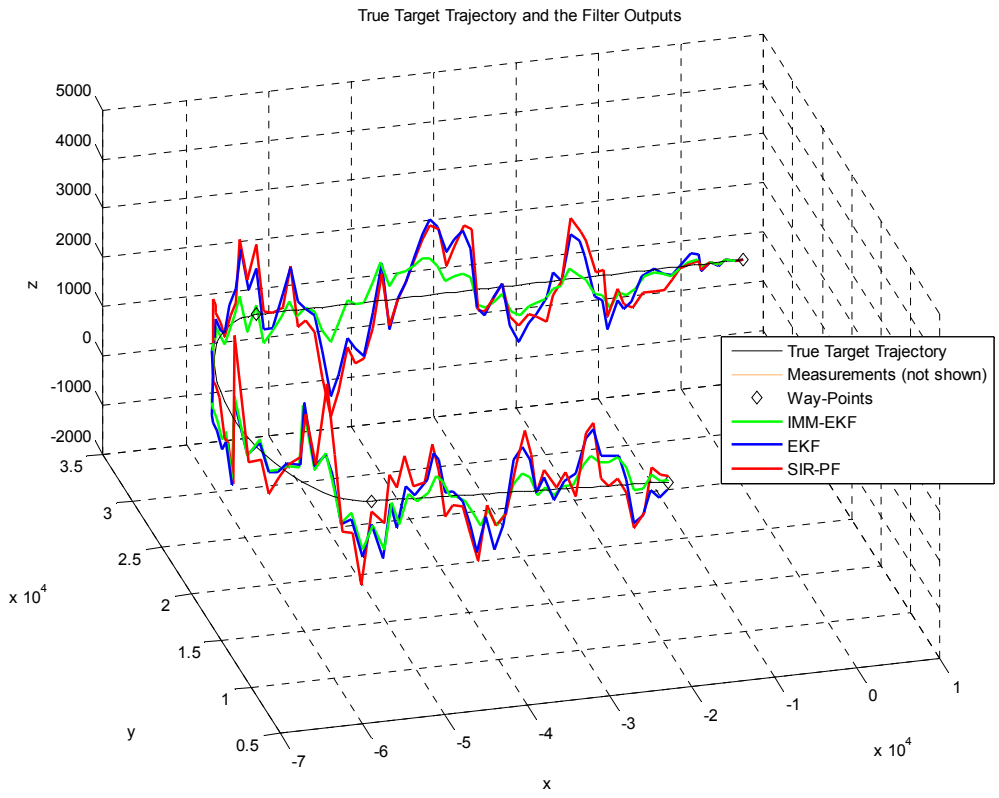

Figure 5-23: True target trajectory and the filter outputs shown in 3D for 5000 particles.

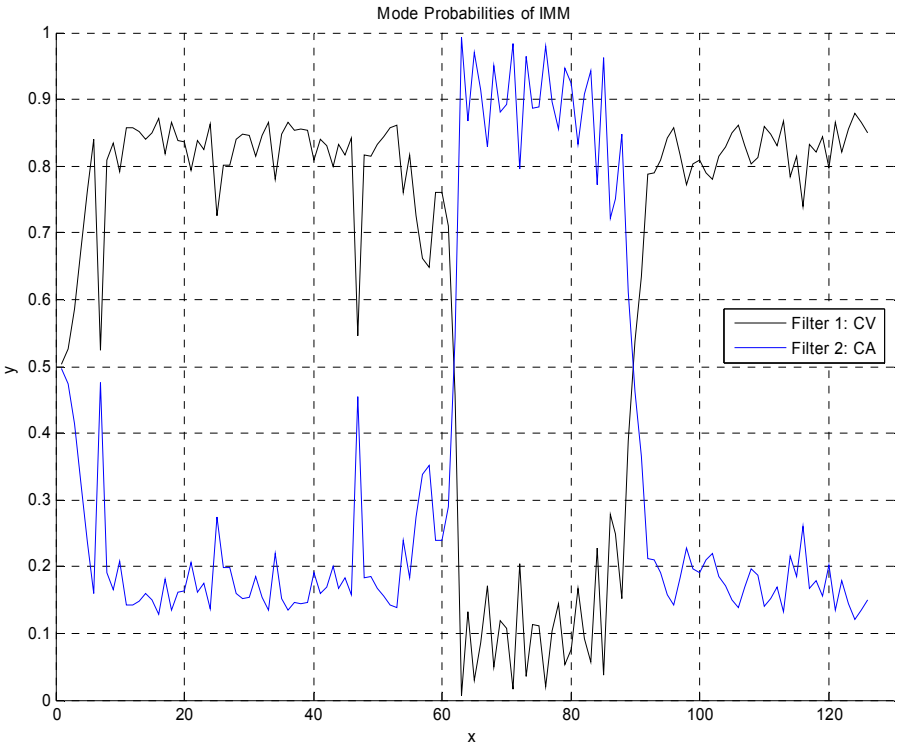

Figure 5-24: Mode probabilities of IMM-EKF.

Table 5-18: Simulation result.

	Meas.	**EKF**	**IMM-EKF**	**SIR-PF**
Total RMSE in Position	1.0922×10^3	714.0706	610.6748	691.8079
Computation Time (s)	-	0.040	0.136	288.619

Remarks:

- For different runs SIR-PF can perform worse than the EKF. But, IMM-EKF usually results in the lowest RMSE.

- CA model is more vulnerable to noise. During the maneuver it dominates the CV model as expected (Figure 5-24)

5.4.3.2 Part B

EKF with CV and IMM-EKF with four models (CV, CA, CCW-CT and CW-CT) are compared for 100 Monte Carlo Runs.

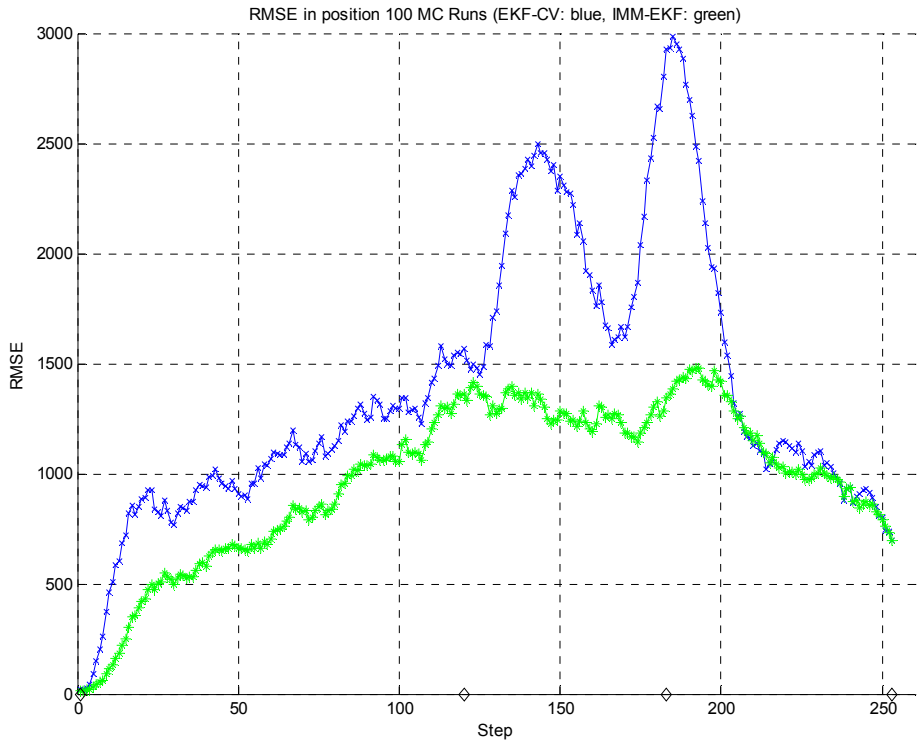

Figure 5-25: RMSE for the 3D scenario in 5.4.2.1 Part A.

Table 5-19: Simulation result.

	Meas.	**EKF-CV**	**IMM**
Total RMSE in Position	~ 3.5285e+003	1.5090e+003	1.0410e+003
Computation Time (s)	-	0.078	0.570

Similarly, EKF with CA and IMM-EKF with four models are compared for 100 Monte Carlo Runs.

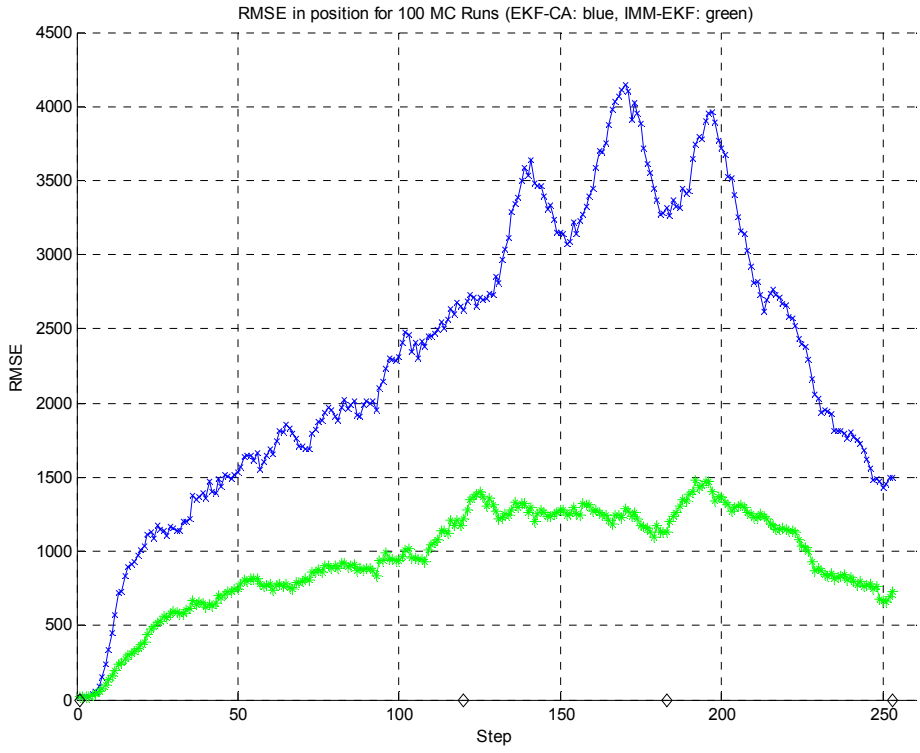

Figure 5-26: RMSE for the 3D scenario in 5.4.2.1 Part A.

