UNIVERSITY OF OKLAHOMA
GRADUATE COLLEGE

IMPROVED NONLINEAR FILTERING FOR TARGET TRACKING

A DISSERTATION
SUBMITTED TO THE GRADUATE FACULTY
in partial fullfillment of the requirement for the
degree of
Doctor of Philosophy

By

YAN ZHAI
Norman, Oklahoma
2007

IMPROVED NONLINEAR FILTERING FOR TARGET TRACKING

A DISSERTATION APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

Dr. Mark Yeary, Chair

Dr. Tian-You Yu

Dr. Joseph Havlicek

Dr. Monte Tull

Dr. Luther White

Dr. John Antonio

ACKNOWLEDGMENTS

I would first like to thank my dissertation advisor Professor Mark Yeary for his assistance and guidance over the past four years. With his encouragement and advisory, I have had the chance to explore interesting and fruitful research topics. I am grateful that Professor Mark Yeary allowed me the flexibility to search for problems in which I was interested, and encouraged me to pursue the novel research. This thesis would not have been possible without the kind support, the trenchant critiques, the probing questions, and the remarkable patience of my dissertation advisor.

I would also like to thank the members on my dissertation committee, Professor Joseph Havlicek, Professor Tian-You Yu, Professor Monte Tull, Professor Luther White and Professor John Antonio. Specifically, I would like acknowledge Professor Joseph Havlicek for his technical assistance which results in valuable research and numerous publications.

I would like to thank my wife Hong Xu for her continuous support and encouragement in all my professional endeavors, and for all the support that she gave me during the years of my graduate study. I also remain indebted to my parents for their unconditional love that guarded me through the ups and downs of my life.

I would also like to thank the University of Oklahoma for providing the remarkable research environment and the opportunities. I would like to express my gratitude to all of those who gave me the possibility to complete this dissertation.

TABLE OF CONTENTS

Acknowledgments iv

Table of Contents v

List of Tables viii

List of Figures ix

Abstract xiii

1 Introduction 1

1.1 Radar Target Tracking 1

1.2 Target Tracking in Sensor Networks 2

1.3 Visual Target Tracking 2

1.4 The State Space Approach and State Estimation 3

1.5 Organization of the Dissertation 4

2 Fundamentals of Particle Filtering 5

2.1 Nonlinear Filtering 5

2.2 Particle Filter 7

2.3 Improving Particle Filters 8

2.3.1 Designing Better Proposal Distributions 9

2.3.2 Resampling and Markov Chain Monte Carlo (MCMC) 11

2.3.3 The Method of Multiple Dynamic Models 14

2.3.4 Using Multiple Measurement Cues or Multiple Sensors 14

3 Particle Filter with the State Partition Technique and Parallel EKFs 16

3.1 The State Partition Technique 16

3.1.1 The Method of SP-PEKF 16

3.1.2 The New Particle Filter Algorithm (PF-SP-PEKF) 20

3.1.3 Example: A Nonlinear Scalar Case 22
3.2 PF-SP-PEKF for Bearings-only Target Tracking 28
3.3 Particle Filter for Target Tracking in Sensor Networks 33
3.3.1 Introduction of Sensor Networks 33
3.3.2 Related Work in Sensor Network Target Tracking 35
3.3.3 Problem Formulation 37
3.3.4 Power-Aware Particle Filter For Target Tracking in Sensor Networks 38
3.3.5 Data Fusion and Tracking Results 42
3.3.6 Summary 50
3.4 Multiple Model Particle Filter 51
3.4.1 The Fundamentals of the Multiple Model Method 51
3.4.2 A Study of Target Dynamic Models 53
3.4.3 The General Form of Multiple Model Particle Filter 57
3.4.4 Multiple Model Bootstrap Filter 60
3.4.5 MM-Bootstrap with Multiple Sensors 62
3.5 Improved Multiple Model Particle Filter for Visual Target Tracking . . 64
3.5.1 An Introduction of Visual Target Tracking 65
3.5.2 Visual Target Tracking Models 67
3.5.3 New Tracking Algorithm: MMPF-SP 72
3.5.4 Computational Complexity Analysis 74
3.6 Laboratory Experiments and Results 76
3.6.1 Single Model PF (PF-SP-PEKF) 76
3.6.2 MMPF-SP: Case No. 1 79
3.6.3 MMPF-SP: Case No. 2 80
3.6.4 MMPF-SP: Case No. 3 80
3.6.5 Summary 81

4 Improving Particle Filter Using Galerkin's Projection Method 88

4.1 Particle Filtering Based On The Galerkin's Method 89

4.1.1 Generating The Proposal Using Galerkin's Method 89
4.1.2 The Proposed New PF Algorithm 93
4.2 Application: Bearings-only Target Tracking 95
4.3 Application: Visual Target Tracking 99

5 A Particle Filter with State Space Discretization 102
5.1 Generating the Proposal Using State Space Discretization 102
5.1.1 A New PF with State Space Discretization 107
5.2 Application: Bearings-only Target Tracking 108
5.3 Application: Target Tracking with Range and Bearings Measurements . 109
5.3.1 The Experimental Results 114
5.4 Application: Visual Tracking with Multiple Measurements 132
5.4.1 The Multiple Measurement Cues 132
5.4.2 The Tracking Algorithm 135
5.4.3 Experimental Results 136

6 Further Analysis and Other Applications 142
6.1 An Analysis of Particle Filter with MCMC 142
6.1.1 The Metropolis-Hastings Algorithms 142
6.1.2 Numerical Analysis 145
6.2 A Particle Filter For Image Denoising 146
6.2.1 The Denoising Algorithm: A Hybrid PF-DWT Method 146
6.2.2 Wavelet Thresholding for PF-DWT 148
6.2.3 Experimental Results 151

7 Conclusion Remarks and Future Research 154

List of References 164

Appendix A The Proof of the Particle Weight Update Equation 165

Appendix B A Demo Code for Visual Tracking Using Bootstrap Filter 168

LIST OF TABLES

1 Algorithm 1: The Single Model Particle Filter (PF-SP-PEKF) 21

2 The Average RMSE of the Filter Estimations 25

3 A Comparison of the PF-SP-PEKF and the Bootstrap Filter 26

4 Algorithm 2: the Multiple Sensor Particle Filter (CPF-SP) 41

5 A Comparison of the CPF-SP and the Bootstrap Filter 44

6 Algorithm 3: The General Form of A Multi-model Particle Filter 59

7 Algorithm 4: Multiple Model Particle Filter with a State Partition . . 75

8 Algorithm 5: The Galerkin Method Based PF Algorithm 94

9 A Comparison of the Projection-based PF and the Bootstrap Filter . . 96

10 Algorithm 6: A State Space Discretization Based Particle Filter 107

11 Algorithm 7: A State Space Discretization Based MMPF 114

12 Algorithm 8: The General Form of Metropolis-Hastings 143

13 Algorithm 9: Random-Walk Metropolis (RWM) 144

14 Algorithm 10: Metropolized Independence Sampler (MIS) 145

15 The Average MSE of Each Sample Set 146

LIST OF FIGURES

1 Representing a PDF with Particles 7
2 Resample the Particles 12
3 Systematic Resampling 13
4 The Block Diagram of the Single Model PF-SP-PEKF 20
5 The PDFs with Different Gains 23
6 The PDF of the System State 24
7 The PDF of the Measurements 24
8 PF-SP-PEKF (Nonlinear Scalar Case): Estimation Errors 26
9 PF-SP-PEKF (Nonlinear Scalar Case): One Realization 27
10 PF-SP-PEKF Bearings-only Tracking (Case No.1): Estimation Errors . 30
11 PF-SP-PEKF Bearings-only Tracking (Case No.1) 31
12 PF-SP-PEKF Bearings-only Tracking (Case No.2) 32
13 Target Tracking in Sensor Networks 35
14 The Multiple Sensor PF Algorithm 40
15 Sensor Nodes Topology 43
16 Sensor Network Tracking Scenario No.1: Root Mean Square Error . . . 44
17 Sensor Network Tracking Scenario No.1: Location Estimation 45
18 Sensor Network Tracking Scenario No.1: Velocity and Acceleration . . . 45
19 Sensor Network Tracking Scenario No.2: Root Mean Square Error . . . 46
20 Sensor Network Tracking Scenario No.2: Location Estimation 47
21 Sensor Network Tracking Scenario No.2: Velocity and Acceleration . . . 47
22 Sensor Network Tracking Scenario No.3: Location Estimaiton 48
23 Sensor Network Tracking Scenario No.3: Velocity and Acceleration . . . 49
24 Sensor Network Tracking Scenario No.3: Root Mean Square Error . . . 49
25 An Illustration of Model Transition 53
26 The Trajectories of A CT Model with Different Turn Rates 55
27 MMPF Estimations for Tracking A Maneuvering Target 60

28 Model Transition Probability in Maneuvering Tracking Application . . 61
29 A MMPF Using Multiple Sensors 62
30 Bearings-Only Target Tracking Using A MMPF 63
31 The Ellipse Template 69
32 Equally Spaced Rays from the Target Center 70
33 Edge Detection and the Background Clutters 71
34 The Block Diagram for the MMPF-SP 74
35 Visual Tracking Results Using the Condensation Method (Case 1) . . . 77
36 Visual Tracking Results Using the PF-SP-PEKF 78
37 MMPF-SP Tracking (Case No. 1): Overall Location Estimation Errors 82
38 MMPF-SP Tracking (Case No. 1): Overall Velocity Estimation Errors . 82
39 Visual Tracking Results of the Condensation Method (case 1) 83
40 MMPF-SP Visual Tracking (Case No. 1) One Realization 84
41 MMPF-SP (Case 1) One Realization Location Estimation 85
42 MMPF-SP and Condensation Visual Tracking (Case No. 1): Velocities. 85
43 MMPF-SP and Condensation Visual Tracking (Case No. 2) 86
44 MMPF-SP Visual Tracking (Case No. 3) 87
45 Tracking Errors of the Projection Based PF (Case 1) 97
46 The Estimated Trajectory Using the Projection Based PF (Case 1) . . 97
47 The Estimated Trajectory Using the Projection Based PF (Case 2) . . 98
48 Visual Tracking Using the Condensation Method 100
49 Visual Tracking Using the Projection Based PF 101
50 The Target Initial Distribution 104
51 The Target Posterior Distribution 105
52 The Particles Generated from the New Proposal Distribution Based on State Space Discretization 106
53 The Estimation of the PF Based State Space Discretization 108
54 The Estimation Errors of the PF Based State Space Discretization . . . 109
55 Updated Cell Centers Using the Multi-Model Method 112

56 Particles Generated From Weighted Cells 112
57 The Multiple Model Particle Filter Based on State Space Discretization 113
58 MMPF Based State Space Discretization (Scenario No.1 Case A): Estimated Location And Estimation Errors 117
59 MMPF Based State Space Discretization (Scenario No.1 Case A): Estimated Velocity And Estimation Errors 118
60 MMPF Based State Space Discretization (Scenario No.1 Case A): Estimated Acceleration And Estimation Errors 119
61 MMPF Based State Space Discretization (Scenario No.1 Case B): Estimated Location And Estimation Errors 120
62 MMPF Based State Space Discretization (Scenario No.1 Case B): Estimated Velocity And Estimation Errors 121
63 MMPF Based State Space Discretization (Scenario No.1 Case B): Estimated Acceleration And Estimation Errors 122
64 MMPF Based State Space Discretization (Scenario No.2 Case A): Estimated Location And Estimation Errors 123
65 MMPF Based State Space Discretization (Scenario No.2 Case A): Estimated Velocity And Estimation Errors 124
66 MMPF Based State Space Discretization (Scenario No.2 Case A): Estimated Acceleration And Estimation Errors 125
67 MMPF Based State Space Discretization (Scenario No.2 Case B): Estimated Location And Estimation Errors 126
68 MMPF Based State Space Discretization (Scenario No.2 Case B): Estimated Velocity And Estimation Errors 127
69 MMPF Based State Space Discretization (Scenario No.2 Case B): Estimated Acceleration And Estimation Errors 128
70 MMPF Based State Space Discretization (Scenario No.3): Estimated Location And Estimation Errors 129

71 MMPF Based State Space Discretization (Scenario No.3): Estimated Velocity And Estimation Errors 130
72 MMPF Based State Space Discretization (Scenario No.3): Estimated Acceleration And Estimation Errors 131
73 The New MMPF Using Multi-Cues for Visual Tracking 136
74 MMPF Multi-Cue Tracking Scenario No.1 138
75 Bootstrap Filter Tracking For Color Video 139
76 MMPF Multi-Cue Tracking Scenario No.2 140
77 MMPF Multi-Cue Tracking Scenario No.3 141
78 Block Diagram of the Spatial Particle Filter 149
79 Image Denosing Example 1 152
80 Image Denosing Example 2 153

ABSTRACT

The objective of this research is to develop robust and accurate tracking algorithms for various tracking applications. These tracking problems can be formulated as nonlinear filtering problems. The tracking algorithms will be developed based on an emerging promising nonlinear filter technique, known as *sequential importance sampling* (nickname: *particle filtering*). This technique was introduced to the engineering community in the early years of 2000, and it has recently drawn significant attention from engineers and researchers in a wide range of areas. Despite the encouraging results reported in the current literature, there are still many open questions to be answered. For the first time, the major research effort will be focusing on making improvement to the particle filter based tracking algorithm in the following three aspects: (I) refining the particle filtering process by designing better proposal distributions (II) refining the dynamic model by using multiple-model method, (*i.e* using switching dynamics and jump Markov process) and (III) refining system measurements by incorporating a data fusion stage for multiple measurement cues.

CHAPTER 1: INTRODUCTION

Target tracking is a fundamental component of many modern engineering applications [1]-[3]. In this dissertation, the research effort is focused on developing advanced tracking algorithms for various applications by utilizing the emerging nonlinear filtering technique, known as particle filtering. In real-world applications, the tracking problem may have very different forms because of the following two reasons: (1) the specific type of target under track, (2) the type and the number of sensors employed for detection and tracking. In this research, the contributions were made to the following three tracking applications: (1) radar target tracking, (2) target tracking in wireless sensor-networks, and (3) visual target tracking (*i.e.* tracking a target in image sequences using a single camera). Although these applications look quite different, they share some common fundamental theories.

1.1 Radar Target Tracking

The problem of radar target tracking, especially with monopulse techniques, can be divided into two categories: *viz.* active tracking and passive tracking [1] [3]. Active radar target tracking uses both range and bearings measurements. This is the most common type of tracking in real applications. Traditionally, the Kalman filter and its variations have been used for active tracking. However, when the target under track can perform maneuvers, multiple dynamic models will be adopted to describe the target. These models usually involve strong nonlinearities. In these cases, Kalman filter based methods will not provide accurate estimations. Passive tracking is different from active tracking, where only the bearings measurement, *i.e.* the direction of arrival (DOA), is available. Because of this reason, bearings-only tracking is also called *DOA tracking*. The problem of bearings-only tracking arises in a variety of important practical applications in surveillance, guidance or positioning systems [3]. Typical examples are aircraft surveillance (using a radar in a passive mode or an electronic warfare device),

underwater target tracking using passive sonar [3] [5] [6]. The purpose of bearings-only tracking is to estimate the kinematics (the positions and velocities) of a target using its noise-corrupted bearings measurements. Due to the inherent nonlinearities in the observation model, bearings-only target tracking has become a standard nonlinear filtering problem that receives intensive investigations.

1.2 Target Tracking in Sensor Networks

Target tracking using sensor networks is an emerging new tracking application [7] [8] [9]. Sensor networks are typically composed of one or more central information processing units with a large number of sensor nodes, which have limited communication and computation capabilities and limited power supply. During recent years, the continuing miniaturization of computing and (wireless) communication circuitry as well as sensor devices has enabled mass production of intelligent wireless micro-sensors at a low cost. Tracking targets with geographically dispersed and cooperating sensors is attractive for several reasons. First, it can improve the robustness of tracking algorithms; sensors deployed close to targets would result in more reliable signal readings. Also, it can be more cost effective. The challenge is to design a tracking method which ingeniously reconciles the two defining characteristics, abundance in quantity and inferiority in quality, to realize the desired robustness. One important application of wireless sensor networks is target tracking, where the target of interest ranges from moving objects in civil and military surveillance applications, to changes in light, temperature, pressure, and acoustics in environmental monitoring [10]. The type of signals to be sensed is determined based on the types of objects to be tracked. In this dissertation, a multiple sensor fusion and tracking algorithm based on particle filtering was developed for ground vehicle tracking using acoustic sensor networks [11] [12].

1.3 Visual Target Tracking

Visual target tracking is also an important issue in many applications, such as intelligent video surveillance systems [13], human computer interfacing [14], smart

room and teleconferencing [15], and others [16]. The objective of visual target tracking is to estimate the target's position and the velocity in the image plane or develop its trajectory over time by using a combination of the objects appearance and movement characteristics. Visual target tracking remains a challenging research topic not only because of the target dynamics can be highly nonlinear and its distribution can be non-Gaussian, but also the target may have changes in scale (zooming), illumination, pose, and possible occlusion. In this dissertation, several particle filter based visual tracking algorithms were developed by exploring the method of multiple target dynamic models and multiple measurement cues to achieve reliable and accurate estimations, as published in the works of the dissertation's author [17] [18] [19].

1.4 The State Space Approach and State Estimation

In a tracking problem, a target is usually described by a state space model, which contains a state process model and an observation model (or measurement model). The state space approach to time series modeling focuses the attention to the *state vector* of a system. The state vector contains all relevant information required to describe the system. For example, in tracking problems, the system state could be the kinematic characteristics of the target (*i.e.* position, velocity, *etc.*). In the current literature, the target tracking problem is always formulated as a state estimation problem, which is also known as filtering [1] [3]. In state estimation, the Kalman filter is widely used when the model is linear and the additive noise is Gaussian. But the real world dynamic systems are often more complex, typically involving nonlinear and non-Gaussian elements. Estimating the states of such systems is a difficult problem in general, and the optimal solution cannot be expressed in a closed form. Sub-optimal methods are widely used in these situations. In these sub-optimal methods, various approximation schemes have been used, which includes the extended Kalman filter (EKF), Gaussian sum approximations and grid-based method. More recently, a new sequential Monte Carlo method has been introduced [21] [22] [23]. This method is also widely known as particle filtering (PF). The particle filter algorithm is based on the approximation

of successive prediction and filtering density functions by many particles that can be considered as an independent realization from these distributions. The main advantage of this scheme is that it can be applied to nonlinear models with non-Gaussian noise.

1.5 Organization of the Dissertation

This dissertation is focused on developing advanced particle filter tracking algorithms by designing better proposal distributions and utilizing the multiple model method with multiple sensors (or multiple measurement cues). The remainder of this dissertation is organized as follows: Chapter 2 discusses the fundamental theories of particle filtering, problems associated with particle filters and the strategies for improving particle filters. Chapter 3 presents the general framework of an improved particle filter using the state partition technique. Chapter 4 discusses improving the particle filters using Galerkin's method, while Chapter 5 provides an analysis of improving particle filters using the pseudo-grid based method. Further discussions about particle resampling and other applications are provided in Chapter 6.

CHAPTER 2: FUNDAMENTALS OF PARTICLE FILTERING

This chapter discusses the fundamentals of particle filtering, the problems associated with particle filters, and the strategies for improving particle filters.

2.1 Nonlinear Filtering

To lay the theoretical foundations for the analysis in this chapter, the setup of the nonlinear filtering problem is now briefly reviewed. Consider the following general form of a nonlinear system:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{f}(\mathbf{x}_{t-1}) + \mathbf{w}_{t-1}, \\ \mathbf{z}_t &= \mathbf{h}(\mathbf{x}_t) + \mathbf{v}_t, \end{aligned} \tag{1}$$

where $\mathbf{x}_t$ and $\mathbf{z}_t$ denote the states and the measurements of the system at time t, respectively. The function $\mathbf{f}(\cdot)$ denotes the state process or the dynamic model, while $\mathbf{h}(\cdot)$ denotes the measurement process. The variables $\mathbf{w}_t$ and $\mathbf{v}_t$ denote the process and measurement noises, respectively. The system given in equation (1) is usually assumed to be Markovian, nonlinear and non-Gaussian. The state sequence is $\{\mathbf{x}_t;\ t \in \mathbb{T}\}$, which is a hidden Markov process with an initial distribution of $p(\mathbf{x}_0)$ and transition distribution $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. The measurements of this system $\{\mathbf{z}_t; t \in \mathbb{T}\}$ are conditionally dependent on the hidden states. For the notational convenience, let $\mathbf{x}_{0:t} \triangleq \{\mathbf{x}_0,\ \cdots \mathbf{x}_t\}$ and $\mathbf{z}_{1:t} \triangleq \{\mathbf{z}_1,\ \cdots \mathbf{z}_t\}$, where t represents that the system response is up to time t. The goal of the filtering problem is to estimate the following distributions:

$$p(\mathbf{x}_{0:t}|\ \mathbf{z}_{1:t}) \quad \text{or} \quad p(\mathbf{x}_t|\ \mathbf{z}_{1:t}) \tag{2}$$

and the expectation:

$$I(\mathbf{f}) = \mathbb{E}_{p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})}[\mathbf{f}(\mathbf{x}_{0:t})] \triangleq \int \mathbf{f}(\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) d\mathbf{x}_{0:t} \quad . \tag{3}$$

It is assumed that the function $\mathbf{f}(\cdot)$ is integrable with respect to $p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})$. The posterior distribution of this system is given by Bayes' theorem:

$$p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_{1:t}|\mathbf{x}_{0:t})\ p(\mathbf{x}_{0:t})}{\int p(\mathbf{z}_{1:t}|\mathbf{x}_{0:t})\ p(\mathbf{x}_{0:t})\ d\mathbf{x}_{0:t}} \quad . \tag{4}$$

The recursive formula for the joint distribution $p(\mathbf{x}_{0:t+1}|\mathbf{z}_{1:t+1})$ is given in [4] as:

$$p(\mathbf{x}_{0:t+1}|\mathbf{z}_{1:t+1}) = p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})\frac{p(\mathbf{z}_{t+1}|\mathbf{x}_{t+1})p(\mathbf{x}_{t+1}|\mathbf{x}_t)}{p(\mathbf{z}_{t+1}|\mathbf{z}_{1:t})} \quad . \tag{5}$$

From the above equation, the marginal distribution $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ can be evaluated by the following two equations:

The prediction distribution (the Chapman-Kolmogorov equation):

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})\ d\mathbf{x}_{t-1} \tag{6}$$

The updated distribution (the Bayesian equation):

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})}{\int p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})d\mathbf{x}_t} \tag{7}$$

The above expressions are known as the "conceptual solution," and are generally impossible to solve except for some special cases. The problem of nonlinear filtering has been intensively studied. Among the many nonlinear filtering algorithms, the extended Kalman filter (EKF) is the most widely used. Other methods include the Gaussian sum filter, the unscented Kalman filter (UKF) and others. However, these methods require either the linearization of the nonlinear system or the Gaussian assumption, or both. Because of these restrictions, the aforementioned methods are not able to provide accurate estimations for highly nonlinear systems with non-Gaussian noise in general [3].

2.2 Particle Filter

Recently, an alternative nonlinear filtering method, known as *sequential importance sampling* (SIS), has been introduced to provide better estimations with more flexibilities for the nonlinear filtering problem. The rationale behind this method is to approximate the posterior distribution of the system via a set of weighted samples (also called particles) as illustrated in Figure 1. This is how this technique gets its popular nickname "particle filter." Within a sequential framework, a set of particle

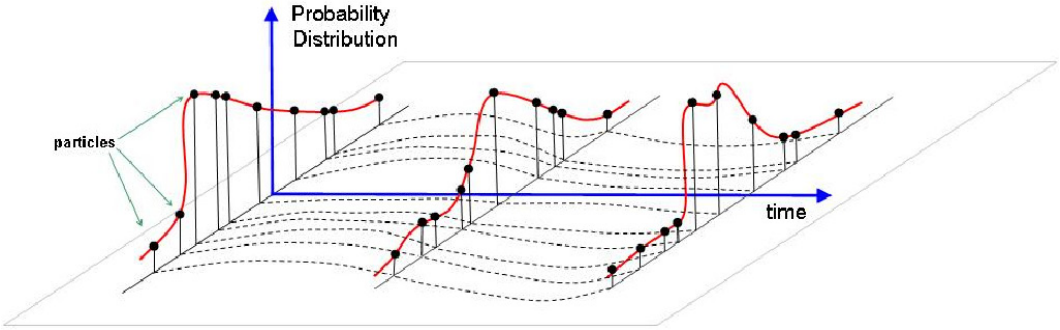


Figure 1: *Representing a PDF using weighted particles.*

are initially drawn from an arbitrary distribution $q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})$, known as the *proposal distribution*, which has the support of the true posterior distribution $p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})$. The particles are then propagated through the system, and the corresponding weights are calculated whenever the most recent measurements are received. More specifically, if our purpose is to estimate $I(\mathbf{f})$ given in equation (3), then we can rewrite the equation into the following form:

$$I(\mathbf{f}) = \frac{\int \mathbf{f}(\mathbf{x}_{0:t})\omega(\mathbf{x}_{0:t})q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})d\mathbf{x}_{0:t}}{\int \omega(\mathbf{x}_{0:t})q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})d\mathbf{x}_{0:t}} \quad , \tag{8}$$

where $\omega(\mathbf{x}_{0:t})$ is known as the *importance weight* or the *particle weight* given by:

$$\omega(\mathbf{x}_{0:t}) = \frac{p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})} \quad . \tag{9}$$

This weight is considered as the ratio of the *true posterior* to the *proposal*. Next, if we can draw N_s i.i.d particles $\{\mathbf{x}^i_{0:t}; i = 1, \cdots, N_s\}$ according to $q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})$, then the

estimate of the integral is:

$$\hat{I}_{N_s}(\mathbf{f}) = \frac{\frac{1}{N_s}\sum_{i=1}^{N_s}\mathbf{f}(\mathbf{x}_{0:t}^{(i)})\omega(\mathbf{x}_{0:t}^{(i)})}{\frac{1}{N_s}\sum_{j=1}^{N_s}\omega(\mathbf{x}_{0:t}^{(i)})} = \sum_{i=1}^{N_s}\mathbf{f}(\mathbf{x}_{0:t}^{(i)})\tilde{\omega}_t^{(i)} \quad , \tag{10}$$

$\tilde{\omega}_t^{(i)}$ is the *normalized importance weight*,

$$\tilde{\omega}_t^{(i)} = \frac{\omega(\mathbf{x}_{0:t}^{(i)})}{\sum_{j=1}^{N_s}\omega(\mathbf{x}_{0:t}^{j})} \quad . \tag{11}$$

In order to construct a sequential estimation method, proposal distributions of the following form can be used:

$$q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) = q(\mathbf{x}_{0:t-1}|\mathbf{z}_{1:t-1})q(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{z}_{1:t}) \quad . \tag{12}$$

This equation allows us to evaluate the importance weight recursively in time:

$$\tilde{\omega}_t^{(i)} \propto \tilde{\omega}_{t-1}^{(i)}\frac{p(\mathbf{z}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)},\mathbf{z}_{1:t})} \quad . \tag{13}$$

The first practical implementation of particle filter was reported in 1993 [24]. At that time, Gordon solved the degeneracy problem. This made the sequential Monte-Carlo method possible to run on modern computers of that time. After this, particle filtering was relatively undiscovered in the engineering community until 2002. Since then it has drawn increasing attention due to its superior performance in various nonlinear/non-Gaussian estimation problems, such as radar target tracking, robot localization and mapping, communications, visual target tracking, and others [3] [4] [23] .

2.3 Improving Particle Filters

Particle filters provide a flexible framework for nonlinear filtering in general. However, to design a robust and accurate particle filter for real applications, several practical implementation issues have to be carefully studied. In this section we will discuss

the problems associated with implementing particle filters and different strategies for improving particle filters.

2.3.1 Designing Better Proposal Distributions

A major problem of designing a particle filter is to choose a good proposal distribution [23]. Due to the fact that the particles are drawn from this distribution, and the particle weights are also related to this distribution, the performance of a particle filter is strongly influenced by the choice of the proposal distribution. A standard scheme is to choose the state transition prior $p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})$ as the proposal distribution. This algorithm is known as the bootstrap filter or the condensation method [3]. The advantage of this method is that it's efficient to implement. However, this proposal (the transition prior) simply relies on one-step ahead predictions, and does not take into account of the current measurements. When the likelihood distribution is narrow with respect to the transition prior distribution, many particles will receive negligible weights, which means an abundance of computation will be wasted. Designing a good proposal is a challenging task, and it's also the major topic of this research. A good proposal needs to have enough support from the posterior distribution of the true state and has to be easy to sample from.

An optimal proposal distribution was proposed in [21], where optimal is in the sense of minimizing the conditional variance of the importance weights defined as $\omega_t^{*i} = \frac{p(\mathbf{x}_k^i|\mathbf{z}_{1:t})}{q(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i,\mathbf{z}_t)}$. However, as indicated in [21][23], the optimal proposal distribution is only useful for two special nonlinear systems, *i.e.* the discrete state space systems and the Jump-Markov linear systems. This is because the optimal proposal distribution is required to sample from the distribution $q(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i, \mathbf{z}_t)$ and integrate over the new state, where it's impossible for most nonlinear systems. To design better proposal distributions for particle filters which are applicable to more applications, the local linearization technique was first introduced [21]. This strategy can be divided into two categories: First, linearization of the state space model. In this method, an extended Kalman filter (EKF) is used to generate the proposal distribution. Then, particles

are drawn from a normal distribution in which the mean and variance are provided by the EKF. The proposal distribution obtained by using an EKF includes the most recent measurement information, which shows the advantage over the transition prior proposal. Nevertheless, the linearization stage introduces modeling errors, which can yield large estimation errors if the system is highly nonlinear. The second method is to linearize the optimal proposal, which will also result in a Gaussian proposal distribution. The unscented Kalman filter (UKF) was proposed to replace the EKF to generate the proposal. This type of particle filter is known as the unscented particle filter (UPF) [25]. The basic idea is to use multiple deterministically generated sigma points to approximate the systems state, then prorogate these points through the actual nonlinear system. An UKF does not require the linearization of the state space model, but it requires the Gaussian assumption. As reported in [25], the UPF can achieve better performance when compared to the bootstrap filter and the PF with EKF proposals. Other improvements for designing a proposal also include the auxiliary particle filter [26], the gradient proposal [27] and others.

A major task of this research is to design better proposals for various particle filter based target tracking algorithms. Three new proposal distributions have been developed, which are (1) the technique of state-partition with parallel EKF bank, (2) Galerkin's method and (3) the state space discretization method. These methods have been applied to different tracking applications and scenarios. It has been demonstrated through extensive experiments that better results can be achieved by using these new methods, which will be thoroughly discussed in Chapter 3, 4 and Chapter 5, respectively.

2.3.2 Resampling and Markov Chain Monte Carlo (MCMC)

Another problem associated with a particle filter is the so called "degeneracy problem," which means the variance of the importance weights increases over time [4] [22] [23]. This implies that after a few iterations, all but one weight will converge to zero. In this case, a resampling scheme is introduced to avoid the degeneracy problem. The effective sample size N_{eff} was introduced in [28] as the measure of degeneracy, and it was defined as follows:

$$N_{\text{eff}} = \frac{N_s}{1 + Var(\omega_t^{*i})} \tag{14}$$

where N_s is the sample size and $\omega_t^{*i} = p(\mathbf{x}_k^i|\mathbf{z}_{1:t})/q(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i, \mathbf{z}_t)$. As indicated in [23], N_{eff} usually cannot be exactly calculated. In this case, the estimated effective sample size $\widehat{N_{\text{eff}}}$ can be used, which is defined as: $\widehat{N_{\text{eff}}} = 1/\sum_{i=1}^{N_s}(\omega_t^i)^2$. To mitigate the particle degeneracy problem, a resampling stage is always adopted in the practical implementation of particle filters. Particle resampling is based on the idea of discarding the particles with small weights and focusing on the particles with more significant weights. In other words, the resampling stage of a particle filter is a process that generates a set of uniformly weighted particles from a set of weighted particles, where these two sets of particles have the same discrete PDF. This idea is illustrated in Figure 2. In this figure, the blue circles on the top symbolize the resampled particles at previous time index. The same size of these particles represents that they are equally weighted. After the particles are propagated through the system, their weights will be evaluated. The particles with large weights are represented as large circles (*i.e.* the red circle has the largest weight). After resampling, the particles with large weights will generate multiple children particles. For example, in this figure the particle with the largest weight will have four equally weighted children particles.

There have been many resampling methods reported in current literature. The most basic approach among them is the *sampling-importance resampling and multinomial resampling* [25], where it is required to construct the discrete cumulative distribution function (CDF). Then, the uniformly drawn sampling indices are projected onto the distribution domain. The intersection with the distribution domain gives the resampled

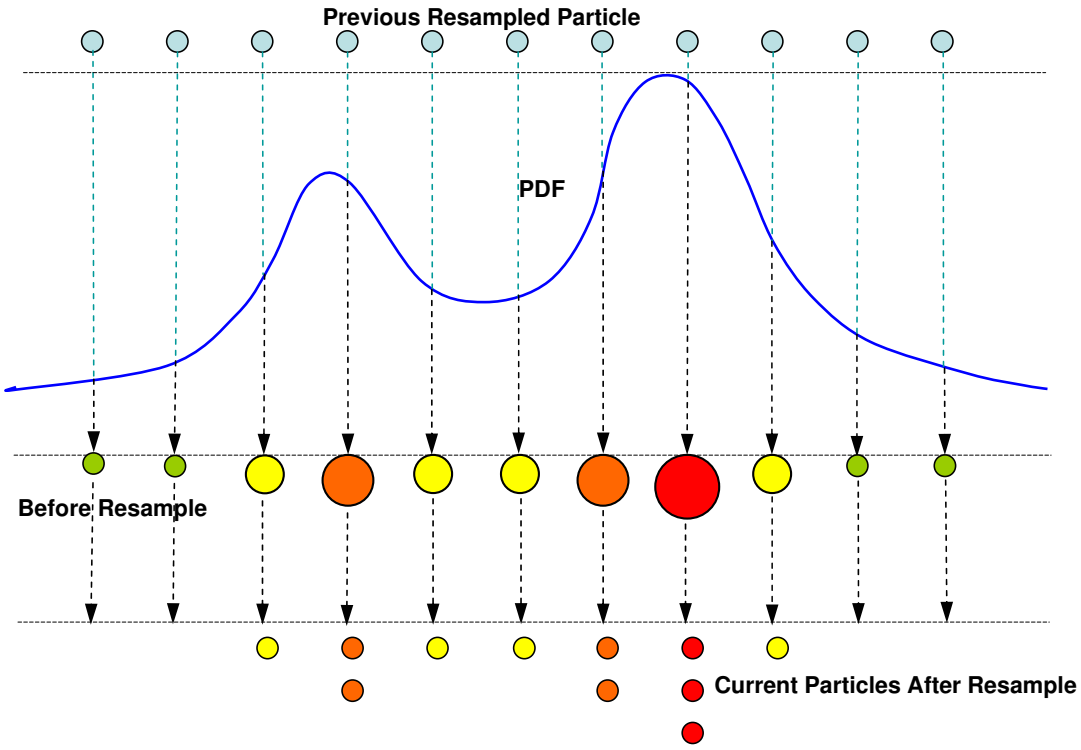


Figure 2: *Resample the particles*

index. This method has $\mathcal{O}(N_s)$ of operations. Another resampling method is called *residual resampling*, which was introduced to improve the resampling step by reducing the particle weight variance [28]. Residual resampling involves two stages: first, set the number of each resampled particle using $N_i = \lfloor N\omega_t^i \rfloor$, second, for the remaining $N_r = N_s - \sum_{i=1}^{N_s} N_i$ particles, use the SIR algorithm. The residual resampling can provide a better sample population and requires less computation compared to SIR. Another resampling method is called "systematic resampling" [23]. This method is the most widely used resampling algorithm because of its computational simplicity and good empirical performances. In addition, it can also provide minimum Monte Carlo

variations. The basic idea of systematic resampling is to draw a set of samples from $[0,\ 1]$ with each has a distance of ${N_s}^{-1}$ apart. Then project these samples to the CDF. The resampled index is given by the number of intersections inside each step of the CDF. This method is illustrated in Figure 3.

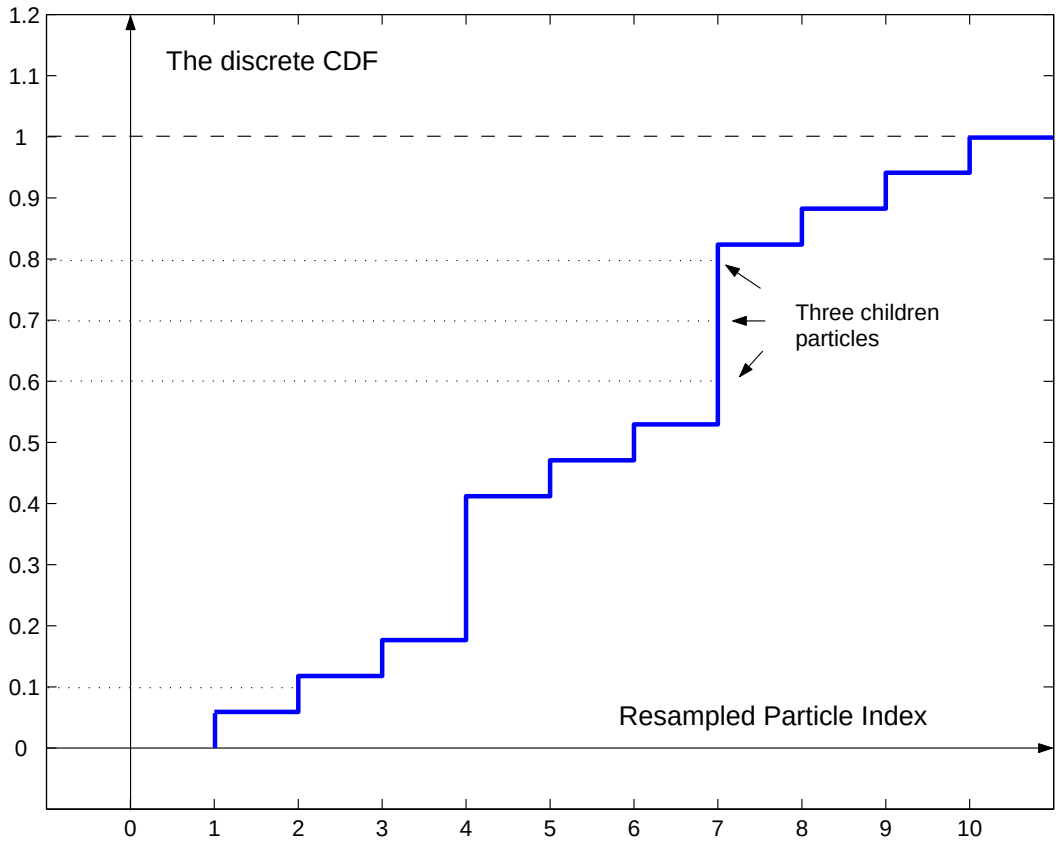


Figure 3: *An illustration of the systematic resampling algorithm.*

In the resampling stage, some new particles (the children particles) will be repeated copies of a previous particle (the parent particle). The worst-case scenario is that virtually all new particles were repeated copies of a single parent particle with significant weight. This leads to a loss of particle diversity, which is often called "sample impoverishment." This phenomenon will be aggravated when the process noise variance is small. A strategy to mitigate this situation is to introduce the Markov Chain Monte Carlo (MCMC) steps after resampling. The motivation is that if the particles are distributed according to the posterior $p(\tilde{x}_{0:t}|z_{1:t})$, after applying a Markov chain transition kernel $\mathcal{K}(x_{0:t}|\tilde{x}_{0:t})$, the resultant set of particles are still distributed according to the

same posterior distribution. But the new particles will be distinct with with respect to each other. The Metropolis-Hastings (M-H) method is a major group of algorithms belonging to the class of Markov Chain Monte Carlo (MCMC), and the performance of particle filters with MCMC is closely related to the choice of the specific M-H algorithm. Chapter 6 provides a thorough analysis of using various M-H algorithms for particle filters, which is also reported in our previous work [29].

2.3.3 The Method of Multiple Dynamic Models

For most of the nonlinear filtering problems, estimations can also be improved by refining the dynamic model that approximates the true system and improves the measurement model to provide more reliable measurements. In target tracking problems, multiple dynamic models have been used to describe the targets which have complex motions. More specifically, in these cases, the targets are assumed to operate according to one model from a finite set of models at each sampling rate. This set of models captures all the possible dynamics of the target. The switching process from one model to another is governed by a model transition probability matrix. The multiple model particle filter (MMPF) is a significant extension of the single model particle filter, and it's proven to be a powerful tracking framework for many applications. The multiple model particle filter is also a research focus of this dissertation, and will be discussed in Chapter 3 and Chapter 5, where two improved MMPF algorithms have been successfully developed and applied to various tracking scenarios.

2.3.4 Using Multiple Measurement Cues or Multiple Sensors

It has been reported that the filtering performance can also be improved by incorporating multiple measurement cues [16]. One important advantage of particle filtering is that it allows the information from different measurement sources to be fused in a principled manner without increasing the dimension of the state vector. This is especially important for visual target tracking, since a single imaging sensor can provide different measurement cues, *e.g.* edge information, color information and others. These

independent measurements will greatly improve the estimation performance. In this case, special signal processing is needed for extracting these features. For radar and acoustic sensor network applications, multiple sensors will also help to improve estimation accuracy. The last research topic of this dissertation is to develop tracking algorithms which can synergistically fuse multiple measurement cues or measurements from multiple sensors for various tracking applications. In Chapter 3 and Chapter 5, it is shown that with the newly designed proposal distribution with multiple measurement cues, the improved MMPF can provide greatly improved estimation accuracy and robustness for complicated tracking problems.

CHAPTER 3: PARTICLE FILTER WITH THE STATE PARTITION TECHNIQUE AND PARALLEL EKFS

The sequential importance sampling (SIS) technique, with the nickname of particle filtering (PF), has emerged as a promising filtering technique for nonlinear and non-Gaussian systems in which state estimation is the ultimate objective. In a particle filter framework, the accuracy of the estimation depends on the choice of the proposal distribution. As indicated in [3] [22][23] and our analysis in Chapter 2, using the state transition prior as the proposal (also known as bootstrap filter) cannot produce precise estimates for systems which have strong nonlinearities and with severe noise corruption, since this proposal does not include the most recent measurements of the system. In this chapter, a novel particle filtering algorithm is developed based on state partitioning and a bank of extended Kalman filters to render a more accurate proposal distribution and hence yield a more precise estimation of the state. Moreover, because of the improved proposal distribution, the new filter can achieve a given level of performance using fewer samples than its conventional SIS counterparts. Our results show that this new approach yields significantly improved estimates of the state. We name this algorithm as a particle filter with state partition and parallel EKFs (PF-SP-PEFK) [30].

3.1 The State Partition Technique

In this section, we will first discuss the method of state partition and parallel EKFs (SP-PEKF), and then we will show how it can be incorporated within the particle filter framework. Next, we will provide an example, which the PF-SP-PEKF algorithm is applied to a nonlinear scalar system.

3.1.1 The Method of SP-PEKF

For notational convenience, we denote each component of the state vector $\mathbf{x}_t$ as $x(t)$, and use the subscript to represent the filter channel. The rationale of SP-PEKF is as

follows: for each component $x(t)$ of $\mathbf{x}_t$, we generate a set of samples $x_i(t)$, $i \in [1, N]$ denotes the *i-th* filter, according to the given initial distribution $\mathcal{N}(x_i(0), R(0))$. The variable N denotes the total number of filter banks. These samples are partitioned as $x_i(t) \triangleq x_{n_i}(t) + x_{r_i}(t)$, where $x_{n_i}(t)$ and $x_{r_i}(t)$ denote the nominal and residual parts of the true state, respectively. After partitioning, the samples are propagated through a bank of parallel filters and the estimated states are calculated using the weighted sum of filtered samples. Initially, samples $x_i(0)$ are generated according to

$$x(0) = x_{n_i}(0) + x_{r_i}(0) \tag{15}$$

$$\hat{x}(0) = \hat{x}_{n_i}(0) + \hat{x}_{r_i}(0) \tag{16}$$

$$R(0) = R_{n_i}(0) + R_{r_i}(0) \quad . \tag{17}$$

and

$$\begin{aligned} \hat{x}_{n_i}(0) &= \hat{x}(0) & R_{n_i}(0) &= R(0) \\ \hat{x}_{r_i}(0) &= 0 & R_{r_i}(0) &= 0 \quad . \end{aligned}$$

The scalar variables R_{n_i} and R_{r_i} represent the covariance associated with the nominal state and the residual state, respectively. In the case when $\hat{x}(0)$ and $R(0)$ are known, the nominal state propagates through the system as follows:

$$x_{n_i}(t) = f(x_{n_i}(t-1)) + \mathrm{w}_{n_i}(t-1) \tag{18}$$

where $\mathrm{w}_{n_i}(t)$ has the same distribution as $\mathrm{w}(t)$, which is the process noise. But $\mathrm{w}_{n_i}(t)$ is a different realization. Following that, the system is linearized about $x_{n_i}(t)$ as:

$$\begin{aligned} x(t) \approx & f(x_{n_i}(t-1)) + \mathrm{F}(x_{n_i}(t-1)) \cdot [x(t-1) - x_{n_i}(t-1)] + \mathrm{w}(t-1) \\ = & f(x_{n_i}(t-1)) + \mathrm{F}(x_{n_i}(t-1)) \cdot x_{r_i}(t-1) + \mathrm{w}(t-1) \end{aligned}$$

where F is the Jacobian of the state process given as:

$$\mathrm{F}(x_{n_i}(t)) = \left.\frac{\partial(f(x))}{\partial x}\right|_{x=x_{n_i}(t)} \quad . \tag{19}$$

The above equation can be simplified as:

$$x_{r_i}(t) \approx \mathrm{F}(x_{n_i}(t-1)) \cdot x_{r_i}(t-1) + \mathrm{w}(t-1) - \mathrm{w}_{n_i}(t-1) \ , \tag{20}$$

Furthermore, the following equation is constructed in a similar way:

$$z(t) \approx h(x_{n_i}(t)) + \mathrm{H}(x_{n_i}(t)) \cdot x_{r_i}(t) + \mathrm{v}(t) \tag{21}$$

where

$$\mathrm{H}(x_{n_i}(t)) = \left.\frac{\partial(h(x))}{\partial x}\right|_{x=x_{n_i}(t)} \quad . \tag{22}$$

Equations (20) and (21) form an approximate dynamic model to the nonlinear system given by (1). This approximate model is linear to the residual part of the state $x_{r_i}(t)$. A bank of EKFs is then applied to update $x_{r_i}(t)$ as follows. For further readings about SP-PEKF filters, see [31] [32] [33].

To begin, the one step ahead prediction of the residual state $\hat{x}_{r_i}(t|t-1)$ and the filtered estimate of $\hat{x}_{r_i}(t|t)$ are given by:

$$\hat{x}_{r_i}(t|t-1) = \mathrm{F}(x_{n_i}(t-1)) \cdot \hat{x}_{r_i}(t-1|t-1) - \mathrm{w}_{n_i}(t-1) \tag{23}$$

and

$$\hat{x}_{r_i}(t|t) = \hat{x}_{r_i}(t|t-1) + G_i(t) \cdot \hat{z}_i(t|t-1) \ , \tag{24}$$

where $G_i(t)$ is the *i-th* filter gain given as:

$$G_i(t) = R_i(t|t-1)\mathrm{H}(x_{n_i}(t))^{\mathrm{T}} R_{z_i}(t|t-1)^{-1} \quad . \tag{25}$$

The residual state prediction covariance and its update covariance are:

$$R_{r_i}(t|t-1) = \mathrm{F}(x_{n_i}(t-1))R_i(t-1|t-1)\mathrm{F}(x_{n_i}(t-1))^{\mathrm{T}} + \sigma_{\mathrm{w}}^2 \tag{26}$$

and

$$R_{r_i}(t|t) = \left[\mathrm{I} - G_i(t)\mathrm{H}(x_{n_i}(t))\right] R_i(t|t-1) \quad .$$

The pseudo-innovation process and its covariance are:

$$\hat{z}_i(t|t-1) \approx z(t) - h(\hat{x}_i(t|t-1)) \tag{27}$$

$$R_{z_i}(t|t-1) = \mathrm{H}(x_{n_i}(t))R_i(t|t-1) \times \mathrm{H}(x_{n_i}(t))^{\mathrm{T}} + \sigma_{\mathrm{v}}^2 \quad .$$

The variables σ_{w}^2 and σ_{v}^2 denote the variance of the process noise and measurement noise, respectively. The estimation of the state at time t is given by the following weighted sum:

$$\hat{x}_{total}(t|t) = \sum_{i=1}^{\mathrm{N}} \hat{x}_i(t|t) \cdot \psi_i(t) \tag{28}$$

where $\hat{x}_i(t|t) = x_{n_i}(t) + \hat{x}_{r_i}(t|t)$. The weight for each filter is given as:

$$\psi_i(t) = \frac{L_i(t|t) \cdot \psi_i(t-1)}{\sum_{i=1}^{\mathrm{N}} L_i(t|t) \cdot \psi_i(t-1)} \quad , \tag{29}$$

where

$$L_i(t|t) = |R_{z_i}(t|t-1)|^{-0.5} \exp\left[-\frac{1}{2} \left\|\hat{z}_i(t|t-1)\right\|^2 \cdot R_{z_i}^{-1}(t|t-1)\right].$$

Finally, the overall estimation error covariance R_{total} is:

$$R_{total}(t|t) = \sum_{i=1}^{N} \left\{ R_i(t|t) + \left[\hat{x}(t|t) - \hat{x}_i(t|t)\right] \times \left[\hat{x}(t|t) - \hat{x}_i(t|t)\right]^{\mathrm{T}} \right\} \cdot \psi_i(t) \quad . \tag{30}$$

The above filtering method is based on the idea of using a bank of EKFs to generate multiple state trajectories to simulate the true state trajectory. The weighted sum

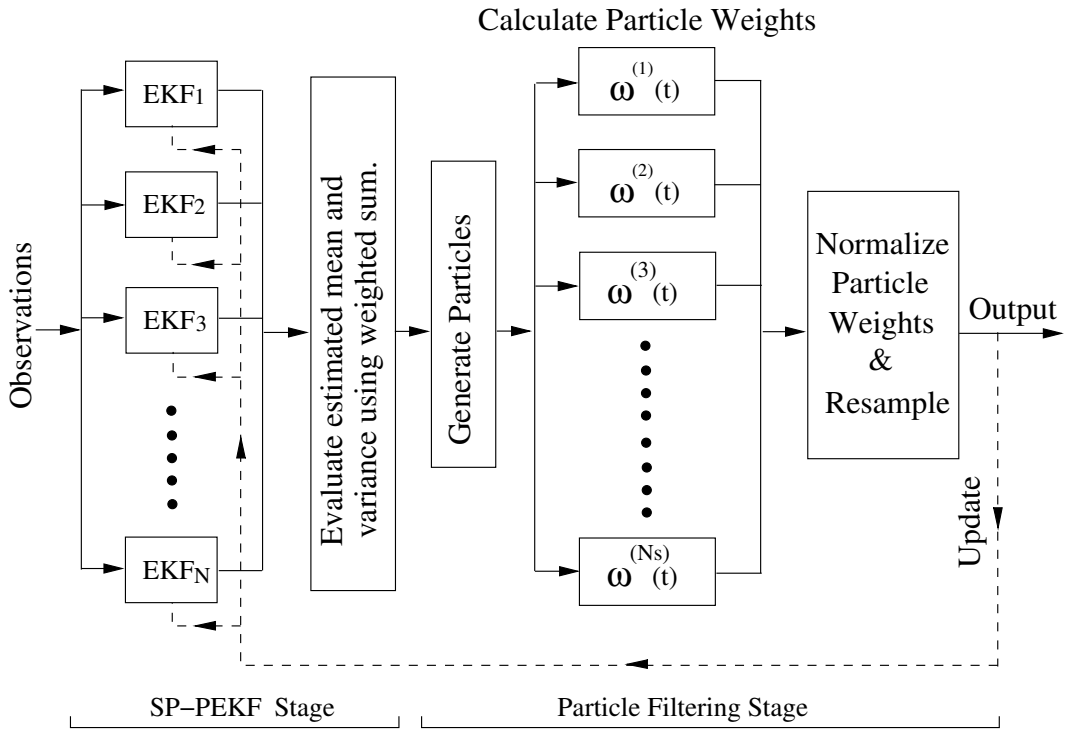


Figure 4: *The block diagram of the single model PF-SP-PEKF algorithm.*

gives the estimation of the system state, and the weights are adaptively updated as time evolves.

3.1.2 The New Particle Filter Algorithm (PF-SP-PEKF)

The SP-PEKF filter uses multiple trajectories to simulate the true state, which can provide more accurate estimates of a nonlinear system than the EKF, which uses only one trajectory. However, the SP-PEKF still rests on the assumption that the state distribution is Gaussian; on the other hand, the PF algorithms can accommodate arbitrary distributions, but an appropriate proposal is needed. To take advantage of the strength of both PF and SP-PEKF, we suggest a new particle filter algorithm (PF-SP-PEKF) which uses the SP-PEKF filter as its proposal. More specifically, in each iteration of the PF-SP-PEKF, the particles are drawn from a normal distribution with a mean and variance given by equations (28) and (30). Then, these particles are feed into the PF algorithm and propagated according to equations (11) and (13) with a resampling step. The new algorithm is illustrated in Figure 4 and summarized in Table 1.

Table 1: Algorithm 1: The Single Model Particle Filter (PF-SP-PEKF)

- Sequential Importance Sampling (SIS) Step:

  ◇ Sample from the proposal distribution according to:

  $$x^{(i)}(t) \sim q\left(x^{(i)}(t)|x^{(i)}(0:t-1\ )\ , z(1:t)\right),$$
  $$= \mathcal{N}\big(\ \hat{x}_{total}(t|t),\ R_{total}(t|t)\ \big)$$

  ◇ Evaluate the particle weight using the weight update equation given in (13), then normalize the particle weights ;

- Resampling Step:

  ◇ Generate a new set of particles $x^{i\star}(t)$ from $x^{(i)}(t)$ by sampling N_s times the approximate distribution of $p(x(t)|y(1:t))$ so that

  $$Pr\left(x^{i\star}(t) = x^{(j)}(t)\right) = \tilde{\omega}^{(j)}(t);$$

- Output and Update Step:

  ◇ Estimate $x(t)$ according to $\hat{x}_{PF}(t) \approx \frac{1}{N_s}\sum_{i=1}^{N_s} x^{i\star}(t)$

  ◇ Update the proposal.

Two additional remarks are as follows: First, in order to take the advantage of the estimates derived from the particle filter, the final estimate is recursively feed back into the SP-PEKF filter at the end of each iteration and repartitioned as: $\hat{x}^{\star}_{r_i}(t|t) = \hat{x}_{PF}(t) - x_{n_i}(t)$, where $\hat{x}^{\star}_{r_i}(t|t)$ will be used to replace $\hat{x}_{r_i}(t|t)$ as the residual state at time t. This re-partition step serves as a "correction" step for the SP-PEKF filter at each interaction. Second, as indicated in [31], a memoryless scheme can produce better estimates for highly nonlinear systems. More explicitly, in the memoryless scheme, the filter weight does not depend on its previous value, *i.e.* $\psi_i(t-1)$ in equation (29) is omitted. Thus, it is assumed that all of the filters are equally weighted at the previous step. The bank of weighted EKFs (SP-PEKF) are only used to generate the proposal distribution from where the particles are drawn. This proposal includes the most recent observation $y(t)$, which gives more support from the true posterior distribution. The

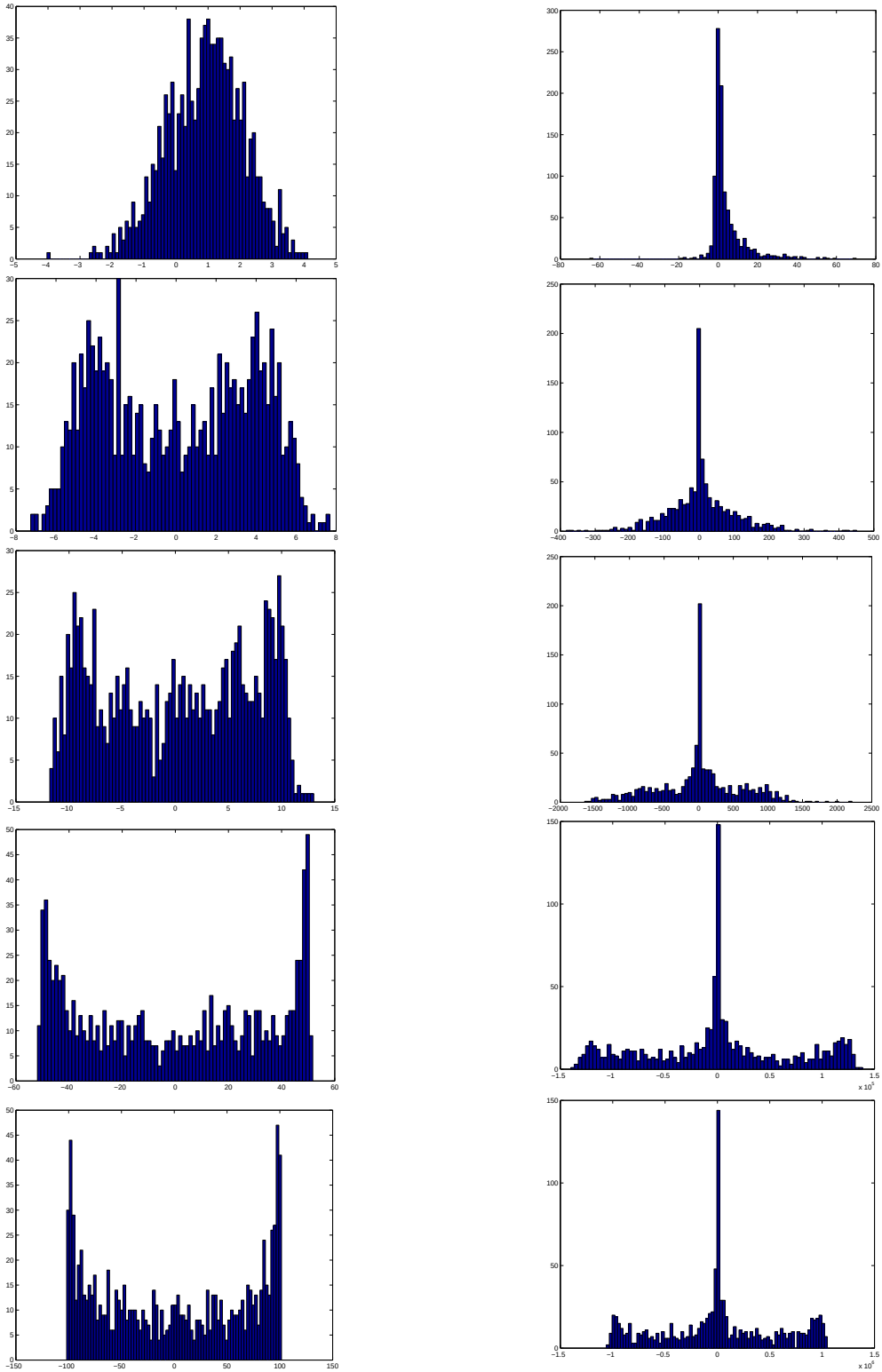
PS-SP-PEKF algorithm differs from the interacting multiple model extended Kalman filter (IMM-EKF) method in the sense that PS-SP-PEKF applies multiple EKFs to a single dynamic model to generate a proposal distribution. However in the IMM-EKF, a system is assumed to operate according to multiple dynamic models, and each EKF is applied to a different model. In addition, we typically choose the number of the filter bank channels N in equations (28)-(30) much smaller than the number of particles N_s; our experience is that taking $10 \leq N \leq 20$ is generally sufficient to obtain significantly improved performance compared to bootstrap filters (PF taking transition prior as its proposal).

3.1.3 Example: A Nonlinear Scalar Case

In this section, we provide one simple example to illustrate the superior performance of the new designed particle filter (PF-SP-PEKF). Here we apply the new PF to a nonlinear scalar system given in equations (31) and (32), which is similar to the example given in [31] except for more nonlinearities are incorporated to yield a bimodal PDF. The variable $x(t)$ is the state of the nonlinear system at time t, and $z(t)$ denotes the system measurement. The time index is incremented as $t = 1, \cdots, 50$. The variables $\mathrm{w}(t)$ and $\mathrm{v}(t)$ represent the process noise and the measurement noise, respectively. And they are distributed according to $\mathcal{N}(0, 1)$. In this system, the gain of the sinusoidal term in equation (31) has an important impact on both the state PDF and the measurement PDF, which is shown in Figure 5. Figures 6 and 7 illustrate the PDFs evolve with time as the gain is equal to 5.0. It is clear from these figures that the system is highly nonlinear and non-Gaussian in nature.

$$x(t+1) = 1.7 \exp\left(-2x^2(t)\right) + 5.0 \sin(x(t)) + \mathrm{w}(t) \tag{31}$$

$$z(t+1) = x^3(t+1) + \mathrm{v}(t+1) \tag{32}$$



The PDF of the system state $x(t)$ *The PDF of the system measurement $y(t)$*

Figure 5: *This figure illustrates the effects of different sinusoidal gains on the PDF of both the state and the measurement. From the top row, gain is given as 1, 5, 10, 50 and 100, respectively. As seen, with the increase of the gain of the sinusoidal term, the PDFs become more complex.*

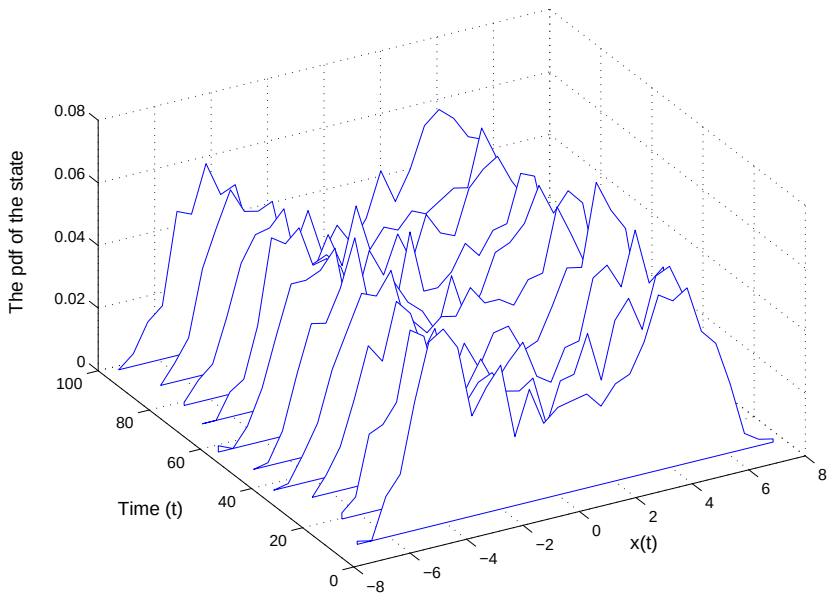


Figure 6: *The pdf of the state evolving with time as the gain is 5.*

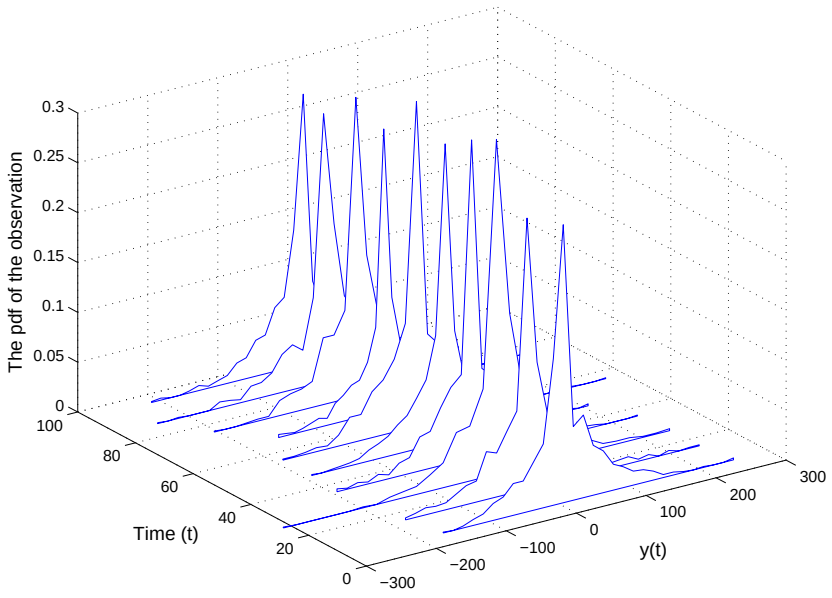


Figure 7: *The pdf of the measurements evolving with time as the gain is 5.*

Three different filters are implemented: (a) EKF, (b) bootstrap filter, (c) PF-SP-PEKF. Moreover, the systematic resampling scheme discussed in [23] was used since it is efficient to implement and minimizes the MC variation. In addition, for the PF-SP-PEKF algorithm, the memoryless scheme is applied, and 20 sub-filters are used, *i.e.* $N = 20$. Considering the ensemble average, one thousand Monte Carlo runs were implemented to produce the statistical performance, the root mean square error (RMSE), which is shown in Figure 8. In addition, the mean and variance of the estimation RMSE are summarized in Table 2. Moreover, Figure 9 illustrates the true states vs. the estimates from one realization of the simulation. As indicated in Figure 8 and Table 2, the PF-SP-PEKF algorithm gives fairly accurate estimations compared to that of the bootstrap filter and the EKF. The improvement of PF-SP-PEKF over EKF in terms of the mean RMSE is about 78%. Furthermore, PF-SP-PEKF also has the minimum variance of the RMSE.

Table 2: The Average RMSE of the Filter Estimations

Filters	Mean RMSE	Var of RMSE	Time (s)
EKF	2.9232	0.3175	0.0067
Bootstrap	1.5580	0.0558	0.7967
PF-SP-PEKF	0.6442	0.0247	1.5217

Table 3 provides a comparison of the PF-SP-PEKF and the bootstrap filter in terms of sample size and run time. As indicated in this simulation, even using 2000 particles, the bootstrap filter still cannot achieve the same estimation accuracy as that of the PF-SP-PEKF with 200 particles. However, the bootstrap with 2000 particles takes significantly longer time compared to the PF-SP-PEKF with 200 particles, which clearly demonstrates the strength of the PF-SP-PEKF.

Table 3: A Comparison of the PF-SP-PEKF and the Bootstrap Filter

Filter (Sample Size)	RMSE		Time (s)
	Mean	Var	
Bootstrap (200)	1.5581	0.0558	0.7967
Bootstrap (1000)	1.3339	0.0485	4.1495
Bootstrap (2000)	1.1803	0.0344	7.3011
PF-SP-PEKF (200)	0.6442	0.0247	1.5217

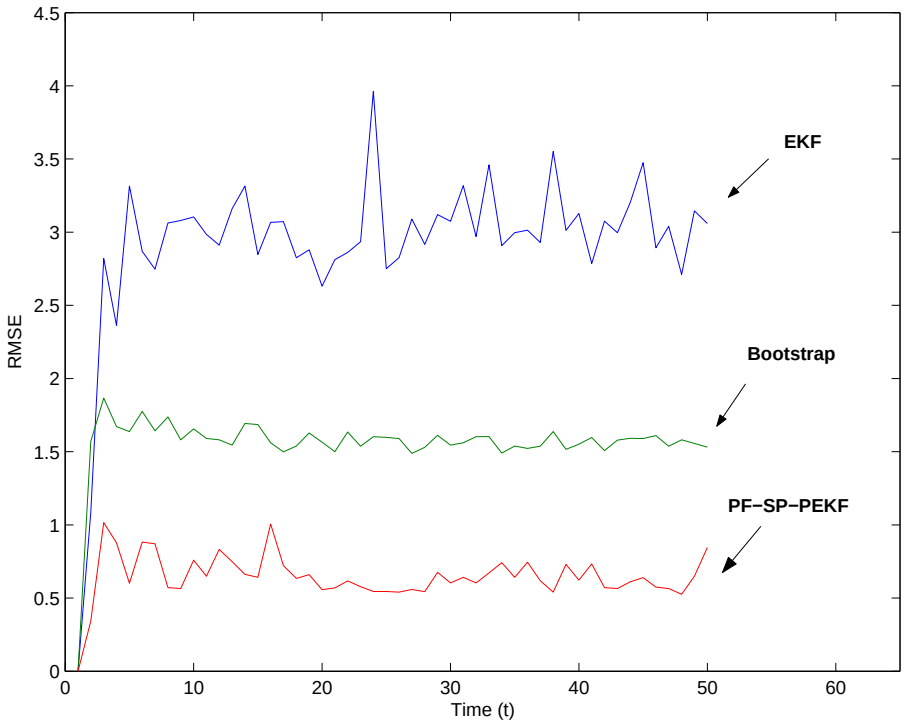


Figure 8: *The RMSE generated from the ensemble average of the different filtering algorithms: EKF, bootstrap and PF-SP-PEKF.*

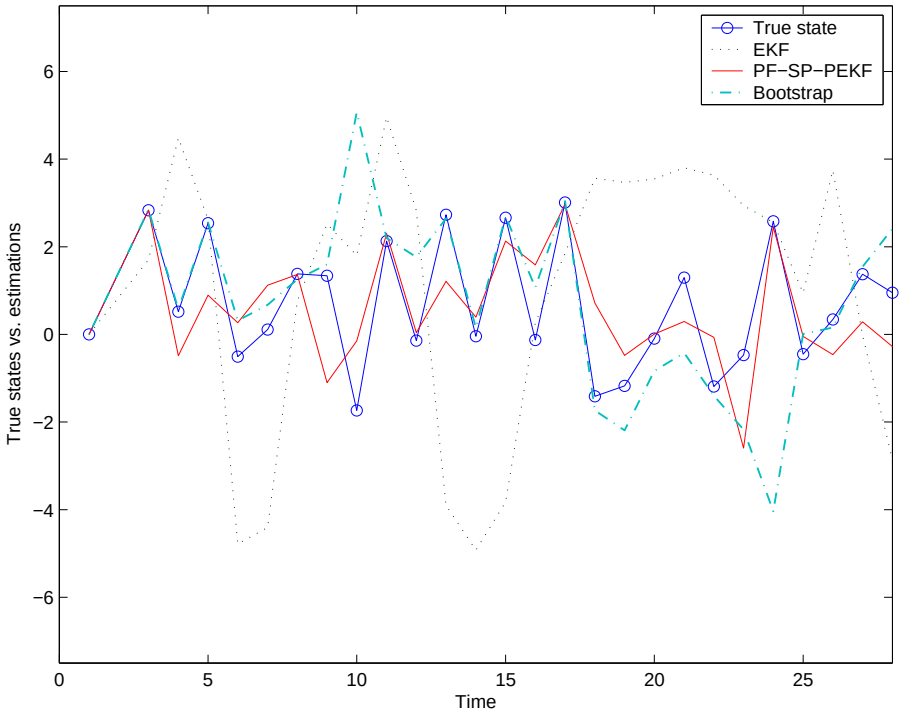


Figure 9: *The true states vs. the estimated states of one realization from the different filtering algorithms: EKF, bootstrap and PF-SP-PEKF.*

3.2 PF-SP-PEKF for Bearings-only Target Tracking

Next, the PF-SP-PEKF algorithm is applied to a radar target tracking problem. Target tracking is a fundamental component of surveillance, guidance or positioning systems, whose objective is to determine the number, position, and velocity of a moving target. This section focuses on tracking a non-maneuvering target using bearings-only measurement. The problem of bearings-only tracking is also known as direction of the arrival (DOA) tracking, whose objective is to estimate the kinematics (the positions and velocities) of a target using its noise-corrupted measurements. Bearings-only tracking has many practical applications, such as passive sonar applications and aircraft surveillance [3]. Nonlinearities in the measurement equation preclude the use of conventional linear analysis, although pseudo-linear formulations are possible. The DOA sensor, which is also known as ownship, could be stationery or mobile. In this section we assume it is stationery. In the case of a single-sensor, the bearings measurements are extracted from only one sensor, which makes the tracking problem ill-posed in the general case of arbitrary target motion. Indeed, unique solutions exist only for non-maneuvering targets. Despite the aforementioned difficulties, numerous techniques have been developed for bearings-only target tracking, which include EKF [34], modified polar coordinate EKF [3], bootstrap filter [24] and others. Recently, this challenging problem has become a standard nonlinear problem that many people have investigated [31] [35] [36] .

Here, we analyze the state space model for the bearings-only tracking problem. Assume the target is a non-maneuvering target (maneuvering target tracking will be analyzed later), then target can be represented by a vector given as:

$$\mathbf{x}_t = \begin{bmatrix} x(t) & y(t) & v_x(t) & v_y(t) \end{bmatrix}^T \quad , \tag{33}$$

The variables $(x(t), y(t))$ describe the target location in a Cartesian coordinate system. The variables $v_x(t)$ and $v_y(t)$ denote the target velocities along the x-axis and y-axis, respectively. The discrete time model for the kinematics of a non-maneuvering target

is given as [3]:

$$\mathbf{x}_t = \mathbf{F} \cdot \mathbf{x}_{t-1} + \Gamma \cdot \mathbf{m}_{t-1} \quad , \tag{34}$$

where

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \Gamma = \begin{bmatrix} T^2/2 & 0 \\ 0 & T^2/2 \\ T & 0 \\ 0 & T \end{bmatrix}$$

where T is the sampling rate. The process noise $\mathbf{m}_t$ is a 2-by-1 noise vector with a distribution $\mathbf{m}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_m)$, where

$$\mathbf{Q}_m = \begin{bmatrix} \sigma_m^2 & 0 \\ 0 & \sigma_m^2 \end{bmatrix}$$

This model is called the constant velocity model, where the target has a rectilinear motion. The measurement of the tracking system is the angle between the observation platform (the coordinate origin) and the location of the target referenced to the y-axis, which is given by:

$$z(k) = \arctan\left(\frac{x(t)}{y(t)}\right) + n(t) \quad , \tag{35}$$

where the scalar variable $n(t)$ denotes the measurement noise which is distributed as $n \sim \mathcal{N}(0, \sigma_n^2)$. Equations (34) and (35) constitute the state space model for the bearings-only tracking problem, in which the observation equation contains nonlinearities. In addition, the velocities $v_x(t)$ and $v_y(t)$ are the hidden states of the system, which do not have direct measurements. Three different filters are applied to this nonlinear tracking model, which are an extended Kalman filter, a bootstrap filter (PF using transition prior as its proposal) and the proposed new particle filter algorithm (PF-SP-PEKF). The initial conditions of this simulation are given as

$$\mathbf{x}_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T$$

and

$$\mathbf{Q}_m = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix} \quad , \qquad \sigma_n^2 = 0.01 \quad .$$

The time index is incremented as $t = 1, \cdots, 100$. To evaluate the the tracking accuracy, we define the distance between the estimated location and the true location as the estimation error, which is given by

$$e(t) = \sqrt{[x(t) - \hat{x}(t)]^2 + [y(t) - \hat{y}(t)]^2} \quad . \tag{36}$$

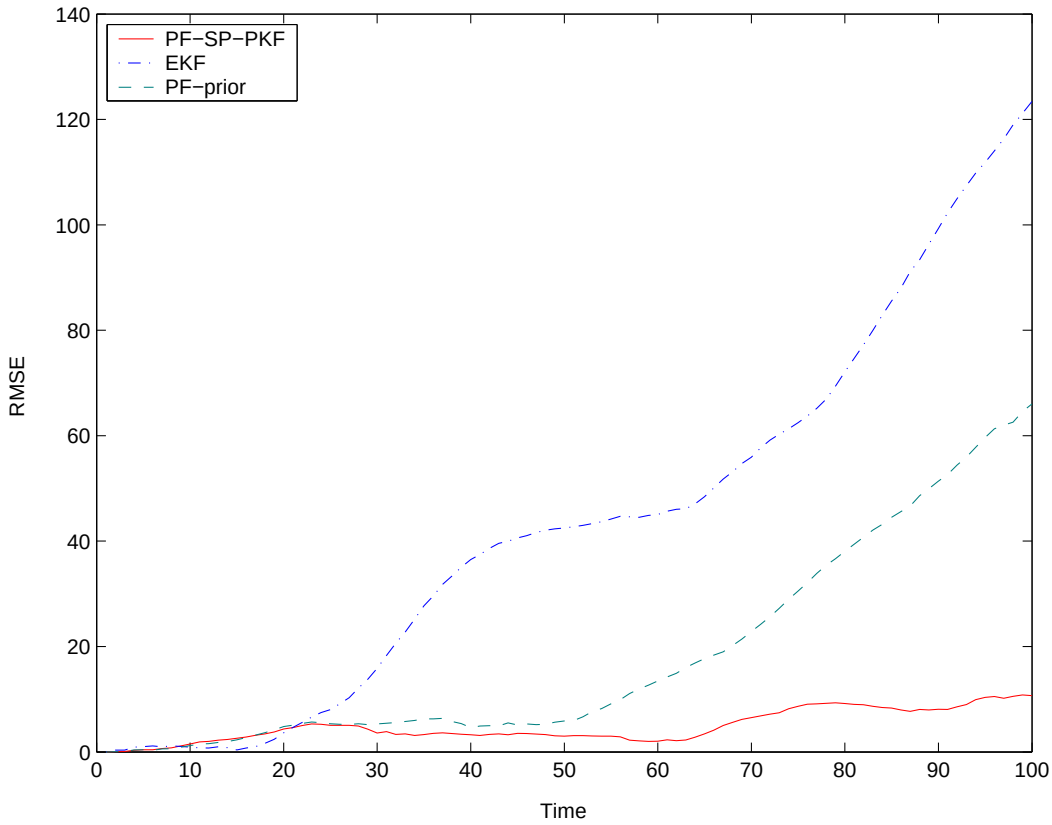


Figure 10: *The estimation RMSE of 100 MC runs: The dash-dot line ($-\cdot$) represents the RMSE of EKF; the dash line ($--$) represents the RMSE of PF-prior; the dotted line ($\cdots$) represents the the RMSE of PF-SP-PEKF. As seen, the our proposed PF-SP-PEKF method has the least error.*

Then, one hundred Monte Carlo runs of the tracking simulation were implemented, and the RMSE of the estimation errors was evaluated as the performance index. Figure 10 shows the estimation RMSE, in which it is indicated that the extended Kalman filter loses the target after $t = 30$, and the estimation error increased dramatically. In addition, although the bootstrap filter keeps the tracking, the estimation error is large and non-convergent. However, the proposed new PF algorithm (PF-SP-PEKF) can

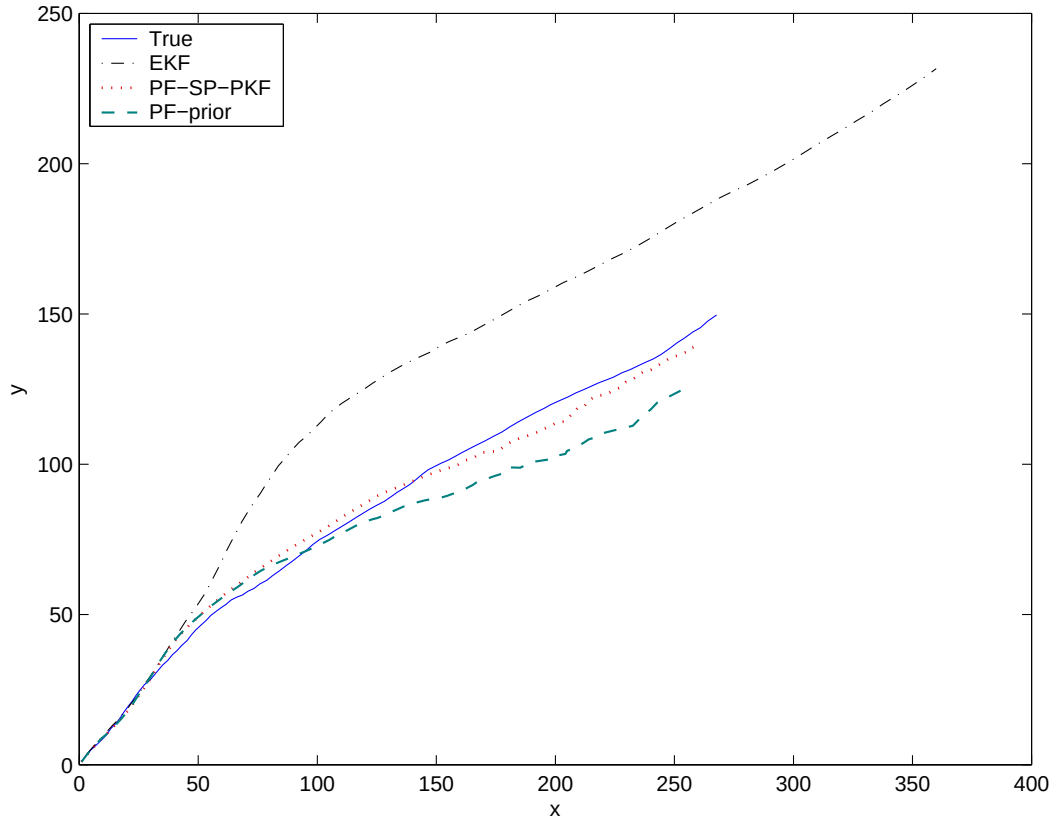


Figure 11: *One realization of targeting trajectories in the Cartesian coordinates: The solid line represents the true trajectory (–); the dash-dot line (–·) represents the estimates of EKF; the dash line (– –) represents the estimates of PF-prior; the dotted line (· · ·) represents the the estimates of PF-SP-PEKF.*

follow the target throughout the whole simulation with small estimation error. Figure 11 shows one realization of the tracking simulation, which is typical in the Monte Carlo runs. It is clear from this figure that the EKF loses the target, and the tracking error subsequently becomes huge. On the contrary, both the bootstrap filter and the PF-SP-PEKF maintains close tracking of the moving target. However, the PF-SP-PEKF gives much more accurate estimations of the target positions. Next, we present another example of bearings-only tracking using the PF-SP-PEKF. The data in this experiment shows the air-traffic at a major airport (Oklahoma City International Airport), as measured by the National Weather Radar Testbed (NWRT) in Norman, Oklahoma. This data was taken on September of 2005. Here, detections were made within a 90 degree sector, looking towards the airport, and the origin establishes the location of

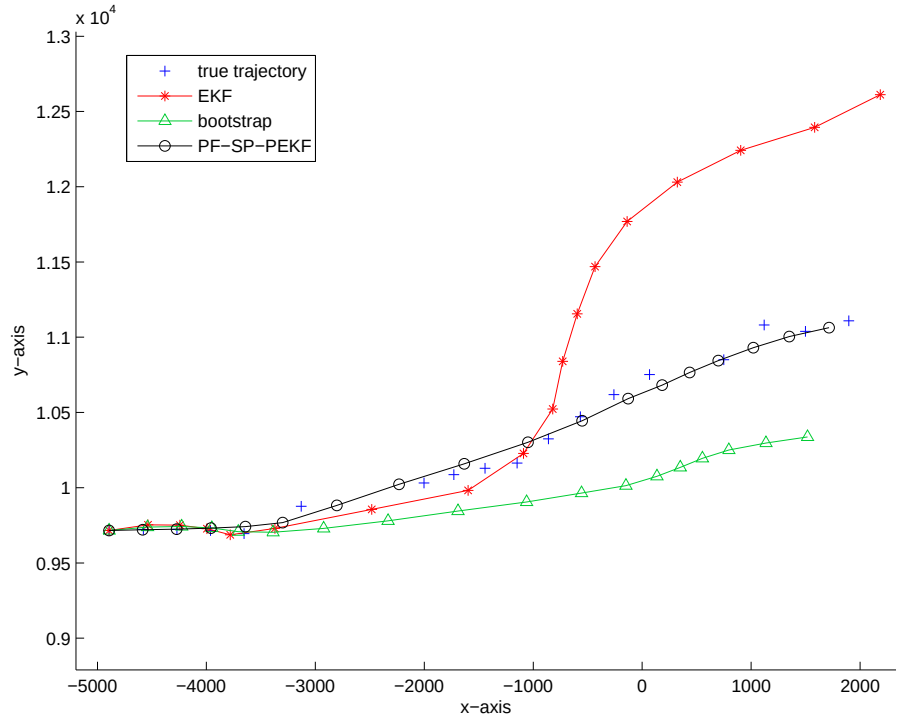


Figure 12: *The plus marks indicate the true locations. The line with the circles indicates our estimates with the new method. The line with the triangles denotes the output of the traditional bootstrap filter. Finally, the line with the stars indicates the estimates with the extended Kalman filter. As the results indicate, the new approach yields very close estimates of the true states, while the others weakly follow or lose track.*

the radar. We apply three tracking algorithms to a specific path. These algorithms are an extended Kalman filter (EKF), a bootstrap filter, and the PF-SP-PEKF. Figure 12 depicts the experimental results of one typical realization of the tracking algorithm. As indicated in the figure, our new algorithm gives better estimates of the target locations, as compared to the EKF and the bootstrap filter. Also, this example shows that a small motion perturbation can cause the EKF estimates to diverge from the true trajectory, although the target has a rectilinear motion in general. This experimental result is important because tracking based on single DOA measurements has traditionally been a difficult task.

3.3 Particle Filter for Target Tracking in Sensor Networks

In this section, we address the problem of target tracking in collaborative acoustic sensor networks. To cope with the inherent characteristics and constraints of wireless sensor networks, we present a novel target tracking algorithm with power aware concerns. The underlying tracking methodology is described as a multiple sensor tracking and fusion technique based on particle filtering (PF). As discussed in the most recent literature, particle filtering is defined as an emerging Monte-Carlo state estimation technique with proven superior performance in many target tracking applications. More specifically, in the proposed method, each activated sensor transmits the acoustic intensity and the direction of arrival (DOA) of the target to the sensor fusion center (a dedicated computing and storage platform, such as a micro-server). In addition, each sensor node uses its DOA to generate a set of estimations based on the state partition technique as described earlier. Meanwhile, a set of sensor weights are calculated at the fusion center based on the acoustic intensity transmitted from each of the activated sensors. Next, the weighted sum of the estimates are used to generate the proposal distribution for the particle filter implemented at sensor fusion center. We name this algorithm as the centralized particle filter with state partition (CPF-SP). This technique renders a more accurate proposal distribution, hence yields more precise and robust estimations of the target with using fewer samples than that of the traditional bootstrap filter. In addition, since the majority of the signal processing resides efficiently on the fusion center, thus the computation load at the sensor nodes is limited, which is desirable for power aware systems. Finally, the performance of the new tracking algorithm in various tracking scenarios is thoroughly studied and compared to standard tracking methods. As shown in the theory and demonstrated by our experimental results, the CPF-SP reliably outperforms traditional methods in all experiments.

3.3.1 Introduction of Sensor Networks

During recent years, with the dramatic advances in the design of micro-electro mechanical systems (MEMS) and ad-hoc networking protocols, there has been an emerging

trend towards the use of sophisticated wireless networks of unattended sensor devices for a variety of new monitoring and control applications [7][8][9]. Sensor networks provide the monitoring for the physical world via many distributed wireless sensing devices deployed in the region of interest. Each of these low-cost tiny devices, also known as a sensor node, has limited processing and communication capabilities with a limited power supply. The sensing information is processed collaboratively between the sensor nodes and at sensor fusion center (*e.g.* micro-server), which has significantly greater bandwidth, computation and energy capabilities compared to sensor nodes. Locating and tracking moving stimuli or targets is a primary task for a sensor network in many practical applications, such as robot navigation, security surveillance and battlefield awareness [7]. In this section, we consider tracking a ground vehicle in a wireless acoustic sensor network using the DOAs as the measurements. Similar research can be found in many works, including [39] [40] [44]. The sensor network studied here consists of a large number of sensor nodes and one mobile sensor fusion center, which has the access to the information gathered by the sensor nodes and is installed on a monitoring vehicle. The tracking scenario is illustrated in Figure 13. For tracking targets in a sensor network, it is important to design a sequential method that can dynamically fuse the information sent by multiple sensors without requiring significant processing capabilities at the sensor nodes and excessive communication within the network. In general, the selection of data routing scheme and communication protocol is out of the scope of this study.

This section is based our previous work [11][12] and is organized as follows: in sub-section 3.3.2, we briefly review the work of target tracking in sensor network. Sub-section 3.3.3 analyzes the mathematical model for target tracking in an acoustic sensor network. The new tracking algorithm is formulated in sub-section 3.3.4. Finally, simulation results and a summary are provided in sub-section 3.3.5.

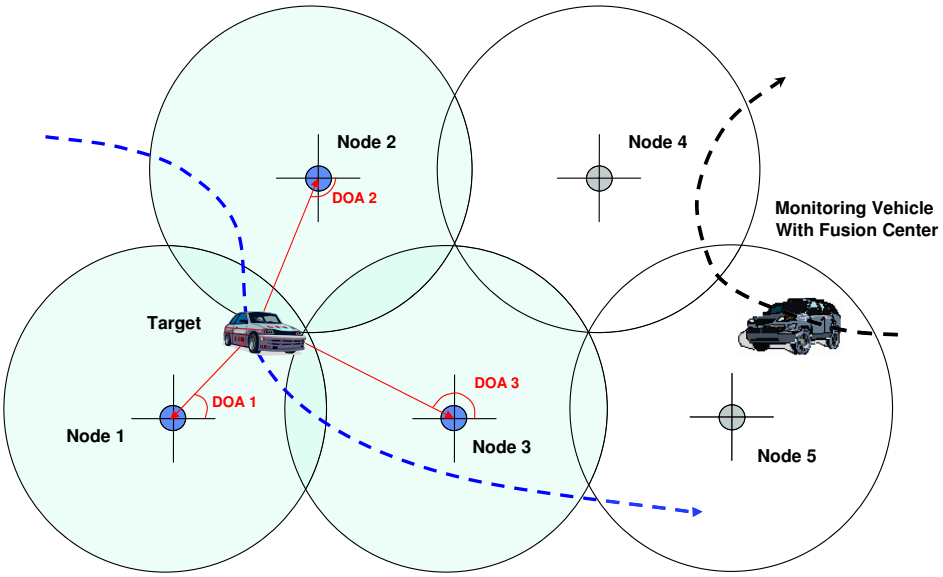


Figure 13: *Target tracking in a wireless sensor network: Sensor nodes 1-3 are in the active mode, and sensor nodes 4 and 5 are in the sleep mode. The fusion center is installed on the monitoring vehicle, which has the access to all sensor nodes.*

3.3.2 Related Work in Sensor Network Target Tracking

The earliest work of target tracking using multiple sensors involves various Kalman filter based methods, such as the extended Kalman filter (EKF), the interacting multiple model IMM-EKF, and the probability data association (PDA) methods [37]. Later, the hidden Markov model (HMM) based methods were studied. In [38], an IMM tracking algorithm was developed based on a discretized target state space; meanwhile, the "theory of evidence" was used in [38] to construct the observation model. In addition, a maximum likelihood source localization method based on acoustic sensor readings was also proposed for target tracking in sensor networks [39]. But as indicated in [40], this method is sensitive to the parameter perturbation and the computational complexity is high for multi-target location estimation. Recently proposed methods also include the "dynamic convoy tree" target tracking [41] as well as others. More recently, particle filter based methods are receiving increased attention. For example, a PF tracking method was developed in [40], in which a simple form of particle filtering

(known as the bootstrap filter) was used and its particle weights were calculated based on the acoustic energy readings. However, it is well known that a bootstrap filter takes the state transition prior as the proposal distribution, which does not take into account of the current measurements – it simply relies on one-step ahead predictions. When the likelihood distribution is narrow with respect to the transition prior distribution, many particles will receive negligible weights, which means an abundance of computation will be wasted. In [6], a new PF tracking algorithm was proposed with an improved Gaussian proposal and a more complex measurement model.