Table 5-20: Simulation result.

	Meas.	**EKF-CA**	**IMM-EKF**
Total RMSE in Position	~ 3.5954×10^3	2.5242×10^3	1.0117×10^3
Computation Time (s)	-	0.078	0.574

Remarks:

- EKF with CV model performs better than the EKF with CA model. Although it is expected that CA would provide improvement since it is a more sophisticated model, it may not be the case in practice. To be able to track targets having constant acceleration the acceleration increment should be relatively small. However acceleration can change rapidly at the beginning or at the end of the maneuver, causing problems for EKF with CA model.

5.4.3.3 Part C

The implemented IMM-EKF includes CV (Constant Velocity), CA (Constant Acceleration), CCW-CT (Counter-Clockwise Coordinated Turn) and CW-CT (Clockwise Coordinated Turn) models. The results of 5.4.3.2 Part B show that the EKF with CV model outperforms the EKF with CA model. Therefore, IMM-EKF is compared with EKF and SIR-PF that have CV models inside.

Table 5-21: Simulation parameters.

Noise in the measurements	Azimuth (rad): Gaussian, σ^2=0.0052 Elevation (rad): Gaussian, σ^2=0.00087 Range (m): Uniform on [-20, 20]
Measurement noise model (covariance) in the filters	Azimuth and Elevation: For all filters same as the actual covariance Range for EKF: Gaussian, σ^2=400 Range for IMM filters: Gaussian, σ^2=400 Range for SIR-PF: Uniform on [-20, 20]
Process noise model (covariance) in the filters	EKF: CV Model → 4 * I_3 IMM: CV Model → 0.1 * I_3 CA Model →0.01 * I_3 CCW-CT & CW-CT Models → For acceleration components: 0.001 * I_3 For turn rate (rad/s) component: 0.000025 SIR-PF: 4 * I_3
IMM Transition Prob.	[0.85 0.05 0.05 0.05 ; 0.05 0.85 0.05 0.05 ; 0.05 0.05 0.85 0.05 ; 0.05 0.05 0.05 0.85];
Time Step (s)	5
Particle Count for SIR-PF	5000

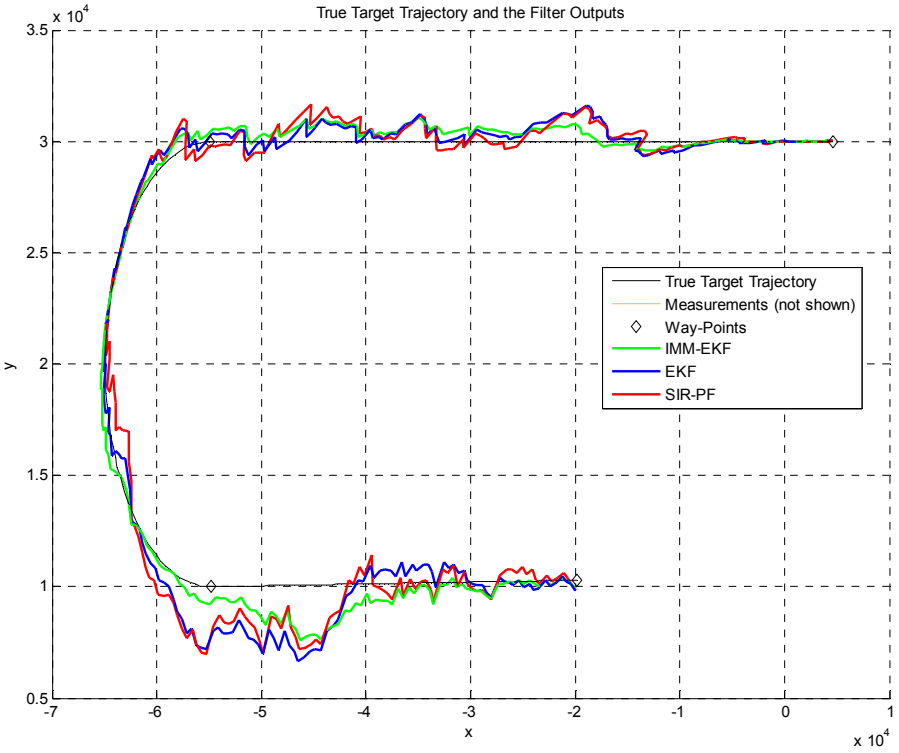

Figure 5-27: True target trajectory and the filter outputs for 5000 particles.

Figure 5-28: Mode probabilities of IMM-EKF.

Table 5-22: Simulation result.

	Meas.	**EKF**	**IMM-EKF**	**SIR-PF**
Total RMSE in Position	3.5667×10^3	1.1677×10^3	815.8345	1.1533×10^3
Computation Time (s)	-	0.078	0.568	449.065

Remarks:

- SIR –PF and the EKF's performance is close in terms of RMSE. IMM-EKF has superiority over both of them since it also includes the coordinated turn (CT) models. As seen in Figure 5-28 filter can adopt itself to the mode changes of the target.

- The turn rate of the true trajectory is determined according to the Equation (4-32). Since the linearized version of the same equation is also in the CT models, target state can accurately be estimated. The behavior of the IMM-EKF under different turn rates should be investigated. In that case, performance improvement with respect to EKF and SIR-PF would not be that much.

5.4.4 Simulation Set 4

5.4.4.1 Part A

The MM-PF described in Section 3.4.3 is compared with the EKF and IMM-EKF by using *Trajectory 3*. While EKF has the CV model, IMM-EKF and MM-PF include the same four state transition models: CV, CA CCW-CT and CW-CT.

Table 5-23: Simulation parameters.

Noise in the measurements	Azimuth (rad): Gaussian, σ^2=0.0052 Elevation (rad): Gaussian, σ^2=0.000087 Range (m): Uniform on [-20, 20]
Measurement noise model (covariance) in the filters	Azimuth and Elevation: For all filters same as the actual covariance Range for EKF: Gaussian, σ^2=400 Range for IMM filters: Gaussian, σ^2=400 Range for MM-PF: Uniform on [-20, 20]
Process noise model (covariance) in the filters	EKF: CV Model → 4 * I_3 IMM: CV Model → 0.01 * I_3 CA Model → 0.1 * I_3 CCW-CT & CW-CT Models → For acceleration components: 0.001 * I_3 For turn rate (rad/s) component: 0.000025 MM-PF: Same as IMM
IMM & MM-PF Transition Probability Matrix	[0.85 0.05 0.05 0.05 ; 0.05 0.85 0.05 0.05 ; 0.05 0.05 0.85 0.05 ; 0.05 0.05 0.05 0.85];
Time Step (s)	5
Particle Count for MM-PF	2000

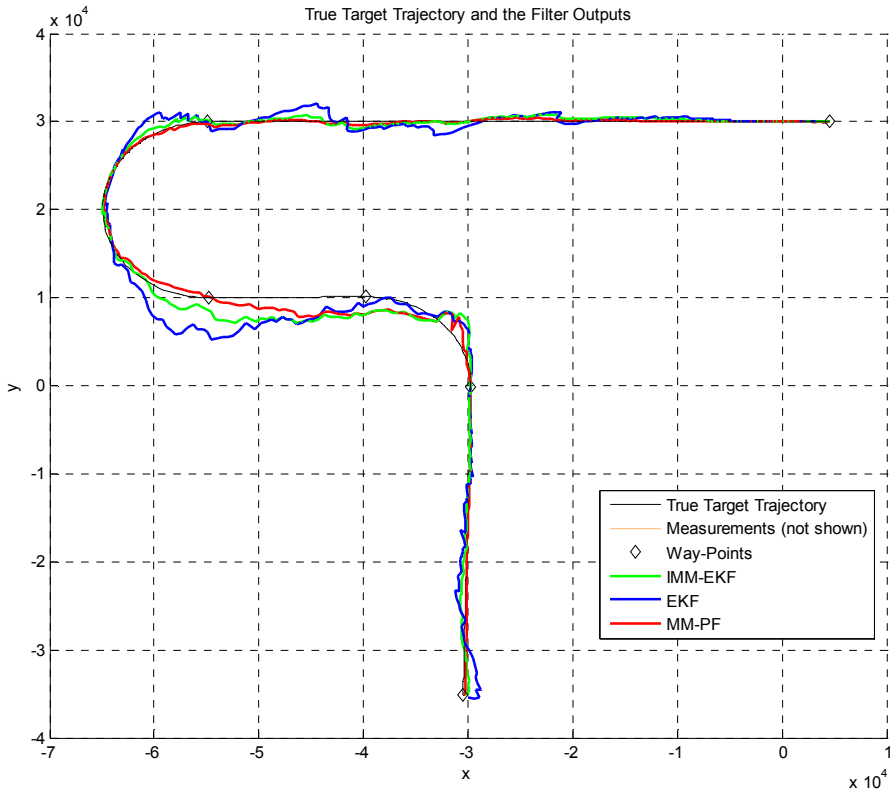

Figure 5-29: True target trajectory and the filter outputs for 2000 particles.

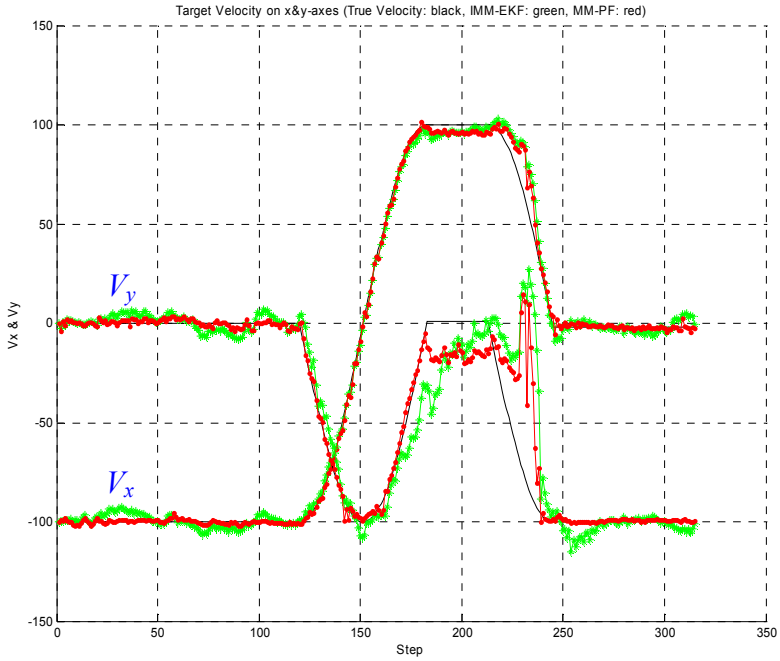

Figure 5-30: True target velocities and the filter outputs for 2000 particles.