In general, PF tracking algorithms for sensor networks can be categorized as *centralized* or *decentralized*. The aforementioned works [6][40] are examples of centralized versions. Centralized particle filters (CPF) implies that most signal processing tasks are carried out at the fusion center. On the other hand, various de-centralized (or distributed) particle filter (DPF) tracking algorithms are discussed in the current literature. In [42], a DPF algorithm is proposed, but this algorithm demands significant communication in the sensor network to update the complete particle set and also involves complicated learning procedures. In [44], the sensors are divided into disjoint "cliques," and the particle filters associated with each clique run in parallel. Following this, a Gaussian mixture model is used to approximate the whole particle set. Although this strategy does not require significant communications between sensor nodes, it requires computationally intensive filtering algorithms to be implemented at the sensor node, whose computation capability and power supply are very limited in many applications. Finally, the work in [44] depends on many sensors to be turned on to achieve their desired performance. In general, many tracking methods discussed here either require the sensor nodes to have substantial signal processing capabilities or require a more complex sensor network architecture. For example, the recent work of [41] requires an elaborate multi-node structure that includes multiple "lead nodes" to track the target. To rectify these problems, this section suggests a new particle filter tracking algorithm which is based on the state partition technique and parallel EKFs (SP-PEKFs). As demonstrated in our simulation results, the new algorithm

renders considerably accurate tracking results while requiring only a small amount of communication protocol in the network and limited processing tasks at each sensor node, which in turn reduces the power consumption.

3.3.3 Problem Formulation

Here we describe the tracking problem in two steps. The acoustic sensor model is first introduced, then the dynamic model of the moving target is discussed.

Acoustic Wave Intensity Decay Model

Here, a distributed wireless acoustic sensor network is one which is composed of deterministically or randomly deployed sensors whose positions are known to the fusion center, and this fusion center can be either stationary or mobile. The sensor locations could be computed by using methods proposed in [45]-[48] or using an on board GPS system. An acoustic sensor is capable of detecting the DOA of a perambulating (or moving) target when the target comes into the neighborhood (*i.e.* effective range) of this sensor. A sensor has limited computing capabilities to calculate the received acoustic intensity, and it also serves as a transmitter/receiver. We assume the sensor nodes are dense enough such that during each sampling period there is at least one sensor is in the active mode. Assuming that the acoustic source can be treated as a point target and sound propagation is isotropic, the acoustic intensity received by sensor nodes can be modeled as follows [7] [49] [50]:

$$\mathrm{P}(t) = \frac{\mathrm{S}(t)}{\|\mathrm{x}(t) - r(t)\|^{\frac{\alpha}{2}}} + \varepsilon(t) \quad , \tag{37}$$

where S(t) denotes the acoustic intensity emitted from the source (target) located at x(t), P(t) represents the acoustic intensity received by a sensor located at $r(t)$. The variable α is an attenuation coefficient, and $\varepsilon(t)$ is the additive noise assumed to have a zero-mean Gaussian distribution. In the sensor network, the sensors are designed to switch to their active mode when the received acoustic intensity is above a certain threshold. Once a sensor is in the active mode, the acoustic intensity received by the

sensor will be calculated and sent to the fusion center. At the fusion center, a set of sensor weights are calculated as $\mathcal{W}_k(t) = \mathrm{P}_k(t) / \left(\sum_{k=1}^{K} \mathrm{P}_k(t) \right)$, where $\mathcal{W}_k(t)$ denotes the sensor weight for sensor k, where K is the total number of the active sensors, and finally where $\mathrm{P}_k(t)$ is the acoustic intensity received by sensor k.

Target Dynamic Model

In this section, we focus on wireless acoustic sensor networks, where the sensor node will only provide the bearings measurement (*i.e.* DOA), and there is no range measurement available. As the target moves through the network field, sensors which are located in the neighborhood of the target will be turned on and will send out the DOA of the target to the sensor fusion center. For each single activated sensor, this is a bearings-only tracking problem, where the target is represented by the vector given as:

$$\mathbf{x}_t = \left[\, x(t) \;\; v_x(t) \;\; y(t) \;\; v_y(t) \, \right]^T \quad , \tag{38}$$

where $(x(t), y(t))$ is the target location in a *local* Cartesian coordinate system which takes the sensor location as its origin. The variables $v_x(t)$ and $v_y(t)$ denote the target velocities along the x-axis and y-axis in the sensor local coordinate system, respectively. The state space model is given by equations (34) and (35), in which the observation equation contains nonlinearities. An interesting feature of this problem is the system will have multiple DOA measurements at each sampling period, and the total number of measurements that are available to the fusion center is not fixed. Also, these measurements are not equally "reliable" because of the sensor noise and clutter.

3.3.4 Power-Aware Particle Filter For Target Tracking in Sensor Networks

The New Scheme for Tracking in Sensor Networks

As indicated in Section 3.3.1, the sensor nodes in the network have both limited computation and communication capabilities. In addition, it is difficult or even impossible to replenish the power supply frequently in many sensor networks. All of

these constraints pose new challenges for the target tracking problem. On the other hand, particle filtering has emerged to be a promising tracking algorithm in various applications. The design of the proper proposal distribution is crucial to implement a particle filter. This distribution governs the weights of the particles that approximate the posterior distribution. In light of this, we propose a novel particle filtering algorithm with its proposal generated from a weighted sum of estimates calculated by using each sensor measurement. More specifically, at the fusion center, the DOAs of each activated sensor are used to formulate a set of individual trackers. These trackers can generate fast but perhaps not very accurate estimates. However, these estimates should remain in the neighborhood of the true target location. In addition, a set of sensor weights $\mathcal{W}_k(t)$ are calculated by using the acoustic intensity received by each active sensor node. Then, a weighted sum is calculated based on the these estimates and the sensor weights. Finally, by taking the weighted sum of the estimates as the proposal distribution, a particle filter is applied for target sensor fusion. Here in this section, we analyze the use of the state partition technique with a bank of EKFs (SP-PEKFs) to generate individual trackers using each DOA, and discuss how the particle filter framework can be applied to achieve the task of sensor fusion and target tracking.

Particle Filter (PF) Based Sensor Network Tracking

Recently, numerous particle filter algorithms have been developed for sensor network applications [6] [39] [40] [44] which provide promising results. In this section, we will present a new multiple sensor particle filter tracking algorithm (CPF-SP) which is based on the state partition technique and parallel EKFs as we discussed earlier. More specifically, an independent tracker (*i.e.* a SP-PEKF tracker) is implemented at each active sensor node, which returns the local DOA information, the received acoustic intensity and estimated target distribution, *i.e.* the mean and variance given in equations (28) and (30), to the fusion center. The fusion center will evaluate the sensor weights based on the received acoustic intensities and the total number of active sensors. Then the fusion center will construct a weighted Gaussian sum or a Gaussian mixture as

the proposal distribution. Next, a particle filter is implemented at the fusion center, in which the particles are drawn from the aforementioned proposal distribution. The particle weights will be calculated based on the multiple DOA measurements. Finally, the fusion center will send the terminal estimates to the active sensors to update the next iteration. Our new multiple sensor CPF-SP algorithm is illustrated in Figure 14 and is summarized in Table 4.

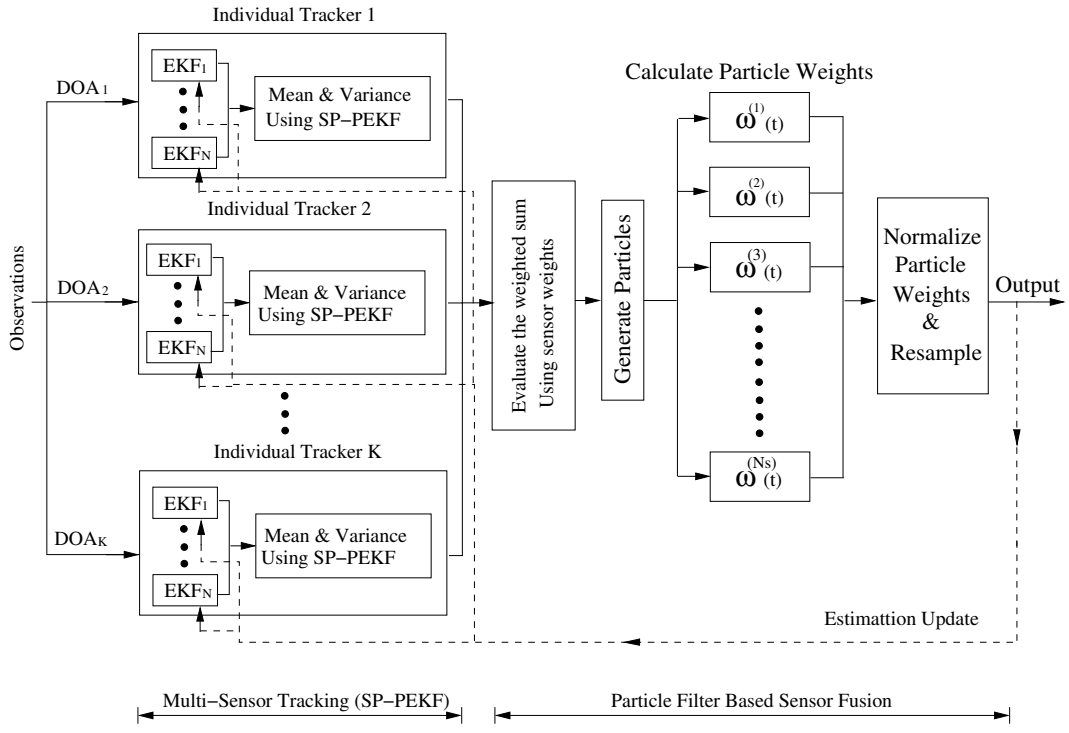


Figure 14: *The PF-based sensor tracking/fusion with power-aware design.*

Table 4: Algorithm 2: the Multiple Sensor Particle Filter (CPF-SP)

- At each active sensor node:

  ◇ Implement an individual SP-PEKF algorithm using the local DOA to provide a normal distribution $\mathcal{N}(\hat{\mathbf{x}}_k(t), \hat{\mathbf{R}}_k(t))$

  ◇ Calculate the received acoustic intensity $\mathrm{P}_k(t)$

  ◇ Sent local DOA, local estimates and acoustic intensity to the fusion center

- At the fusion center:

  ◇ Calculate the sensor weights based on the number of active sensors and the received acoustic intensities

  ◇ Construct a weighted Gaussian sum or a Gaussian mixture based on the sensor weights

  ◇ Sample from the weighted Gaussian sum

$$\begin{aligned} \mathbf{x}^{(i)}(t) &\sim \sum_{k=1}^{K} \mathcal{W}_k(t) \cdot \mathcal{N}\left(\hat{\mathbf{x}}_k(t|t), \hat{\mathbf{R}}_k(t|t)\right) \\ &= \mathcal{N}\big(\hat{\mathbf{x}}_{all}(t|t), \hat{\mathbf{R}}_{all}(t|t)\big) \end{aligned}$$

  ◇ Calculate and normalize the particle weights according to equation (13)

  ◇ Resampling: generate a new set of particles $\mathbf{x}^{i\star}(t)$ from $\mathbf{x}^{(i)}(t)$ by sampling N_s times the proposal distribution so that

$$Pr\left(\mathbf{x}^{i\star}(t) = \mathbf{x}^{(j)}(t)\right) = \tilde{\omega}^{(j)}(t);$$

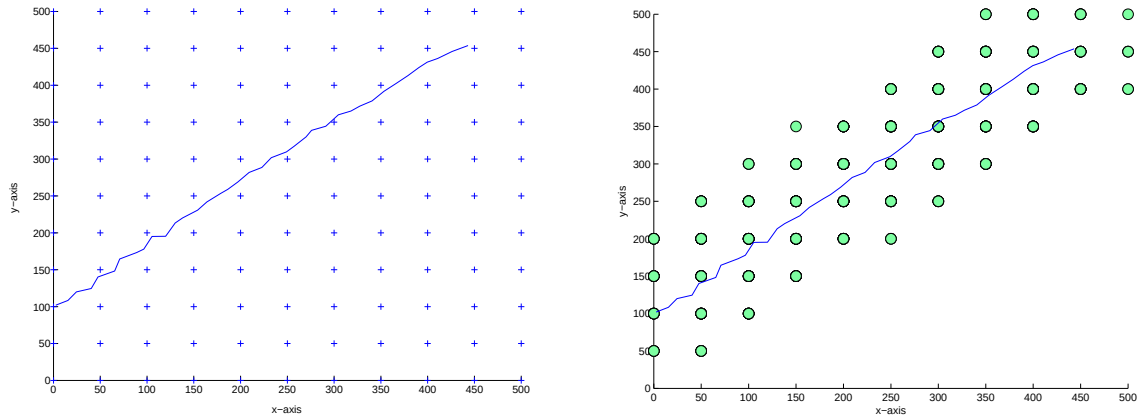
3.3.5 Data Fusion and Tracking Results

Extensive experiments have been conducted to test the accuracy and robustness of the new proposed algorithm. The experimental results are presented in this section. Next, we examine the application of the CPF-SP for ground vehicle tracking in an acoustic sensor network for three different tracking scenarios: (1) a rectilinear motion disturbed by random noise, (2) motion that involves sharp coordinate turns with periodically changing velocities and accelerations, and (3) a dual-mode motion trajectory that is initially described by a constant acceleration then followed by a constant deceleration. Based on our research, these three motion models are the major components that represent the true motions of a ground vehicle.

Tracking Scenario No. 1

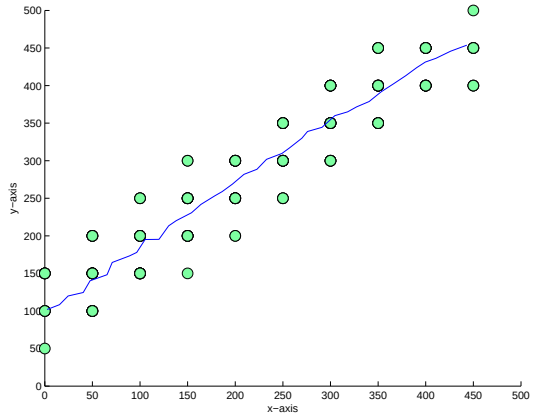
In this tracking scenario, we analyze tracking a target, which has a rectilinear motion disturbed by random noise. The target trajectory together with a grid sensor topology are shown in Figure 15(a). In all the simulations in this study, we assume that the moving target, *i.e.* a ground vehicle, has a noise level of 70dB, which is a typical value for this type of target. The sensor threshold is set to 72%, 75%, and 78% of the source acoustic intensity, respectively. Once the acoustic intensity received by a sensor exceeds these thresholds, the sensor will be switched to the active mode and will be able to transmit the target's DOA to the fusion center. The activated sensors that above thresholds are plotted in Figure 15(b)-(d), respectively. As illustrated in this figure, when the threshold is set to 78% of the source intensity, only one or two sensors are in the active mode during each sampling period. This indicates that 78% is approximately the highest threshold that can be used for this particular network topology, since it is assumed that at least one sensor is active during each sampling period.

Next, a CPF-SP and a centralized bootstrap filter (a standard PF) are applied to this problem for a comparison. The sensor threshold is set to 75% of the source acoustic intensity. One hundred Monte Carlo runs are implemented to generate the

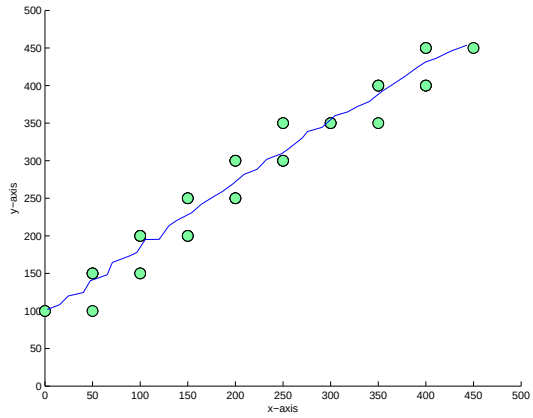


(a)Sensor Topology

(b) Threshold: 72%



(c) Threshold: 75%



(d) Threshold: 78%

Figure 15: *This figure shows: (a) The grid sensor topology and the true target trajectory, (b)-(d) The activated sensors with sensor thresholds setting to 72%, 75%, and 78% of the source energy, respectively.*

statistical performance index, the root mean square error (RMSE). The RMSEs for both the x-coordinate and the y-coordinate are shown in Figure 16. As indicated in this figure, the CPF-SP gives very accurate estimations while the traditional bootstrap filter yields huge estimation error. This is mainly because use the bootstrap filter takes the target state transition prior as the proposal distribution for the particle filter, which does not include the current measurement. When the likelihood distribution is narrow with respect to the state transition prior distribution, many particles will receive negligible weights, and large estimations errors will occur. Moreover, the overall mean and variance of the RMSEs are shown in Table 5. From this table, it is clear that the CPF-SP not only gives small estimation error, but also gives small estimation

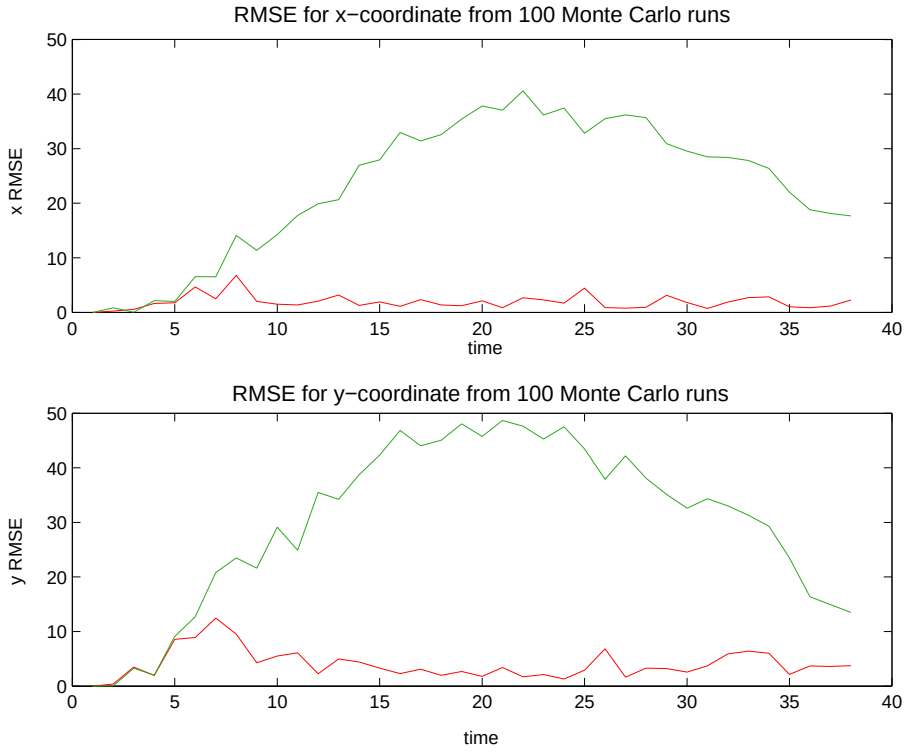


Figure 16: *Root mean square errors (RMSE) of Scenario No.1: The red line denotes the RMSE from the CPF-SP, while the green line represents the RMSE from the bootstrap filter.*

error variance, which means that the CPF-SP gives much more consistent estimations compared to the bootstrap filter. It should be also noted that the CPF-SP renders these good estimations with only 200 particles, however the bootstrap filter uses 2000 particles. The execution time for each algorithm is also provided in Table 5.

Table 5: A Comparison of the CPF-SP and the Bootstrap Filter

	RMSE for x		RMSE for y		Time
	Mean	Var.	Mean	Var.	(Sec.)
CPF-SP	1.909	1.693	4.004	6.978	2.161
Bootstrap	23.180	154.63	30.060	223.291	8.109

Figure 17 and 18 illustrate a typical realization of this tracking scenario. As demonstrated in this figure, the CPF-SP algorithm keeps a close track of the target throughout the whole experiment. But for the bootstrap filter, although it can follow the general

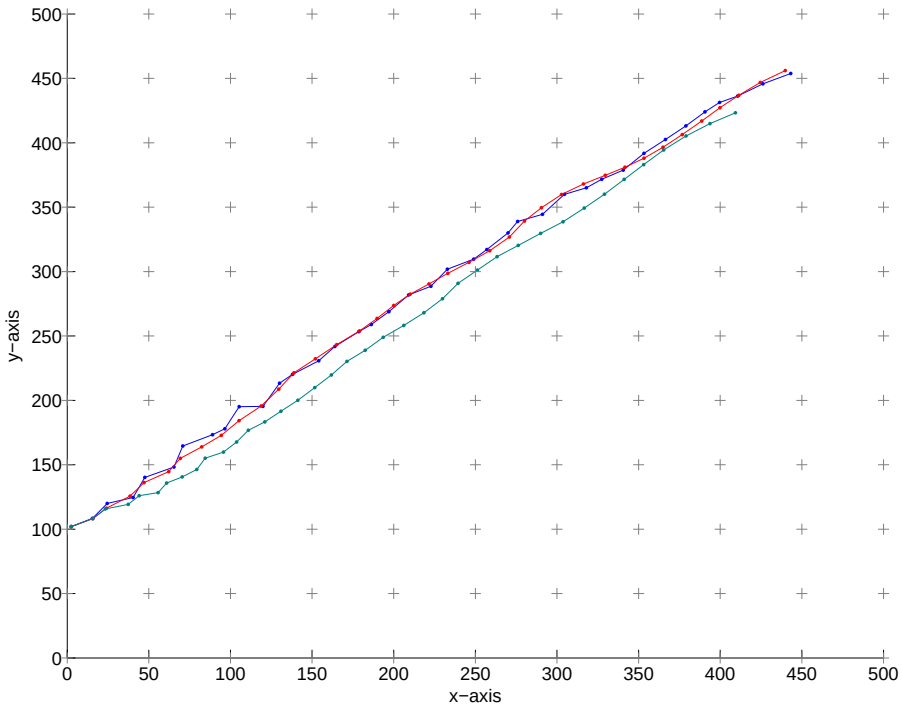


Figure 17: *The location estimations from one typical realization of the scenario No.1: blue line: true target trajectory, red line: the CPF-SP estimations, green line: the bootstrap filter estimations.*

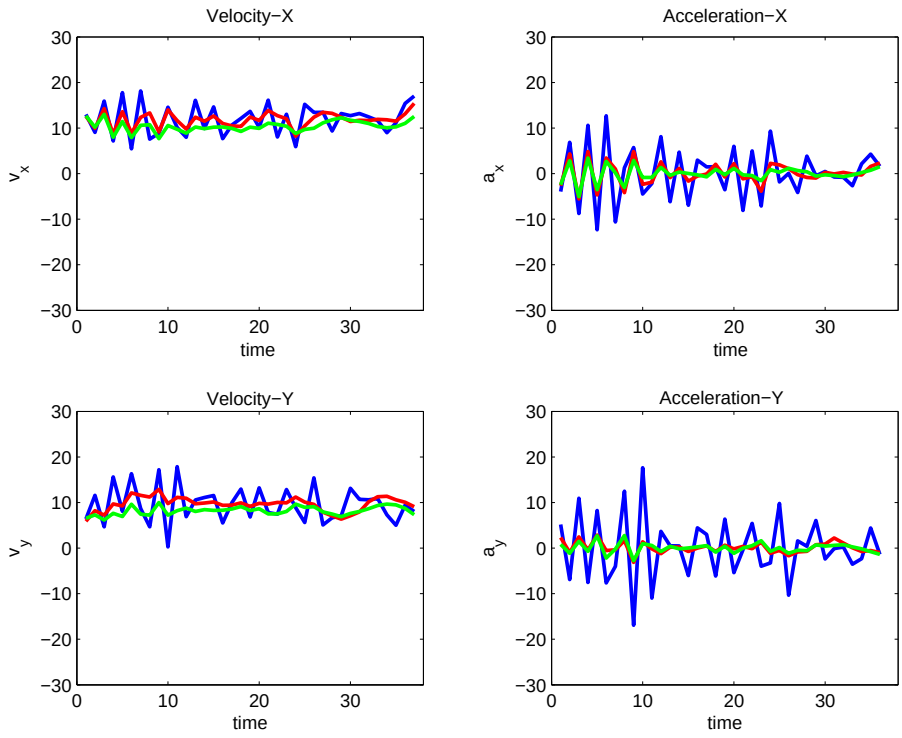


Figure 18: *The velocity and acceleration estimations from one typical realization of the scenario No.1: blue line: true target, red line: the CPF-SP estimations, green line: the bootstrap filter estimations.*

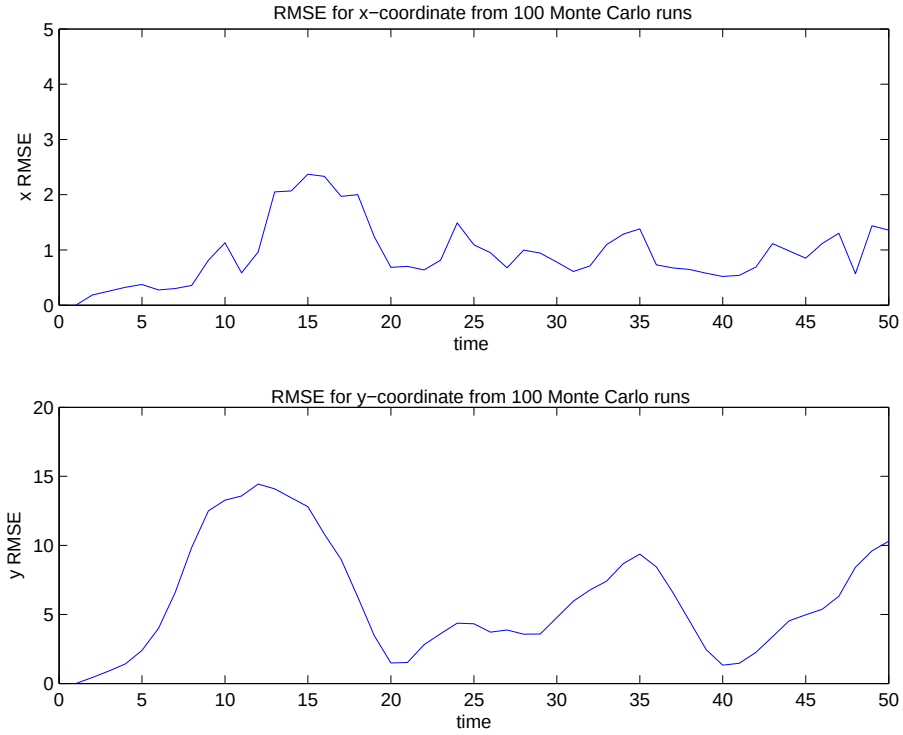


Figure 19: *Root mean square errors (RMSE) of Scenario No.2.*

direction of the target, it gives large estimation errors when compared to that of the CPF-SP, as shown in Figure 17. In addition, this research also shows that when setting the threshold to 78%, the CPF-SP can provide estimation results similar to the aforementioned, which demonstrates that this algorithm has the advantage to reduce the computation and communication load at the sensor node required to give accurate results.

Tracking Scenario No. 2

In the second scenario, we apply the tracking algorithm to a target whose motion trajectory has two sharp coordinate turns with periodic changing velocities and accelerations. In the target tracking community, this kind of target is classified as a *maneuvering target* [52]. By definition, this implies that target's x and y velocities are non-constant as it makes a path. This aspect presents additional tracking challenges, as compared to tracking non-maneuvering targets. Traditionally, multiple-model techniques are used to track maneuvering targets [37]. However, to test the algorithm

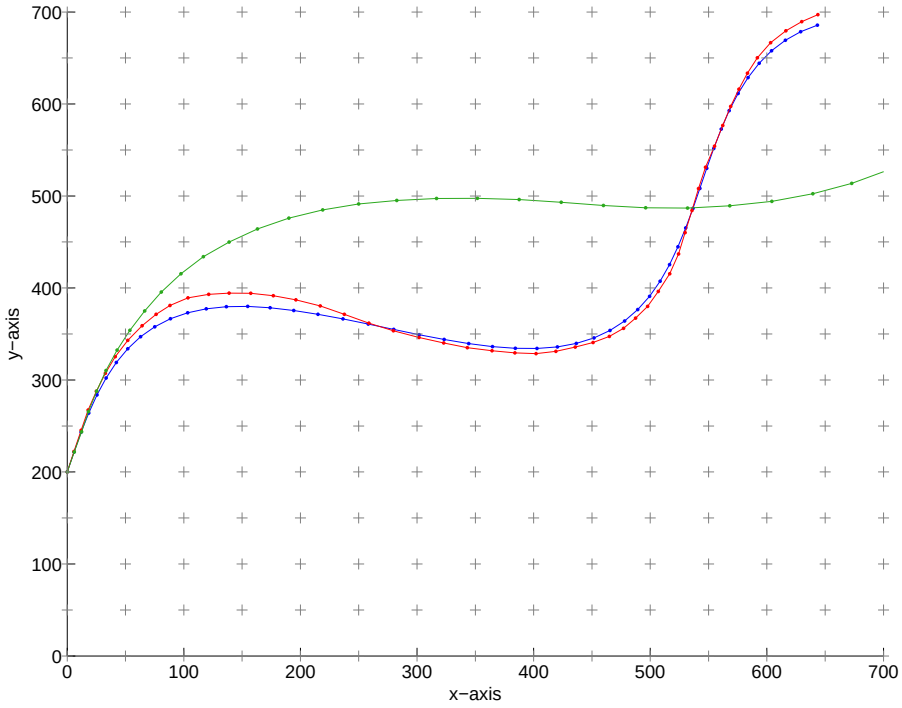


Figure 20: *The location estimations from one typical realization of the scenario No.2: blue line: true target trajectory, red line: the CPF-SP estimations, green line: the bootstrap filter estimations*

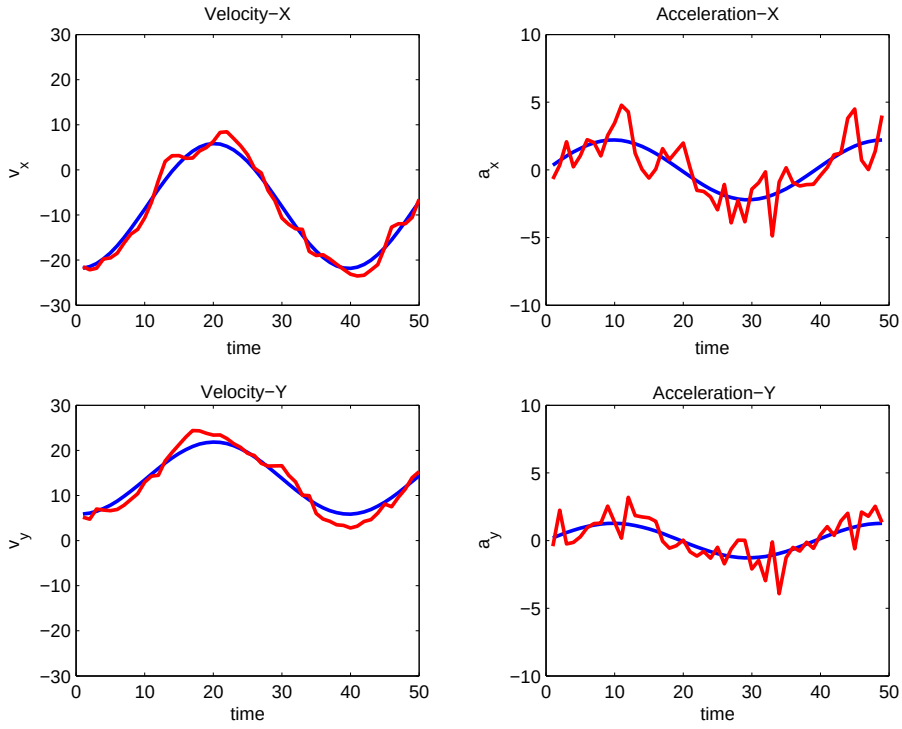


Figure 21: *The velocity and acceleration estimations from one typical realization of the scenario No.2: blue line: true target, red line: the CPF-SP estimations.*

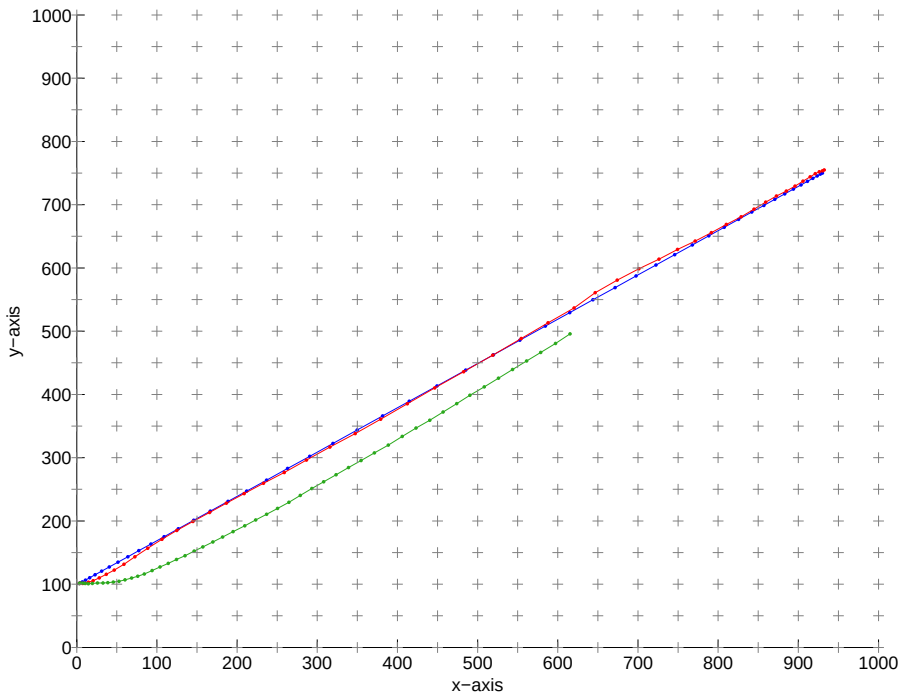


Figure 22: *The location estimations from one typical realization of the scenario No.3: blue line: true target trajectory, red line: the CPF-SP estimations, green line: the bootstrap filter estimations*

robustness for model uncertainties, we still use the *single non-maneuvering model* for the CPF-SP. Similar to the previous example, 100 Monte Carlo runs were implemented. As shown in the experiments, the CPF-SP can keep a close track in every realization. However, the bootstrap filter looses the target in more than 80% of the 100 realizations. Due to fact that most of the bootstrap filters lose track, the RMSE for the CPF-SP is only provided, as depicted in Figure 19. A typical realization is shown in Figure 20 and 21. As it is evident from this simulation, the CPF-SP follows the target closely despite the model utilized here is a *single non-maneuvering model*. This fact demonstrates that the CPF-SP can achieve certain robustness under model uncertainties. Moreover, this robust tracking is achieved by using only 200 particles, which is a rather small sample size compared to the standard bootstrap particle filter which uses 2000 particles.

Tracking Scenario No. 3

In the third scenario, tracking a target with a dual-mode motion trajectory that is initially described by a constant acceleration then followed by a constant deceleration is

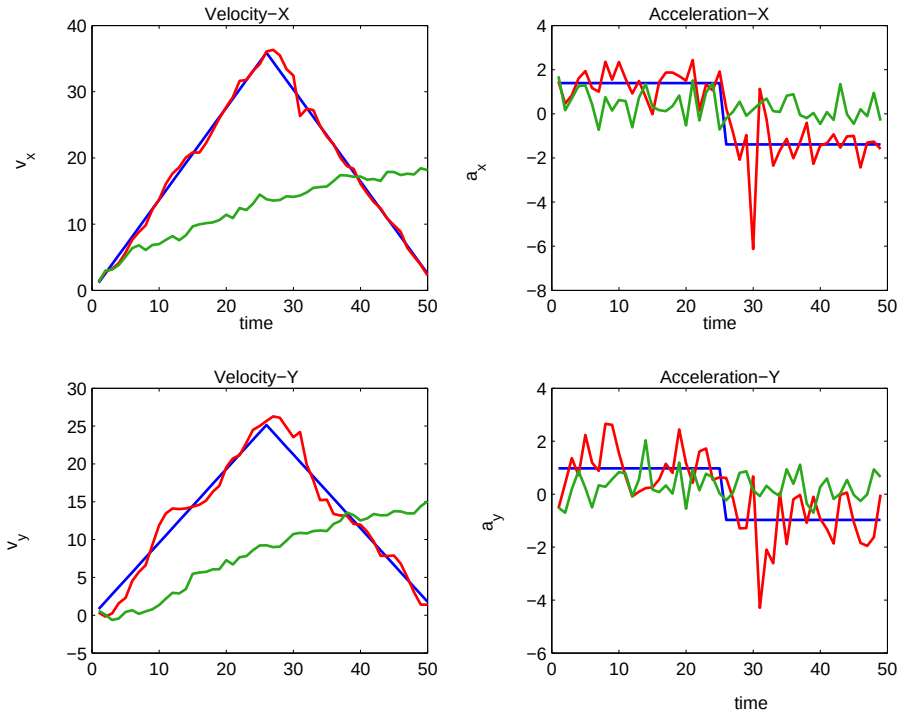


Figure 23: *The velocity and acceleration estimations from one typical realization of the scenario No.3: blue line: true target, red line: the CPF-SP estimations, green line: bootstrap filter estimations.*

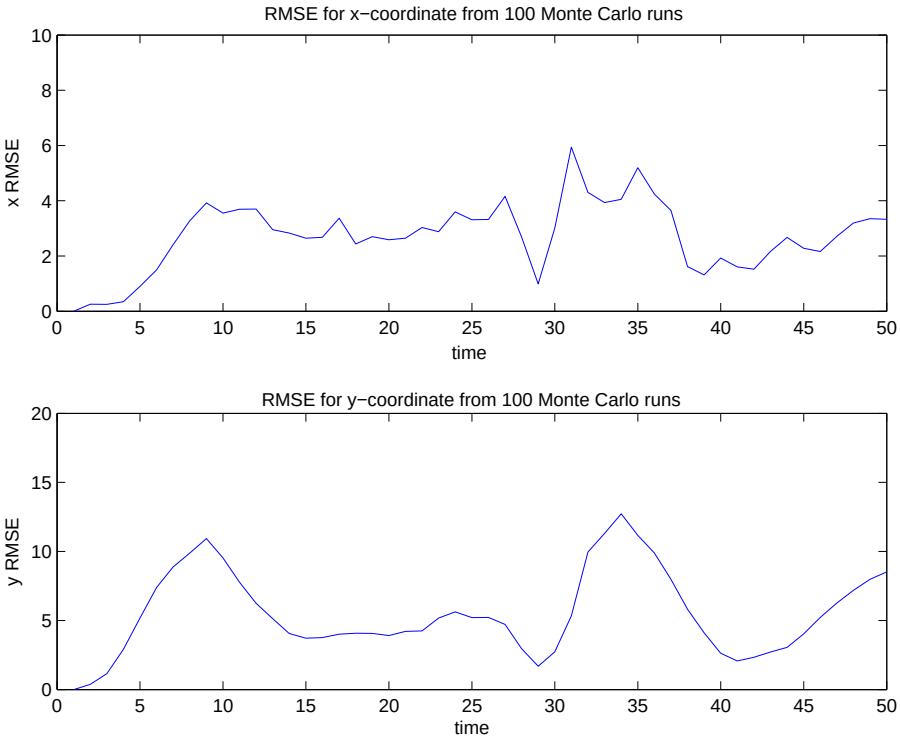


Figure 24: *Root mean square errors (RMSE) of Scenario No.3.*

investigated. Due to the instantaneous change in the velocity and the acceleration, the target makes a significant maneuver operation, although its motion is still rectilinear. As indicated in the 100 Monte Carlo runs, the bootstrap filter can follow the general direction of the target, but it fails to detect the instantaneous acceleration change and finally it loses the target. Figure 22 and 23 depicts one realization. It is very clear in Figure 23 that the bootstrap filter cannot detect the velocity and acceleration change. On the other hand, the CPF-SP yields very good estimates for both target locations and velocities. In addition, the RMSEs of the Monte Carlo runs are shown in Figure 24.

3.3.6 Summary

This section proposed a new sensor network target tracking method based particle filtering and state partition techniques. Simulation results demonstrate that this technique is capable to yield accurate and robust estimations while reducing the computations and channel requirements in the sensor network. In particular, the dynamic model of a moving target via an acoustic sensor network has been analyzed. The network has been defined as a distributed wireless acoustic sensor network which is composed of deterministically or randomly deployed sensors whose positions are known to the fusion center which is either stationary or mobile. The work here has provided contributions by showing that improved sequential Monte-Carlo methods can be used to efficiently generate precise estimations for targets which may has complex motion patterns, and that sensor nodes can be switched between the active mode and the sleep mode adaptively based on the requirements of power consumption and estimation accuracy.

3.4 Multiple Model Particle Filter

As indicated in Section 2.3.3, the estimation accuracy can be also improved by refining the modeling process. In many real world problems, the system under study is so complex that a single model can not fully describe its behavior. In these cases, multiple dynamic models are often utilized to approximate the true system. The system under study is assumed to have a switching structure that the system may change from one model to anther during each sampling period. This kind switching process is known as a jump Markov process. In tracking problems, the target can be divided into two categories: non-maneuvering targets and maneuvering targets, where a maneuvering target represents the target that can make sudden and dramatic changes in motion patterns. Recently, various particle filter based algorithms have been developed for tracking maneuvering targets. Among them, multiple model particle filter (MMPF) has been proven to be successful [3] [55] [56]. Based on the choice of different proposal distributions and different target models, MMPFs may have many different forms. In this section, we will first analyze the fundamental concepts of the multiple model method. Next, we will provide a study of target dynamic models used for maneuvering target tracking. Then, we will present the general form of a multiple model particle filter (MMPF) with two examples. These are the MM-bootstrap and the multi-sensor MM-bootstrap, respectively. Finally, in the next section, we will develop a more advanced MMPF, called MMPF-SP, and apply it to visual target tracking problems.

3.4.1 The Fundamentals of the Multiple Model Method

When tracking a target which can perform maneuvers, the estimations generated by a bootstrap filter based on the aforementioned model, equations (34) and (35) are not always accurate enough [3][54]. The problem of maneuvering target tracking is often referred to as a jump Markov process [3] [52], in which the system is assumed to operate according to one model from a finite set of hypothetical models, known as regimes or modes. In this scenario, the general form of the state space model for the

hybrid system is defined as:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{f}_{t-1}(\mathbf{x}_{t-1}, \mathbf{w}_{t-1}, r_t) \\ \mathbf{z}_t &= \mathbf{h}_t(\mathbf{x}_t, \mathbf{v}_t, r_t) \end{aligned} \tag{39}$$

where $\mathbf{x}_t$ is the system state vector at time t, and r_t indicates the system operates according to *r-th* model at time t. The variable r_t is a function of time, and its transition is governed by a regime transition matrix or model transition matrix defined as:

$$\Pi = \begin{bmatrix} \pi_{11} & \pi_{12} & \cdots & \pi_{N_r 1} \\ \pi_{21} & \pi_{22} & \cdots & \cdot \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{1N_r} & \cdots & \cdots & \pi_{N_r N_r} \end{bmatrix} \tag{40}$$

where

$$\pi_{ij} = Pr\Big(r_t = j \;\Big|\; r_{t-1} = i \Big) \; i, j \in 1 \cdots N_r \tag{41}$$

and $\sum_{j=1}^{N_r} \pi_{ij} = 1$. In other words, π_{ij} represents the probability of model i at time $t-1$ switching to model j at time t. And the variable N_r is the number of all possible models. In addition, the state vector can be taken as $[\mathbf{x}_t \quad r_t]^{\mathrm{T}}$. The variable $\mathbf{x}_t$, which describes the target kinematics, has a high dimensional continuous state space. The regime variable r_t (or the model variable), which represents the current system model, has a one dimensional discrete state space. Figure 25 provide an example of model switching structure for a 4-model case. The switching structure is based on an initial model distribution and the model transition matrix. In this example, we initially generate a uniformly distributed model particles (10,000 samples), *i.e.* each model has approximately the same probability. Then, this uniform model distribution is updated based on the following transition matrix:

$$\Pi = \begin{bmatrix} 0.7 & 0.1 & 0.1 & 0.1 \\ 0.4 & 0.5 & 0.05 & 0.05 \\ 0.4 & 0.3 & 0.2 & 0.1 \\ 0.4 & 0.3 & 0.2 & 0.1 \end{bmatrix}$$

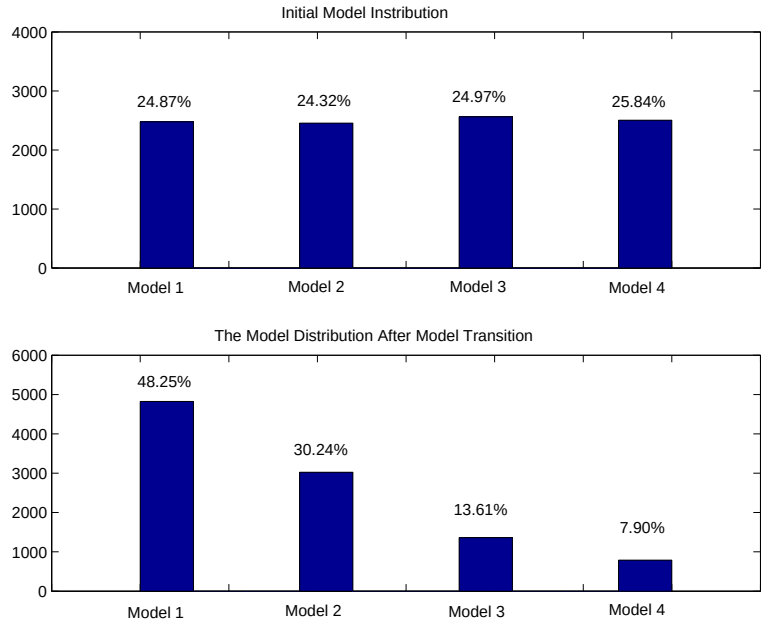


Figure 25: *A demonstration of model distribution transition.*

It is clear from Π and demonstrated from the figure that model one has highest probability. Model two has a higher probability than model three, while model four has the lowest probability.

3.4.2 A Study of Target Dynamic Models

In this subsection, we study various target dynamic models which are commonly adopted in the current literature. A complete study of dynamic models for target tracking can be found in [52], which covers both 2D and 3D models. In this research, we focus only on 2D models, which can be easily extended to 3D. We have already discussed the dynamic model for the non-maneuvering target, the constant velocity model, which is given by equations (34) and (35). Here we will discuss three more models for describing maneuvering targets: (1) coordinated turn (CT) model, (2) constant acceleration model (CA), and (3) curvilinear motion model. These are below.

The Coordinate Turn (CT) Model

The CT model is used to describe a target which has a circular-type motion with a constant turn rate. This kind of model has two forms: (1) the counter-clockwise CT model and (2) the clockwise CT model. By taking the state vector as $\mathbf{x}_t = [x_t \ y_t \ v_{x_t} \ v_{y_t}]^{\mathrm{T}}$, consider a state space model of the form given as:

$$\mathbf{x}_t = \mathbf{F}^{r_t}(\mathbf{x}_{t-1}) \cdot \mathbf{x}_{t-1} + \Gamma \cdot \mathbf{V}_{t-1} \quad . \tag{42}$$

We use $r = 2$ and 3 to denote the counter-clockwise and the clockwise CT models, respectively. And $r = 1$ is reserved for representing the non-maneuvering CV model. The CT models are given as:

$$\mathbf{F}^r(\mathbf{x}_t) = \begin{bmatrix} 1 & 0 & \frac{\sin \Omega_t^{(r)} T}{\Omega_t^{(r)}} & -\frac{(1-\cos \Omega_t^{(r)} T)}{\Omega_t^{(r)}} \\ 0 & 1 & \frac{(1-\cos \Omega_t^{(r)} T)}{\Omega_t^{(r)}} & \frac{\sin \Omega_t^{(r)} T}{\Omega_t^{(r)}} \\ 0 & 0 & \cos \Omega_t^{(r)} T & -\sin \Omega_t^{(r)} T \\ 0 & 0 & \sin \Omega_t^{(r)} T & \cos \Omega_t^{(r)} T \end{bmatrix} \quad , \tag{43}$$

where $\Omega_t^{(r)}$ denotes the model-conditioned turning rate given as:

$$\begin{aligned} \Omega_t^{(2)} &= \frac{a_m}{\sqrt{v_{x_t}^2 + v_{y_t}^2}} \\ \Omega_t^{(3)} &= \frac{-a_m}{\sqrt{v_{x_t}^2 + v_{y_t}^2}} \quad . \end{aligned}$$

The variable a_m in the above equation represents the maneuver acceleration. Figure 26 illustrates the possible turns result from various maneuver accelerations with the target initial condition given as: $\mathbf{x}_t = [0 \quad 0 \quad 1 \quad 0]^{\mathrm{T}}$. As indicated above, the two CT models have strong nonlinearities. In addition, for the bearings-only tracking problem, the measurement model is same as equation (35).

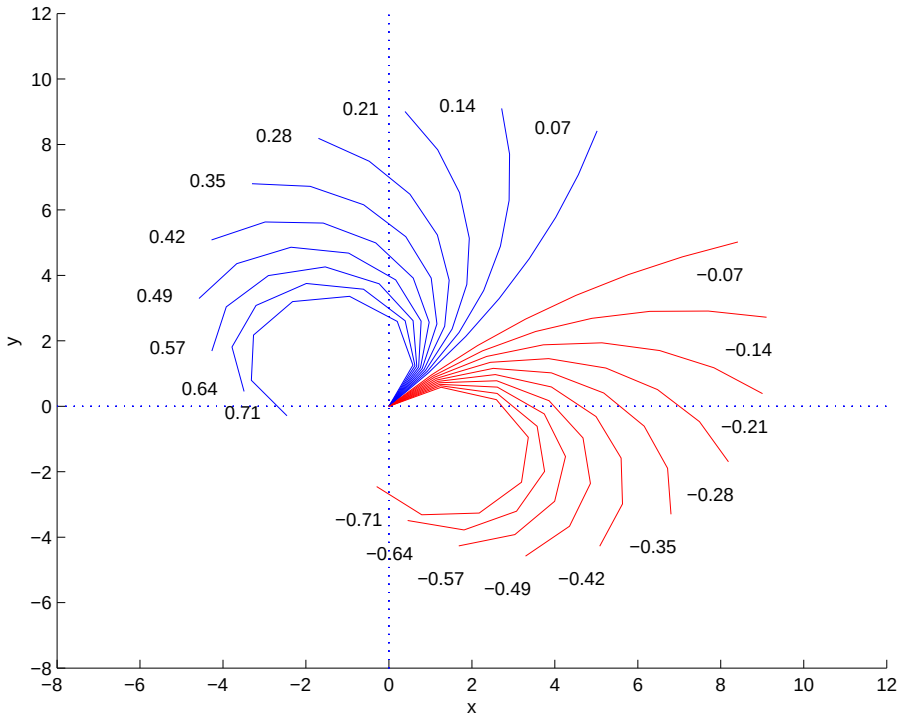


Figure 26: *The coordinated turn (CT) models with different turn rate.*

The Constant Acceleration Model

This model assumes that the target under track has a constant acceleration, which is corrupted noise. There are two types of acceleration models: (1) the white-noise acceleration model, and (2) the constant acceleration (CA) model [52]. The white-noise constant acceleration model is the most basic form of acceleration model. In this model, the target's acceleration is deemed as an independent process corrupted by white noise. This model is similar to the CV model, except that it has a higher noise level, which is used to model unpredictable maneuver operations. Strictly speaking this kind of model is not a constant acceleration model. The white-noise acceleration model is also known as white-noise jerk model (the second derivative of velocity). The second type acceleration model is the constant acceleration model, where the acceleration is a process with "independent increments." In other words, the CA model assumes that the acceleration increment is an independent process. This model is also called the Wiener-process acceleration model. But the studies of [52], the authors indicate that the acceleration process is not necessarily a Wiener process. In acceleration models,

the state vector has the following form:

$$\mathbf{x}_t = [x_t \ y_t \ v_{x_t} \ v_{y_t} \ a_{x_t} \ a_{y_t}]^{\mathrm{T}}$$

and the dynamic model is:

$$\mathbf{x}_t = \mathbf{F}^4 \left(\mathbf{x}_{t-1}\right) \cdot \mathbf{x}_{t-1} + \Gamma \cdot \mathbf{w}_{t-1} \quad . \tag{44}$$

where

$$\mathbf{F}^{(4)} = \begin{bmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{45}$$

and for white-noise acceleration model:

$$\Gamma = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^{\mathrm{T}}$$

for constant acceleration model:

$$\Gamma = \begin{bmatrix} \frac{T^2}{2} & 0 & T & 0 & 1 & 0 \\ 0 & \frac{T^2}{2} & 0 & T & 0 & 1 \end{bmatrix}^{\mathrm{T}}$$

Curvilinear motion model

Various curvilinear motion models were reported in current literature [52][51]. Here we introduce the basic form of the curvilinear model. Taking the state vector as $\mathrm{x}_t = [x_t \ y_t \ v_{x_t} \ v_{y_t}]^{\mathrm{T}}$, the curvilinear motion model is defined as:

$$\mathbf{x}_t = \mathbf{F}^1 \cdot \mathbf{x}_{t-1} + \mathbf{B}(\mathbf{x}_{t-1}) \cdot \mathbf{a}_t + \mathbf{w}_t \tag{46}$$

where

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

and

$$\mathbf{B}(\mathbf{x}_t) = \begin{bmatrix} 0 & 0 \\ \dfrac{v_{x_t}}{\sqrt{v_{x_t}^2 + v_{y_t}^2}} & -\dfrac{v_{y_t}}{\sqrt{v_{x_t}^2 + v_{y_t}^2}} \\ 0 & 0 \\ \dfrac{v_{y_t}}{\sqrt{v_{x_t}^2 + v_{y_t}^2}} & \dfrac{v_{x_t}}{\sqrt{v_{x_t}^2 + v_{y_t}^2}} \end{bmatrix}$$

with $\mathbf{A}_t = [A_{tangent}\ A_{normal}]^{\mathrm{T}}$. The matrix $\mathbf{B}(\mathbf{x}_t)$ is the curvilinear motion transition matrix. The vector $\mathbf{a}_t$ represents the maneuver acceleration along the tangential and normal directions, respectively. It is clear that this model has strong linearities. Another improved curvilinear motion model was reported in [51], which is one of the most sophisticated 2D motion model in target tracking [52].

3.4.3 The General Form of Multiple Model Particle Filter

Maneuvering target tracking is a *hybrid state estimation* problem. By definition, hybrid state estimation is the estimation of a quantity that has both continuous and discrete components [53]. In maneuvering target tracking, the state components that describe the target kinematics has continuous values, while the model variable only has discrete values. In the problem of maneuvering target tracking, the multiple model method has been widely used and is considered as the mainstream approach [53]. Currently, the most widely used method is the interacting multiple model extended Kalman filter (IMM-EKF) [59]-[62]. The basic idea of IMM-EKF is to use a separate EKF for each hypothetical model, then the final estimate is the weighted sum of the all components. The filter weight is a function of the regime transition probabilities and the filter weight at previous time index. The approximation error of an IMM-EKF comes from two sources. First of all, each EKF approximates a nonlinear system by linearizing it about a certain operating point. This procedure introduces the linearization error. For

example, some models, such as the coordinated turn model and the curvilinear motion model, have strong nonlinearities. Linearizing these models produces large modeling error. Secondly, the IMM approximates the exponentially growing Gaussian mixture with a finite Gaussian mixture which results in an approximation error. These two kinds of errors may cause the filter to diverge from the true target trajectory in some scenarios. More recently, inspired by IMM-EKF, some researchers have developed various interacting multiple model particle filters (IMM-PF) [63][66] [67]. However, these algorithms still use the same method as the IMM-EKF that approximates the exponentially growing Gaussian mixture with a finite Gaussian mixture.

An improvement over the IMM-PF is the recently introduced multiple model particle filter (MMPF). Actually, MMPF is a flexible framework and is a superior alternative to the IMM-EKF to perform nonlinear filtering with switching dynamic models [54][55] [56][57]. The MMPF framework can be modified to couple with different applications. For example, in [65], a MMPF with gating and data association was designed for tracking multiple targets. As a novel extension of the single model particle filter, the MMPF generates state estimates based on all models. More specifically, at each iteration, a set of model samples $\{r_t^{(i)}\}_{i=1}^{Ns}$ are generated based on the model transition matrix Π defined in equation (40) and the model samples at previous time $\{r_{t-1}^{(i)}\}_{i=1}^{Ns}$ such that:

$$Pr\left(r_t^{(i)} = j | r_{t-1}^{(i)} = i\right) = \pi_{ij} \quad , \tag{47}$$

where Ns is the sample size. Then, the state particles are drawn from the proposal $q(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, r_t^{(i)}, z_t)$, and the particle weights are calculated as follows:

$$\omega_t^{(i)} = \omega_{t-1}^{(i)} \frac{p(z_t|\mathbf{x}_t^{(i)}, r_t^{(i)}) p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, r_t^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, r_t^{(i)}, z_{1:t})} \tag{48}$$

where $\mathbf{x}_t^{(i)}$ and z_t denote the state and the measurement, respectively. This equation is similar to single model particle filter, except that the model variable $r_t^{(i)}$ is incorporated. In addition, the proposal distribution $q(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, r_t^{(i)}, z_{1:t})$ also involves the

Table 6: Algorithm 3: The General Form of A Multi-model Particle Filter

- Generating regime samples (particles) based on regime transition matrix:
  - generate $\{r_t^{(i)}\}_{i=1}^{Ns}$ based on $\{r_{t-1}^{(i)}\}_{i=1}^{Ns}$ such that
    $$Pr\left(r_t^{(i)} = j | r_{t-1}^{(i)} = i\right) = \pi_{ij} \quad ,$$
    where π_{ij} is given in equation (40) and Ns is the sample size.
- Evaluating particle weights:
  - Draw samples $\mathbf{x}_t^{(i)}$ according to
    $$\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t^{(i)} | \mathbf{x}_{t-1}^{(i)}, r_t^{(i)}, z_{1:t})$$
  - Calculate particle weights based on equation (48).
  - Normalize the weights as:
    $$\tilde{\omega}_t^{(i)} = \frac{\omega_t^{(i)}}{\sum_{j=1}^{Ns} \omega_t^{(j)}} \quad .$$
- Resampling and updating the estimates:
  - Generate a new set of particles $\mathbf{x}_t^{(i\star)}$ from $\mathbf{x}_t^{(i)}$ so that
    $$Pr\left(\mathbf{x}_t^{(i\star)} = \mathbf{x}_t^{(i)}\right) = \tilde{\omega}^{(i)}(t) \quad .$$
  - Output and update: The final extimate is given as $\mathbf{x}(t) \approx \frac{1}{N_s} \sum_{i=1}^{N_s} \mathbf{x}^{(i\star)}(t)$.

model variable, which means that the proposal is generated based on the current dynamic model. After this, the particle weights are normalized followed by a resampling step. The aforementioned algorithm is a general framework of MMPF. For the sake of completeness, this algorithm is summarized in Table 6.

3.4.4 Multiple Model Bootstrap Filter

Based on different choices proposal distribution, various MMPF algorithms can be developed. Among them, the multi-model bootstrap filter is a special case, in which the proposal distribution of the PF, $q(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, r_t^{(i)}, z_{1:t})$, is taken as the state transition prior $p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, r_t^{(i)})$. In this case, the particles weights only depend on the system's likelihood $p(z_t|\mathbf{x}_t^{(i)}, r_t^{(i)})$ as indicated in previous subsection.

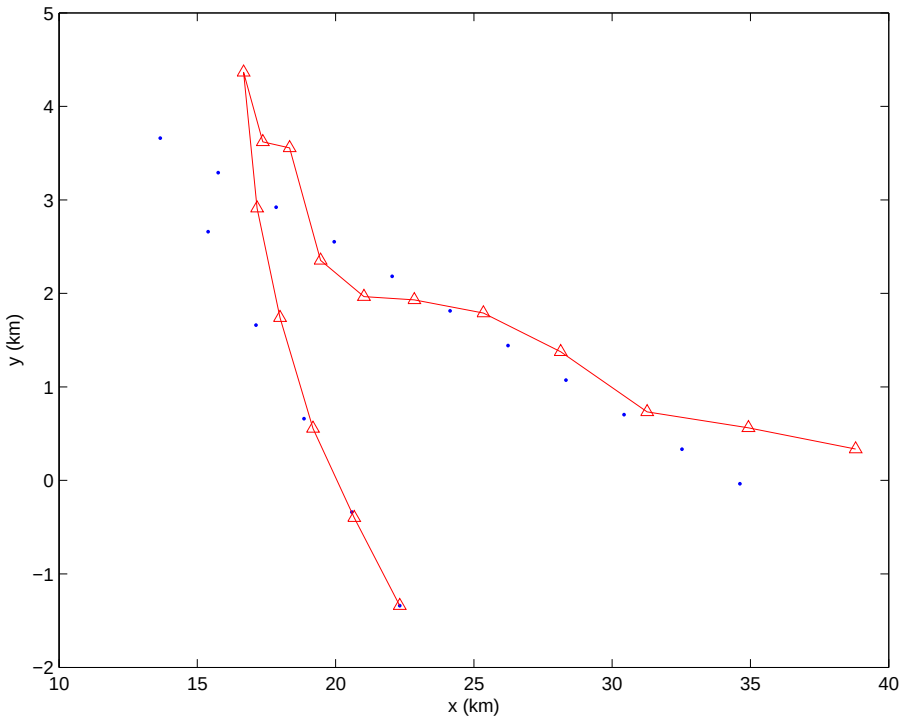


Figure 27: *Tracking Maneuvering target using MMPF: The blue dots represent the true target location. The red triangle-line represents the estimated location.*

An example to demonstrate the strength of the MM-bootstrap is now provided. In Figure 27, the blue dots represent the ground truth of a target trajectory which makes a dramatic maneuver, a clockwise sharp turn at $t = 7$. In addition, we assume this

is a bearings-only tracking scenario. The challenge of this problem comes from two sources: first the target is a maneuvering target, second only the angle measurement is available. A multi-model bootstrap filter was implemented for this tracking problem. The estimated trajectory is represented by the red line. From this example, we can see that MM-bootstrap can follow the general direction of the target, even when the target makes a sharp turn. In this problem, three dynamic models were used, *i.e.* one CV model and two CT models. Figure 28 shows the number of particles of each model after resampling, which is proportional to the model probability. The blue line represents the constant velocity (CV) model, while the green line and red line denote the counter-clockwise and the clockwise CT model, respectively. It is obvious that the counter-clockwise has a higher probability than the other models at time $t = 7$, when the target makes the sharp counter-clockwise turn. This figure fully demonstrates the strength of the multiple model approach.

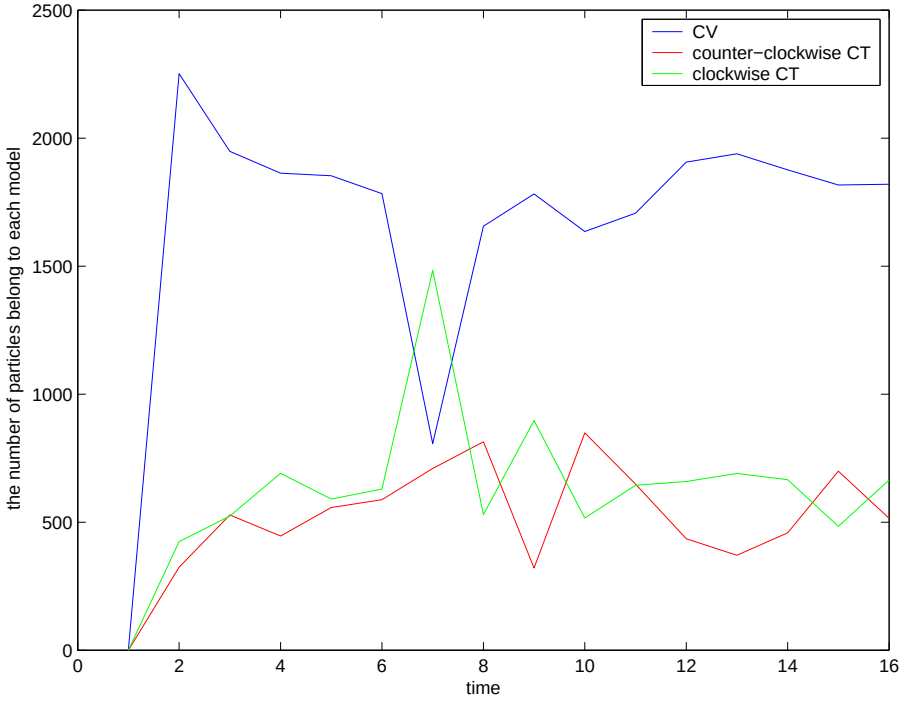


Figure 28: *The number of particles of each model after resampling. The blue line: the constant velocity (CV) model, the red line: the clockwise coordinated turn model, the green line: the counter-clockwise coordinated turn model.*

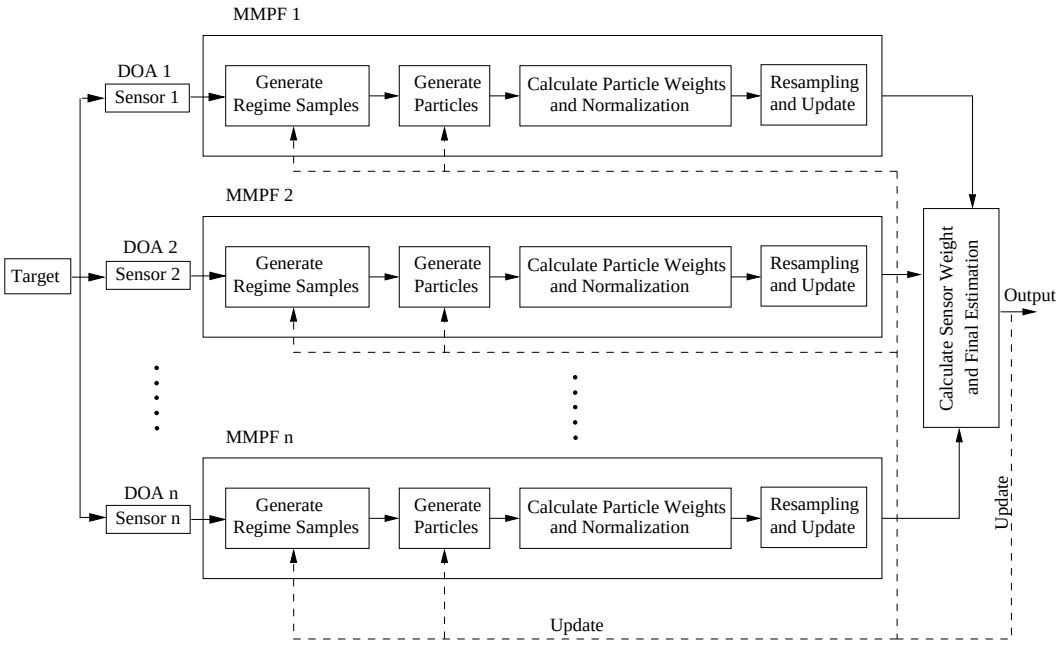


Figure 29: *This figure illustrates the multiple model particle filter (MMPF) tracking algorithm with using the DOA measurements from multiple sensors.*

3.4.5 MM-Bootstrap with Multiple Sensors

Moving further, we have developed a multiple-sensor multi-model bootstrap filter to track a maneuvering target. In this tracking scenario, we have applied separate multi-model bootstrap filters using DOA measurements from each sensor, then the overall estimation is given by a weighted sum of these individual estimates. The sensor weights are chosen as the normalized reflectivity received by each radar, which is proportional to the inverse of the distance squared. The algorithm of the MMPF using multiple sensors is illustrated in Figure 29. In addition, a single-sensor single-model bootstrap filter and a single-sensor MM-bootstrap filter are also implemented for the purpose of comparison. The estimation results of these three tracking algorithms are shown in Figure 30.

In this figure, the true trajectory is denoted by the plus signs (+). The estimates from the single-sensor single-model bootstrap filter are represented by the line with a cross ($-\!\!\times\!\!-$), while estimates from MMPF with a single sensor and with multiple sensors are depicted by the upward triangle line ($-\!\!\triangle\!\!-$) and the line with dots ($-\!\!\bullet\!\!-$), respectively. In addition, each PF algorithm uses 5000 particles. As indicated in the figure, although

the single-model single-sensor bootstrap filter can follow the general direction of the target, it failed to detect the small turns made by the target. On the other hand, the multiple-sensor MMPF algorithm is able to detect the maneuvers made by the target and outperforms single-sensor MMPF.

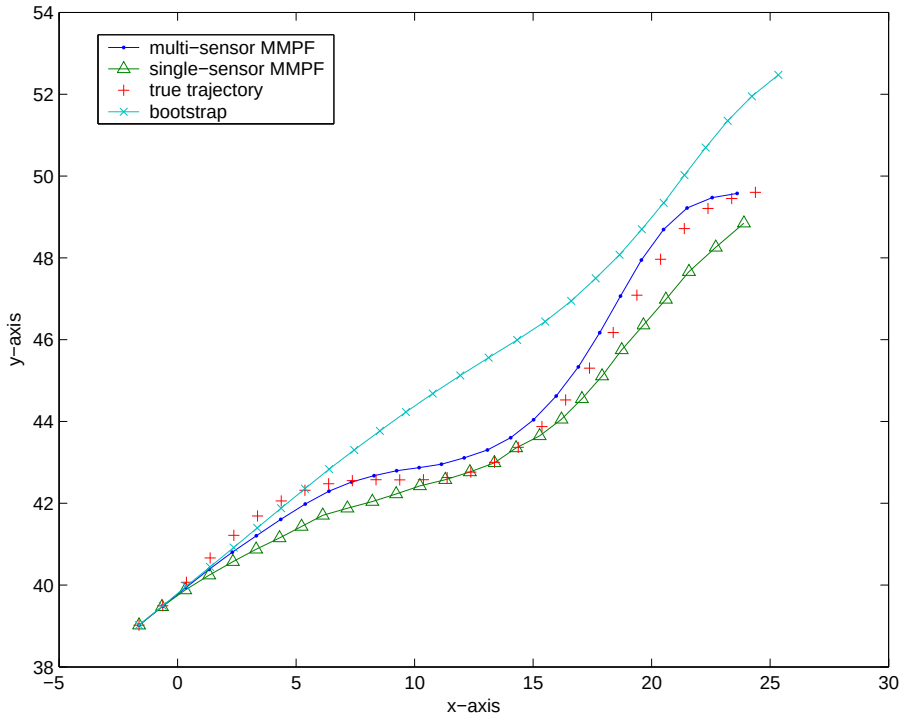


Figure 30: *This figure illustrates the tracking results of three algorithms: single-sensor single-model bootstrap filter (−×−), single-sensor MMPF (−△−) and multi-sensor MMPF (−●−). In addition, the true target location is represented by plus signs (+). It is clearly demonstrated from this figure that the MMPF algorithm with multiple sensors yield very accurate estimates.*

3.5 Improved Multiple Model Particle Filter for Visual Target Tracking

Robust, accurate visual object tracking is fundamental to a variety of computer vision applications including robotics, human tracking and biometric identification, intelligent transportation systems, smart rooms, and military targeting systems. Usually, the objects of interest in a video sequence are represented by state space models that may involve strong nonlinearities. In addition, the noise and background clutter are almost always present in real-world video sequences which make the visual tracking problem particularly challenging. Conventional methods for rectifying nonlinear and non-Gaussian problems include the extended Kalman filter (EKF), the Bayesian multiple-hypothesis filters, and similar variants including the probability data association filter (PDAF) and the joint probability data association filter (JPDAF) [69]. Hidden Markov models (HMM) have also been widely used [72][76].

As it is evident from the works of many recent authors, the particle filtering framework has revolutionized probabilistic visual target tracking. In this section, a new particle filter tracking algorithm is developed, in which it combines switching multiple dynamic models and the technique of state partitioning with parallel filter banks. This chapter is based on the previous work of the author's in [19][20]. Traditionally, most tracking algorithms assume the target operates according to a single dynamic model. However, the single model assumption causes the tracker to become unstable, especially when the target has complex motions, and the camera has abrupt ego-motions. In our new tracking algorithm, the target is assumed to operate according to one dynamic model from a finite set of models. The switching process from one model to another is governed by a so-called *jump Markov process*. This strategy can effectively capture the target's dynamics. In addition, the state partition technique and a parallel bank of extended Kalman filters (SP-PEKF) are used to generate the proposal distribution used in the particle filter to achieve further estimation accuracy. The main purpose of this research is to improve the visual tracking performance for a given type of measurement cue. Extensive tests have been conducted to evaluate the new tracking algorithm, and key outcomes are given in the results section. For the first time,

it has been demonstrated by our experiments that this new approach yields a significantly improved estimate of the state, enabling the new particle filter to effectively track human subjects in a video sequence where the standard condensation filter fails to maintain track lock.

3.5.1 An Introduction of Visual Target Tracking

Visual target tracking is an important task for many industrial applications, such as intelligent video surveillance systems [13] [70] [71], human machine interfaces [14][16], remote sensing and defense systems [72] [73], and others. Despite the ubiquitous applications, visual target tracking still remains a challenging problem. This is due to the fact that besides the traditional tracking challenges (such as complex target motions and background clutters), the targets in visual data are always subject to deformation, occlusion, camera ego-motion, changes in scale, and various illuminations.

In general, visual target tracking algorithms can be divided into two categories, *i.e.* (1) deterministic methods and (2) stochastic methods. In the deterministic approach, a cost function is always designed based on features generated/extracted from visual data. Then, the tracking problem is reduced to an optimization problem: seeking a target location which minimizes the cost function. The gradient descent [74] and mean shift [75] are two common methods. In the second category, the stochastic approach, the objects of interest are represented by state space models and the tracking problem is then formulated as a filtering or a state estimation problem. Early works of stochastic target tracking were based on the Kalman filter or its variants. These methods take the Gaussian assumption and require linearization of nonlinear systems. In addition, hidden Markov model (HMM) based methods were also reported [76]. More recently, a sequential importance sampling (SIS) technique, known as particle filtering, has been recognized as a significant target tracking algorithm because of its accuracy, robustness and flexibility in non-linear and non-Gaussian systems [2][4].

Various methods have been proposed in the current literature to improve the tracking results by either (1) developing more advanced dynamic models and refining tracking algorithms or (2) designing better detection method to improve measurement accuracy. Using the first strategy [80], an unscented particle filter is used for contour-based human face tracking, where an unscented Kalman filter is used in this algorithm to generate a better proposal distribution. In [81], an auxiliary particle filter was implemented for tracking a target in infrared videos. In the work of [17], the state partition technique was incorporated into the PF framework to achieve better tracking result. Also, in [82] a motion-based particle filter was proposed, where an "optimal implementable" proposal was designed based on the Kullback-Leibler (KL) measure. On the other hand, some researchers focus on the second strategy, which is designing better observation models to improve the tracking algorithms. For example, in [83], the target under track is represented by its histogram. Then, a kernel object tracking algorithm based on the Bhattacharyya coefficient and Epanechnikov profile was developed to achieve robust tracking. In [16], multiple measurement cues were fused using a unified particle filter framework. These measurement cues include: color cues, motion cues and sound cues. Each cue is used both for generating the proposal and calculating the particle weight. Of course these two strategies can be combined. For instance, in [84], a particle filter with both adaptively varying sample size and model noise variance is developed. This algorithm also uses intra frame information to adaptively update the gray scale template online.

In this section, a new particle filter tracking algorithm is presented by focusing on the first strategy, *i.e.* by developing more advanced dynamic models and refining tracking algorithms. The contribution of this work is twofold:

1. A target operates according to one of a finite set of dynamic models is considered. The switching process from one model to another is governed by the so called *jump Markov process* with a predefined model transition matrix. As noted in current literature, most tracking algorithms assume the target operates according to a single dynamic model. However, this assumption can cause the tracker to become

unstable especially when the target has complex motions and the camera has strong ego-motions. With our new modeling strategy, the tracker can effectively capture the target's dynamics.

2. A new design strategy is offered that adopts the state partitioning technique and a bank parallel EKFs to generate the proposal distribution used in the particle filter to achieve further estimation accuracy.

3.5.2 Visual Target Tracking Models

In this subsection, we briefly review the target model before introducing the new tracking algorithm. In many visual target tracking applications, the objective is to track the target's centroid, while the target maybe rigid or deformable. For example, in problem of people face tracking, the objective is usually to detect the face and track face center. In these cases, the target centers are treated as a point target similar to radar target tracking, although the measurement model is fundamentally different. The research in this section is focused on human face tracking, where the face is modeled as a moving ellipse.

Target Dynamic Model

Here the row and column index of the ellipse center in the image plane $(\text{row}, \text{col}) = (r, s)$ is taken as the target's centroid. Then, the Langevin process is used to approximate the centroid's dynamics. Used in various current research [76][80], the Langevin process is a modified constant velocity model, which is given below:

$$\begin{bmatrix} r(t+1) \\ s(t+1) \\ \dot{r}(t+1) \\ \dot{s}(t+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & a_r & 0 \\ 0 & 0 & 0 & a_s \end{bmatrix} \begin{bmatrix} r(t) \\ s(t) \\ \dot{r}(t) \\ \dot{s}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ b_r \\ b_s \end{bmatrix} m(t) \ ,$$

where $x(t) = [r(t)\ s(t)\ \dot{r}(t)\ \dot{s}(t)]^T$ is the state vector. Other parameters are given as follows.

$$\begin{aligned} a_r &= \exp(\beta_r \Delta T) \\ a_s &= \exp(\beta_s \Delta T) \\ b_r &= \bar{v}_r \sqrt{1 - a_r^2} \\ b_s &= \bar{v}_s \sqrt{1 - a_s^2} \end{aligned}$$

The variables β_r and β_s are rate constants, ΔT is the discretization time step, $\bar{v}_r$ and $\bar{v}_s$ are the steady-state root-mean-square velocity, and $m(t)$ is the process noise. Later we will develop a multiple model particle filter, where the target has the ability to operate according to coordinated turn models and constant acceleration models. These models have been discussed in Section 3.4.

Target Measurement Model for Visual Target Tracking

In general, the measurement models for visual target tracking are based on different type of features (also called measurement cues) extracted from each frame. Commonly used features include edge information, grayscale texture-type information, color information. Other more advanced features include localized frequency information. The purpose of this section is to provide robust visual target tracking methods for a given same type of measurement cue. For this reason, only the edge information was used here. More sophisticated algorithms using multiple measurement cues will be discussed in Chapter 5. Edge cues were widely used in visual target detection, tracking and other image processing applications [77][78]. This is because edges carry significant information of the image content and those measurement cues are robust to change in the illuminations and target pose. Also, they are relatively straight forward to extract. As discussed earlier, in this section we use an ellipse to model the target under track, which is a person's face. The ellipse serves as a target template. This type template is one

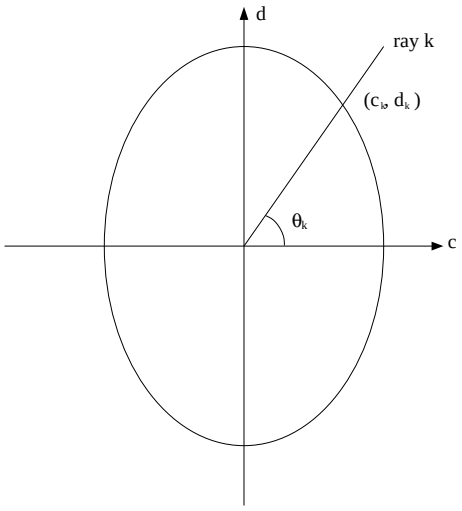


Figure 31: An ellipse is used a template to model human's head.

of the simplest form of parametric templates, yet it can provide robust detections as demonstrated in [76][80]. More advanced templates use B-splines to model the target boundary, which are also known as active contours [2][16][79].

To begin, an ellipse, shown in Figure 31, is used to model human's head. The boundary of the ellipse is utilized as the observation given the estimated states (the estimated center of the ellipse). Then, K equally spaced rays are drawn from the center of the ellipse. The intersections of these rays with the ellipse boundary are taken as observations. In the local coordinates, the intersection point of the ellipse with the k^{th} ray is obtained according to

$$\begin{aligned} c_k &= \sqrt{a^2 b^2 / (b^2 + a^2 \tan^2 \theta_k)} \\ d_k &= \tan \theta_k \cdot \sqrt{a^2 b^2 / (b^2 + a^2 \tan^2 \theta_k)} \end{aligned} \tag{49}$$

and by solving the ellipse equation

$$\frac{(c_k - m)^2}{a^2} + \frac{(d_k - n)^2}{b^2} = 1$$

and the ray equation $d_k = c_k \tan \phi_k$, where a and b denote the major and minor axes of the ellipse, respectively. By letting $\mathbf{z}$ represent the observation and converting the

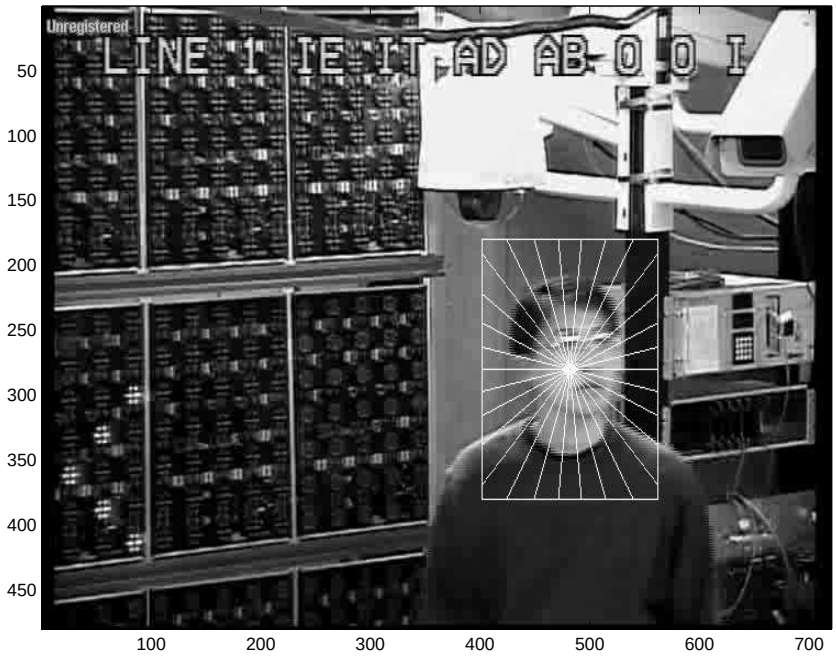


Figure 32: Equally space rays from the target center.

local coordinates back to image coordinates, the observation model is given by

$$\mathbf{z}_t = \begin{bmatrix} c_k + x_t \\ d_k + y_t \end{bmatrix} + \mathbf{v}_t \tag{50}$$

where $\mathbf{v}_t$ is a 2-by-1 noise vector. Figure 32 shows the equally spaced rays overlaid on the target in an image, which is a frame from one of our test videos. Then the Canny edge detection is implemented to the image to extract the image edge features. The Canny edge detection algorithm is known to many as the *optimal* edge detector. The details of this algorithm can be found in [77]. The target edge information can be easily contaminated by background clutters, which is demonstrated in Figure 33. In this figure, we implement the edge detection algorithm inside the tracking aid window. In order to to develop robust tracking algorithm, the background clutter rejection has to be considered. To reject clutter, we apply edge detection, and search along each ray. The edges are labeled as $j = 1 \cdots J_k$, where J_k is the total number of edges detected along the k^{th} ray. The likelihood of each ray is then

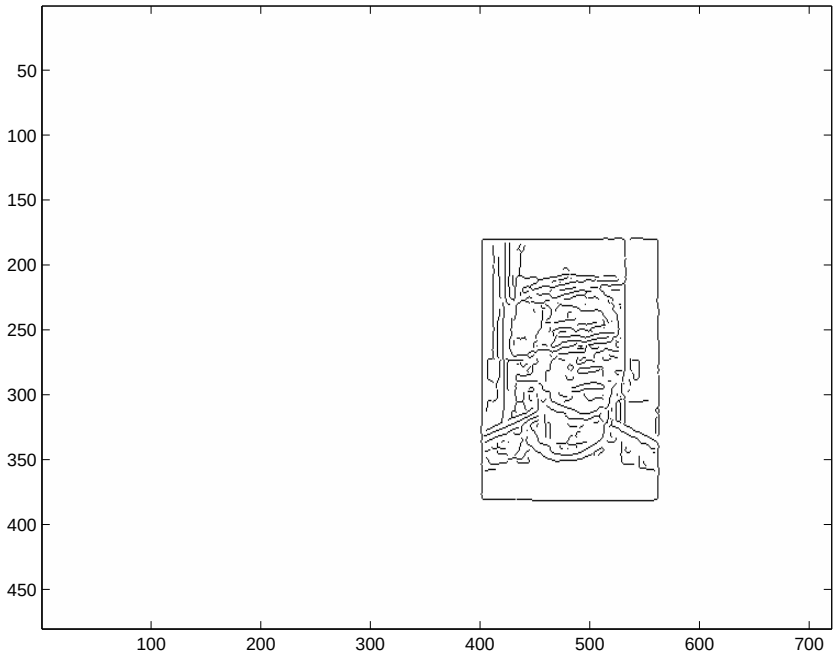


Figure 33: *The edge detection algorithm is implemented inside the tracking aid window. This figure shows how the target edge information can be contaminated by the background clutter.*

$$p_k(\mathbf{z}_t|\mathbf{x}_t) = N_m \sum_{j=1}^{J_K} \mathcal{N}\left((u_k, v_k), \sigma_{kj}^2\right) \quad , \tag{51}$$

where N_m is a normalizing factor. The overall likelihood is

$$p\left(\mathbf{z}_t|\mathbf{x}_t\right) = \prod_{k=1}^{K} p_k\left(\mathbf{z}_t|\mathbf{x}_t\right) \ , \tag{52}$$

which is a simplified version of the model used in [76][80].