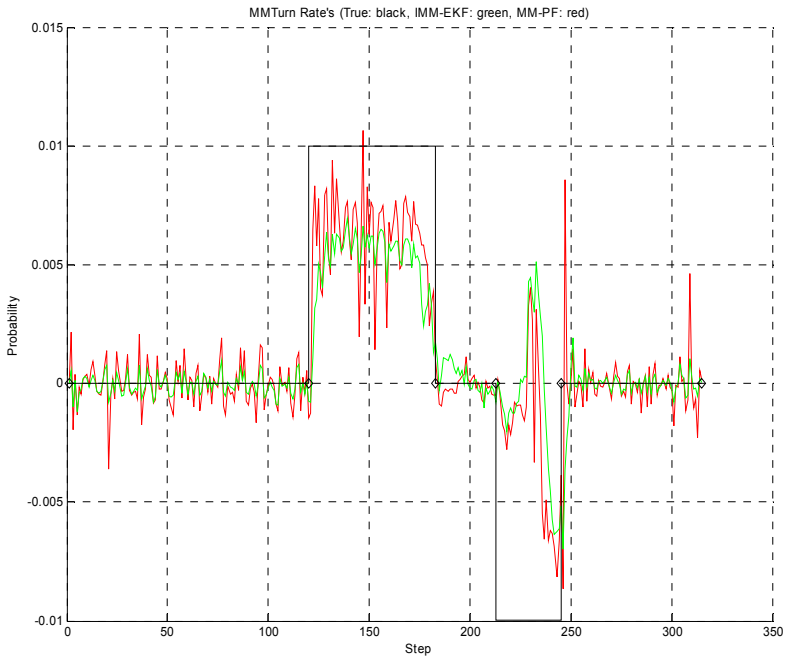

Figure 5-31: Turn rate estimate of IMM-EKF and MM-PF.

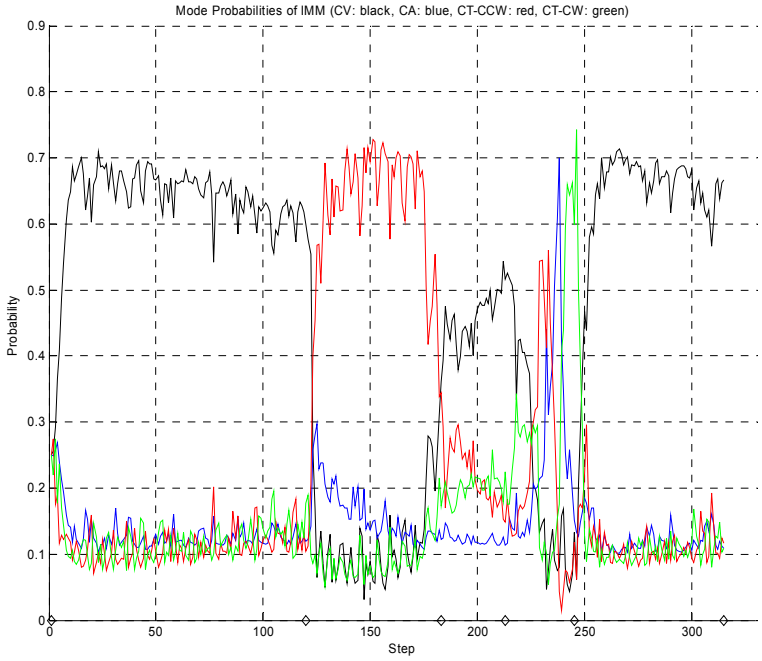

Figure 5-32: Mode probabilities of IMM-EKF.

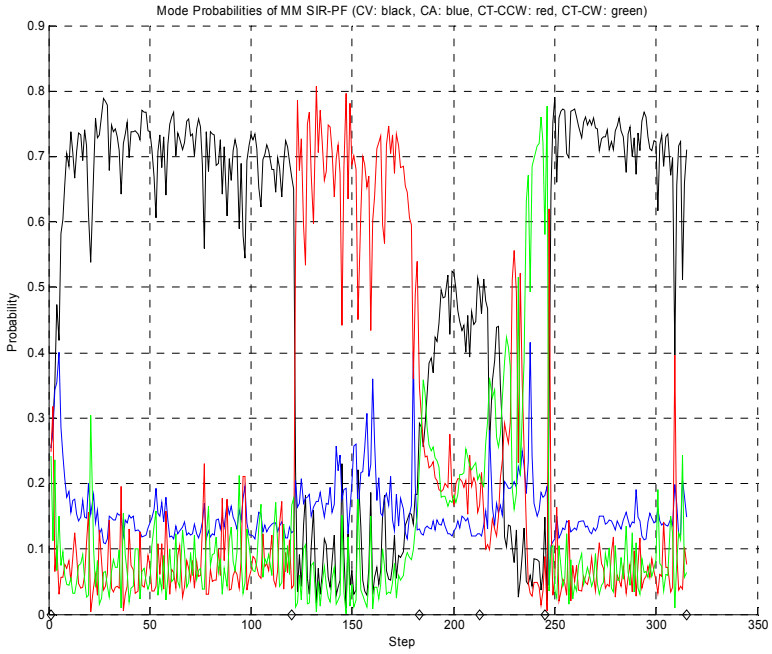

Figure 5-33: Number of selected particles from each model of MM-PF.

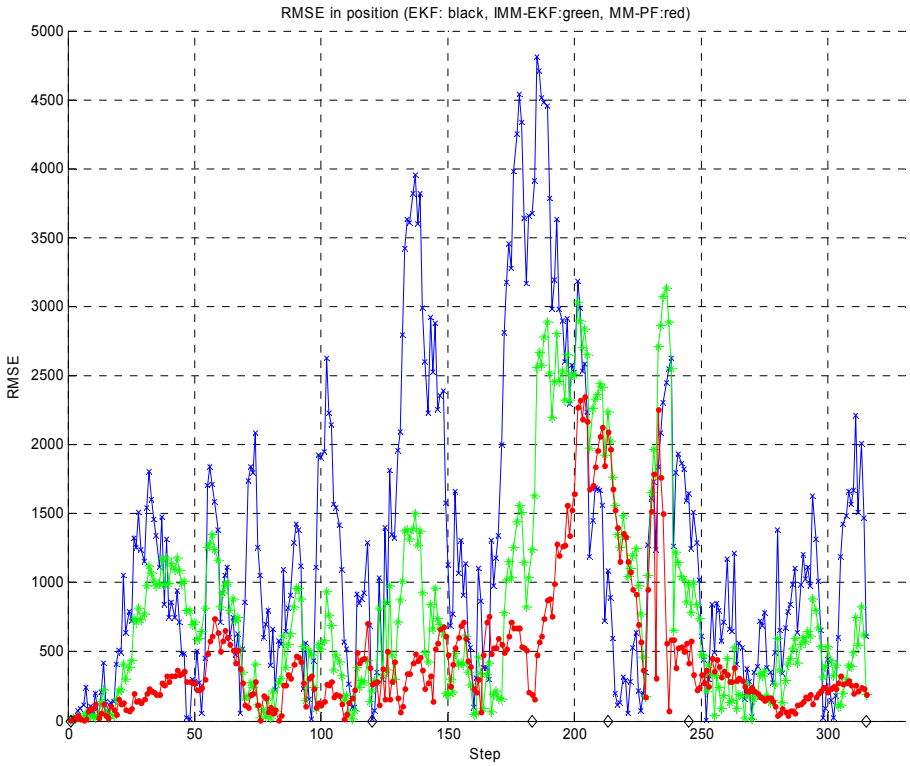

Figure 5-34: RMSE of EKF, IMM-EKF and MM-PF's for 2000 particles.

Table 5-24: Simulation result.

	Meas.	**EKF**	**IMM-EKF**	**MM-PF**
Total RMSE in Position	3.3181×10^3	1.6907×10^3	1.1083×10^3	677.0086
Computation Time (s)	-	0.095	0.679	649.664

Remarks:

- The adaptive structure of IMM can easily be observed from Figure 5-32. The mode related to the ongoing maneuver is automatically become dominant. Similarly MM-PF's response can also be viewed in Figure 5-33. For each time step, the figure shows the ratio of the number of selected particles generated by each dedicated model over the total number of particles. Therefore, for example, during the left turn the particles drawn from the CCW-CT model are statistically selected more.

- MM-PF's performance is better than the IMM-EKF even for a lower particle size (2000) compared to previous simulation sets. This improvement can also be observed from the mode probabilities. In MM-PF the mode probabilities of the true model are usually higher than the ones in IMM-EKF.

- The estimated turn rate in Figure 5-31 also shows that both filters recognize the maneuver.

5.4.4.2 Part B

The filters in Part A are applied to the same trajectory (*Trajectory 3*). In this case time step is 10s instead of 5s, and the RMSE results are averaged over 10 Monte Carlo runs.

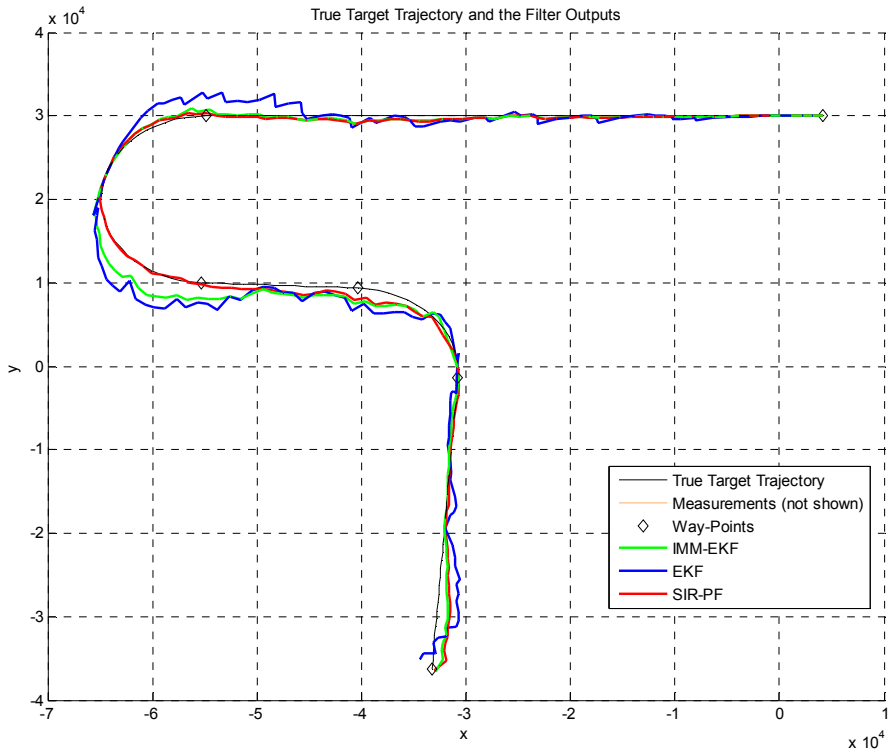

Figure 5-35: True target trajectory and the filter outputs for 2000 particles. Graph is for only one run in the MC simulation.