3.5.3 New Tracking Algorithm: MMPF-SP

We have discussed the SP-PEKF technique and the improved particle filter algorithm (PF-SP-PEKF). We also introduced the multiple model method and the MMPF. In this section, we develop a new particle filter algorithm, which is a novel extension of single model PF-SP-PEKF. Four models are used in this algorithm: constant velocity model, clockwise coordinated turn model, counter-clockwise coordinated turn model and constant acceleration model. More specifically, at each iteration, a set model samples $\{r_t^{(i)}\}_{i=1}^{Ns}$ are generated based on the model transition matrix Π defined in (40) and the model samples at previous time $\{r_{t-1}^{(i)}\}_{i=1}^{Ns}$ such that: $Pr\left(r_t^{(i)}=j|r_{t-1}^{(i)}=i\right) = \pi_{ij}$, where Ns is the sample size. Then, the state particles are drawn from the proposal $q(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, r_t^{(i)}, \mathbf{z}_{1:t})$, and the particle weights are calculated as follows:

$$\omega_t^{(i)} = \omega_{t-1}^{(i)} \frac{p(\mathbf{z}_t|\mathbf{x}_t^{(i)}, r_t^{(i)}) p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, r_t^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, r_t^{(i)}, \mathbf{z}_{1:t})} \quad . \tag{53}$$

It should be noted that the proposal distribution $q(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, r_t^{(i)}, \mathbf{z}_{1:t})$ in the above equation is evaluated based on the partitioning method discussed in the previous subsection. More specifically, using the SP-PEKF to generate the proposal, we have the following formula:

$$q(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, r_t^{(i)}, \mathbf{z}_t) = \mathcal{N}\left(\hat{\mathbf{x}}_{total}^{r_t}(t|t), \mathbf{R}_{total}^{r_t}(t|t)\right) \tag{54}$$

where $\mathbf{x}_{total}^{r_t}(t|t)$ and $\mathbf{R}_{total}^{r_t}(t|t)$ can be evaluated based on model r_t and equations (28) and (30). For the two coordinated turn (CT) models, to implement the parallel EKF, we need the Jacobian matrices which are given in [3] and are also provided here:

$$\tilde{\mathbf{F}} = \begin{bmatrix} 1 & 0 & \frac{\partial f_1^{(j)}}{\partial v_{x_t}} & \frac{\partial f_1^{(j)}}{\partial v_{y_t}} \\ 0 & 1 & \frac{\partial f_2^{(j)}}{\partial v_{x_t}} & \frac{\partial f_2^{(j)}}{\partial v_{y_t}} \\ 0 & 0 & \frac{\partial f_3^{(j)}}{\partial v_{x_t}} & \frac{\partial f_3^{(j)}}{\partial v_{y_t}} \\ 0 & 0 & \frac{\partial f_4^{(j)}}{\partial v_{x_t}} & \frac{\partial f_4^{(j)}}{\partial v_{y_t}} \end{bmatrix}, \qquad j = 2, 3 \tag{55}$$

where

$$\frac{\partial f_1^{(j)}}{\partial v_{x_t}} = \frac{\sin(\Omega_t^{(j)} T)}{\Omega_t^{(j)}} + g_1^{(j)}(t) \frac{\partial \Omega_t^{(j)}}{\partial v_{x_t}} \tag{56}$$

$$\frac{\partial f_1^{(j)}}{\partial v_{y_t}} = \frac{-(1 - \cos(\Omega_t^{(j)} T))}{\Omega_t^{(j)}} + g_1^{(j)}(t) \frac{\partial \Omega_t^{(j)}}{\partial v_{y_t}} \tag{57}$$

$$\frac{\partial f_2^{(j)}}{\partial v_{x_t}} = \frac{(1 - \cos(\Omega_t^{(j)} T))}{\Omega_t^{(j)}} + g_2^{(j)}(t) \frac{\partial \Omega_t^{(j)}}{\partial v_{x_t}} \tag{58}$$

$$\frac{\partial f_2^{(j)}}{\partial v_{y_t}} = \frac{\sin(\Omega_t^{(j)} T)}{\Omega_t^{(j)}} + g_2^{(j)}(t) \frac{\partial \Omega_t^{(j)}}{\partial v_{y_t}} \tag{59}$$

$$\frac{\partial f_3^{(j)}}{\partial v_{x_t}} = \cos(\Omega_t^{(j)} T)) + g_3^{(j)}(t) \frac{\partial \Omega_t^{(j)}}{\partial v_{x_t}} \tag{60}$$

$$\frac{\partial f_3^{(j)}}{\partial v_{y_t}} = -\sin(\Omega_t^{(j)} T) + g_3^{(j)}(t) \frac{\partial \Omega_t^{(j)}}{\partial v_{y_t}} \tag{61}$$

$$\frac{\partial f_4^{(j)}}{\partial v_{x_t}} = \sin(\Omega_t^{(j)} T) + g_4^{(j)}(t) \frac{\partial \Omega_t^{(j)}}{\partial v_{x_t}} \tag{62}$$

$$\frac{\partial f_4^{(j)}}{\partial v_{y_t}} = \cos(\Omega_t^{(j)} T)) + g_4^{(j)}(t) \frac{\partial \Omega_t^{(j)}}{\partial v_{y_t}} \tag{63}$$

with

$$g_1^{(j)}(t) = \frac{T\cos(\Omega_t^{(j)} T) v_{x_t}}{\Omega_t^{(j)}} - \frac{\sin(\Omega_t^{(j)} T) v_{x_t}}{(\Omega_t^{(j)})^2} - \frac{T\sin(\Omega_t^{(j)} T) v_{y_t}}{\Omega_t^{(j)}} + \frac{(-1 + \cos(\Omega_t^{(j)})) v_{y_t}}{(\Omega_t^{(j)})^2}$$

$$g_2^{(j)}(t) = \frac{T\sin(\Omega_t^{(j)} T) v_{x_t}}{\Omega_t^{(j)}} - \frac{(1 - \cos(\Omega_t^{(j)})) x_{y_t}}{(\Omega_t^{(j)})^2} + \frac{T\cos(\Omega_t^{(j)} T) v_{y_t}}{\Omega_t^{(j)}} - \frac{\sin(\Omega_t^{(j)} T) v_{y_t}}{(\Omega_t^{(j)})^2}$$

$$g_3^{(j)}(t) = -\sin(\Omega_t^{(j)} T) T v_{x_t} - \cos(\Omega_t^{(j)} T) T y_{x_t}$$

$$g_4^{(j)}(t) = \cos(\Omega_t^{(j)} T) T v_{x_t} - \sin(\Omega_t^{(j)} T) T y_{x_t}$$

$$\frac{\partial \Omega_t^{(j)}}{\partial v_{x_t}} = \frac{(-1)^{j+1} a_m v_{x_t}}{[(v_{x_t})^2 + (v_{y_t})^2]^{3/2}}$$

$$\frac{\partial \Omega_t^{(j)}}{\partial v_{y_t}} = \frac{(-1)^{j+1} a_m v_{y_t}}{[(v_{x_t})^2 + (v_{y_t})^2]^{3/2}} \qquad for \quad j = 2, 3$$

where a_m and $\Omega_t^{(j)}$ are defined in subsection 3.4.2

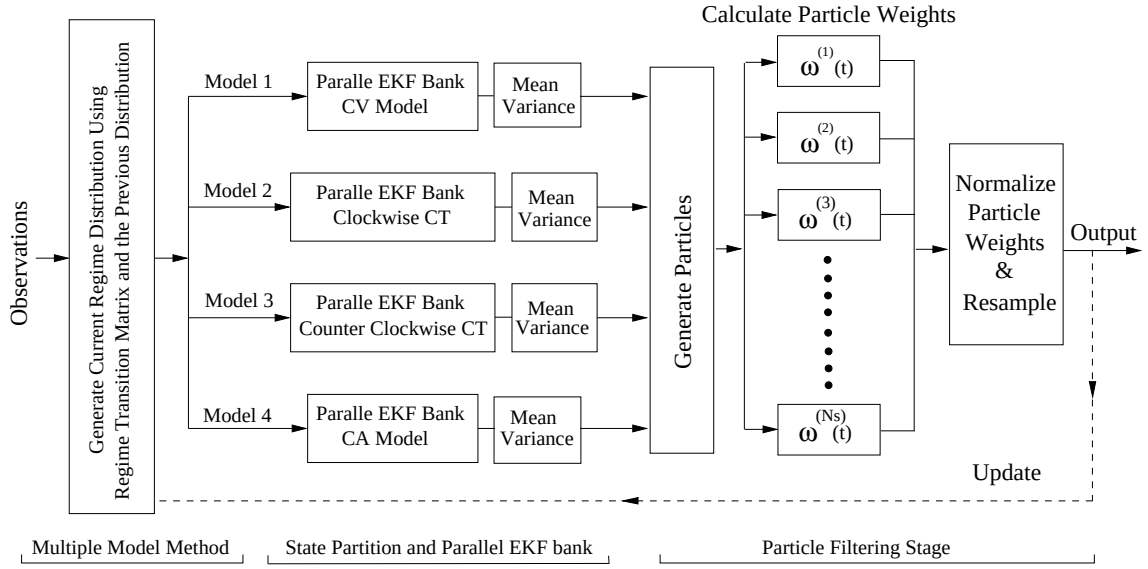


Figure 34: *The block diagram of the proposed multiple model particle filter (MMPF-SP).*

Once the particles are drawn from the aforementioned proposal distribution, the particle weights are evaluated based on the likelihood model discussed earlier. Next, the particle weights are normalized followed by a resampling step. In this research work, the systematic resampling was used. The new MMPF-SP algorithm is illustrated in Figure 34 and also summarized in Table 7.

3.5.4 Computational Complexity Analysis

As noted in the current literature, many researchers approximate the number of operations required for their algorithms as they prepare for real-time implementations [85]-[88]. We follow a similar approach and also approximate the number of operations required for our method. As indicated in Figure 34, the MMPF-SP contains three stages. The first stage analyzes the previous model associated with each particle in the previous iteration, and calculates the current model based on the regime transition matrix defined in equation (40). There are approximately $\mathcal{O}(N_s)$ operations in this stage, where N_s is the sample size, *i.e.* the number of particles. In the second stage, the state partition technique is applied to the four hypothetical models (or regimes).

Table 7: Algorithm 4: Multiple Model Particle Filter with a State Partition

- Step 1: Generating a set of model samples using model transition matrix Π and regime samples at previous iteration, see equation (40) and Section 3.4.1.

- Step 2: Generate particles based on the four models given in equations (34),(42),(44) and using SP-PEKF (28)-(30). Then calculate the particle weights based on the state transition prior, the likelihood and the proposal using equation (53).

- Step 3: Resampling the particles by using the same procedure discussed earlier, and output the estimation result $x(t)$ according to $x(t) \approx \frac{1}{N_s}\sum_{i=1}^{N_s} x^{i\star}(t)$ and update the proposal.

Assuming that each EKF has K operations and this number is model-dependent, then there are $\mathcal{O}(4NK)$ operations in the second stage, where N is the number of filter bank channels. In the last stage, the particle weights are calculated with approximately $\mathcal{O}(N_s)$ operations, and if the basic systematic resampling method is used, as indicated in [85], there will be another $\mathcal{O}(N_s)$ operations. So there are approximately a total of $\mathcal{O}(4NK) + 3\mathcal{O}(N_s)$ operations in the MMPF-SP algorithm.

This proposed algorithm is highly parallizable, which will be beneficial for fast hardware implementation. More specifically, the parallelism can be implemented at three different levels: (1) top level: parallelizing the state partition method applied to the four hypothetical models, (2) medium level: inside each state partition method, parallelizing the filter bank channels, and (3) low level: parallelizing the calculation of particle weights. However, the focus of this chapter is to propose the theoretical foundation for the new particle filter tracking algorithm (MMPF-SP). Further analysis about computational complexity, code optimization, and hardware parallel implementation will be presented in future research.

3.6 Laboratory Experiments and Results

In this section, the problem of tracking a human moving in cluttered environments is studied. This topic has major applications in intelligent video surveillance systems [13] and human computer interfacing [16]. Furthermore, our algorithm can certainly be applied to other applications, such as target tracking in infrared data, by only minor modifications of the observation model. Extensive experiments have been conducted to test the accuracy and robustness of the new proposed algorithm. The experimental results are presented in this section. In order to show the effectiveness of the new algorithm, first the new single model particle filter (PF-SP-PEKF) is applied to a test video which was recorded in a standard laboratory, and compare it to the common condensation method (also known as the bootstrap method). Next, the new MMPF is tested for the following three different cases: (1) tracking a target which has significant complex motions, such as sudden maneuvers, (2) tracking a target with complex motions and with partial occlusions, and (3) tracking a football player with strong background clutter, occlusions, change of poses and camera ego-motions.

3.6.1 Single Model PF (PF-SP-PEKF)

In the first example, we use the single model PF-SP-PEKF algorithm to track a person walking in a highly cluttered laboratory environment. As for a comparison, the standard *condensation* method is also implemented. Tracking results for the two filters are shown in Figure 35 and Figure 36, respectively, where the estimated target centroid is indicated by a white cross. For the first 20 frames the motion was nearly rectilinear with minimal acceleration and both filters performed well. Beginning around frame 25, however, the motion became more complicated with nontrivial accelerations and increased background clutter around the head. While the PF-SP-PEKF filter maintained a consistent track lock throughout the video sequence, however, it can be seen in Figure 35 that the condensation filter progressively diverged in frames 30 through 50, ultimately locking onto the clutter structure and losing the tracked object all together. These results demonstrate the performance advantage that can be gained

with the PF-SP-PEKF approach by explicitly considering the most recent observation when constructing the proposal distribution.

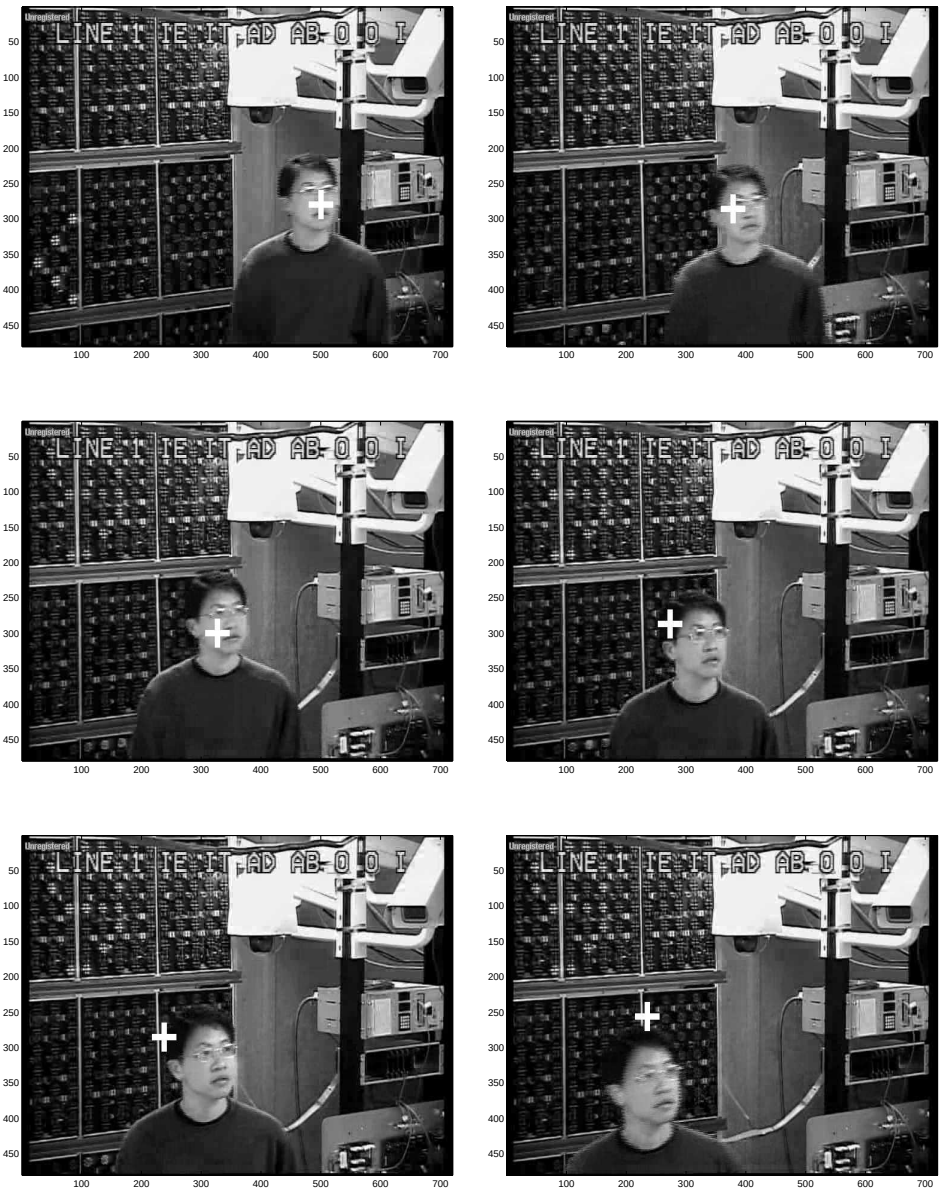


Figure 35: *Using the condensation method: frames 1, 10, 20, 30, 40, 50 are shown here. Divergence is observed in frame 30 with track loss by frame 50.*

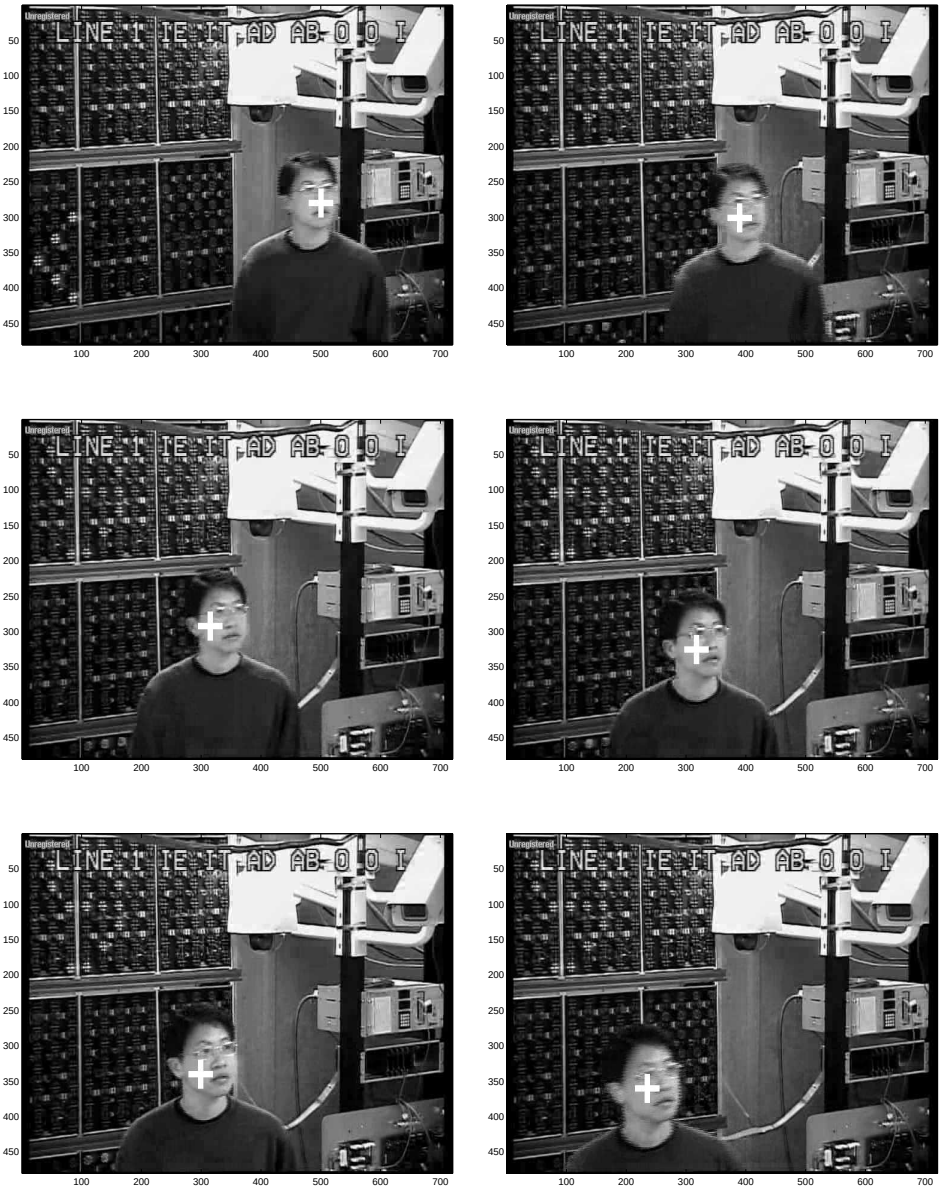


Figure 36: *Using the single model particle filter (PF-SP-PEKF): frames 1, 10, 20, 30, 40, 50 are shown here. The track is maintained throughout the video.*

3.6.2 MMPF-SP: Case No. 1

In this section, the new multiple model particle filter (MMPF-SP) is applied to the visual target tracking problem. In addition, the condensation method is also used for a comparison. The video data used in this simulation is recorded in a standard office. The purpose in this experiment is to track the person's head as it takes four maneuvers in this video. More specifically, in about the first 10 frames in this video, the target has an approximate rectilinear motion. At this stage, both trackers keep a close track. The first maneuver takes place between frames 11 and 15, when the target begins to have a moderate acceleration with a change of the heading direction. Then, the target takes the second maneuver between frames 30 to 40 as it de-accelerates and stops. The third maneuver happens from frame 55, when target begins a sudden and dramatic move to the left side in the image plane, and waves back during frames 70-80. This maneuvering is illustrated by a large positive acceleration. Finally, the fourth maneuver takes places as the target suddenly stops around frame 85.

The new proposed tracking algorithm has been extensively tested. One hundred Monte Carlo runs were implemented to generate the statistical performance, and the root mean square error (RMSE) was used as the performance index. In the simulations, the condensation method gives unstable performances. More specifically, sometimes the condensation method can keep a good track up to frame 55 but it fails when the target makes the third maneuver. But in some other realizations, the condensation method can only track for about 10 frames. On the other hand, the new tracking algorithm keeps a close track in every realization. The RMSE of estimated locations and velocities generated by the MMPF-SP are shown in Figure 37 and Figure 38. As demonstrated from the simulation results, the new tracking algorithm gives very accurate estimations for both target locations and velocities.

In addition, the simulation results from one typical realization are provided here. Figure 40 depicts the tracking results of the MMPF-SP algorithm, while Figure 39 depicts the results of the condensation method for a comparison. The condensation

method loses track after the first few frames, so only a few frames are shown. As indicated in these figures, the MMPF-SP tracker always maintain a close track throughout the entire sequence of frames, in spite of sudden maneuvers made by the target. Here a variety of frames are illustrated to highlight the dynamics of the maneuvers. Figure 41 illustrates the estimated target center and its true locations, while Figure 42 exemplifies the relation between the estimated and true velocities. Next, to further demonstrate the accuracy and the robustness of the new tracking algorithm, two more examples are discussed in the subsequent subsections.

3.6.3 MMPF-SP: Case No. 2

There are two targets present in the second example. We track one of the two targets, and treat another as background clutter. The challenge of tracking the target in this video comes from two sources: first, the target under track has relatively complex motions. It is still for about the first ten frames, then it moves to the left side with a nontrivial acceleration followed by a deceleration, and moves back. Second, the target under track was partially occluded for several times. The upper portion of Figure 43 demonstrates one realization of the MMPF-SP method, while the lower portion of Figure 43 depicts one realization of the condensation method. As indicated in this figure, the condensation method lost the target when the two targets crossing over, *i.e.* when the occlusion occurs. Despite there is no special algorithm to deal with the occlusion problem, our new method can effectively handle the disturbances caused by the partial occlusions (frames 20-27 and frames 80-90), and the tracker closely followed the target throughout the whole video.

3.6.4 MMPF-SP: Case No. 3

Recently, there has been extensive research on pursuit of individuals within video sequences. In practical settings there are many factors that contribute towards the uncertainty in an object's exact location and configuration [89]. One such setting is the tracking of athletes on an outdoor playing field. A variety of researchers around the

world have benchmarked their tracking algorithms in these complicated environments [83, 90, 91]. The new tracking algorithm proposed in this paper is tested in a similar manner. The video was obtained from the website of the University of Oklahoma. The objective in this case is to track the head of one specific player running while crossing the football field. This video is especially challenging due to the constantly changing back ground, substantial background clutter, occlusions, changes in the target scale and pose, and strong camera ego-motions. More specially, after a couple of occlusions at the beginning of the video, the target starts to move at a high speed. During this processing, the background, which has similar color to the target, changes dramatically. Also during this process, strong camera ego-motions occurred due to the cameraman try to keep the player at the center of the image plane. Meanwhile, the target gradually changes it pose. Despite all of these challenges, our new tracker can always maintain a close track. The simulation results are shown in the Figure 44. The standard condensation method is also tested for this video. However, as demonstrated by the simulation result, the condensation method fails to provide a close track for every implementation.

3.6.5 Summary

In this section we propose a new visual target tracking algorithm which can to be applied to an intelligent surveillance system. The novelty of this algorithm lies in the fact that unlike other visual target tracking methods which use single dynamic model, the new algorithm uses a switching state space model and a jump Markov process to approximate the true target dynamics. In addition, the technique of state partition and parallel EKFs are used to generate the proposal distribution used in the particle filter, which in turn further improves the tracking results. This algorithm was extensively tested on different tracking scenarios, which contain complex target motion, strong background clutter, occlusion, changes of pose and strong camera ego-motion. When compared to the standard condensation method, our new algorithm gives a much more robust and reliable tracking result for all of these videos.

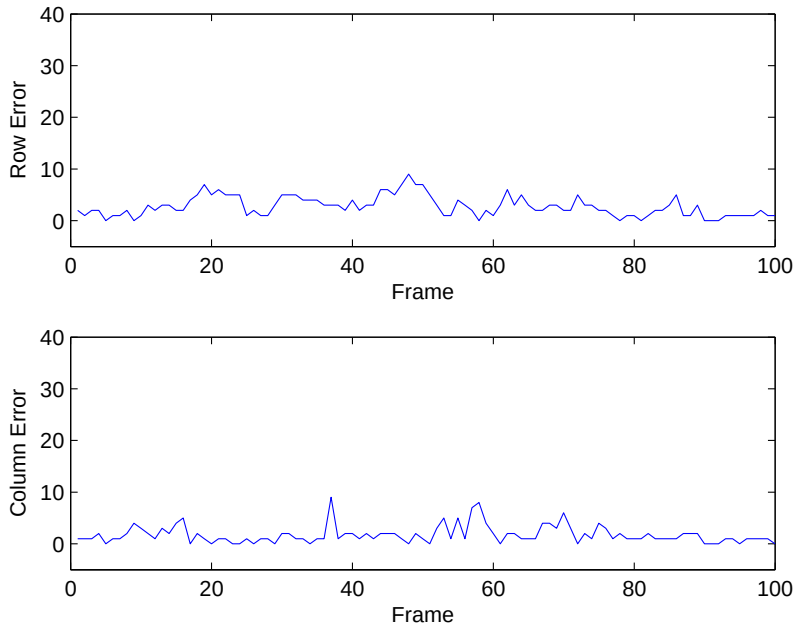


Figure 37: *MMPF-SP (Case 1): Estimation errors for the row and the column indices. The errors are terms of pixels.*

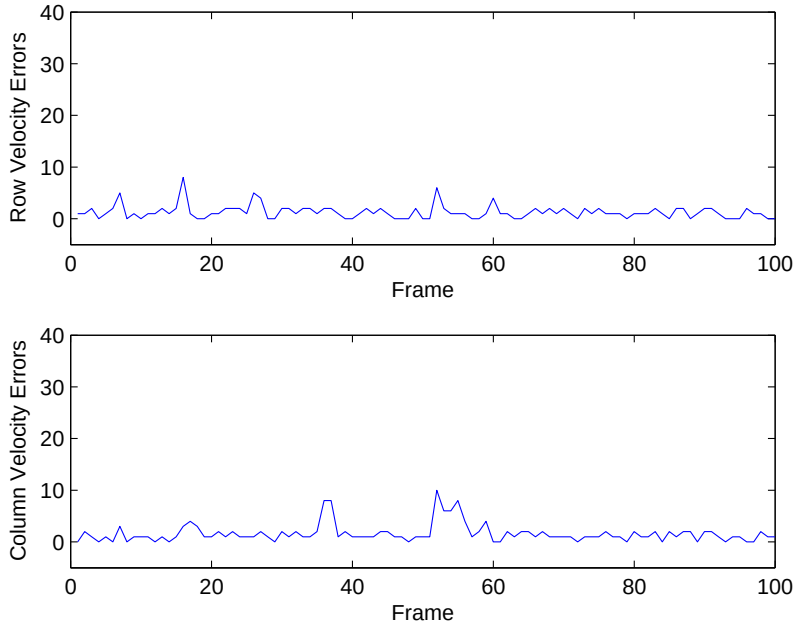


Figure 38: *MMPF-SP (Case 1): Estimation errors for the velocities. The errors are terms of pixels.*

Figure 39: *The tracking result for the second visual tracking example using the condensation method. Frames 1, 5, 11, 13, 15, 20 are provided here. As shown in these figures, the condensation method fails to provide reliable track.*

Figure 40: *MMPF-SP (Case 1): The tracking result for the second visual tracking example. These illustrate the tracking results using our MMPF-SP. Frames 1, 10, 15, 20, 30, 40, 50, 57, 60, 67, 75, 78, 80, 87, 97 are shown here. As shown here, the MMPF-SP algorithm keeps close track, while the condensation method fails to provide reliable track.*

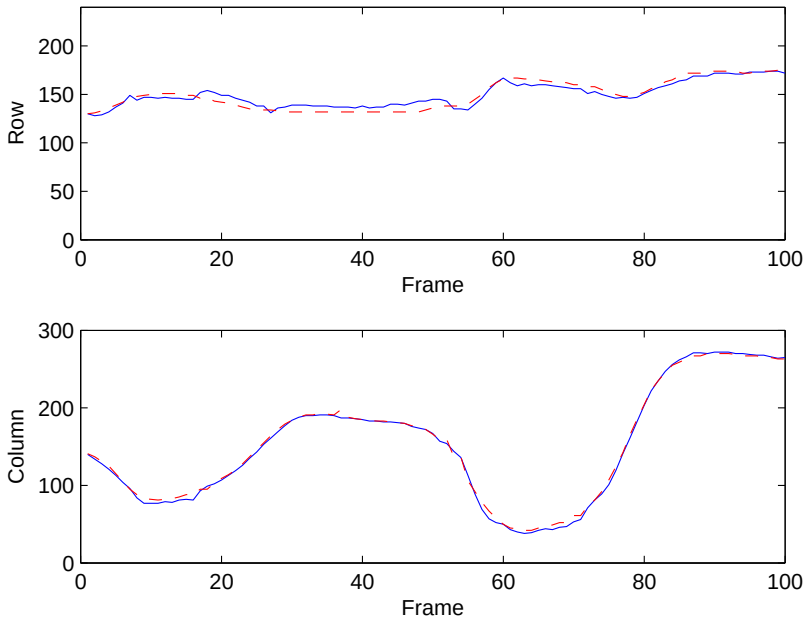


Figure 41: *MMPF-SP (Case 1): the estimated target centers (solid blue line) vs. the true target centers (red dashed line).*

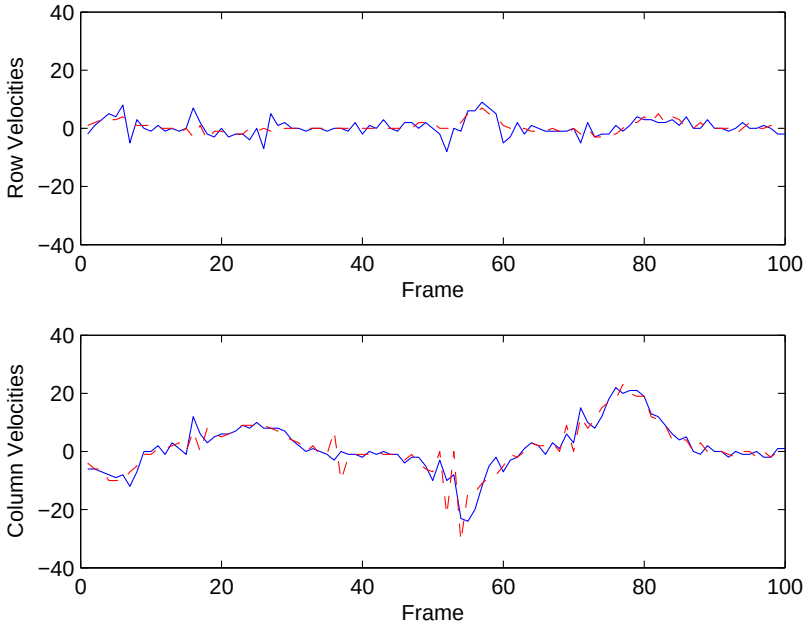


Figure 42: *MMPF-SP (Case 1): the estimated velocities (solid blue line) vs. the true velocities (red dashed line).*

The tracking results using the MMPF-SP algorithm.

Frames 10, 20, 24, 27, 58, 78, 86, 92, 115 are shown here.

The tracking results using the condensation method.

Frames 10, 15, 22, 25, 30, 40 are shown here.

Figure 43: *MMPF-SP (Case No. 2): The tracking result for the second example. The top two rows illustrate the tracking results using our MMPF-SP, while the bottom two row show the results by using traditional condensation method. As shown here, the MMPF-SP algorithm can closely follow the target despite partial occlusions, while the condensation tracker diverges when occlusion occurs.*

Figure 44: *MMPF-SP (Case No. 3): Tracking one football player in the video. Despite the substantial background clutter, occlusions, and strong camera ego-motions, a robust track lock is maintained. Frames 2, 7, 9, 18, 20, 30, 35, 40, 46, 50, 60, 65, 70, 75, 78 are shown here.*

CHAPTER 4: IMPROVING PARTICLE FILTER USING GALERKIN'S PROJECTION METHOD

In Chapter 3, a new particle filter framework was developed based on the technique of state-partitioning and parallel EKFs. Also, this chapter provides two modified particle filter algorithms for multiple sensor and multiple measurements, respectively. However, there are problems associated with these algorithms. First, to implement the parallel EKFs, the nonlinear system has to be linearized, which may result in large modeling error. Second, the proposal distribution of these particle filters is constructed by using the weighted Gaussian sum generated for the SP-PEKF. This idea assumes the proposal is a Gaussian distribution, which is not true in general. In this chapter, we present a new particle filter (PF) algorithm, which uses a mathematical tool known as Galerkin's projection method to generate the proposal distribution. By definition, Galerkin's method is a numerical approach to approximate the solution of a partial differential equation (PDE). By leveraging this method with L^2 theory and the FFT, this new proposal is fundamentally different to various local linearization or Kalman filter based proposals.

To test its performance, we first apply this algorithm to the bearings-only tracking problem, and then use it for a visual target tracking. This chapter is based on our previous work [18][92]. As we discussed earlier, the bearings-only target tracking is a fundamental component of many engineering applications. Due to the inherent nonlinearities in the observation model, bearings-only target tracking has become a standard nonlinear filtering problem that receives intensive investigations. Traditionally, various Kalman filter based tracking techniques have been used. However, for many cases these methods cannot provide satisfactory results. More recently, particle filtering techniques have received increasing attention. Inspired by the work [24], many particle filter based tracking algorithms were proposed [3][23]. Bearing the nature of sequential importance sampling and the Monte Carlo approach, particle filtering has emerged as a superior alternative to the traditional tracking methods.

This chapter discusses a new particle filter algorithm in which the Galerkin's projection method is used to generate the proposal distribution. Applying the Galerkin's method to the nonlinear filtering problem has been reported in [93] [94]. The rationale behind Galerkin's method is to assume the state posterior distribution is in L^2 space. In this case, this distribution can be approximated by its projection onto a finite set of orthogonal basis vectors. At each iteration, it only needs to update the projection on each basis vector to approximate the true proposal distribution. In addition, by choosing a set of special exponential basis vectors, the projection can be approximated by the FFT which is computationally efficient to implement. Finally, we use this approximated distribution as the proposal in the new PF algorithm. This proposal does not require any local linearization of the nonlinear system and does not have any Gaussian assumption of the systems' states when calculating the proposal, which differs fundamentally to various Kalman filter based PF algorithms in [21][25][95]. As shown in the theory and indicated by our simulations, this proposal renders more support from the true posterior distribution, thereby significantly enhanceing the estimation accuracy compared to standard bootstrap filters. In addition, because of this improved proposal distribution, the new particle filter can achieve a given level of performance with less sample size.

4.1 Particle Filtering Based On The Galerkin's Method

In this section, we present a new particle filter algorithm based the Galerkin's projection method. First, this section provides a brief review of the Galerkin's projection method and shows how it can utilized to construct the proposal distribution for a particle filter. Then, the new particle filter algorithm will be formulated.

4.1.1 Generating The Proposal Using Galerkin's Method

Galerkin's method is a numerical approach to approximate the solution of a partial differential equation (PDE) [93][94]. We let

$$\mathcal{P}(x,t) = 0$$

denotes a PDE, which is a function of both temporal variable t and spatial variable x. The basic idea of Galerkin's method is to assume that $p(x,t)$ is the *solution* of the above PDE, and it is in the L^2 space, such that it can be decomposed by the following equation:

$$p(x,t) = \sum_{l=0}^{\infty} \epsilon_l(t)\phi_l(x) \quad , \tag{64}$$

where $\{\phi_l(x)\}_{l=0}^{\infty}$ is a set of complete orthogonal basis of the L^2 space and $\epsilon_l(t)$ is the projection of $p(x,t)$ onto basis $\phi_l(x)$ at time t defined by the inner product $\langle p(x,t), \phi_l(x)\rangle$ as:

$$\langle p(x,t), \phi_l(x)\rangle = \int p(x,t)\phi_l(x)^* dx \quad . \tag{65}$$

Our objective is to find an approximation of $p(x,t)$, denoted as $\hat{p}(x,t)$, such that

$$\hat{p}(x,t) = \sum_{l=0}^{N-1} c_l(t)\phi_l(x) \quad . \tag{66}$$

Note that the approximation error arises from the replacement of infinite basis with a set of finite orthogonal basis. The projections $c_l(t), l = 0, \cdots, N-1$, are the values to be determined. Having this setup, we can project $\mathcal{P}(x,t)$ onto the subspace $\mathrm{span}\{\phi_l(x)\}_{l=0}^{N-1}$ as:

$$\langle \mathcal{P}(x,t),\ \phi_l(x)\rangle = 0, \qquad l = 0, \cdots, N-1 \quad . \tag{67}$$

By doing this, the problem of solving the PDE is converted into solving N ordinary differential equations (ODE). Next we apply this method to the nonlinear filtering problem defined in equations (6) and (7) and also provide here.

The prediction distribution (the Chapman-Kolmogorov equation):

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})\ d\mathbf{x}_{t-1} \tag{68}$$

The updated distribution (the Bayesian equation):

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})}{\int p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})d\mathbf{x}_t} \tag{69}$$

Initially, it is assumed that $p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$ has the following form:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \sum_{l=0}^{N-1} \tilde{c}_l(t)\phi_l$$

where $\tilde{c}_l(t)$ will be determined later. For simplification of notations, we drop the variable x. Then we apply Galerkin's method to equation (69) by projecting it onto $\text{span}\{\phi_l(x)\}_{l=0}^{N-1}$ as:

$$\begin{aligned}\langle p(\mathbf{x}_t|\mathbf{z}_{1:t}), \phi_k\rangle &= \sum_{l=0}^{N-1} c_l(t)\langle\phi_l, \phi_k\rangle \\ &= \frac{\sum_{l=0}^{N-1} \tilde{c}_l(t)\langle p(\mathbf{z}_t|\mathbf{x}_t)\phi_l, \phi_k\rangle}{\sum_{l=0}^{N-1} \tilde{c}_l(t)\langle p(\mathbf{z}_t|\mathbf{x}_t), \phi_l^*\rangle}\end{aligned} \tag{70}$$

where $k = 0, \cdots, N-1$. For simplification, the above equation can be written in a matrix form as:

$$\mathbf{C}(t) = \frac{\boldsymbol{\Upsilon}_t\tilde{\mathbf{C}}(t)}{\upsilon_t^T\tilde{\mathbf{C}}(t)} \tag{71}$$

where $\boldsymbol{\Upsilon}_t$ is a $N \times N$ matrix with the element at k^{th} row and l^{th} column given by

$$[\boldsymbol{\Upsilon}_t]_{k,l} = \langle p(\mathbf{z}_t|\mathbf{x}_t)\phi_l, \phi_k\rangle \quad .$$

The variables $\mathbf{C}(t)$, $\tilde{\mathbf{C}}(t)$ and υ_t are $N \times 1$ vectors, with

$$[\upsilon_t]_l = \langle p(\mathbf{z}_t|\mathbf{x}_t), \phi_l^*\rangle \quad .$$

Now we choose the exponential basis as the following form:

$$\phi_l(x) = \frac{1}{\sqrt{b-a}} \exp\left(j2\pi l\frac{x-a}{b-a}\right)$$

where a and b are the integral limits. It has been shown in [93] that by using this set of basis the inner product can be approximated by FFT as follows:

$$\begin{bmatrix} \langle p(x), \phi_0 \rangle \\ \vdots \\ \langle p(x), \phi_{N-1} \rangle \end{bmatrix} \approx \frac{\sqrt{b-a}}{N} \text{FFT}[p(x)] \quad ,$$

$$\begin{bmatrix} \langle p(x), \phi_0^* \rangle \\ \vdots \\ \langle p(x), \phi_{N-1}^* \rangle \end{bmatrix} \approx \sqrt{b-a}\ \text{IFFT}[p(x)] \quad ,$$

see [93] [94] for details. Then, equation (71) can be approximated by using FFT as follows:

$$[\boldsymbol{\Upsilon}_t]_l = (\sqrt{b-a}/N)\ \text{FFT}\ [p(\mathbf{z}_t|\mathbf{x}_t)\ \phi_l] \tag{72}$$

$$\upsilon_t = \sqrt{b-a}\ \text{IFFT}\ [p(\mathbf{z}_t|\mathbf{x}_t)] \quad . \tag{73}$$

To evaluate $\tilde{c}_l(t)$, we apply the Galerkin's method in a similar way to equation (6). Then we have:

$$\tilde{c}_l(t) \approx (\sqrt{b-a}/N)\text{IFFT}_l\big[c_l(t-1)\text{FFT}_l[p(\mathbf{x}_t|\mathbf{x}_{t-1})]\big] \ , \tag{74}$$

where the $\text{FFT}_l[\cdot]$ represents the l^{th} bin of FFT of the argument. In addition, $\tilde{c}_l(t)$ can also be calculated by:

$$\tilde{\mathbf{C}}(t) = \text{FFT}\left[\sqrt{b-a} \cdot p(\mathbf{x}_t|\mathbf{x}_{t-1})\ \text{IFFT}[\mathbf{C}(t-1)]\right] \quad . \tag{75}$$

Moreover, the prediction distribution and posterior distribution can be calculated by:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) \approx (N/\sqrt{b-a})\ \text{IFFT}[\tilde{\mathbf{C}}(t)] \tag{76}$$

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) \approx (N/\sqrt{b-a})\ \text{IFFT}[\mathbf{C}(t)] \quad . \tag{77}$$

As a summary, in order to approximate the posterior distribution $p(\mathbf{x}_t|\mathbf{z}_{1:t})$, we only need to update the vector $\mathbf{C}(t)$ at each iteration.

4.1.2 The Proposed New PF Algorithm

This section shows how Galerkin's method can be incorporated within the particle filter framework. More specifically, at each iteration, we use $\tilde{\mathbf{C}}(t)$ and $\mathbf{C}(t)$ to approximate the posterior distribution leveraging the IFFT defined in (77), then draw particles from this approximated distribution. After that, the particles' weights will be evaluated. The final step is the resampling and update stage. Since the proposal is generated by projecting the true posterior distribution onto a subspace of L^2 space, the accuracy of the proposal is guaranteed by choosing appropriate the number of basis. As indicated in the simulations, using a limited number of basis is good enough to generate accurate proposal which in turn significantly reduces the number particles used to achieve a given level of performance. The detailed algorithm is summarized in Table 8.

Table 8: Algorithm 5: The Galerkin Method Based PF Algorithm

- Sequential Importance Sampling (SIS) Step:
  - Calculate the parameters $\tilde{\mathbf{C}}(t)$ and $\mathbf{C}(t)$ using equations (72) to (74) or (75);
  - Generate the proposal distribution using equation (77);
  - Sample from the proposal distribution according to

$$\begin{aligned} x_t^{(i)} &\sim q(x_t^{(i)}|x_{0:t-1}^{(i)}, y_{1:t}) \\ &\approx p(\mathbf{x}_t|\mathbf{z}_{1:t}) \\ &\approx (N/\sqrt{b-a})\ \text{IFFT}[\mathbf{C}(t)]; \end{aligned}$$

  - Evaluate and normalize the importance weights according to equation (13);
- Resampling Step: Generate a new set of particles $x_t^{i\star}$ from $x_t^{(i)}$ by sampling N_s times the approximate distribution of so that $Pr\left(x_t^{i\star} = x_t^{(j)}\right) = \tilde{\omega}_t^{(j)}$;
- Output and Update Step: Approximate x_t by $\hat{x}_t \approx \frac{1}{N_s}\sum_{i=1}^{N_s} x^{i\star}(t)$ and update the proposal.

4.2 Application: Bearings-only Target Tracking

In this section, the proposed particle filter is applied to a single sensor bearings-only tracking problem. The constant velocity model is used in this section, which is discussed in Section 3.2. Here, we provide two tracking examples to test the performance of the proposed particle filter tracking algorithm. In the first example, the target has an approximate rectilinear motion. While the target in the second example has a maneuver operation. In addition, for the purpose of comparison, an extended Kalman filter (EKF) and a bootstrap filter (a PF algorithm using state transition prior as proposal) are also implemented. The initial conditions of for the first example are given as

$$\mathbf{X}_1 = \begin{bmatrix} 0 & 50 & 1 & -1 \end{bmatrix}^T$$

and

$$\mathbf{Q}_v = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix} \quad \sigma_n^2 = 0.01 \quad .$$

In the first example, the new PF algorithm uses 100 basis and 200 particles, *i.e.* $N = 100, N_s = 200$. While the bootstrap filter uses the sample size $N_s = 200, 500, 1000, 2000$, respectively. To evaluate the tracking performance of these filters, 200 Monto Carlo runs were implemented. In addition, the root mean square error (RMSE) along both x-axis and y-axis are used as performance indices. The tracking results of the three filters are shown in Figure 45, Figure 46 and Table 9. In Figure 45, the RMSE's vs. time from the 200 Monte Carlo run are plotted. It is obvious from this figure, the new PF gives much more accurate estimation compared to the bootstrap filter using the same sample size. In addition, the tracking results from a typical realization is shown in Figure 46. As shown in this figure, the bootstrap filter can follow the general direction of the target, and it fails to detect small maneuvering operations made by the target. On the other hand, due to the improved proposal distribution, the new proposal PF can keep a close track throughout the whole simulation. Next, for a comparison, bootstrap filters with different sample size are implemented 200 times to generate the performance index. Table 9 summarizes the mean and the variance of the RMSE's

for each bootstrap filter. It is observed from this table that even a bootstrap with a sample size of 2000 still cannot achieve the estimation accuracy of the new proposed PF algorithm.

In the second example, three different filters were implemented to track a target which makes a sharp turn approximately at the location (x=72, y=6). This kind of target is always referred as a maneuvering target. Tracking a maneuvering target is more challenging than tracking a target with constant velocities. Traditionally, multiple model methods are used in this case. However, to test the accuracy and robustness of our new PF, we still use the single constant velocity model given before. The tracking result of one typical realization is shown in Figure 47. As indicated in this figure, both the EKF and the bootstrap filter diverges from the true trajectory when the target makes the sharp turn. However, the new PF still can keep a close track.

Table 9: A Comparison of the Projection-based PF and the Bootstrap Filter

Filter (Sample Size)	Mean RMSE		Var RMSE	
	e_x	e_y	Var e_x	Var e_y
Bootstrap (200)	4.8312	4.0215	6.7798	5.7492
Bootstrap (5000)	4.0599	3.8300	4.0483	6.0063
Bootstrap (1000)	3.6181	3.6961	3.3728	5.8839
Bootstrap (2000)	3.6199	3.6489	3.3991	6.0310
New PF (200)	1.1391	1.1554	0.0535	0.0576

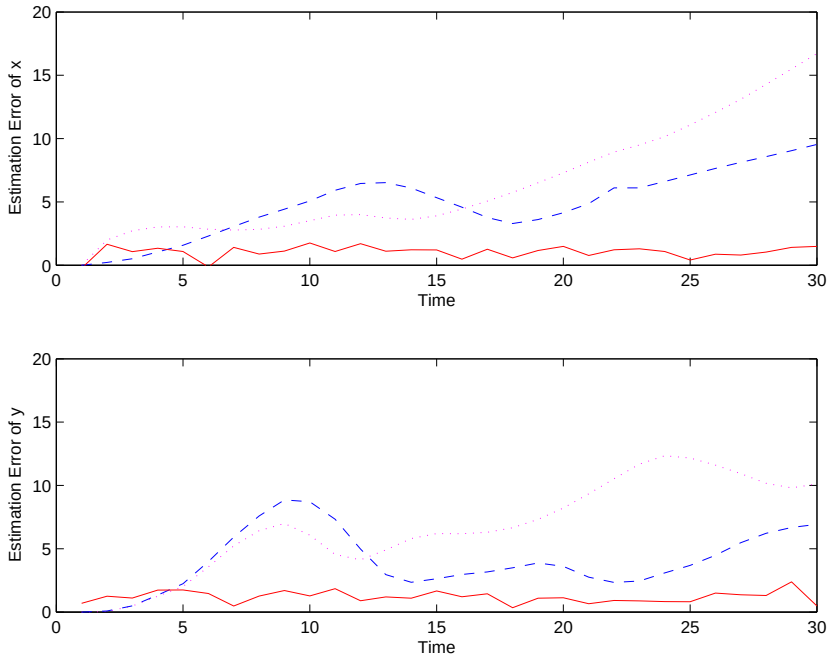


Figure 45: *The estimation RMSE vs. time for both x-axis and y-axis: the solid line(——) denotes the RMSE of the projection based PF; the dashed line (– –) denotes the RMSE of the bootstrap filter with 200 particles; the dotted line (. . .) represents the RMSE of the EKF.*

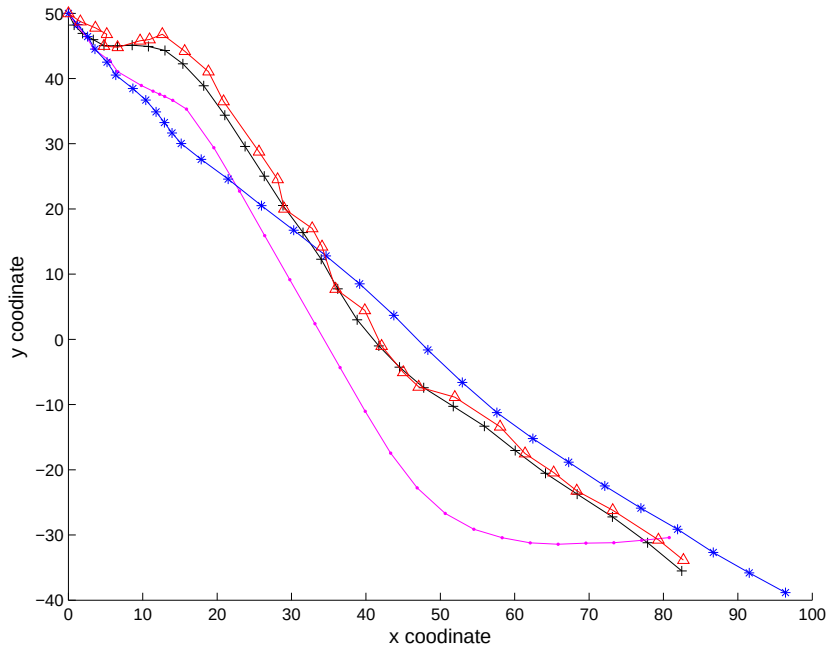


Figure 46: *The first example: the line with a plus sign (–+–) represents the true trajectory; the line with a triangle (–△–) represents the estimated trajectory from the new PF; (–●–) represents the tracking results from EKF; (–∗–) represents the tracking results from bootstrap filter. As seen, the our proposed PF method has the closest track.*

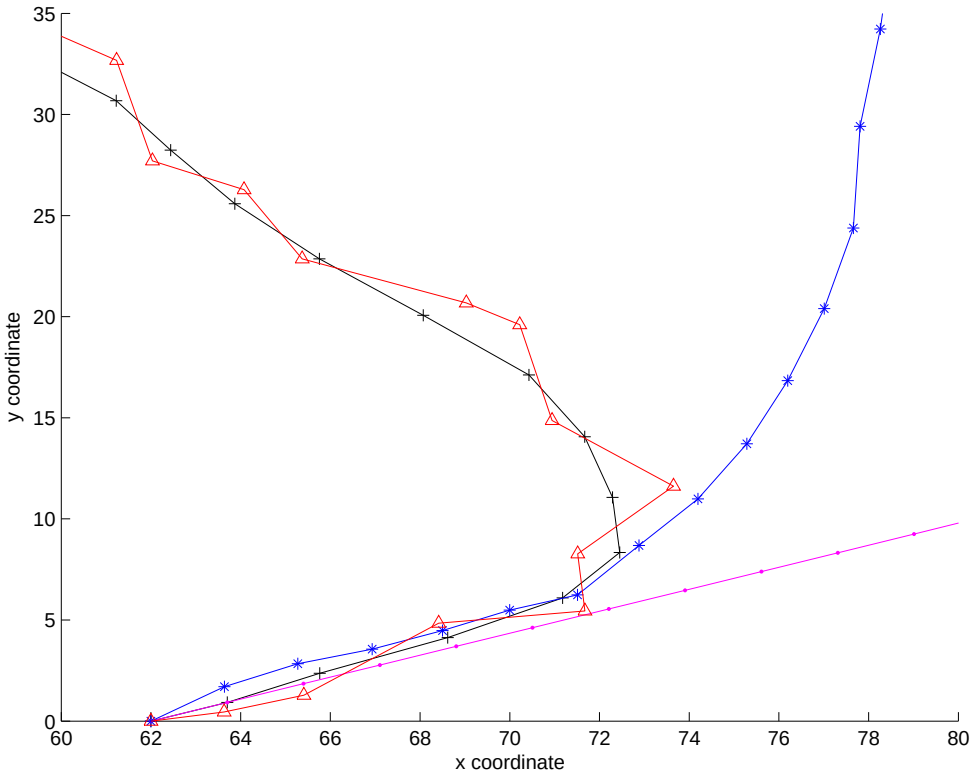


Figure 47: *The second example: The line with a plus sign (−+−) represents the true trajectory; the line with a triangle (−△−) represents the estimated trajectory from the new PF; (−●−) represents the tracking results from EKF; (−∗−) represents the tracking results from bootstrap filter with 500 particles. As seen, the our proposed PF method has the closest track.*

4.3 Application: Visual Target Tracking

As an illustrative example, the projection based PF algorithm is also applied to track a human in a video sequence by using the Langevin process discussed in subsection 3.5.2. To test the performance of our improved PF algorithm, we choose a video in which the target has relatively complex motions, which is same as the one we used for MMPF-SP. The digital video was recorded in a standard laboratory environment with a frame size of 240×320 pixels. Both the *condensation* method and the improved PF algorithm were implemented for a comparison. Tracking results for the two filters are shown in Figure 48 and Figure 49, where the estimated target centroid is indicated by a white cross. In the first 10 frames the target's motion was nearly rectilinear with a constant velocity and both filters performed well. However, from frame 11 to 15, the target begins to have obvious a negative acceleration followed by positive acceleration. The condensation method (with 100 particles) begins to lose its target around frame 11, and it keeps a constant velocity. Finally, the estimated centroid moves out of the image domain. On the other hand, our improved PF (with 100 particles) keeps tracking the target throughout the whole video. Moreover, the target makes another maneuver around frame 35 to frame 40, and the proposed PF algorithm still keeps a close track. Figure 48 and 49 illustrate one typical realization of our simulations. It is demonstrated that because of the improved proposal distribution, the new proposed PF algorithm is able to yield accurate and robust estimations when tracking a target with complex motions. Also, we can see from this example that the particle filter with Galerkin's method can produce approximately the same result as the MMPF-SP for the same test video.

Figure 48: *Visual tracking by using the condensation method. These frames illustrates a dramatic motion. Frames 1, 5, 11, 13, 15, 20 are provided here.*

Figure 49: *Visual tracking using the projection based PF. Frames 1, 5, 11, 13, 15, 20, 25 and 45 are provided here. As shown in this figure, even under several maneuvers, tracking is maintained.*

CHAPTER 5: A PARTICLE FILTER WITH STATE SPACE DISCRETIZATION

In Chapter 4, a particle filter algorithm based on Galerkin's projection method was developed. As demonstrated in our simulations, this algorithm can provide quality estimation results. On the other hand, this algorithm has its own limitations: it requires the computation over the entire state space, where most of the operations are focused on low probability regions. In this chapter, we present a more efficient particle filter framework that only focuses on the high probability regions in the state space. The algorithms of this framework are based on the idea of state-space discretization, which is similar to the well know grid-based method [3]. More specifically, the state space is first partitioned into finite cells, and the cell centers are used to represent the "discretized" state space. Next, the weight of each cell is calculated based on the target's initial distribution. Then only a small number of cell centers are propagated through the target's dynamic system, provided that their weights are above certain threshold. At each iteration, the cells with a large weight represent the high probability area in the target distribution – this implies that the distribution of the state space is nicely supported. The samples (or particles) are drawn from this area, which is serve as the proposal distribution in the new particle filter. As shown in the developed theory and indicated by our simulations, this proposal renders more support from the true posterior distribution, thereby significantly improving the estimation accuracy.

5.1 Generating the Proposal Using State Space Discretization

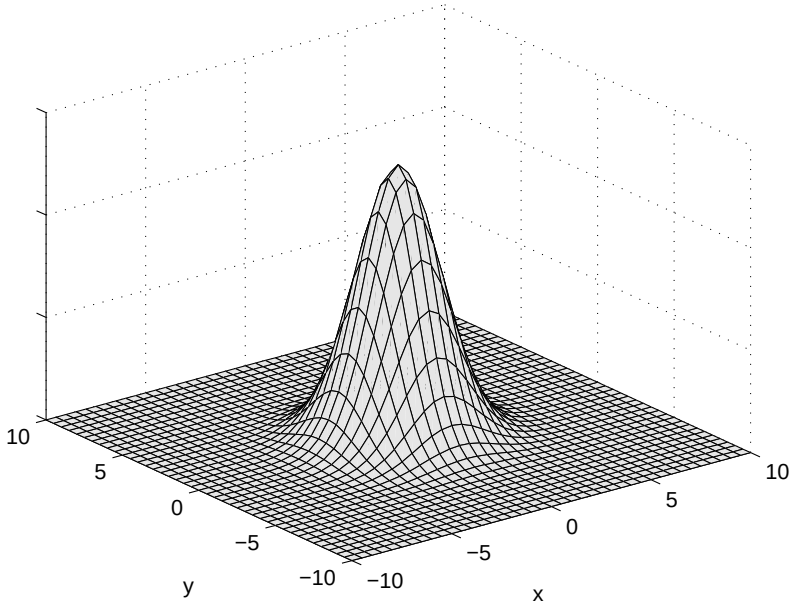
As discussed in the previous chapters, designing a proposal distribution is an important step for implementing a particle filter. Since this distribution governs the particles and their weights, which will be used to represent the true posterior distribution. In this section, a new strategy is presented for calculating the proposal distribution which can effectively capture the high probability area in the target state space without significantly increasing the computation complexity. More specially, we first partition the

target state space into a set of square cells. Each cell is represented by its cell center. For a given target initial distribution, we can evaluate the probability of each cell center, and use the probability as the weight for that cell. Usually, many cells will have negligible weight. In this case, we choose a threshold and only consider the cells whose weights are above this threshold. Then, the cell centers are propagated through the nonlinear system given by (1) to get a new set of cell centers at the next time index. Next, the weights for the new cell centers are evaluated using equations (6) and (7). Since we only consider a finite number of cells, the integrals in these equations can be replaced by summations, and equations (6) and (7) can be rewritten as:

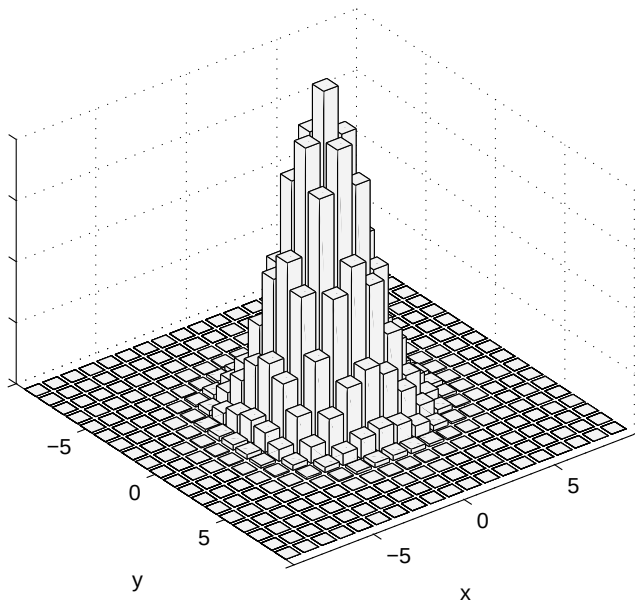
$$\widetilde{W}_{t,j} = \sum_{i=1}^{N_c} p(\mathbf{x}_{t,j}|\mathbf{x}_{t-1,i}) W_{t-1,i} \quad , \tag{78}$$

$$W_{t,j} = \alpha \; p(\mathbf{z}_t|\mathbf{x}_{t,j}) \widetilde{W}_{t,j} \quad . \tag{79}$$

where $\mathbf{x}_{t,i}$ and $W_{t,i}$ represents the center location of the i-*th* cell and its weight at time t, respectively. The variable N_c denotes the total number of cells, and α is a normalizing factor. At this stage, we have a set of cells (*i.e.* N_c cells) centered at $\mathbf{x}_{t,i}$ with their weights defined by $W_{t,i}$, where $i = 1, ..., N_c$. These cells will serve as the proposal distribution of a particle filter. The samples (or particles) will be uniformly drawn from each cell, and the sample number within each cell is proportional to the cell weight. In this way, the 2-D histogram of these samples will have a similar shape of the cells. Now, an example is provided to illustrate the implementation of this algorithm. Assuming a bearings-only tracking scenario, the target is initially located at the origin [0 0] with a 2D Gaussian distribution, and it has an initial velocity of [10 5] units per time increment. Then, we construct a discretized initial condition by using a set of cells and their weights. Figure 50(a) illustrates continuous initial distribution and Figure 50(b) depicts the discrete initial distribution, which is represented by a set of cells and their weights. Next, these cells are propagated through the system by using (34) and (35). The true posterior can be constructed when the DOA is available.

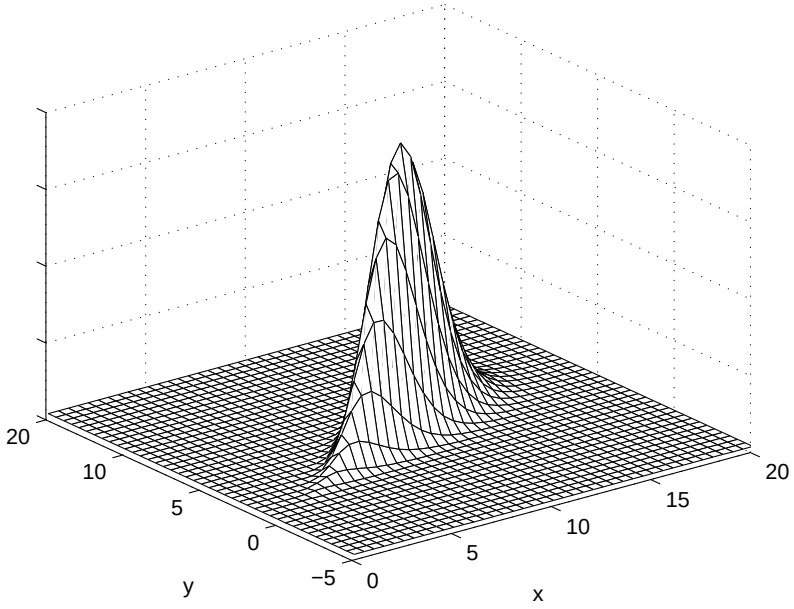


(a)

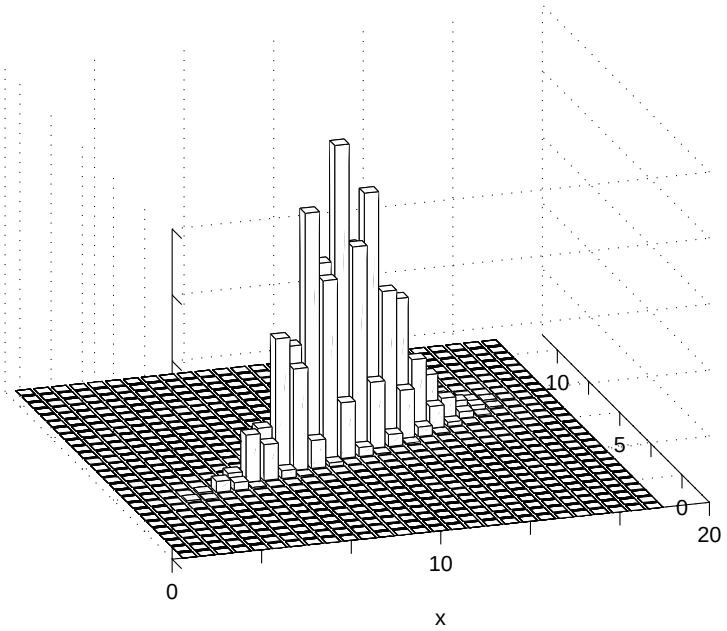


(b)

Figure 50: *The target initial condition: (a) the continuous initial distribution (b) the discretized initial distribution presented by a set of cells and their weights.*



(a)



(b)

Figure 51: *The target posterior distribution: (a) the true posterior distribution (b) the approximate discrete posterior distribution using the weighted cells.*

Figure 51(a) illustrates the true posterior distribution, and Figure 51(b) depicts the approximate discrete posterior distribution. For these figures the weighted cells are used as the proposal distribution. Then particles are uniformly drawn from each cell, and the sample number within each cell is proportional to its weight. The particles are shown in Figure 52. Since the only measurement of the system is the DOA angle, the majority of the particles will be drawn along this angle. As shown in Figure 51 the proposal distribution has significant support from the true posterior distribution. Figure 52 indicates that the particles drawn from this proposal sufficiently represent the high probability area in the state space. Also, if we only use a limited number of cells, the computation complexity for calculating the proposal distribution can be controlled.

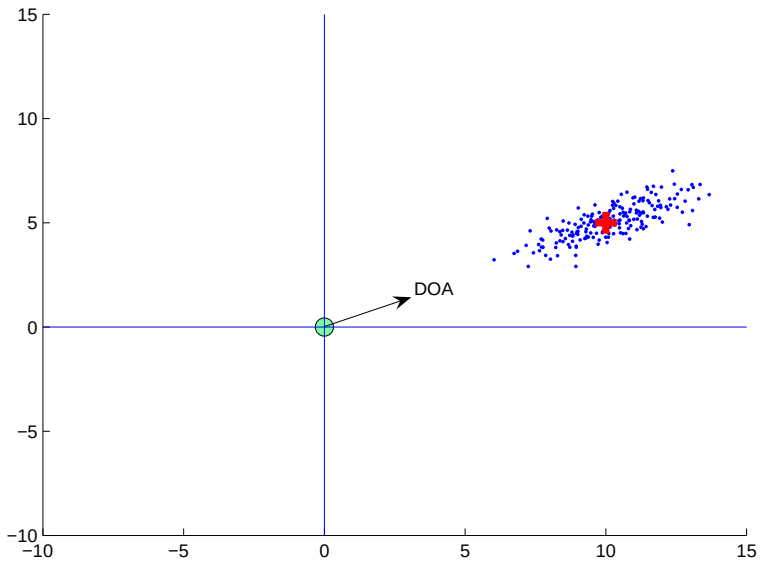


Figure 52: *Particles drawn from the new proposal distribution calculated using the state-space discretization method.*

5.1.1 A New PF with State Space Discretization

As discussed in the previous section, a set of particles will be generated from the proposal distribution, which is actually the discretized posterior distribution. Once the particles are generated, the particle weights will be evaluated based on equation (13). In addition, the target posterior is represented by these weighted particles. When constructing the proposal distribution, it is only required to consider a finite set of cell centers. In this approach, there's no restrictive assumption about the posterior distribution, and there's no need to linearize the system. The accuracy of the proposal distribution and computational complexity can be balanced by choosing different cell sizes and the different thresholds. Also, this algorithm does not require any transforms, in contrast to the particle filter based on the Galerkin's method. The following table summarizes the new particle filter algorithm based on the state-space discretization approach.

Table 10: Algorithm 6: A State Space Discretization Based Particle Filter

- Sequential Importance Sampling (SIS) Step:
  - Calculate the cell centers and the weights using (78) and (79).
  - Construct the proposal distribution using the cells and their weights.
  - Draw particles: uniformly sample within each cell such that the sample number of each cell is proportional to the cell weight.
  - Evaluate particles weights using (13) and normalization.
- Resampling Step: generate a new set of particles $x_t^{i\star}$ from $x_t^{(i)}$ by sampling N_s times the approximate distribution such that: $Pr\left(x_t^{i\star} = x_t^{(j)}\right) = \tilde{\omega}_t^{(j)}$.
- Approximate x_t by $\hat{x}_t \approx \frac{1}{N_s}\sum_{i=1}^{N_s} x^{i\star}(t)$ and update the proposal.

5.2 Application: Bearings-only Target Tracking

In this section, the proposed particle filter is applied to a single sensor DOA tracking problem. In this example, the target has a rectilinear motion, and the DOA measurements are corrupted by additive noise. For the purpose of comparison, an extended Kalman filter (EKF) and a bootstrap filter (a standard PF using the state transition prior as its proposal) are also implemented. In the experiment, the three algorithms are extensively tested. The results are shown in Figures 53 and 54. Figure 53 depicts the target ground truth (the black line) and the estimations from the three filters: the EKF (the green line), the bootstrap filter (the blue line), and the new PF algorithm (the red line). As shown in Figure 53, the EKF displays a non-stable behavior. As indicated in [3], this is because a strong nonlinearity involved in the observation model, and it is a common phenomenon in DOA tracking problem. The bootstrap filter can follow the target in the general direction, but the estimation errors progressively increasing. On the other hand, the new PF gives very accurate estimations. Furthermore, the estimation errors of the bootstrap filter and the new PF are shown in Figure 54.

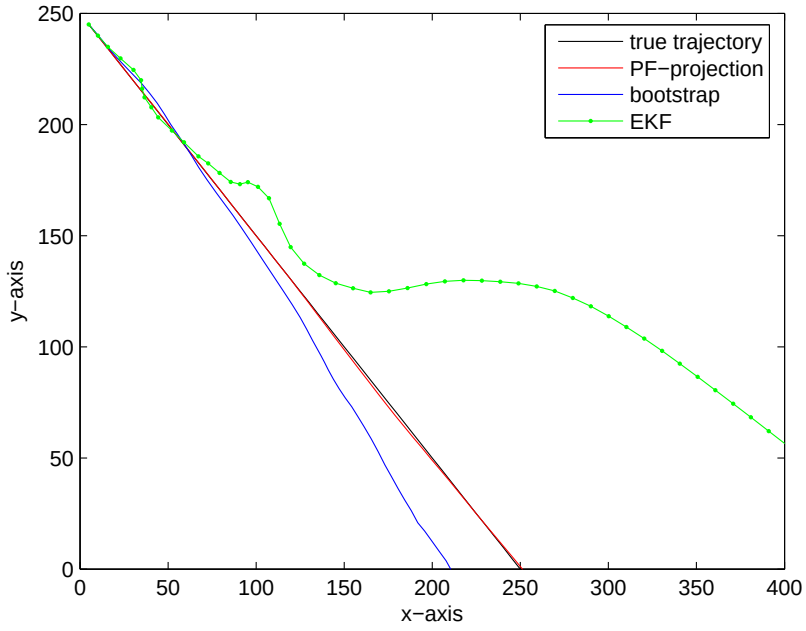


Figure 53: *The true target trajectory (black) and the estimates generated from three filters: EKF (green), bootstrap (blue), the new PF (red).*

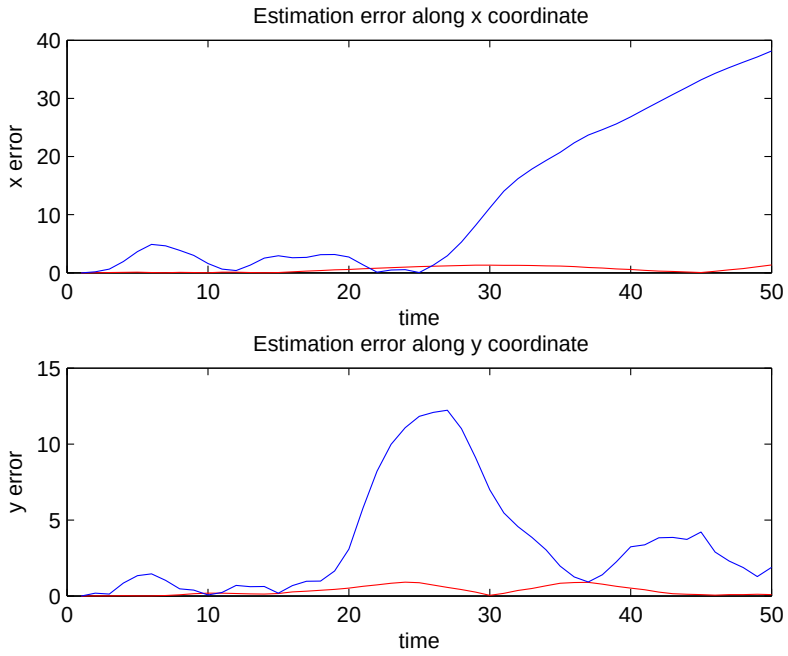


Figure 54: *The estimation errors of the bootstrap filter (blue line) and the new PF algorithm (red line).*

5.3 Application: Target Tracking with Range and Bearings Measurements

In this section, the problem of target tracking with both range and bearings measurement are studied. This is most common type of target tracking. Various coordinate systems have been used in this type of target tracking problem, which include the polar coordinate system, Cartesian coordinate system, and mixed coordinate systems [96]. Among them, the most widely used approach is tracking in the mixed coordinates. More specifically, the target observation model in this method is given as follows:

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \mathbf{v}_t \tag{80}$$

where $\mathbf{x}_t$ is the target state that describes the target kinematics. This state vector and the target motion are defined in the Cartesian system, which are studied in Section 3.4. On the other hand, the system measurement $\mathbf{z}_t$ is defined in the sensor coordinate system, usually a polar coordinate system for a 2D case and a spherical system for the 3D case. Let $(x,\ y)$ be the true position of the target, then the measurement z is defined as $z = [r\ b]^{\mathrm{T}}$, where r and b represent the range and bearings measurement

given as follows:

$$r = \sqrt{x^2 + y^2} \tag{81}$$

$$b = \arctan\left(\frac{x}{y}\right) \quad . \tag{82}$$

As seen, the measurement models are nonlinear and coupled across the Cartesian coordinates, although the measurement noise is usually an additive Gaussian noise, because of the measurement model is in the sensor coordinate systems. The framework of tracking in mixed coordinate systems have been extensively used in most nonlinear estimation and filtering techniques for maneuvering target tracking. As reported in the current literature [96], some research works also adopt the method of tracking in Cartesian coordinate system, which is give as follows:

$$\mathbf{z}_t = \mathbf{H}\mathbf{x}_t + \mathbf{v}_t \quad . \tag{83}$$

In this case, the measurement $\mathbf{z}_t$ is the noise contaminated target location. The advantage of this method is that the measurement model is linear, so a Kalman filter can be directly applied to this problem. However, the linear model is obtained by converting the sensor readings to the Cartesian coordinate system, where the measurement noise is under a nonlinear transform. More specifically, the measurement noise in this case is generally non-Gaussian and state dependent [96]. Considering this, the framework for tracking in this research is the mixed coordinate system.

Next, we present the improved multiple model particle filter based on the method of state space discretization. This algorithm is actually a combination of the aforementioned particle filter algorithm with the multiple model method. In this research we use four dynamic models, *i.e.* one CV, two CTs and one CA. The structure of this PF algorithm is as follows: Firstly, discretize the state space into a set of cells, then calculate the cell center weights based on the initial target distribution. Secondly, propagate the cells through the different dynamic models, and update the cell weights based on the current measurements. In this way, four sets of cells will be generated.

Thirdly, update the model particles based on the model transition matrix and previous model particles. Then count the number of particles that belongs to each model, such as

$$N_s = \sum_{r_t=1}^{N_r} N_s^{(r_t)} \ , \quad \text{with } r_t \in [1, 2, 3, 4] \quad , \tag{84}$$

where $N_s^{(r_t)}$ represent the number of particles that will belong to model r_t. Next, we draw particles from the four sets of cells based $N_s^{(r_t)}$. Finally, we evaluate the particle weight followed by a resampling and an updating stage. The advantages of this filtering algorithm come from two sources: (1) by using the multiple model method, the proposal distribution can effectively capture the complex target dynamics, (2) since only a limited number of cells are used to generate the proposal, the computation complexity this algorithm can be controlled. Figures 55 and 56 demonstrate the strength of this multiple model PF. Figure 55 depicts the cell centers after propagating through the four model system with the given initial condition $\mathbf{x}_0 = [0\ 0\ 3\ 3\ 5\ 5]^{\mathrm{T}}$, where $\mathbf{x}_t$ denotes $[x_t\ y_t\ v_{x_t}\ v_{y_t}\ a_{x_t}\ a_{y_t}]^{\mathrm{T}}$. It is clear that the target has a relatively large acceleration. The cell centers with the color of red, green, blue and yellow represent the cell centers obtained from propagating through the CV, counter-clockwise CT, clockwise CT and CA model, respectively. The green circle symbolizes true target location, while the pink diamond represents the noisy measurement. It is clear that if only one model (such a CV model) is used, the target actual location and the measured location will fall outside of the updated cells. However, if multiple models are used to update the cells, the probability of the target being covered by at least one set of cells will increase dramatically. Figure 56 shows the particles that were draw from this approach. Initially, the model particles are uniformly distributed. The model particles are updated for the next iteration using the transition matrix given as:

$$\Pi = \begin{bmatrix} 0.7 & 0.1 & 0.1 & 0.1 \\ 0.4 & 0.2 & 0.2 & 0.2 \\ 0.4 & 0.2 & 0.2 & 0.2 \\ 0.4 & 0.2 & 0.2 & 0.2 \end{bmatrix}$$

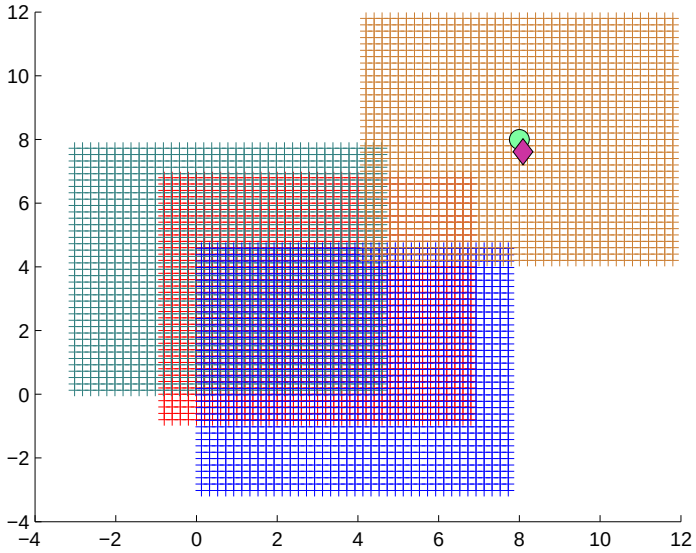


Figure 55: *The cell centers after propagating through multiple models. The cell centers with the color of red, green, blue and yellow represent the cell centers obtained from propagating through the CV, counter-clockwise CT, clockwise CT and CA model, respectively. The green circle symbolizes true target location, while the pink diamond represents the noisy measurement.*

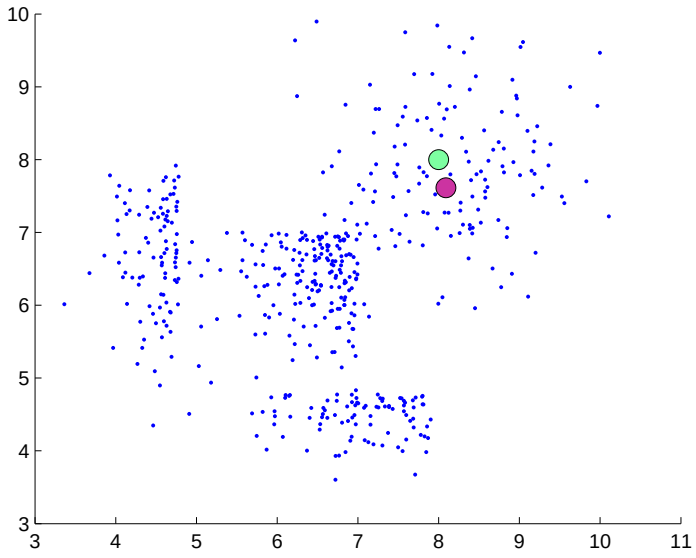


Figure 56: *The particles generated from weighted cells with using the multiple model method.*

It is clear from the transition matrix that the CA model (4-*th* model) has a relatively low probability. Thus in Figure 56 only a small number of particles are generated from the CA model. However, these particles are more closer to the target ground truth and the measured location. In addition, particles generated from the other models are concentrated on the cells which are close to the target. This type irregular particle distribution can only be achieved using the multiple model method. The block diagram of this algorithm is shown in Figure 57, and Table 11 provides a summary of this algorithm.

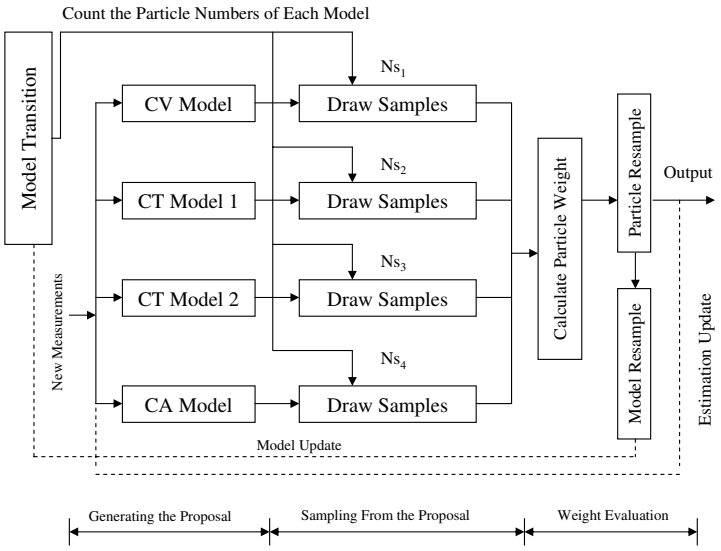


Figure 57: *The multiple model particle filter based on a state space discretization.*

Table 11: Algorithm 7: A State Space Discretization Based MMPF

- Sequential Importance Sampling (SIS) Step:
  - Generate a set of model particles based on the previous model distribution and the model transition matrix.
  - For each dynamic model, generate a set of cells and calculate the cell probability weights based on the discretized state space.
  - Construct the proposal distribution using the cells and their weights.
  - Draw particles: count the total number of model particles from each model, then uniformly draw the corresponding number of particles of each model such that the sample number of each cell is proportional to the cell weight.
  - Evaluate particle weights using (13) and normalization.
- Resampling Step: Generate a new set of particles $x_t^{i\star}$ from $x_t^{(i)}$ by sampling N_s times the approximate distribution such that: $Pr\left(x_t^{i\star} = x_t^{(j)}\right) = \tilde{\omega}_t^{(j)}$.
- Approximate x_t by $\hat{x}_t \approx \frac{1}{N_s}\sum_{i=1}^{N_s} x^{i\star}(t)$ and update the proposal.

5.3.1 The Experimental Results

In this section, the new MMPF algorithm is applied to three different tracking scenarios to evaluate its performance. In the first two tracking scenarios, the algorithm for two different cases is implemented, where each having a different measurement noise level. For the purpose of comparison, a bootstrap filter was also implemented in every tracking scenarios.

Tracking Scenario No.1

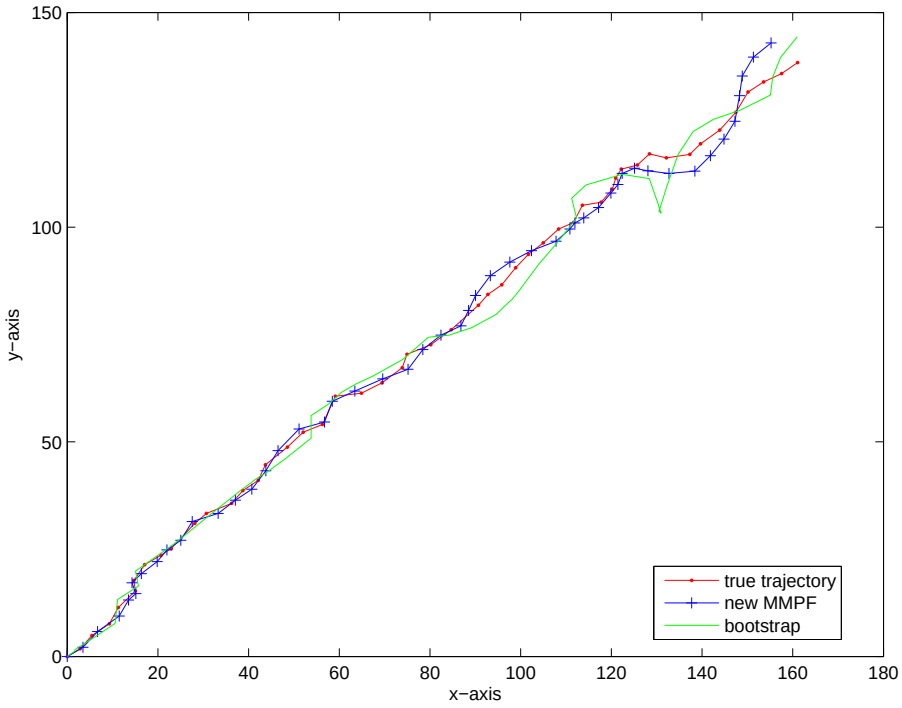
In the first tracking scenario, the target operates according to a constant velocity model, but its motion was disturbed by high level process noise. Strictly speaking, this type of target is not a maneuvering target. However, traditional single model tracking algorithms usually can not provide reliable estimations in this case [97]. As discussed earlier, four dynamic models were used in the new MMPF, which are the constant velocity model, two coordinated turn models, and the constant acceleration model. To illustrate the effect of the measurement noise, the new MMPF algorithm was implemented for two different cases with each has a different measurement noise level. The measurement noise variances of these two cases were set to $\sigma_r^2 = 10$ $\sigma_b^2 = 1$ and $\sigma_b^2 = 20$ $\sigma_r^2 = 10$, respectively. In fact, the bearings noise affects the estimation accuracy more than the range noise. This is because if the target is far from the sensor, even a small amount of bearing or angle noise can result in large errors in the Cartesian coordinates. In this simulation, the unit of bearings measurement is defined in degrees. To facilitate comparisons, the new MMPF is implemented with a standard bootstrap filter, which both use 1000 particles. For the first case, *i.e.* Scenario No.1, *case A* (low noise level case), the target's true trajectory and the estimations from the two filters are shown in Figure 58(a). Figure 58(b) provides the location estimation errors. In addition, the estimated velocities and accelerations together with the estimation errors are shown in Figures 59 and 60, respectively. It can be seen from these figures that the new MMPF provide better estimations compared to the standard single model bootstrap filter. Also, Figure 58(a) shows that as the target moves away from the sensor location (the origin), the estimation error progressively increases, which is consistent with the above discussion about the effect of the bearings noise. The tracking results of *case B* (high noise level case) in this scenario are shown in Figures 61 to 63. As indicated in these figures, both filters can follow the target in the general direction, although the new MMPF provides better estimations.

Tracking Scenario No.2

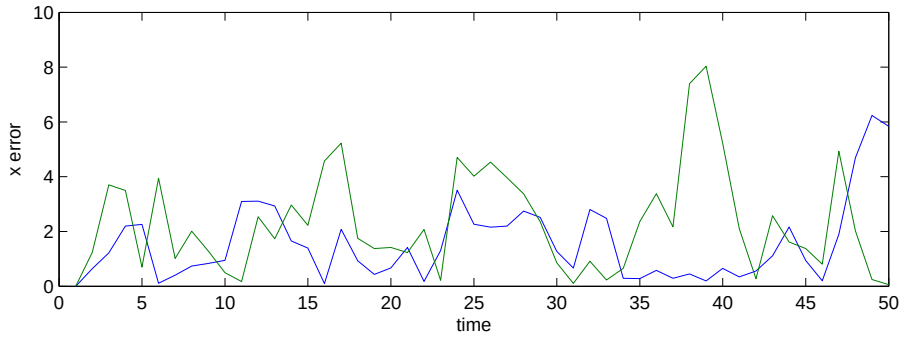
In the second example, the simulation is carried out in a more realistic tracking scenario, where the target takes a sharp counter-clockwise turn as it moves away from the sensor. This scenario is one of the most common case in maneuvering target tracking. Similar to the first tracking scenario, two cases with different measurement noise variances were used to demonstrate the superior performance of the new MMPF. For the low noise level *case A*, the tracking estimations and the relevant estimation errors are shown in Figures 64 to 66. As shown in Figure 64 (a), the new MMPF always keeps a close track of the target in spite of the significant maneuver operations made by the target around $t = 20$ to 25. On the other hand, the bootstrap filter gives large estimation errors during $t = 20$ to 25, when the target made the maneuver. Figures 65 (b) and Figure 66 (b) illustrate that the bootstrap filter give huge estimation errors for the target velocities and accelerations. *Case B* in this scenario further demonstrates the strength of the new MMPF over the standard bootstrap filter. In this case, the MMPF still keeps a good tracking throughout the whole simulation in spite of the increased measurement noise level. However, the bootstrap filter almost diverges from the target as it makes the maneuver operation. In addition, Figures 68(b) and 69(b) reports that, when compared to the new MMPF, the the bootstrap filter gives huge estimation errors for the target's velocities and accelerations.

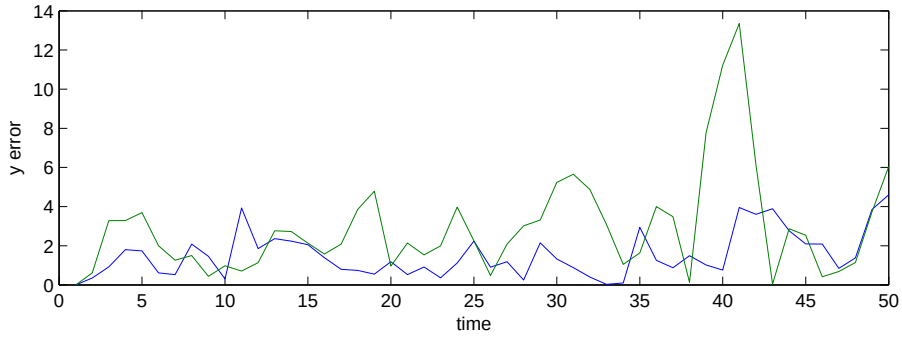
Tracking Scenario No.3

The tracking results in the third scenario are illustrated in Figures 70 to 72. In this example, the target accelerates in a rectilinear motion followed by a sudden de-acceleration. Like the second tracking scenario, this example is also a common case in maneuvering target tracking. It is also challenging due to the dramatic change in the target acceleration. Figure 71 shows that the target's velocity increase linearly until around t=25, when the velocity suddenly decreases. As demonstrated in these figures, consistent with the two previous examples, the new MMPF outperforms the standard bootstrap filter by providing better estimations.