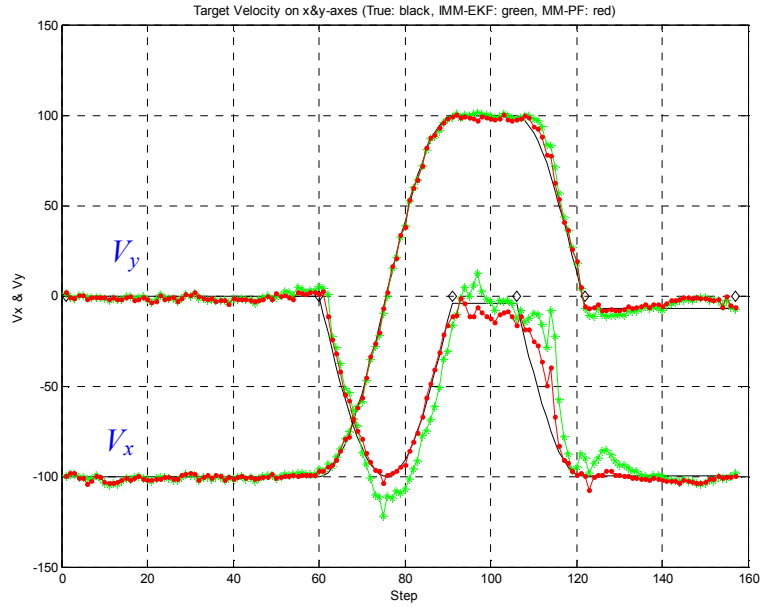

Figure 5-36: True target velocities and the filter outputs for 2000 particles. Graph is for only one run in the MC simulation.

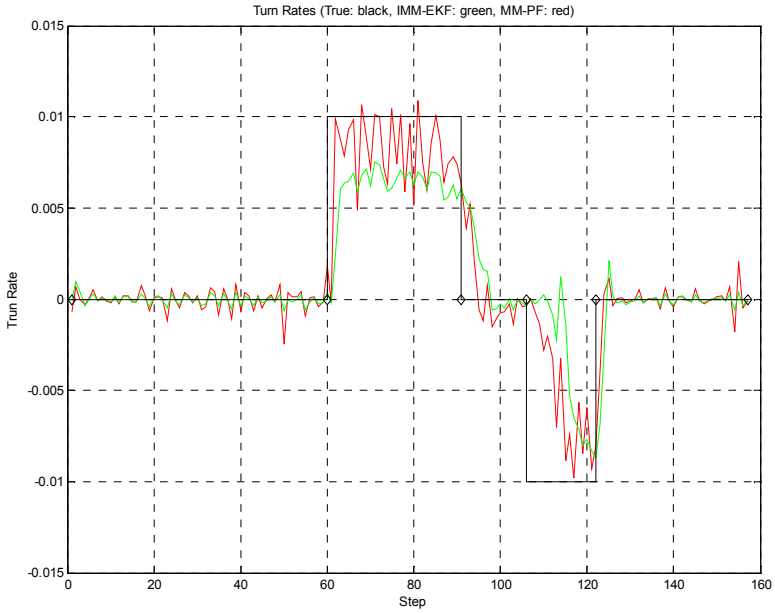

Figure 5-37: Turn rate estimate of IMM-EKF and MM-PF. Graph is for only one run in the MC simulation.

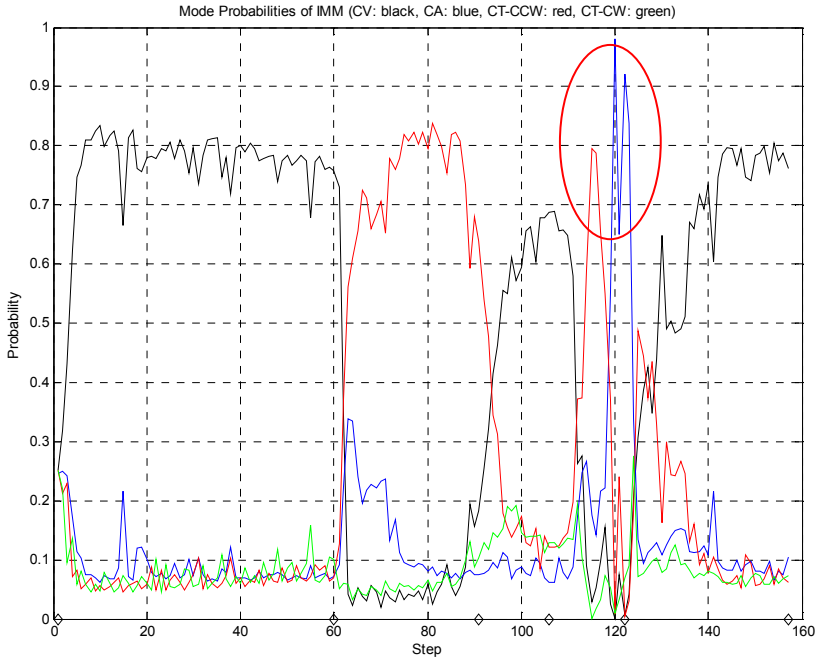

Figure 5-38: Mode probabilities of IMM-EKF.

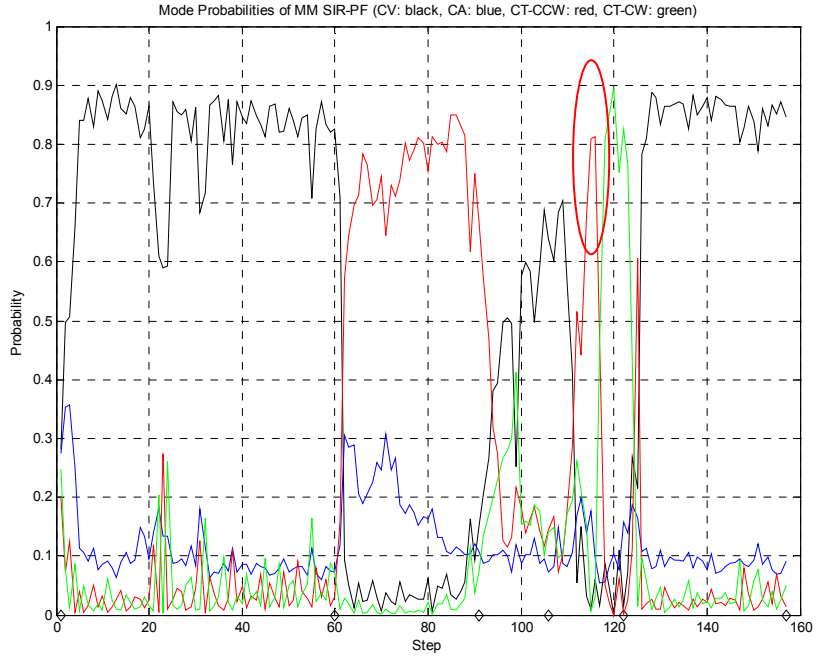

Figure 5-39: Number of selected particles from each model of MM-PF.

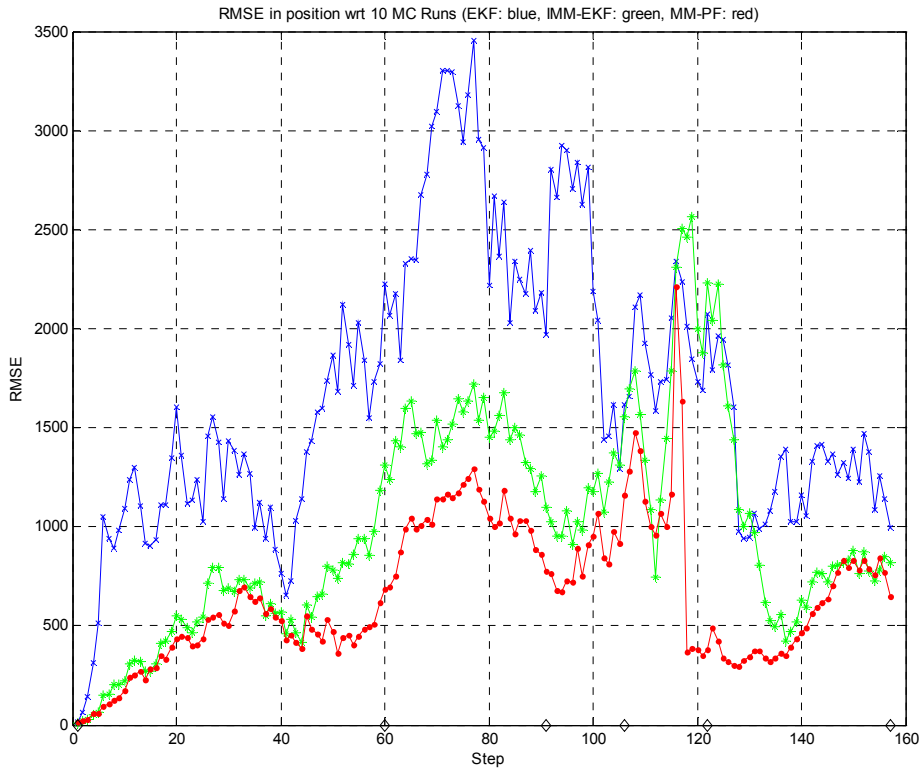

Figure 5-40: RMSE of EKF, IMM-EKF and MM-PF's for 2000 particles and 10 MC runs.