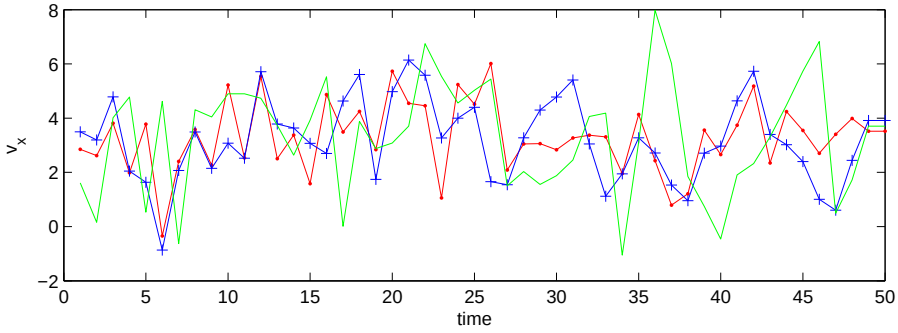
(a)

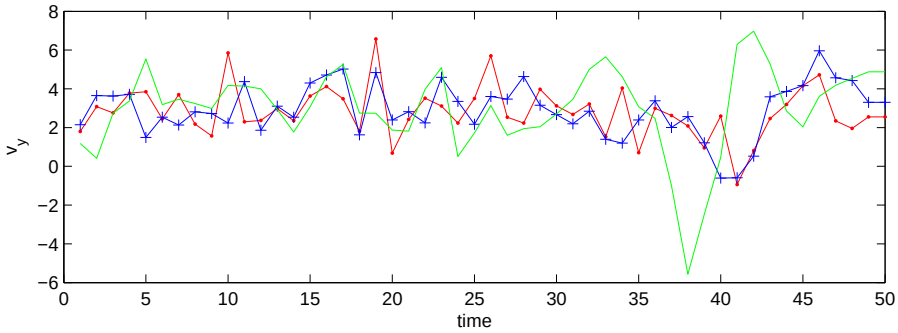




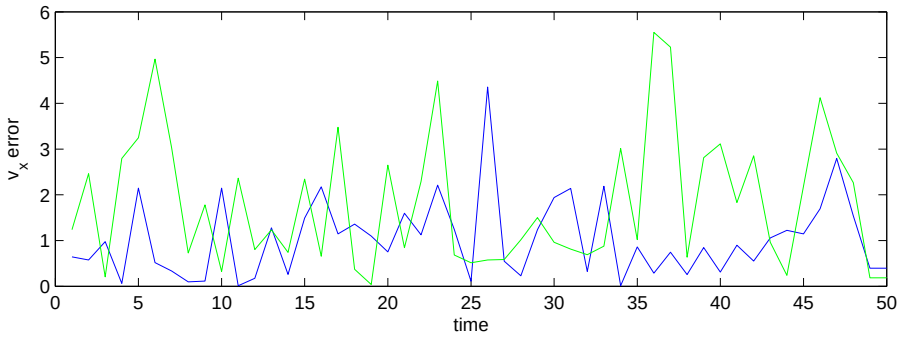
(b)

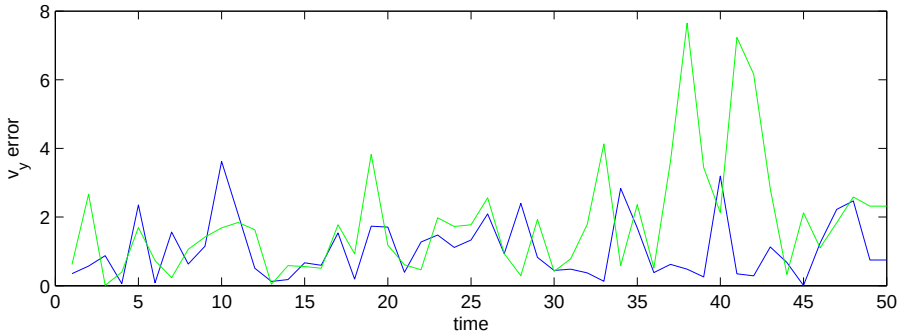
Figure 58: *Scenario No.1 Case A: (a) The true (red) and estimated trajectory of the bootstrap (green) filter and the new MMPF (blue) (b) The estimated location error of the bootstrap (green) filter and the new MMPF (blue).*





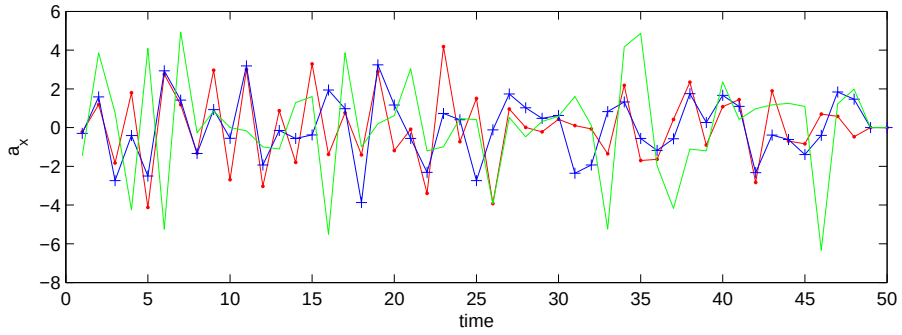
(a)

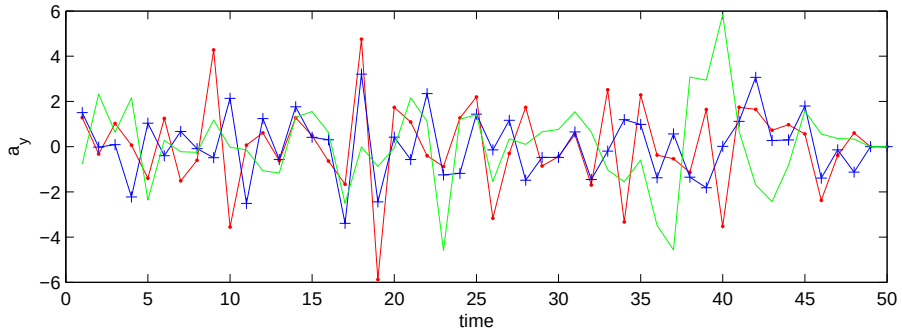




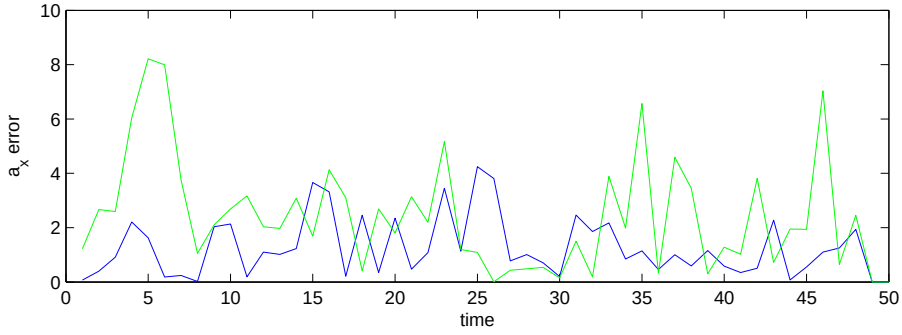
(b)

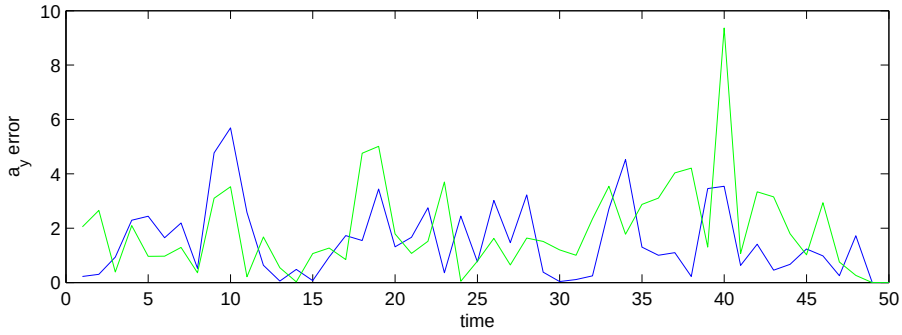
Figure 59: *Scenario No.1 Case A: (a) The true (red) and estimated velocity of the bootstrap (green) filter and the new MMPF (blue) (b) The estimated velocity error of the bootstrap (green) filter and the new MMPF (blue).*





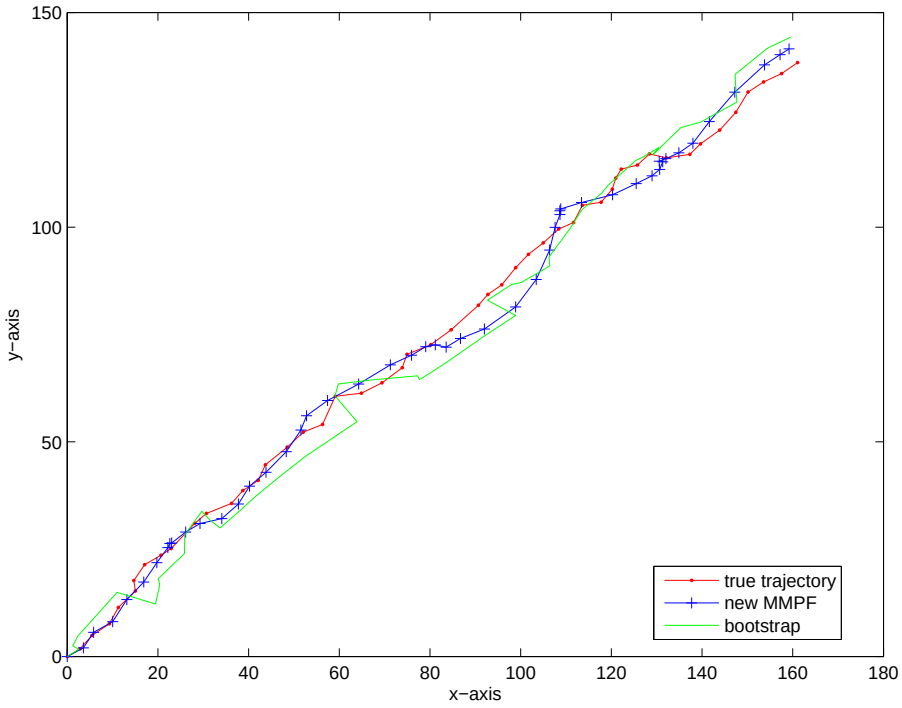
(a)



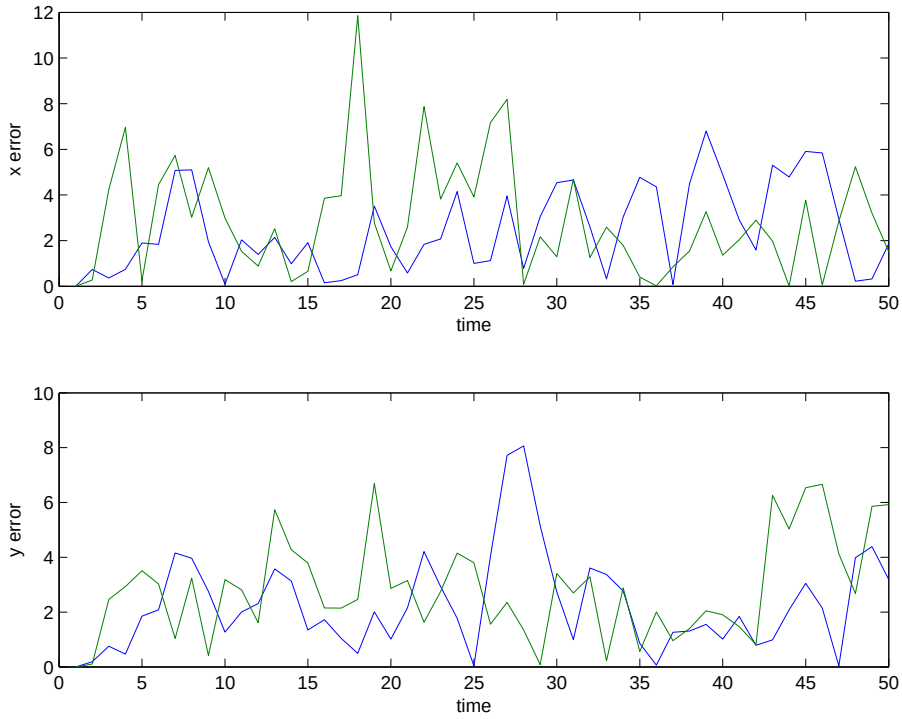


(b)

Figure 60: *Scenario No.1 Case A: (a) The true (red) and estimated acceleration of the bootstrap (green) filter and the new MMPF (blue) (b) The estimated acceleration error of the bootstrap (green) filter and the new MMPF (blue).*

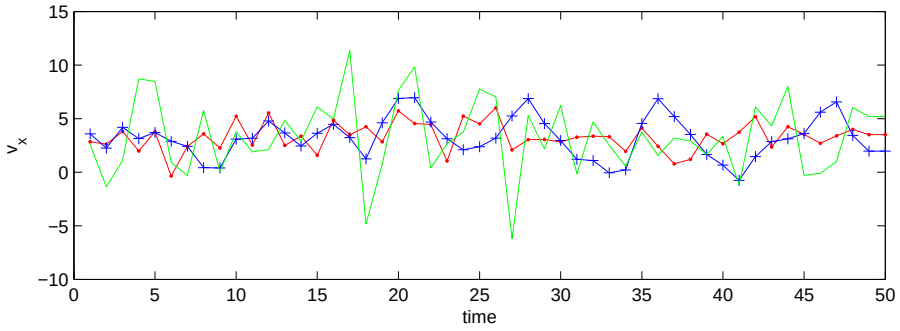


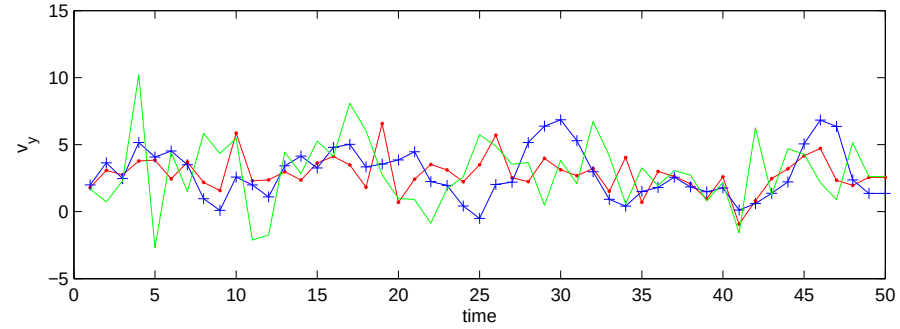
(a)



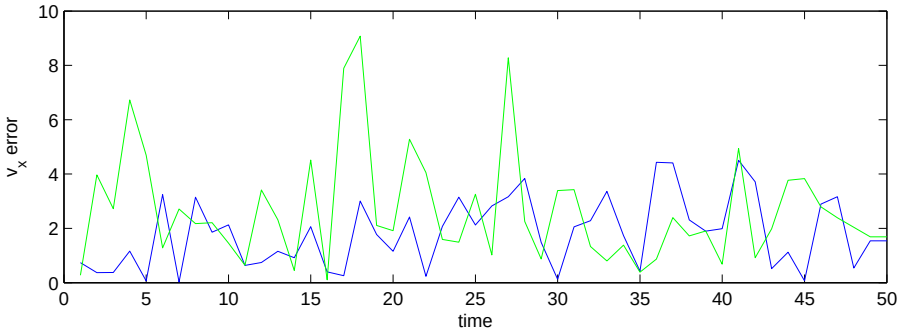
(b)

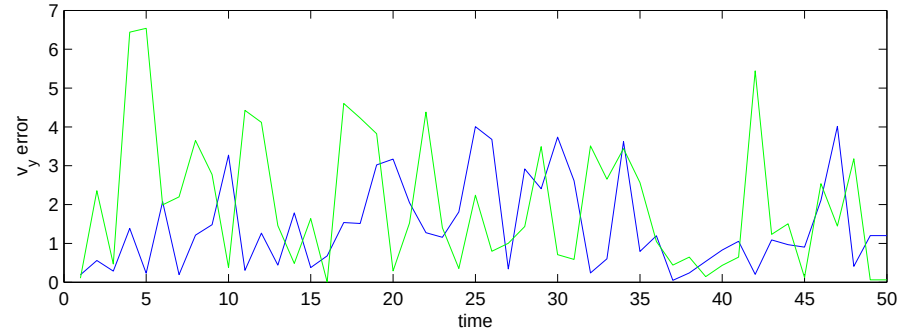
Figure 61: *Scenario No.1 Case B: (a) The true (red) and estimated trajectory of the bootstrap (green) filter and the new MMPF (blue) (b) The estimated location error of the bootstrap (green) filter and the new MMPF (blue).*





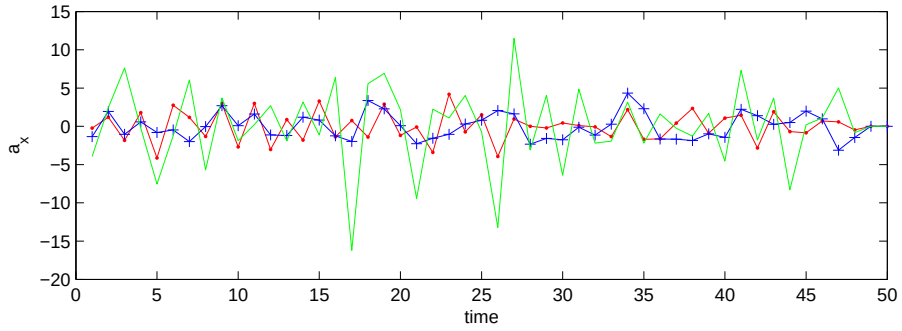
(a)

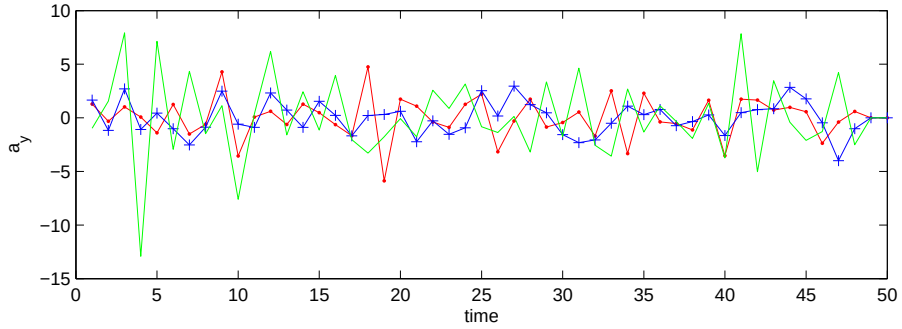




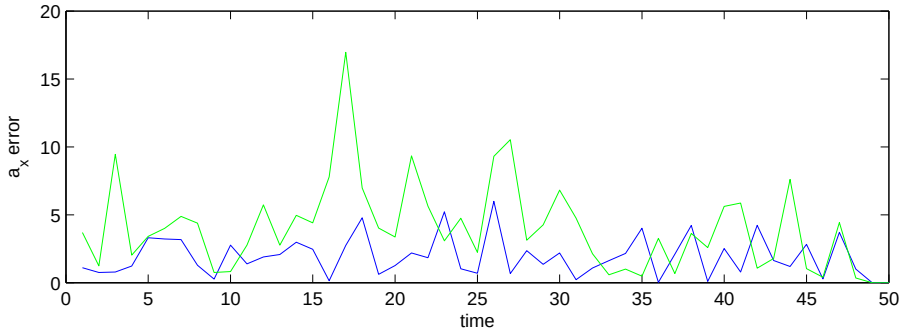
(b)

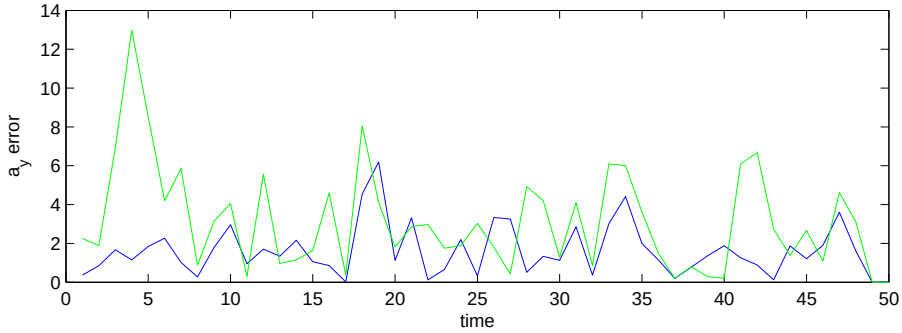
Figure 62: *Scenario No.1 Case B: (a) The true (red) and estimated velocity of the bootstrap (green) filter and the new MMPF (blue) (b) The estimated velocity error of the bootstrap (green) filter and the new MMPF (blue).*





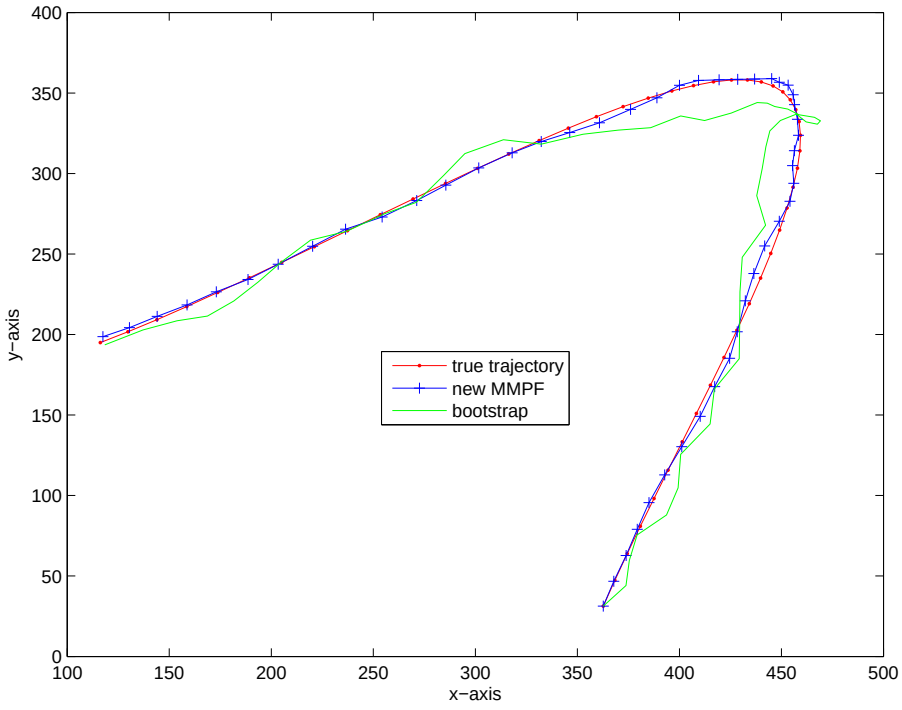
(a)



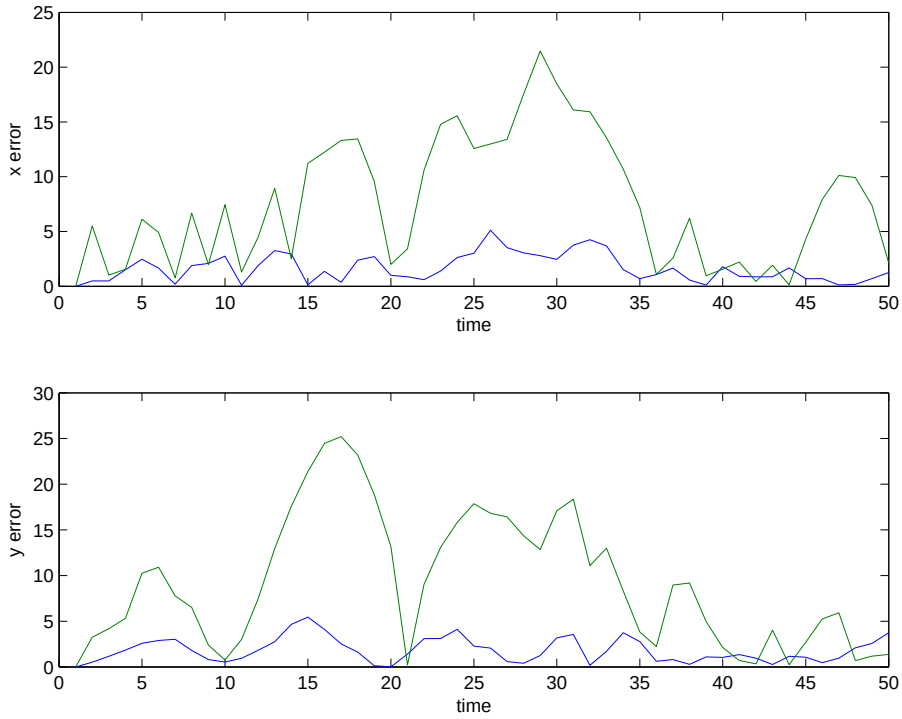


(b)

Figure 63: *Scenario No.1 Case B: (a) The true (red) and estimated acceleration of the bootstrap (green) filter and the new MMPF (blue) (b) The estimated acceleration error of the bootstrap (green) filter and the new MMPF (blue).*

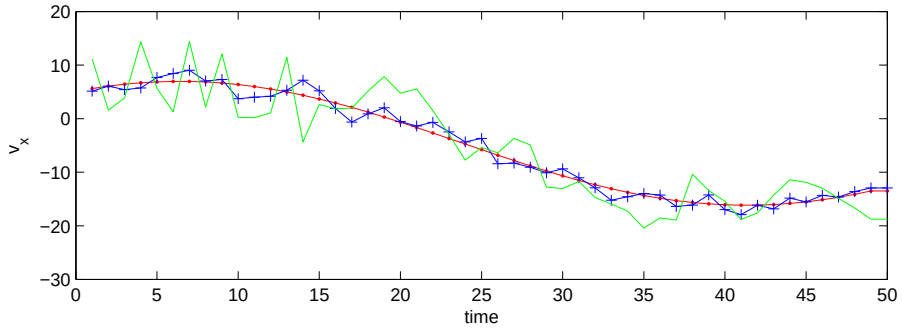


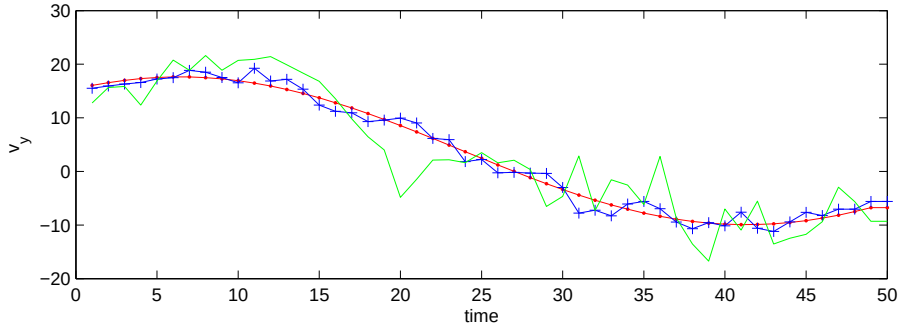
(a)



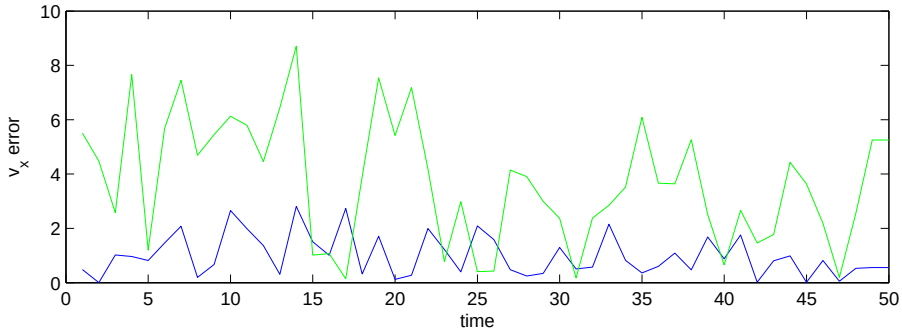
(b)

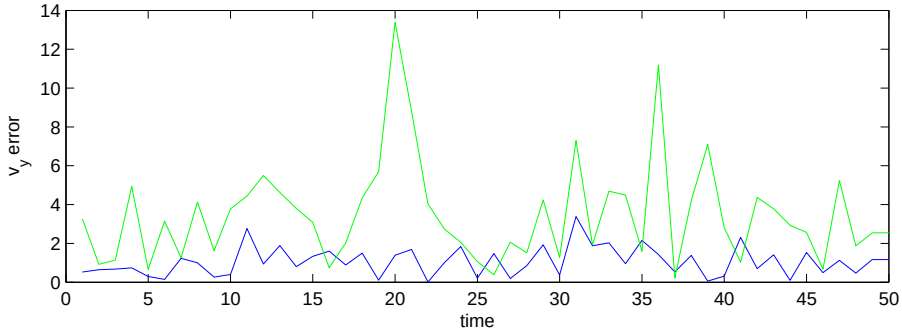
Figure 64: *Scenario No.2 Case A: (a) The true (red) and estimated trajectory of the bootstrap (green) filter and the new MMPF (blue) (b) The estimated location error of the bootstrap (green) filter and the new MMPF (blue).*





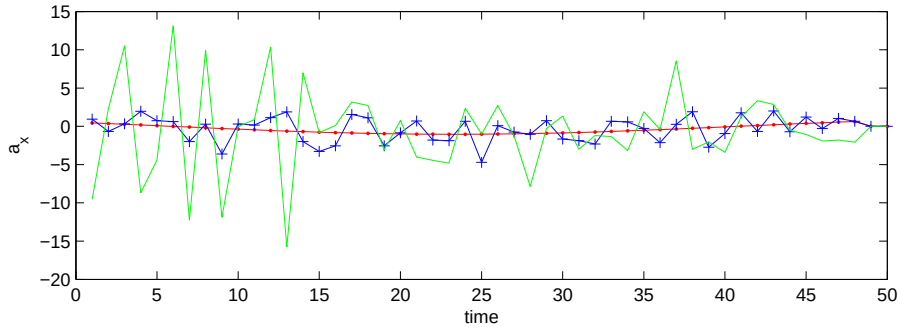
(a)

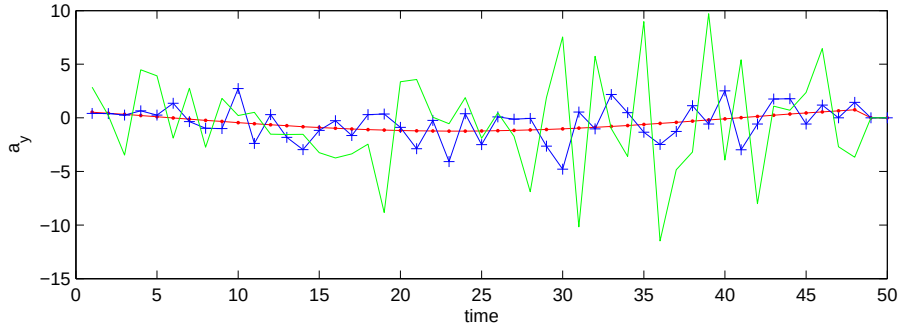




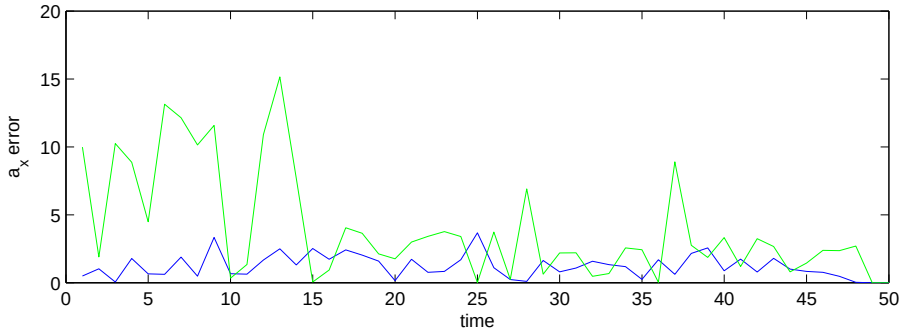
(b)

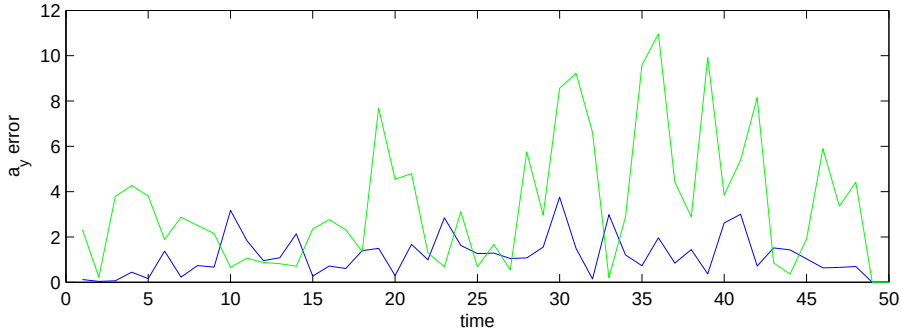
Figure 65: *Scenario No.2 Case A: (a) The true (red) and estimated velocity of the bootstrap (green) filter and the new MMPF (blue) (b) The estimated velocity error of the bootstrap (green) filter and the new MMPF (blue).*





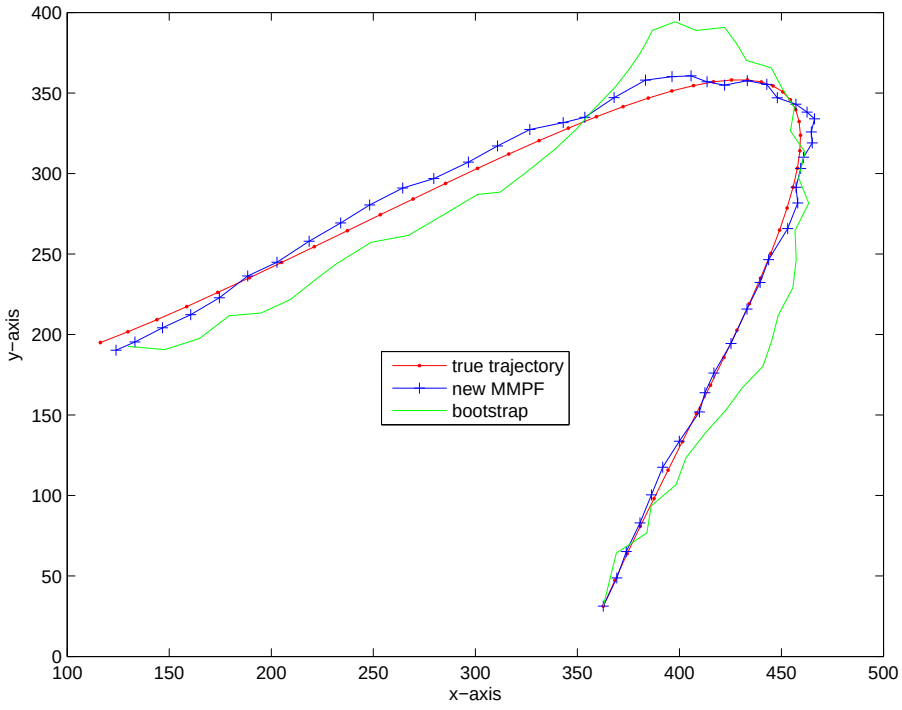
(a)



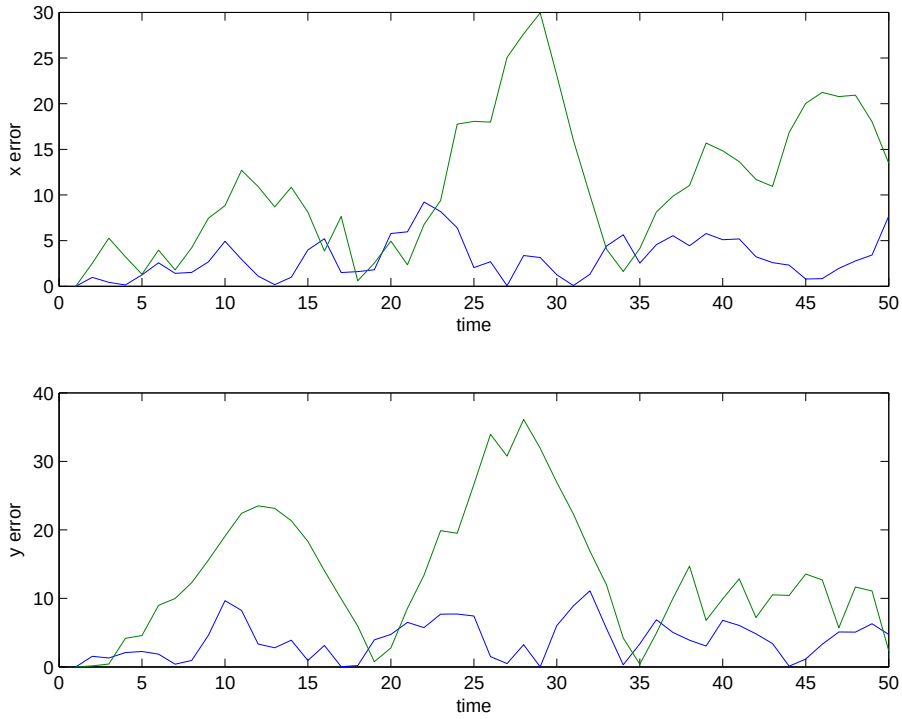


(b)

Figure 66: *Scenario No.2 Case A: (a) The true (red) and estimated acceleration of the bootstrap (green) filter and the new MMPF (blue) (b) The estimated acceleration error of the bootstrap (green) filter and the new MMPF (blue).*

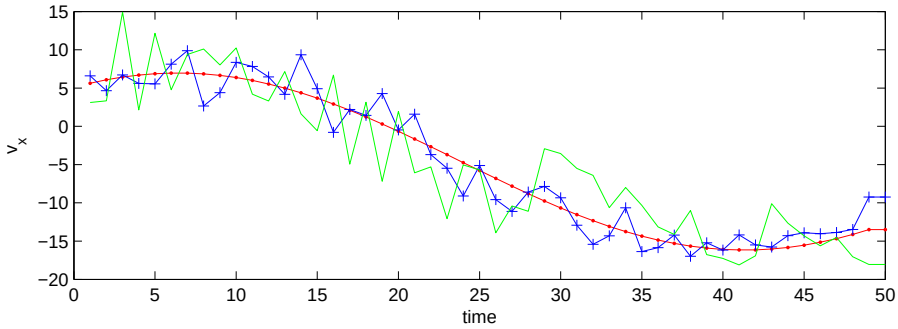


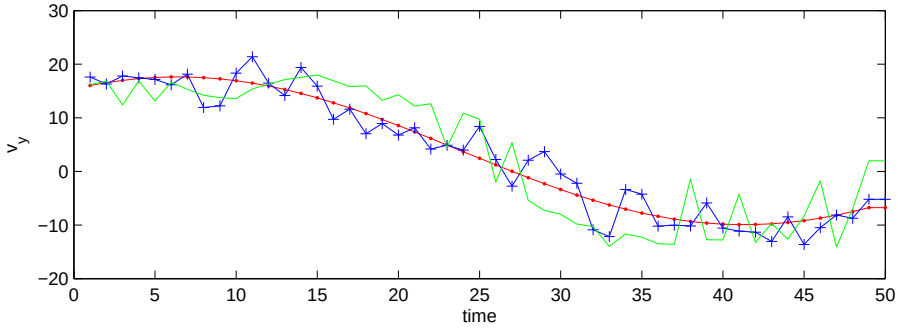
(a)



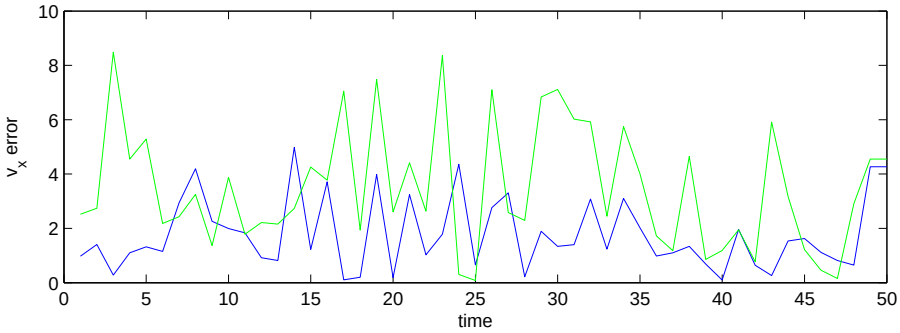
(b)

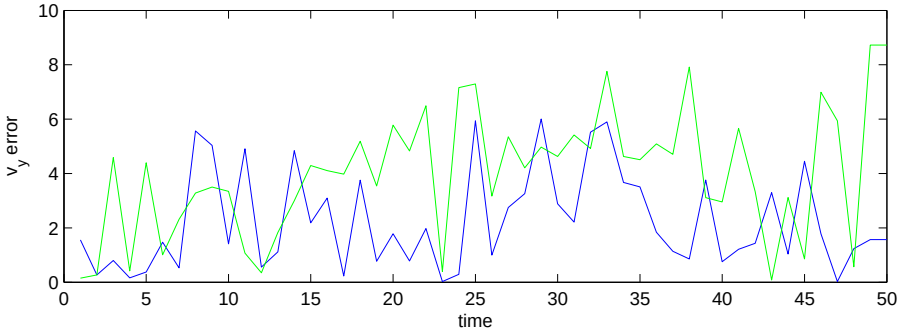
Figure 67: *Scenario No.2 Case B: (a) The true (red) and estimated trajectory of the bootstrap (green) filter and the new MMPF (blue) (b) The estimated location error of the bootstrap (green) filter and the new MMPF (blue).*





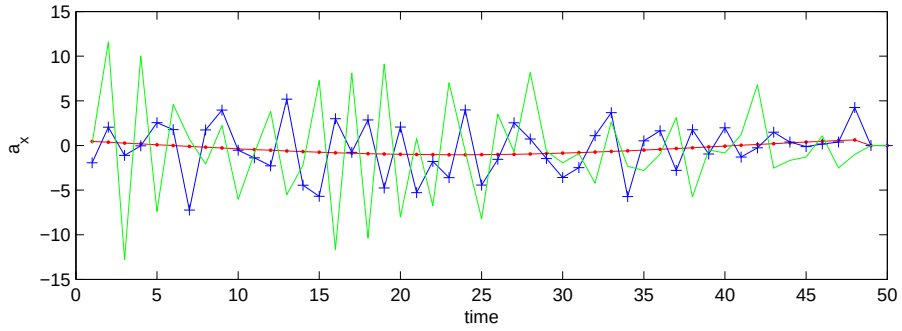
(a)

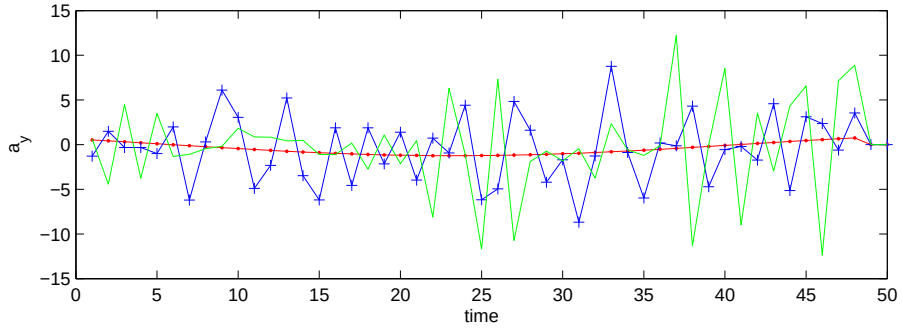




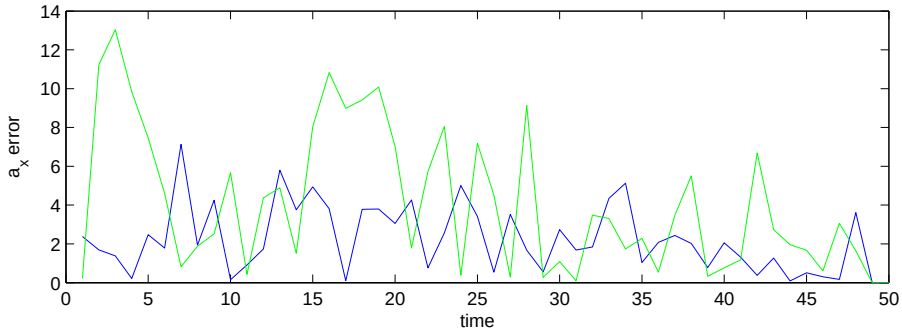
(b)

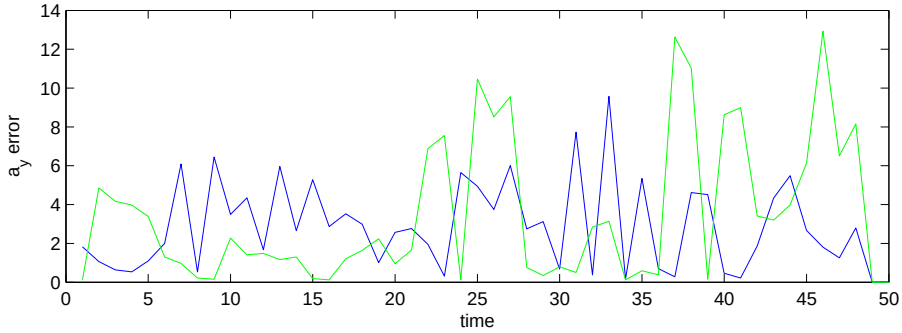
Figure 68: *Scenario No.2 Case B: (a) The true (red) and estimated velocity of the bootstrap (green) filter and the new MMPF (blue) (b) The estimated velocity error of the bootstrap (green) filter and the new MMPF (blue).*





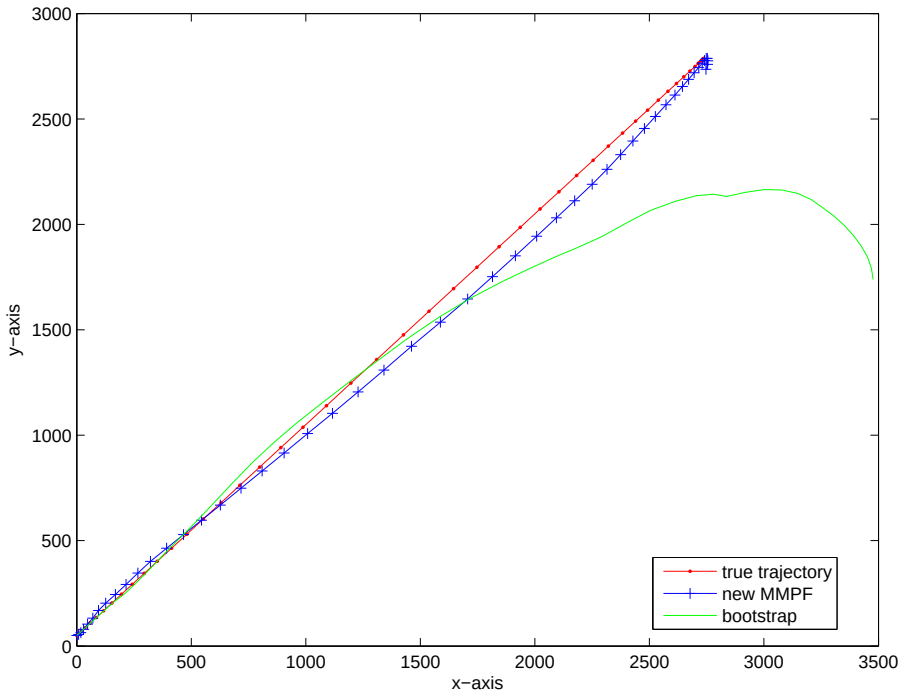
(a)