Table 5-25: Simulation result.

Avg. Values for 10 MC runs	**Meas.**	**EKF**	**IMM-EKF**	**MM-PF**
Total RMSE in Position	3.8152e+003	1.8207e+003	1.1283e+003	753.0373
Computation Time (s)	-	0.048	0.359	347.888

Remarks:

- Mode probabilities of the IMM-EKF (Figure 5-38) and MM-PF (Figure 5-39) are consistent with scenario properties given in Table 5-3, indicating that they both work well. The marked sections in these graphs are faulty mode assumptions; however both filters correct themselves quickly.

- After the Monte Carlo simulations MM-PF's performance is better than the IMM-EKF at almost every step (Figure 5-40). Reasons are further discussed in Section 5.5.3.

5.4.4.3 Part C

The filters in Part A are applied to a modified version of *Trajectory 1*. The duration of the left turn is made ten times longer. Time step is again 10s.

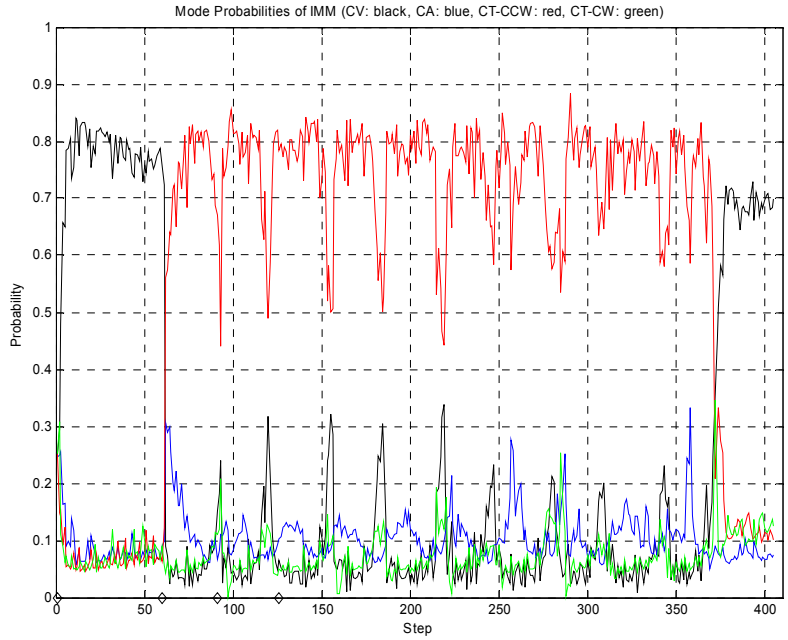

Figure 5-41: Mode probabilities of IMM-EKF.

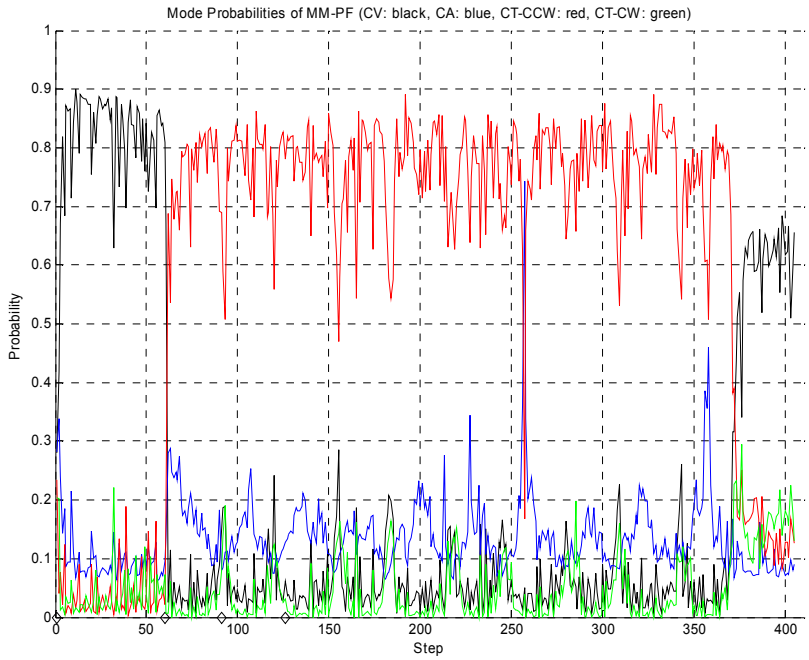

Figure 5-42: Number of selected particles from each model of MM-PF.

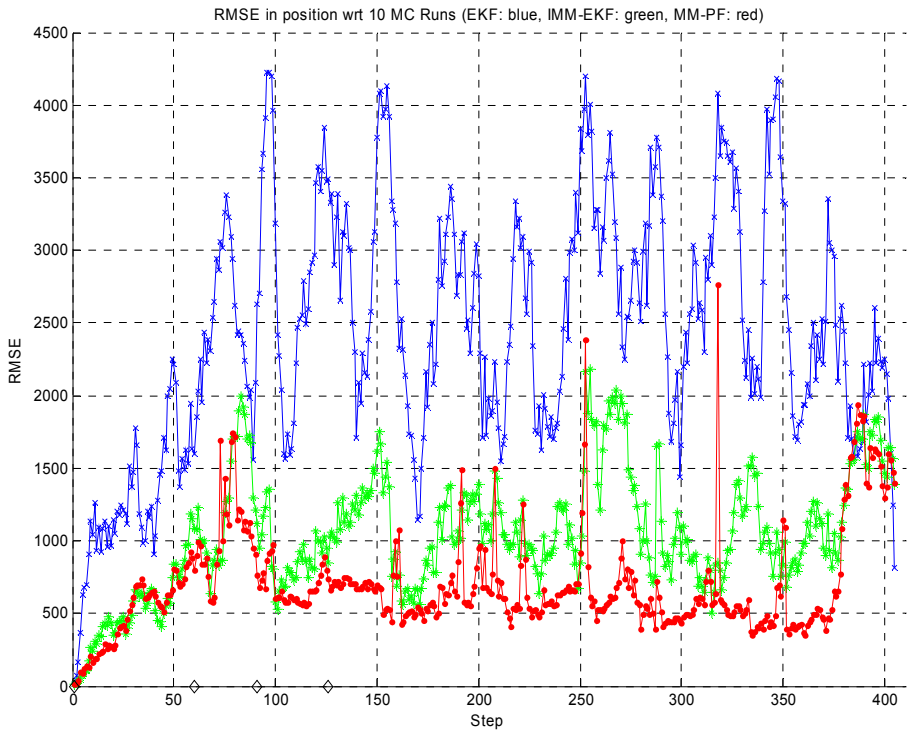

Figure 5-43: RMSE of EKF, IMM-EKF and MM-PF's for 2000 particles and 10 MC runs.

Table 5-26: Simulation result.

Avg. Values for 10 MC runs	**Meas.**	**EKF**	**IMM-EKF**	**MM-PF**
Total RMSE in Position	4.2565×10^3	2.5899×10^3	1.1240×10^3	789.415
Computation Time (s)	-	0.115	0.851	830.343

Remarks:

- MM-PF again provides an improvement in the performance. Reasons are further discussed in Section 5.5.3.

5.5 Discussion

5.5.1 Particle Distribution

Particle filters main property is the ability to represent the distributions more accurately. Figure 5-44 illustrates a non-Gaussian density and its approximations in different filters.

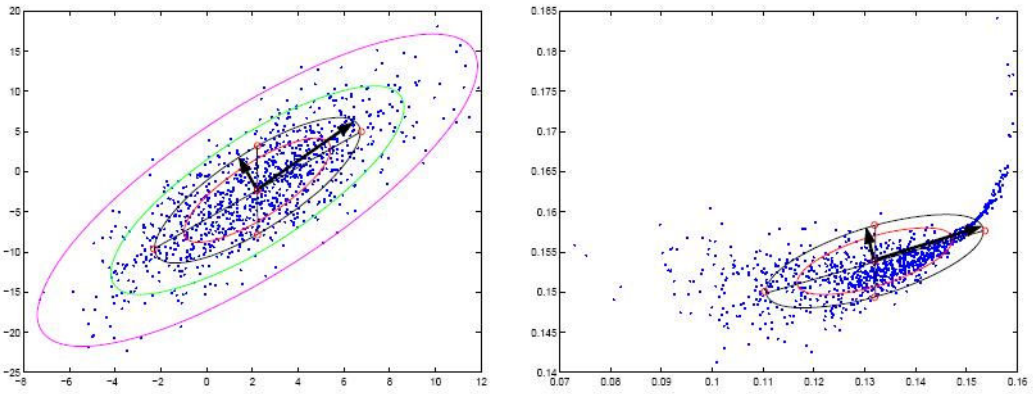

Figure 5-44: Particles representing the distribution [3]. Gaussian and non-Gaussian densities are shown respectively.

The distribution may lose its Gaussianity after passing through a non-linearity, or its original distribution may be non-Gaussian. The following figures include samples of the posterior distributions in the simulations. In those graphs, five simulation

steps are shown but particle distributions are plotted for only two steps. Note that, the duplicated particles are not shown differently.

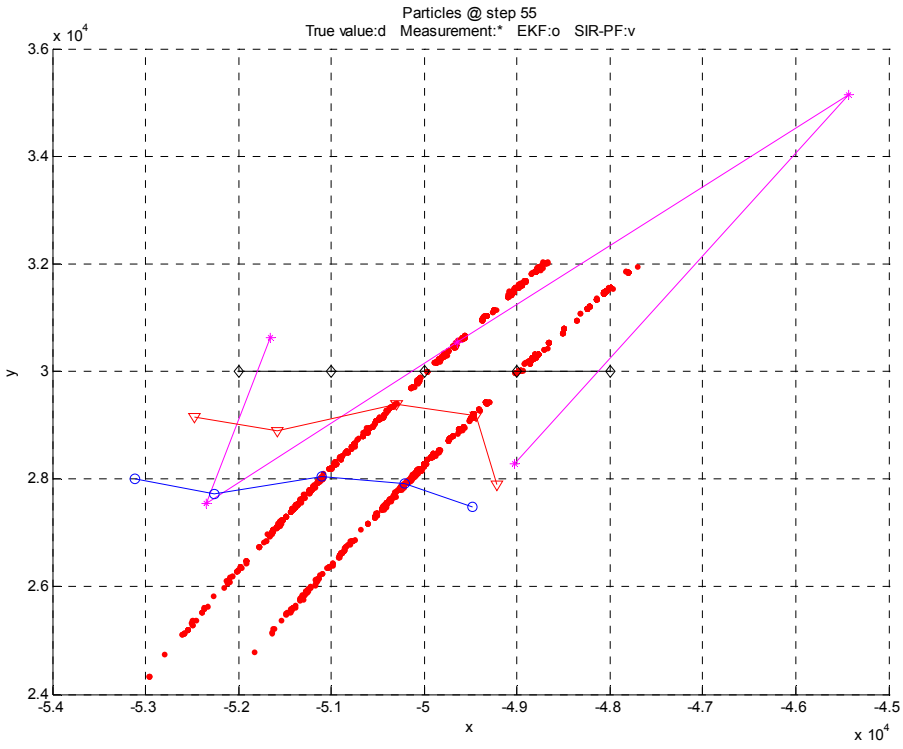

Figure 5-45: Particles from the simulation in 5.4.1.4 Part D (True Trajectory: black, Measurements: pink, EKF: blue, SIR-PF: red).

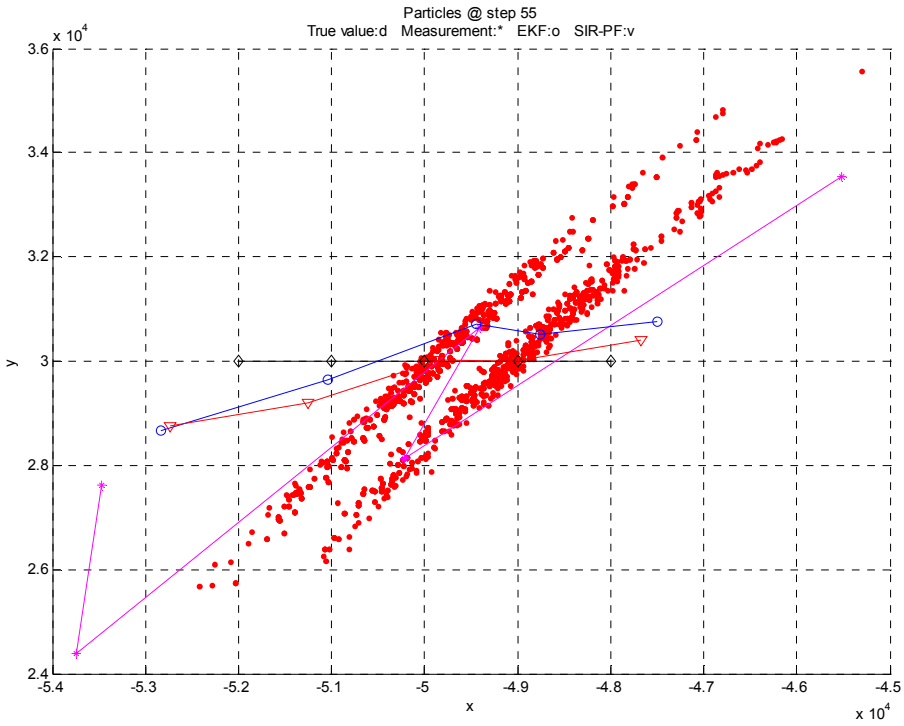

Figure 5-46: Particles from the simulation in 5.4.1.4 Part D. Gaussian range noise is exaggerated to show the distribution (σ^2=10000) (True Trajectory: black, Measurements: pink, EKF: blue, SIR-PF: red).

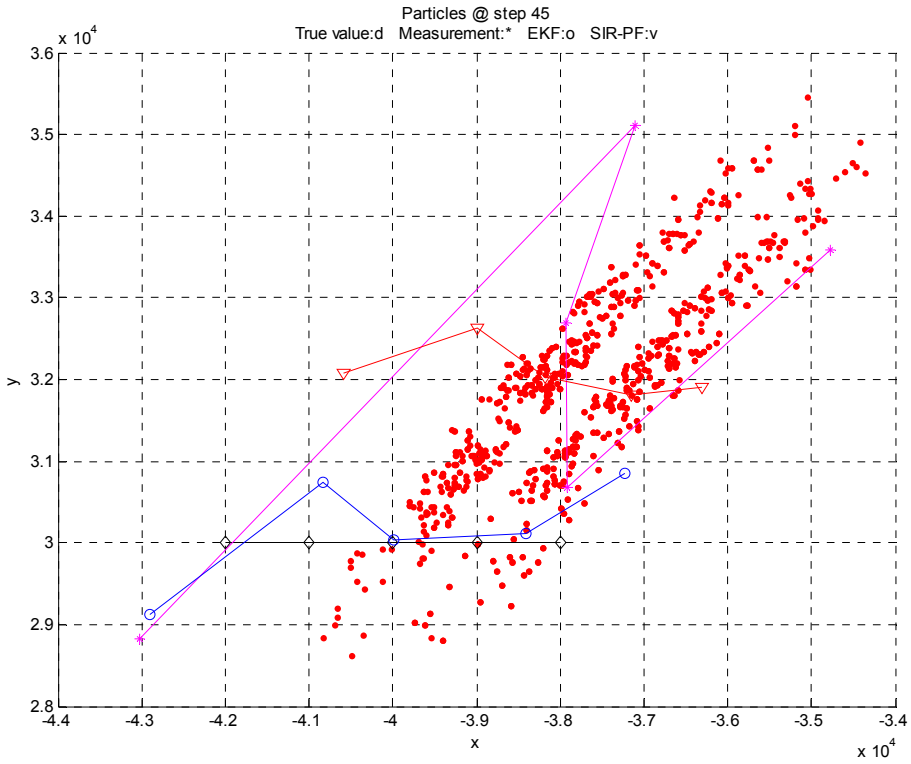

Figure 5-47: Particles from the simulation in 5.4.1.4 Part D. Range noise is uniform and exaggerated to show the distribution ([-500, 500]) (True Trajectory: black, Measurements: pink, EKF: blue, SIR-PF: red).

The degeneracy occurring due to model mismatch or consequent incorrect measurements may lead to divergence. Designer wants to use each sample in the approximation process for maximum computational efficiency. If a couple of particles start dominating the others, then the filter performance gets worse. Degeneracy phenomenon is shown in Figure 5-48 and Figure 5-49. The particle cloud in Figure 5-49 cannot properly do its function, since particles are piled into some regions. However the process noise may give a chance to recover. At each time step particles can spread out and form the desired distribution, unless more noisy measurements cause divergence. For the cases where the process noise is

small, the *annealed prior distribution* [3] approach may be used to broaden the prior.

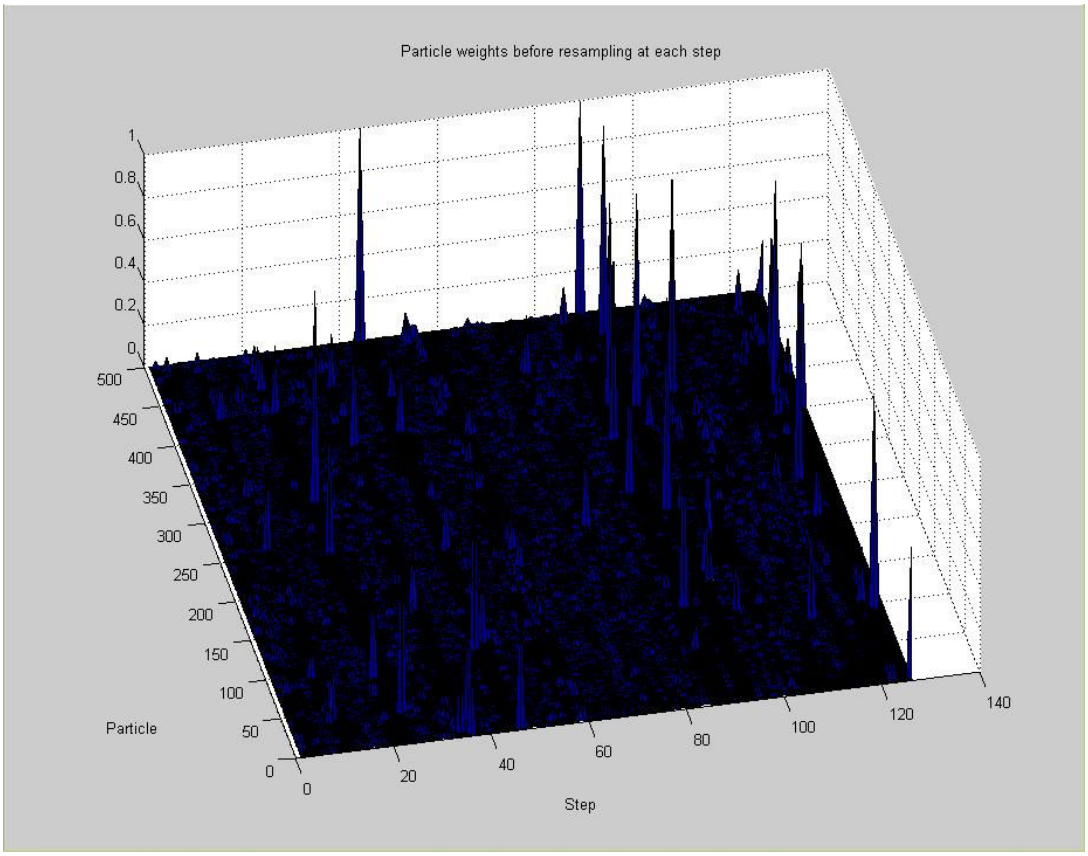

Figure 5-48: Particles weights with respect to time (step).

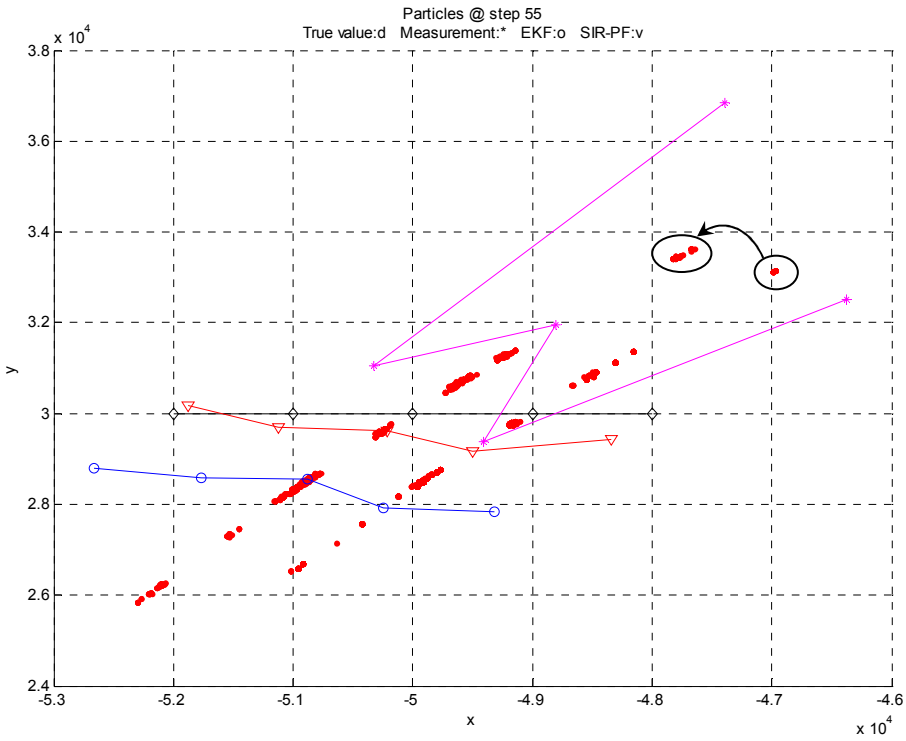

Figure 5-49: The degeneracy in the particles. Graph shows two consecutive steps of the scenario (target moves toward left). Note that, particles start to spread out due to the process noise.

5.5.2 Number of Particles

Particle count plays a very important role in the performance of particle filters. Although there is a lot of innovative way to enhance the algorithm, mostly the real improvement occurs with high number of particles. Therefore for many applications (especially real-time ones) the computational cost can make the particle filter infeasible. The results of the simulation described in 5.4.1.2 Part B for different particle sizes are shown below (Table 5-27 and Table 5-28).

Table 5-27: Simulation results showing the effect of particle size.

For 5000 particles	**Measurement**	**EKF**	**SIR-PF**
Total RMSE in Position	3.4396×10^3	1.4050×10^3	943.4269
Computation Time (s)	-	0.067	406.02

Table 5-28: Simulation results showing the effect of particle size.

For 500 particles	**Measurement**	**EKF**	**SIR-PF**
Total RMSE in Position	3.4724×10^3	1.4183×10^3	1.4773×10^3
Computation Time (s)	-	0.083	33.146

5.5.3 Multiple Model Performance

During the trials in Simulation Set 4, the mode probability graphs indicate that both IMM-EKF and MM-PF can detect the ongoing maneuver and switch to the proper model. Due to their adaptive structure the performance is usually better than a filter with a single model. For the nonlinear problem considered in this study, Monte Carlo runs clearly indicate that MM-PF give better estimation results than the IMM-EKF. Also note that the required number of particles for a certain level of performance is decreased by using multiple model approach.

In the MM-PF the particle cloud is desired to spread according to the different models used. An exaggerated case is shown in Figure 5-50. Although that gives the impression that the new state distribution has three peaks (non-Gaussian), it may not always be the case in the real world. Usually targets do not have discrete maneuver characteristics. Since they do not always turn with the same acceleration component ($\pm a_m$, the typical maneuver acceleration), the turn rates for the same speed may vary. Therefore the regions between the particle clouds (red, blue and green) should also include some particles. The interesting thing is that if the process

noise is large (particle cloud is wide) and turn rate is low, then particles generated by the CV, CA and CT models are mixed. For this case the overall performance would be close to using one particle filter. This effect can be observed from Figure 5-39 and Figure 5-37 where the coordinated turn characteristic cannot be observed (CCW-CT does not dominate since the other models can also follow that trajectory).

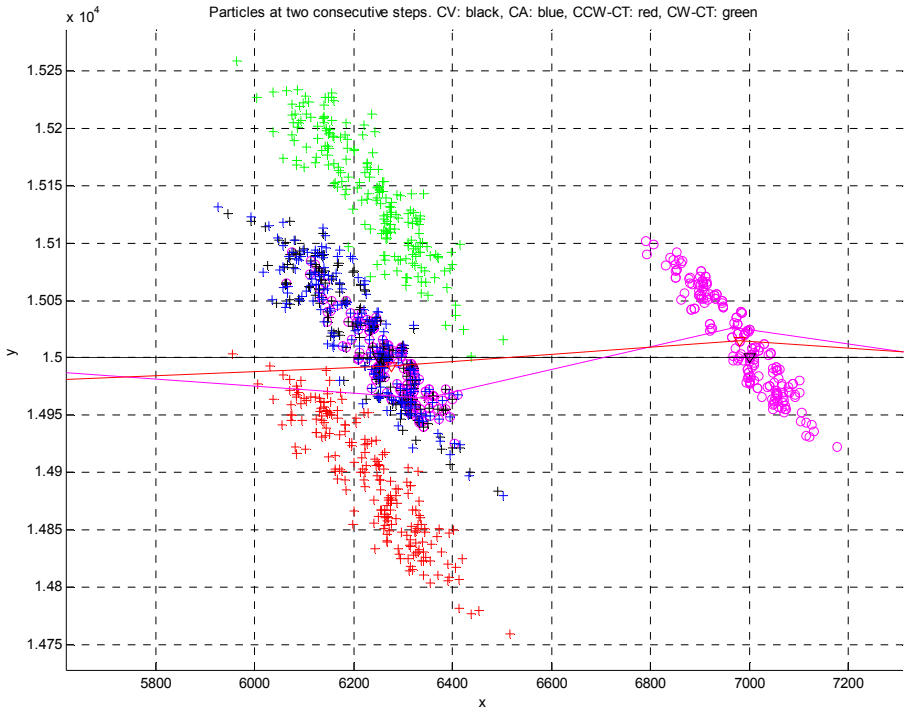

Figure 5-50: An illustration of propagating the particles according to different state transition models.

Particle filter can also be used as a block in the regular IMM framework. In that case it would be a replacement to KF or EKF, and only its output is combined with other filters. However MM-PF algorithm gives the opportunity to directly combine

the particles, which would yield more accurate results than combining only the estimates.

5.5.4 Other Issues

There are many other aspects of particle filtering that needs attention. For example the relative shapes of the prior and likelihood play an important role in particle filter design. One of them being much tighter then the other decreases the efficiency and filter performance. During the simulations there occurred some cases where the particle filter started to outperform other filters when the measurement noise covariance is increased. For low process noise the annealed prior distribution [3] approach for low measurement noise the likelihood particle filter (Section 3.3.5) may be preferred to make the prior and likelihood comparable. If their shapes are very different then the particle filter can diverge or work inefficiently.

Although not used in this study, particle filters has a high capability of incorporating the system constraints. Mostly, evaluating the particle before resampling step is enough to satisfy the constraints. Similar things cannot be done that easily with Kalman filter based approaches. As an example, the elevation angle error covariance in this study is assumed to be very low (even lower than the azimuth angle error covariance) to prevent reading negative angles. However in particle filters unrealistic the samples can easily be rejected. This gives an advantage to the particle filter in terrain aided tracking applications.

In Section 5.4.1.5 the SIR-PF with Gaussian noise assumption has higher RMSE values than the EKF. Considering that the noise characteristic is uniform, the behavior of particle filter under model mismatch needs to be examined further.

CHAPTER 6

CONCLUSIONS

In this thesis, the particle filters are examined using the problem of maneuvering target tracking via radar measurements. In order to gain insight particle filter algorithms are compared with classical Kalman filter based algorithms for different system models and scenarios. Main focus is on observing the performance improvement introduced by particle filtering in non-linear and non-Gaussian systems.

Firstly, the basis for recursive Bayesian estimation is provided and related filtering techniques are presented. Throughout the years many techniques are developed to fight non-linear and non-Gaussian systems. EKF or IMM are examples of the most commonly used methods while dealing with that kind of systems. The designer should know each individual method's advantages and disadvantages and select the proper one that suits the problem.

After the conventional techniques, the sequential Monte Carlo methods, i.e. particle filters, and the theory behind are introduced. The most commonly used algorithms SIS-PF and SIR-PF are derived. Based on these two algorithms, the key points like the degeneracy phenomenon and the choice of importance density are underlined. Proper measures need to be taken in order to overcome the degeneracy problem and design a feasible and robust particle filter. Changing the structure of the particle filter can provide improvements, as well as the good choice of importance density

and resampling algorithm. Some of the important modified versions of the particle filter (APF, RPF, and MM-PF) are explained to provide examples.

The most important contribution of this thesis is the application of particle filtering to tracking systems. To do that first the system models in target tracking are described. It is well known that the estimation filters' performances depend on the characteristics of the model. Expressing the problem in a suitable way increases the resultant performance for any filtering method. In the tracking problem considered here different coordinate axes choices provide different advantages. Augmenting the state with additional parameters like accelerations or turn rate can also be useful. However, extra states do not guarantee better performance. As an example, the EKF with CA model are not as good as the EKF with CV model.

Finally some of those algorithms and system models are realized to investigate the particle filters. The most important point is that particle filtering technique requires high processing power to achieve the promised improvement in the performance. But processing high number of particles does not solve all the problems.

For the tracking problem in this study, the particle filters provide better performance than the Kalman filter based methods with similar modifications. The SIR-PF can outperform EKF but not IMM-EKF. But, when the SIR-PF is used with same multiple model approach, i.e. MM-PF, it can outperform IMM-EKF too. During the simulations, one noticeable result is that the performance improvement of particle filters increases with the amount of nonlinearity in the problem as expected.

The degeneracy is an inevitable problem for particle filters. It not only degrades the performance, but also yields divergence. As explained above, increasing the number of particles is one way to eliminate its effects. However, properly choosing the importance density and modifying the algorithm according to the needs are more elegant and efficient precautions.

The prior distribution is used as the importance density for the sake of simplicity. Then, the resultant particles are weighted according to their likelihoods. In that case the relative shapes of the prior and likelihood distribution affect the performance of the particle filter directly. The particle filter performs well unless one of them is much tighter than the other.

For sufficient number of particles, the particle filter can represent the actual non-Gaussian posterior distribution better than the conventional Kalman filter based methods. However due to the degeneracy phenomenon the number of efficient particles tend to decrease. Resampling step is the basic way to fight with this problem, but the overall performance can also be improved by other methods. Using multiple models to express the system behavior is one of them. More realistic approximations of the posterior and less degeneracy can be achieved since the particles tend to go the more relevant areas of the state-space.

During the implementation, the weaknesses of particle filters, like computational cost and degeneracy phenomenon, should be taken under control; otherwise the resultant performance may be worse. First the analysis of the system should be done and particle filters' potential problems should be addressed. Then the proper modifications should be done on to the regular particle filter algorithm (SIS/SIR). These modifications are not only for assuring convergence but also for reducing the computational cost required for the same performance.

As a future work, verifications should be done with real trajectory and measurement data. Different scenarios should be implemented to investigate the particle filters performance in model mismatched situations. According to the specific tracking problem here, further modifications can be done in particle filter algorithm. Marginalization, regularization and using other filters (EKF, UKF …) to obtain better importance densities are possible.

REFERENCES

[1] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, "Estimation with Applications To Tracking and Navigation", John Wiley & Sons, NY, 2001

[2] Wikipedia, "Kalman Filter", http://en.wikipedia.org/wiki/Kalman_filter, July 8th 2007, Last date accessed: November 2007

[3] Z. Chen, "Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond", http://soma.çrl.mcmaster.ca/~zhechen/ieee_bayes.pdf, 2003

[4] B. Ristic, M. S. Arulampalam, N. Gordon, "Beyond the Kalman Filter, Particle Filters For Tracking Applications", Artech House, 2004

[5] S. M. Herman, "A Particle Filtering Approach to Joint Passive Radar Tracking and Target Classification", Ph.D Thesis. University of Illinois at Urbana-Champaign, 2002

[6] R. Karlsson, "Simulation Based Methods for Target Tracking", M.Sc. Thesis, Linköpings Universitet, Sweden, 2002

[7] K. Akçay, "Performance Metrics For Fundamental Estimation Filters", M.Sc. Thesis, Middle East Technical University, 2005

[8] G. Welch, and G. Bishop, "An Introduction to the Kalman Filter", Transaction of Department of Computer Science University of North Carolina, Chapel Hill, 2002

[9] Wikipedia, "Recursive Bayesian Estimation", http://en.wikipedia.org/wiki/Recursive_Bayesian_estimation, July 9th 2007, Last date accessed: November 2007

[10] V. E. Benes, "Exact finite-dimensional filters with certain diffusion non linear drift", Stochastics, vol. 5, pp. 65-92, 1981

[11] F. E. Daum, "Exact finite dimensional nonlinear filters", IEEE Trans. Automatic Control, vol 31, no. 7, pp. 616-622, 1986

[12] L. Rabiner and B.-H. Juang, “Fundamentals of Speech Recognition”, Englewood”Cliffs, NJ, Prentice Hall, 1993

[13] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp “A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking”, IEEE Transactions on Signal Processing, vol. 50, no. 2, February 2002

[14] L. Tong, “Bayesian Estimation”, Cornell University, NY, http://courseinfo.cit.cornell.edu/courses/ECE564/, 2005

[15] Y. Bar-Shalom, and X. R. Li, “Multitarget-Multisensor Tracking: Principles and Techniques”, Artech House, Boston, 1995

[16] K. V. Ramachandra, "Kalman Filtering Techniques for Radar Tracking", Marcel Dekker, New York, 2000

[17] P. S. Maybeck, “Stochastic Models, Estimation, and Control”, vol. 1, Academic Press, New York, 1979

[18] R. Merwe, “Probabilistic Inference Using Sigma-Point Kalman Filters”, Thesis Proposal, Oregon Health & Science University

[19] B. D. O. Anderson and J. B. Moore, “Optimal Filtering”, Englewood Cliffs, Prentice-Hall, NJ, 1979

[20] F. Gustafsson, and A. J. Isaksson, “Best Choice of Coordinate System for Tracking Coordinated Turns”, Proceedings of the 35th Conference on Decision and Control, Kobe, Japan December 1996

[21] T. Lefebvre, H. Bruyninckx, and J. De Schutter, “Kalman Filters for nonlinear systems: a comparison of performance”, IEEE Transactions on Automatic Control, October 2001

[22] S. Sadhu, S. Mondal, M. Srinivasan, T. K. Ghoshal, "Sigma point Kalman filter for bearing only tracking" Signal Processing 86, Elsevier, 2006

[23] E. A. Wan, and R. Merwe, “Sigma-Point Kalman Filters (Overview)”, OGI School of Science and Engineering at OHSU

[24] E. A. Wan, and R. Merwe, “Sigma-Point Kalman Filters for Nonlinear Estimation and Sensor-Fusion - Applications to Integrated Navigation”, American Institute of Aeronautics and Astronautics

[25] E. A. Wan, and R. Merwe, “The square-root unscented Kalman Filter for State and Parameter-Estimation”, Oregon Graduate Institute of Science and Technology

[26] M. Busch, and S. Blackman, “Evaluation of 1MM filtering for an air defense system application”, SPIE Vol. 2561 pg 435-447, 1995

[27] N. Peach, “Bearings-only tracking using a set of range-parameterised extended Kalman filters”, IEE Proc. -Control Theory Appl., Vol. 142, No. I, pg 73-80, January 1995

[28] S. S. Blackman, R. J. Dempster, M. T. Busch, and R. F. Popoli, “IMM/MHT Solution to Radar Benchmark Tracking Problem”, IEEE Transactions on Aerospace and Electronic Systems Vol. 35, No. 2, pg 730-738, April 1999

[29] L. Song, and H. Ji, “Least Squares Interacting Multiple Model Algorithm for Passive Multi-sensor Maneuvering Target Tracking”, ICNC 2006, Part I, LNCS 4221, pp. 479 – 482, Springer-Verlag Berlin Heidelberg 2006

[30] X. Rong Li, and Yaakov Bar-Shalom, “Design of an Interacting Multiple Model Algorithm for Air Traffic Control Tracking”, IEEE Transactions on Control Systems Technology Vol. 1, No. 3, September 1993

[31] F. Opitz, and T. Kausch, “UKF controlled Variable-Structure IMM Algorithms using Coordinated Turn Models”, Defense and Communications Systems/Air and Naval Defense EADS Deutschland GmbH

[32] Y. Boers, and J.N. Driessen, “Interacting multiple model particle filter”, IEE Proc.-Radar Sonar Navig., Vol. 150, No. 5, pg 344-349, October 2003

[33] R. Karlsson, “Particle Filtering for Positioning and Tracking Applications” Ph.D. Thesis, Linköpings Universitet, Sweden, 2005

[34] G. Liu, E. Gao, and C. Fan, “Multirate Interacting Multiple Model Algorithm Combined with Particle Filter for Nonlinear/Non-Gaussian Target Tracking”, IEEE Proceedings of the 16th International Conference on Artificial Reality and Telexistence—Workshops, 2006

[35] R. Karlsson, and F. Gustafsson, “Range estimation using angle-only target tracking with particle filters”, Proceedings of the American Control Conference Arlington, VA June 25-27, 2001

[36] R. Karlsson, and F. Gustafsson, “Recursive Bayesian estimation: bearings-only applications”, IEE Proc.-Radar Sonar Navig., Vol. 152, No. 5, October 2005

[37] B. R. Mahafza, “Radar Systems Analysis and Design Using MATLAB”, CHAPMAN & HALL/CRC, 2000

[38] M. I. Skolnik, “Radar Handbook”, 2nd ed., McGraw Hill, 1990

[39] Yalçın Tanık, “Yön Bulma Sistemleri”, AFCEA Türkiye, 1995

[40] X. R. Li, and V. P. Jilkov, “A Survey of Maneuvering Target Tracking: Dynamic Models”, Proceedings of SPIE Conference on Signal and Data Processing of Small Targets, Orlando, FL, USA, April 2000

[41] R. A. Singer. “Estimating Optimal Tracking Filter Performance for Manned Maneuvering Targets” IEEE Trans. Aerospace and Electronic Systems, AES-6:473–83, July 1970

[42] S. S. Blackman, "Multiple Target Tracking with Radar Applications", Artech House, 1986

[43] R. Popoli, " Design and analysis of modern tracking systems”, Artech House, 1999

[44] M. S. Arulampalam, B. Ristic, N. Gordon, T. Mansell, “Bearings-Only Tracking of Maneuvering Targets Using Particle Filters”, EURASIP Journal on Applied Signal Processing 15 pg 2351-2365, 2004

[45] R. Karlsson, and N. Bergman., “Auxiliary Particle Filters for Tracking a Maneuvering Target”, Proceedings of the 39:th IEEE Conference on Decision and Control, pages 3891–3895, Sydney, Australia, December 2000

[46] X. R. Li, and V. P. Jilkov, “A Survey of Maneuvering Target Tracking—Part III: Measurement Models”, Proceedings of SPIE Conference on Signal and Data Processing of Small Targets, San Diego, CA, USA, July-August 2001

[47] R. Karlsson, and F. Gustafsson, “Range estimation using angle-only target tracking with particle filters”, Proceedings of the American Control Conference Arlington, VA June 25-27, 2001

[48] Robinson, and Yin “Modified Spherical Coordinates for Radar”, American Institute of Aeronautics and Astronautics, PV1994_3546, 1994

[49] J. D. Hol, T. B. Schön, F. Gustafsson, “On Resampling Algorithms for Particle Filters”, Linköping University, 2004

[50] S. Roth, L. Sigal, M. J. Black, “Gibbs Likelihoods For Bayesian Tracking”, Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’04), 2004

[51] G. Liu, E. Gao, C. Fan, “Multirate Interacting Multiple Model Algorithm Combined with Particle Filter for Nonlinear/Non-Gaussian Target Tracking”, IEEE Proceedings of the 16th International Conference on Artificial Reality and Telexistence--Workshops (ICAT'06), 2006

[52] A. Athalye, S. Hong, P. M. Djuric, ”Distributed Architecture and Interconnection Scheme for Multiple model Particle Filters”, ICASSP, IEEE, 2006

[53] J. Wang, D. Zhao, W. Gao, S. Shan, “Interacting Multiple Model Particle Filter To Adaptive Visual Tracking”, Proceedings of the Third International Conference on Image and Graphics (ICIG’04), IEEE, 2004

[54] M. A. Zaveri, S. N. Merchant, U. B. Desai, “Arbitrary Trajectories Tracking using Multiple Model Based Particle Filtering in Infrared Image Sequence”, Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC’04), IEEE, 2004