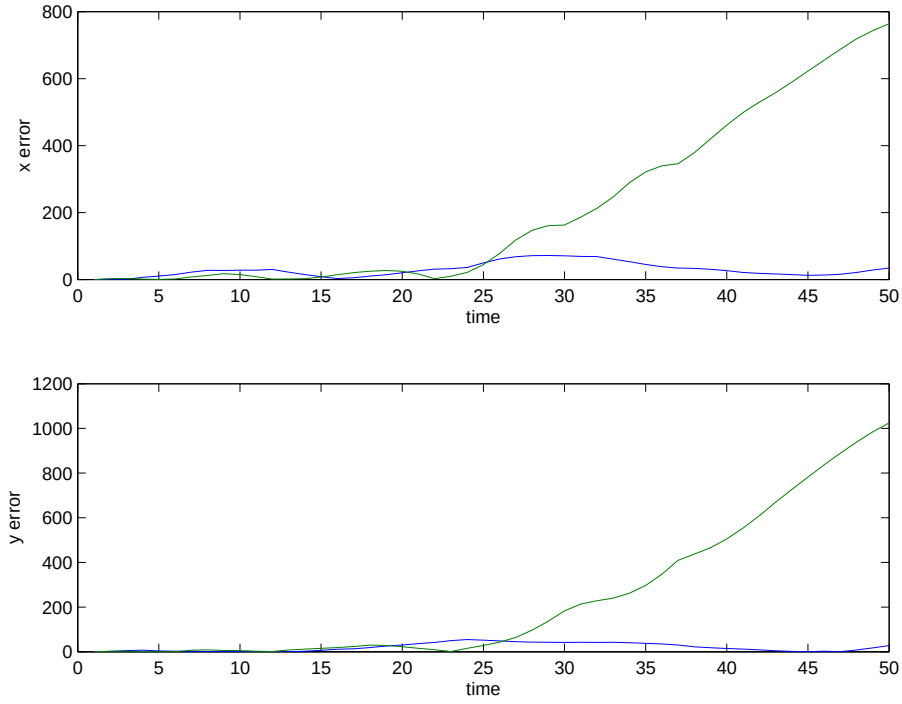


(b)

Figure 69: *Scenario No.2 Case B: (a) The true (red) and estimated acceleration of the bootstrap (green) filter and the new MMPF (blue) (b) The estimated acceleration error of the bootstrap (green) filter and the new MMPF (blue).*

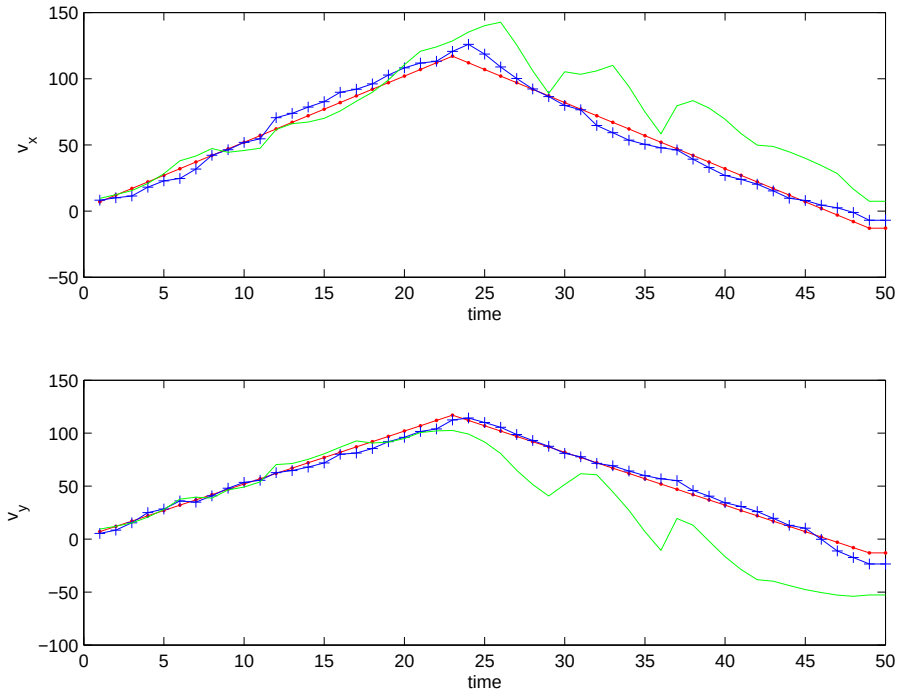


(a)

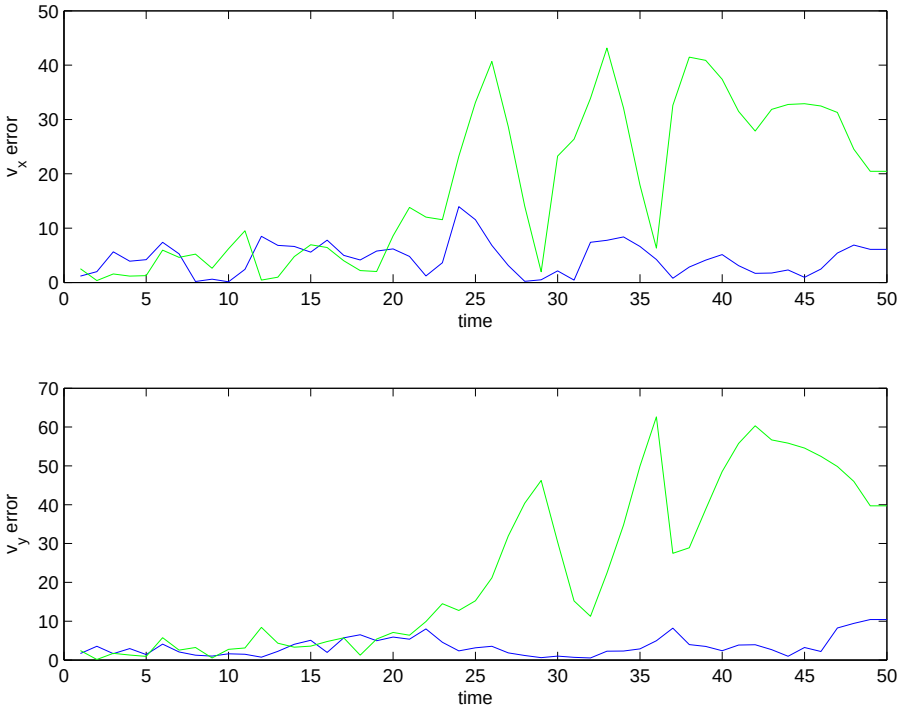


(b)

Figure 70: *Scenario No.3: (a) The true (red) and estimated trajectory of the bootstrap (green) filter and the new MMPF (blue) (b) The estimated location error of the bootstrap (green) filter and the new MMPF (blue).*

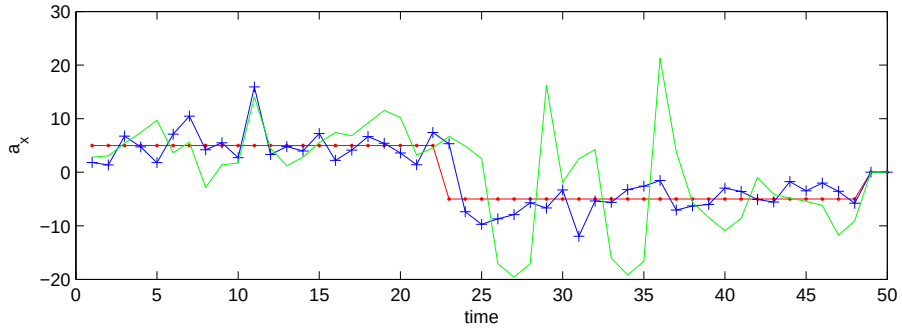


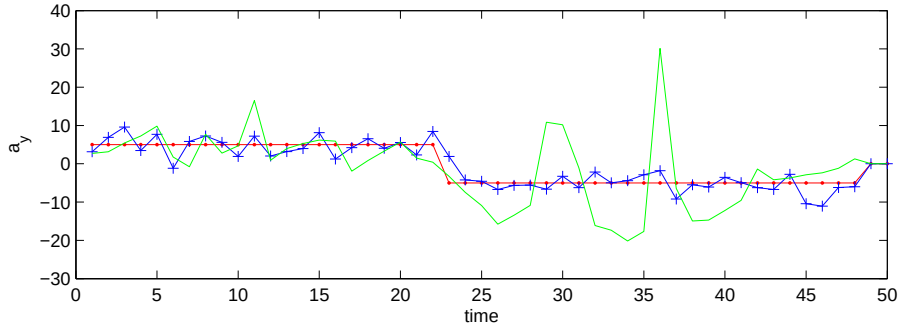
(a)



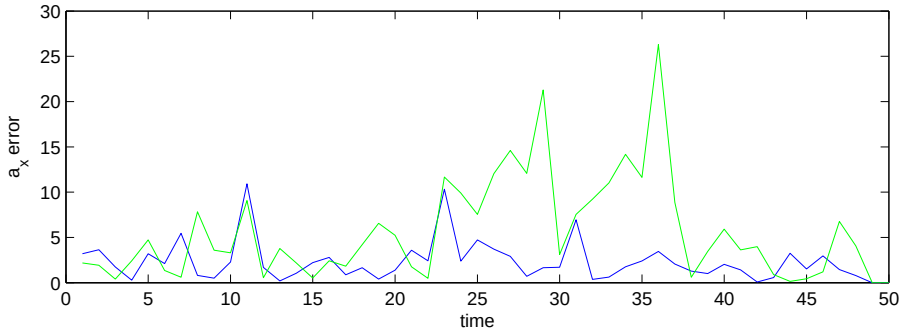
(b)

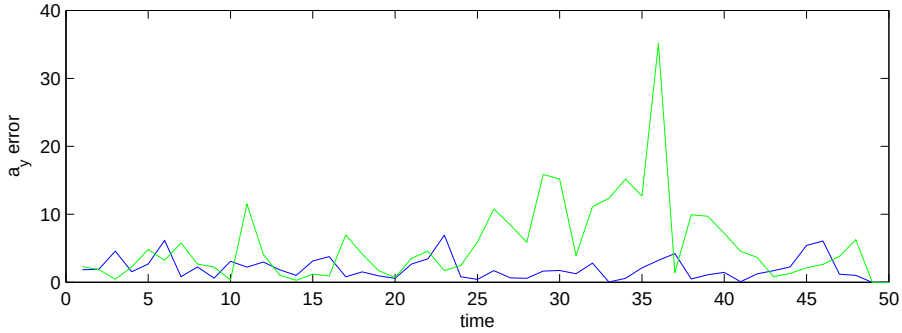
Figure 71: *Scenario No.3: (a) The true (red) and estimated velocity of the bootstrap (green) filter and the new MMPF (blue) (b) The estimated velocity error of the bootstrap (green) filter and the new MMPF (blue).*





(a)





(b)

Figure 72: *Scenario No.3: (a) The true (red) and estimated acceleration of the bootstrap (green) filter and the new MMPF (blue) (b) The estimated acceleration error of the bootstrap (green) filter and the new MMPF (blue).*

5.4 Application: Visual Tracking with Multiple Measurements

In this section, the new MMPF discussed above is further improved for visual target tracking problems. In this new algorithm, multiple measurement cues will be fused within the particle filter framework to design more accurate likelihood models that will result in more accurate tracking estimates.

5.4.1 The Multiple Measurement Cues

As discussed earlier, in the new tracking algorithm multiple measurement features (also known as measurement cues) will be analyzed and extracted to construct the likelihood model. In this research work, three different type of features were used. These are edge features, color features, and the grayscale texture-type features. The methods of feature extraction and selection will be covered in this sub-section.

Adaptive Edge Template

Image edges always carry abundant information of regarding the image content. Many high level image processing and computer vision applications rely on edge detection to provide measurement cues. In section 3.5, the well known Canny edge detection method was adopted, and a parametric edge template was used to approximate human face for face tracking applications. However, this type of template has its limitations. First, it is not convenient to use for targets which have complex shapes. Second, such fixed templates cannot provide good estimations for targets that are subject to change in scales and poses, which is very common in visual tracking problems. To rectify this problem, an adaptive edge template is used in the new algorithm. More specifically, two consecutive frames were considered. Once an estimated target location for the previous frame is obtained, a small window centered at the estimated target location will be used to select the template. Then, the edge detection algorithm is used to generate the edge template, which will be used for the detection within the current frame. The assumption for this method is that as long as the previous estimation is accurate and the video sampling rate is high enough, then the targets in the two consecutive frames

will have a large correlation. For common commercial video sequences, the sampling rate is about 24 frames per second, which is usually high enough. This method has the following three advantages: first, the template is adaptive, which is more robust than the fixed template. Second, this method is efficient to implement. Third, the template only considers the target of interest and a small neighborhood around it. In other words, it does not count on the prior background information, which makes it suitable for moving cameras (*i.e.* pan and zoom) tracking applications where most background substraction algorithms fail. Of course, this template can be further improved by incorporating intra-frame information. In this algorithm, once the template is obtained, the cross correlation of the template and the tracking aid window will be calculated as follows [77]:

$$f(x,y) \circ h(x,y) = \frac{1}{MN} \cdot \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n)h(x+m,y+n) \tag{85}$$

where the variales m, n, x and y represent the pixel index in the image plane. The pixel with the largest cross correlation corresponds to the location with the highest probability that the target is present at this location.

Intra-frame Grayscale Template

The grayscale texture type template has been widely used in image analysis and target detection [77][78]. Actually, the Kalman filter with a grayscale template matching is one of the most classic visual tracking algorithms. The grayscale template can be obtained by using the aforementioned method. However, more advanced on-line adaptive model for particle filtering have been reported in [84], which take the advantage of intra-frame information. This template has two components: (1) one that characterizing the two-frame variations, and (2) the one that preserves the stable structure of the target that persists for long preservations. The intra-frame grayscale template used in this research is a simplified version of the template reported in [84], which consists of several weighted past frames. The weight is normalized based on an "exponential

forgetting parameter" defined by an exponential forgetting envelop function given as [84]:

$$\varepsilon_t(k) = \alpha \cdot \exp(-\tau^{-1}(t-k)) \quad \text{for } k \leq t \tag{86}$$

where $\tau = n_h / \log 2$ and $\alpha = 1 - \exp(-\tau^{-1})$. Also, the variable n_h is called the half life of the envelope in frames. After the template is obtained, the cross correlation will be evaluated inside the tracking aid window using equation (85). Then the pixel with the largest correlation value will be the measured target location in the image plane.

Extracting Color Features

Currently, most video recorded by cameras are color videos. Color can provide very distinctive information of the target in contract to the background. Because of this reason, the color measurement cues should be incorporated into the tracking algorithm to provide extra useful information. Color representation is related to imaging science, where various color models (color space) have been constructed [77]. Among them, the most widely used color space is the RGB color space, where RGB stands for red, green and blue channel, respectively. The RGB system is an additive color system. Traditional color image processing techniques treat these three channels independently. Also, these channels have strong correlation to each other. In this research, besides the RGB color space, the HSI color model is also used, where HSI stands for hue, saturation and intensity. As indicated in [77], the color image processing algorithms implemented in HSI color space can produce better results for color image analysis applications. The conversion from RGB to HSI is given as follows:

$$H = \begin{cases} \theta & \text{if } B \leq G \\ 360 - \theta & \text{if } B > G \end{cases} \tag{87}$$

where

$$\theta = \cos^{-1}\left\{ \frac{\frac{1}{2}\left[(R-G)+(R+G)\right]}{\left[(R-G)^2+(R-B)(G-B)\right]^{1/2}} \right\} \quad .$$

The saturation component is given by:

$$S = 1 - \frac{3}{R+B+G}\left[\min(R,G,B)\right] \quad . \tag{88}$$

The intensity component is given by $I = \frac{1}{3}(R+G+B)$. In the tracking algorithm, the color histogram of the target is first calculated. Then it is compared to the color histogram calculated inside the tracking aid window. Here the Kullback-Leibler divergence (KLD) is adopted as a measure metric. Introduced in information theory to measure the distance between two distributions, the Kullback-Leibler divergence is defined as follows:

$$\mathbb{D}(p_1, p_2) = \sum_i p_1(x) \log\left(\frac{p_1(x)}{p_2(x)}\right) \quad , \tag{89}$$

where $p_1(x)$ and $p_2(x)$ denote the two distributions, and log is the natural logarithm. Once the KLD is calcuated inside the tracking aid window, the pixel with smallest KLD will be the measured target location.

5.4.2 The Tracking Algorithm

The new MMPF algorithm using multiple measurement cues is developed in this subsection. As discussed earlier, three different templates were used, with each template providing a measurement for the target location in the image plane. Assuming the target true location has a 2-dimensional Gaussian distribution, then three different likelihood models can be constructed. Let $\mathbf{x} = [x_r \ x_c]$ represent the target location, and the variable r_t and c_t denote the row and column index, respectively. Suppose the measured target location for the *i-th* template is $\mathbf{y}_t^{(i)} = [y_r^{(i)} \ y_c^{(i)}]$, then the likelihood for the $i-th$ template is given as:

$$\mathcal{L}^{(i)} = \frac{1}{2\pi\sigma^2} \exp\left[\frac{(y_r^{(i)} - x_r)^2 - 2(y_r^{(i)} - x_r)(y_c^{(i)} - x_c) + (y_c^{(i)} - x_c)^2}{\sigma^2}\right] \quad . \tag{90}$$

Finally, the overall likelihood is give as:

$$\mathcal{L} = \prod_{i=1}^{\mathrm{N_m}} \mathcal{L}^{(i)}, \quad \mathrm{N_m} = 3 \quad . \tag{91}$$

where $\mathrm{N_m}$ is the total number of measurement cues. The product in the above equation assumes that all these measurements cues are independent. As a summary, the new MMPF visual target tracking algorithm is given in the following block diagram:

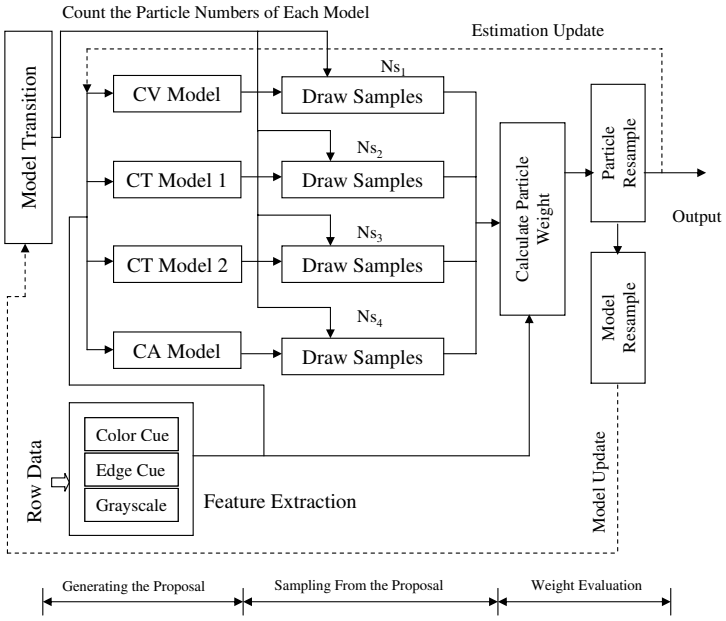


Figure 73: *The mutiple model particle filter with multiple measurement cues.*

5.4.3 Experimental Results

To evaluate the performances of the particle filter using multiple measurement cues, three different tracking scenarios were studied in this section, which contains two outdoor videos and one indoor video. The targets in the two outdoor videos have relatively low image quality and are subject to large scale change, while the target in the indoor video has more complex motions.

Tracking Multiple Measurement Cues: Case No.1

The objective in this scenario is to track a group of people in an outdoor environment. The challenge of this video is that the target is has a relatively small size, which can be bounded in a 16×10 pixel window. Also, the target does not have significantly distinctive color content with respect to the background. In addition, the target background contains a large amount of edge clutter. For a comparison, the new MMPF was implemented with a standard bootstrap filter with grayscale template as the its sole measurement cue. As demonstrated in the simulation and shown in Figure 74 the new MMPF gives fairly accurate tracking results throughout the whole video. However, the bootstrap filter can only keep the track to about 70 frames.

Tracking Multiple Measurement Cues: Case No.2

In the tracking case No.2, the objective is to track a vehicle in an outdoor video, the target has a very different color contents with respect to its background. In this case, the color measurement cue is especially helpful, which makes the target easy to detect. Also in this video the target has significant change in its scale, although it's motion is relatively simple. The new MMPF is applied to this video. As demonstrated by the tracking results that are shown in Figure 76, the new algorithm keeps a close track of the target.

Tracking Multiple Measurement Cues: Case No.3

The last example is the most interesting, in which the target has very complicated motion patterns. However, because this video is recorded indoors, the image quality and the illumination conditions are perfect. Also, there's no large change in the target scales. With the help of multiple measurement cues, the tracking algorithm provides a very good tracking results as shown in Figure 77.

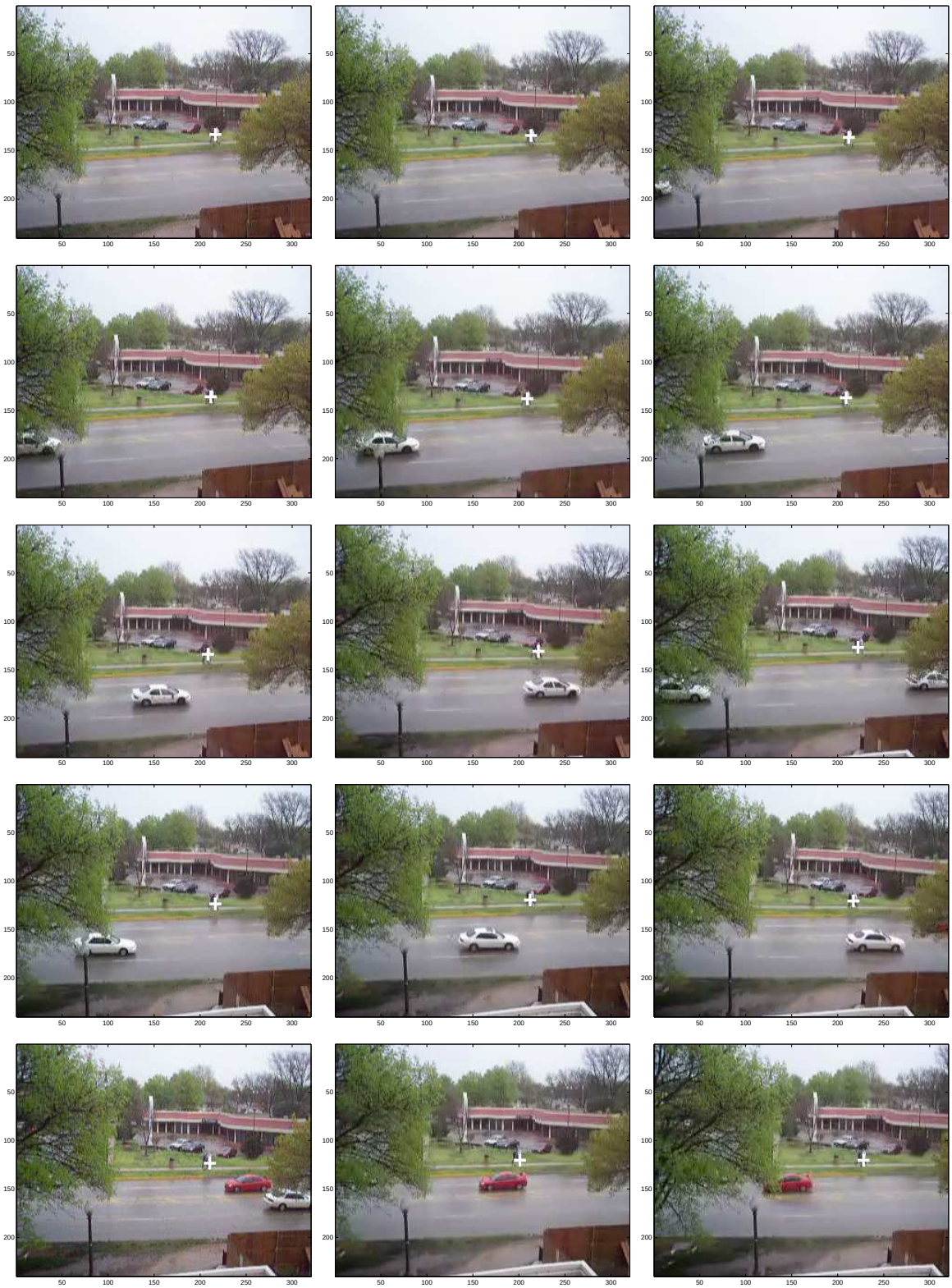


Figure 74: *MMPF Multi-Cue Tracking (Scenario No.1): Using the new MMPF, a close track can be achieved throughout the whole video. Frames 2, 10, 15, 20, 25, 30, 40, 50, 60, 70, 80, 90, 100, 110 and 120 are provided.*

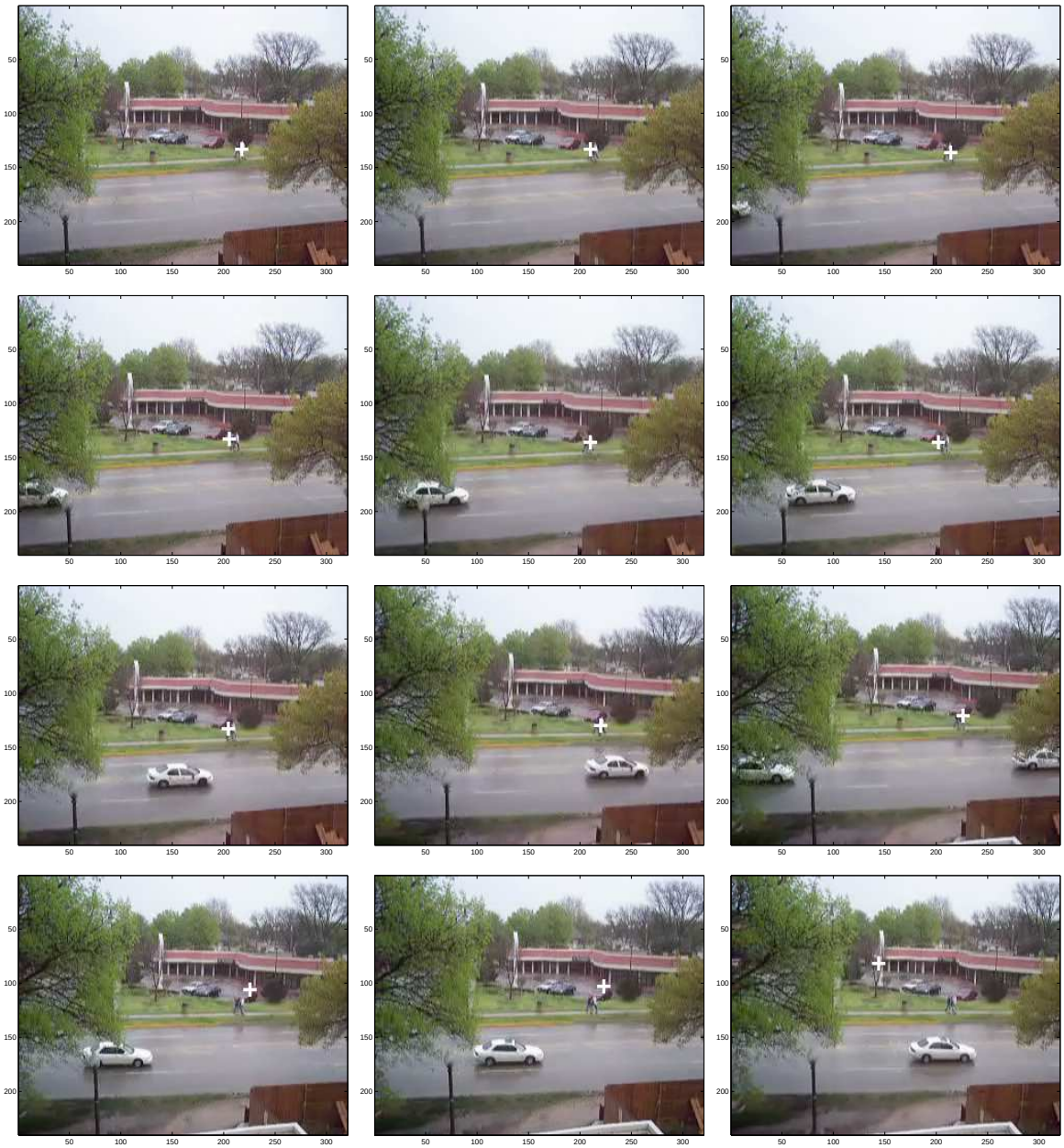


Figure 75: *Bootstrap Filter Tracking (Scenario No.1): The bootstrap filter looses the target around frame 70. Frames 2, 10, 15, 20, 25, 30, 40, 50, 60, 70, 75 and 85 are provided.*

Figure 76: *MMPF Multi-Cue Tracking (Scenario No.2): Using the new MMPF, a close track can be achieved throughout the whole video. Frames 2, 10, 15, 20, 25, 30, 40, 50, 60, 70, 75 and 80 are provided.*

Figure 77: *MMPF Multi-Cue Tracking (Scenario No.3): Using the new MMPF, a close track can be achieved throughout the whole video. Frames 2, 10, 15, 20, 25, 30, 40, 50, 60, 70, 80, 90, 100, 110 and 120 are provided.*

CHAPTER 6: FURTHER ANALYSIS AND OTHER APPLICATIONS

6.1 An Analysis of Particle Filter with MCMC

As discuss in Chapter 2, a resampling step is used in the particle filter framework to mitigate the sample degeneracy problem. However, it can also introduce a new problem called "sample impoverishment." This means that after resampling, it is possible to have a large number of repeated copies of particles, which will result in the loss of the sample diversity. In this case, MCMC methods can be introduced to solve this problem [98], which is known as an MCMC move in a particle filter. The theory behind this strategy is that if an MCMC kernel is applied to the particle set with the invariant distribution same as the posterior, then the resultant new particle set will have the same distribution as the original particle set. However, the new particles will have distinct values. Practically, many algorithms can achieve this task, which include two major groups of MCMC algorithms, *i.e.* the Gibbs sampler and Metropolis-Hastings (M-H) methods. This subsection numerically analyzed the effects of choosing different Metropolis-Hastings methods on the performance of particle filtering. More specifically, two special M-H algorithms, namely the random-walk metropolis (RWM) and the metropolized independence sampling (MIS) techniques, are studied here. At the end of this subsection, simulation results based on these two strategies are provided. Although this study is cannot provide a theoretical foundation of choosing specific M-H algorithm, it provides a new perspective of improving particle filtering.

6.1.1 The Metropolis-Hastings Algorithms

Although the degeneracy problem can be solved by applying a resampling step, this scheme induces the "sample impoverishment" problem, which is after the resampling step, multiple repeated particles are generated from heavily weighted former particles. As a result, the particles of the next generation will lose their diversity. The worst case scenario is that a large number of particles are produced by a single former particle with a large weight. The MCMC algorithms have been proposed to solve this problem

[25][98] [99].

In theory, if a Markov chain with a transition kernel $\mathcal{K}(\mathbf{x}|\mathbf{x}^*)$ satisfies the following condition:

$$\mathcal{K}(\mathbf{x}^*|\mathbf{x})f(\mathbf{x}) = \mathcal{K}(\mathbf{x}|\mathbf{x}^*)f(\mathbf{x}^*) \tag{92}$$

where $f(\cdot)$ denotes the target distribution, then the resultant distribution will be same as the target distribution. The Metropolis-Hastings algorithm is one of the most common MCMC methods, which employs a conditional distribution (known as the instrumental distribution) to generate a Markov chain with an invariant distribution $f(\cdot)$. A Metropolis-Hastings algorithm has the following form summarized in Table 12, see [100] for details:

Table 12: Algorithm 8: The General Form of Metropolis-Hastings

- Choose a starting point $\mathbf{x}_0$, and set $i = 0$.
- Given current state $\mathbf{x}_t$, draw $\mathbf{x}^*$ from $T(\mathbf{x}^*|\mathbf{x}_t)$ and draw a random number u from $U(0, 1)$.
- Accept $\mathbf{x}_t = \mathbf{x}^*$ if

$$u \leq \min\left\{1, \frac{f(\mathbf{x}^*)\,T(\mathbf{x}_t|\mathbf{x}^*)}{f(\mathbf{x}_t)\,T(\mathbf{x}^*|\mathbf{x}_t)}\right\},$$

  otherwise, $\mathbf{x}_{t+1} = \mathbf{x}_t$.
- Set $i = i + 1$, go back to the first step.

The target distribution and the instrumental distribution are denoted by $f(\cdot)$ and $T(\cdot)$, respectively. Usually, the instrumental distribution is chosen to be symmetric, that is $T(\mathbf{x}_t|\mathbf{x}^*) = T(\mathbf{x}^*|\mathbf{x}_t)$, and this promotes a possible displacement of the particles to a better location in the state space. Then an acceptance-rejection rule is applied to generate the chain. It is known that different M-H algorithms will affect the performance of particle filters. To demonstrate this effect, two special M-H algorithms (the RWM and the MIS) are implemented with a bootstrap filter for a simple nonlinear scalar system.

Random-Walk Metropolis (RWM)

The rationale behind the use of this algorithm is to perturb the current state of the chain by adding some "noise", $\mathbf{x}' = \mathbf{x}_t + \epsilon_t$, where $\epsilon_t \sim g_\sigma(\cdot)$ is i.i.d. for different time t. In addition, the new candidates still remain in the neighborhood of the state $\mathbf{x}_t$. Then, it is required to determine if the new value $\mathbf{x}'$ is of interest or not by applying rejection rules. Usually, $\epsilon_t \sim g_\sigma(\cdot)$ is chosen to be a symmetric distribution. For example, the common choice of $\epsilon_t \sim g_\sigma(\cdot)$ is a Gaussian distribution $\mathcal{N}(0, \sigma^2 I)$ or a uniform distribution. In [100], the Random-Walk Metropolis is summarized as follows:

Table 13: Algorithm 9: Random-Walk Metropolis (RWM)

- Choose a starting point $\mathbf{x}_0$.
- Given the current state $\mathbf{x}_t$, generate $\epsilon_t \sim g_\sigma(\cdot)$, then evaluate $\mathbf{x}' = \mathbf{x}_t + \epsilon_t$.
- Draw a random number u from $U(0, 1)$, and accept $\mathbf{x}_{t+1} = \mathbf{x}'$ if

$$u \leq \frac{f(\mathbf{x}')}{f(\mathbf{x}_t)} \ ,$$

  otherwise, $\mathbf{x}_{t+1} = \mathbf{x}_t$.
- Set $i = i + 1$, go back to the first step.

Metropolized Independence Sampler (MIS)

The Metropolized Independence Sampler is another class of special Metropolis-Hastings algorithms, which the instrumental density is independent of the given state. In other words, if $T(\mathbf{x}_t, \mathbf{x}^*)$ in the standard M-H algorithm is chosen to be $T(\mathbf{x}^*)$, then the standard M-H algorithm reduces to the MIS. The accuracy of this algorithm is closely related to the choice of the instrumental distribution. The limitation associated with this algorithm is that the instrumental distribution that has to be related to the target distribution to some extent. However, if carefully selected, it can produce very good results. The MIS algorithm is summarized in Table 14:

Table 14: Algorithm 10: Metropolized Independence Sampler (MIS)

- Choose a starting point $\mathbf{x}_0$.
- Draw samples $\mathbf{x}^*$ from the instrumental distribution $T(\mathbf{x}^*)$.
- Draw samples u from $U(0,1)$.
- Accept $\mathbf{x}_{t+1} = \mathbf{x}^*$ if $u \leq \min\left\{1, \frac{\omega(\mathbf{x}^*)}{\omega(\mathbf{x}_t)}\right\}$, where $\omega(\mathbf{x}) = f(\mathbf{x})/T(\mathbf{x})$ is the sampling weight. Otherwise, $\mathbf{x}_{t+1} = \mathbf{x}_t$.
- Set $i = i + 1$, go back to the first step.

6.1.2 Numerical Analysis

An illustrative example is provided in this subsection to evaluate the performance of a bootstrap filter with the aforementioned two M-H algorithms. Consider following nonlinear scalar system:

$$x_t = 1.7 \cdot exp\ (-2 \cdot x_{t-1}^2) + 5 \cdot cos(x_{t-1}) + w_t \tag{93}$$

$$y_t = x_t^2 + v_t \tag{94}$$

where the initial state x_0 is Gaussian $\mathcal{N}(0, 0.5)$. The process noise w_t and the measurement noise v_t are also assumed to be Gaussian, which are distributed according to $\mathcal{N}(0, 8)$ and $\mathcal{N}(0, 0.5)$, respectively. The time index is incremented as $t = 1, ..., 25$. In this example, we have implemented the particle filters by taking the prior transition distribution $p(x_t|x_{t-1}^{(i)})$ as the proposal distribution. Four particle filters (PF, PF with standard form Metropolis-Hastings, PF with RWM, PF with MIS) are implemented separately. The instrumental distribution of the standard Metropolis-Hastings algorithm is chosen as the transition prior $p(x_t|x_{t-1})$. In the Random-Walk Metropolis, $g_\sigma(\cdot)$ is set to $\mathcal{N}(0, 0.2)$, and in the PF-MIS algorithm, $T(x)$ is chosen as $\mathcal{N}(0, 0.15)$. For each set of samples (N= 100, 200, 500, 1000), the simulation is run for fifty times to produce an ensemble statistical performance index, the average Mean Square Error (MSE). The numerical results are displayed in Table 15. From the simulation results, we can see that the PF-MH gives better performance than the PF, as determined by

Table 15: The Average MSE of Each Sample Set

Samples	PF	PF-MH	PF-RWM	PF-MIS
N=100	5.3494	4.8779	3.5558	3.7227
N=200	5.0470	4.0194	3.3813	3.5506
N=500	4.3868	3.3737	3.2536	3.3580
N=1000	3.4414	3.3238	3.1814	3.2574

the average of MSE. Additionally, as indicated by the results, PF-MH performance can be further improved by implementing the RWM.

6.2 A Particle Filter For Image Denoising

As discussed in the previous chapters, the particle filter is a powerful algorithm for nonlinear filtering. In addition, the particle filter algorithm can also be utilized in other applications, such as image denoising. In this section, a hybrid image restoration method has been developed that effectively combines a particle filter with wavelet shrinkage to achieve robust performance against inhomogeneous noise mixtures. Specifically, the particle filter acts to suppress outlier-rich components of the noise, while in a subsequent step, the wavelet domain shrinkage attenuates any remaining, less heavily tailed noise components. We present recent preliminary examples demonstrating excellent rejection of salt-and-pepper like Cauchy noise mixed with additive white Gaussian noise (AWGN). Although limited in scope, these preliminary results suggest that the combination of particle filters with more traditional restoration techniques is a powerful approach that can provide a new dimension of flexibility for addressing noise mixtures involving difficult nonlinear and non-Gaussian components.

6.2.1 The Denoising Algorithm: A Hybrid PF-DWT Method

When an image is corrupted by noise comprised of both heavily tailed and Gaussian components, we have observed that undesirable blurring artifacts are often introduced if one attempts to suppress both components simultaneously. Therefore, we introduce a two-stage hybrid denoising technique called *PF-DWT*. In the first stage, a spatial PF is used to suppress the heavy tailed component of the noise. In the second stage, wavelet

thresholding is applied to attenuate the remaining "Gaussian-like" noise. The input of the second stage depends on the output of the first stage, thus realizing a synergistic hybrid approach that is robust and effective against a wide variety of inhomogeneous noise mixtures.

Particle Filter For Image Processing (Spatial Case)

To implement the PF, we employ a 2-D state space model similar to the ones used for Kalman filtering in [102, 108, 109].

The Image Model

The image model proposed in [109] is used since it is efficient in the sense of using only a small number of pixels:

$$\begin{aligned}\mathbb{I}(i,j) =& h_1\mathbb{I}(i,j-1) + h_2\mathbb{I}(i-1,j) \\ &+ h_3\mathbb{I}(i-1,j-1),\end{aligned} \tag{95}$$

where $\mathbb{I}(i,j)$ denotes the pixel at the i^{th} row and j^{th} column of the image. In equation (95), h_1, h_2, and h_3 are image parameters which can be estimated by various methods including, *e.g.*, least-squares. Next, the following state space model is constructed:

$$\mathbf{x}(i,j) = C\mathbf{x}(i,j-1) + Eu(i,j) + D\mathbf{w}(i,j), \tag{96}$$

$$y(i,j) = H\mathbf{x}(i,j) + v(i,j), \tag{97}$$

where

$$\mathbf{x} = \begin{bmatrix} \mathbb{I}(i,j) \\ \mathbb{I}(i,j-1) \\ \mathbb{I}(i-1,j+1) \\ \mathbb{I}(i-1,j) \end{bmatrix} \quad C = \begin{bmatrix} h_1 & 0 & h_2 & h_3 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$E = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^T \quad D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}^T$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$

The input term $u(i,j)$ is introduced as the recent estimate of pixel $\mathbb{I}(i-1, j+1)$. The variables $\mathbf{w}$ and v denote the process noise and the measurement noise, respectively.

The Spatial Particle Filter Algorithm

The first step in designing a particle filter is to choose a proposal distribution. While arbitrary in form, the proposal must be supported by the posterior distribution of the true system state and must be easy to sample. For PF-DWT a standard Kalman filter is used to provide the proposal distribution, since the image model is a linear system with non-Gaussian noise. Although the estimates delivered by the Kalman filter are not optimal in this case, they still provide the required support for the true posterior. The 2-D Kalman filter formulation is given by

$$\bar{\mathbf{x}}(i,j) = C\hat{\mathbf{x}}(i,j-1) + Eu(i,j) \tag{98}$$

$$\bar{P}(i,j) = CP(i,j-1)C^T + DQD^T \tag{99}$$

$$K(i,j) = \bar{P}(i,j)H^T[H\bar{P}(i,j)H^T + R]^{-1} \tag{100}$$

$$\hat{\mathbf{x}}(i,j) = \bar{\mathbf{x}}(i,j) + K(i,j)[y(i,j) - H\bar{\mathbf{x}}(i,j)] \tag{101}$$

$$P(i,j) = [I - K(i,j)H]\bar{P}(i,j) \quad , \tag{102}$$

where R and Q denote the covariance of the process noise and measurement noise, respectively. Finally, the spatial particle filter algorithm is illustrated in Figure 78 and is also summarized below.

6.2.2 Wavelet Thresholding for PF-DWT

To begin, the DWT is applied to decompose the data into subbands. A threshold is then applied for noise removal. In the DWT domain, small coefficients in the high frequency subband are assumed to arise primarily from noise. The idea is to "zero out" the high frequency subband coefficients that are less than a particular threshold.

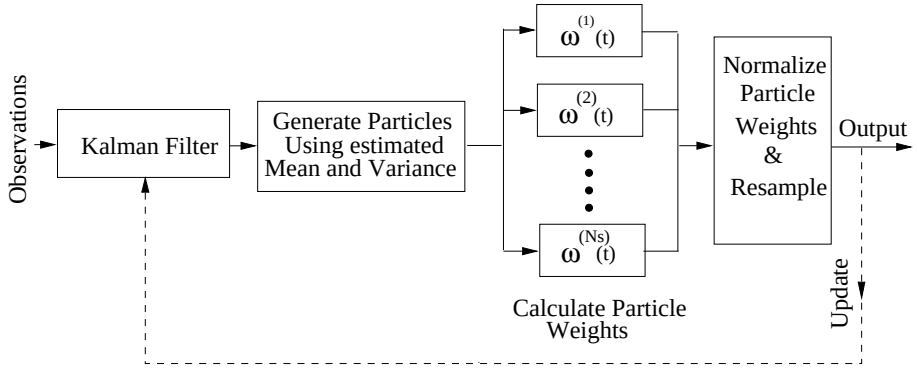


Figure 78: Block Diagram of the Spatial Particle Filter

1. Sequential Importance Sampling (SIS) Step:

   - At each iteration, calculate $\hat{\mathbf{x}}(i,j)$ and $P(i,j)$ according to (98)-(102).
   - Sample from the proposal distribution:

$$x^{(i)}(i,j) \sim \mathcal{N}\left(\hat{\mathbf{x}}(i,j), P(i,j)\right) \quad .$$

   - Evaluate the importance weights according to (13): calculate the transition prior, the likelihood and the proposal.
   - Normalize the importance weights.

2. Resample and Update Step:

   - Generate a new set of particles $x^{i\star}(t)$ from $x^{(i)}(i,j)$,
     so that

$$Pr\left(x^{i\star}(i,j) = x^{(j)}(i,j)\right) = \tilde{\omega}^{(j)}(i,j) \quad .$$

   - The final estimate of $x(i,j)$ is given by

$$x(i,j) \approx \frac{1}{N_s} \sum_{i=1}^{N_s} x^{i\star}(i,j) \quad .$$

   - Update the proposal with the resampled particles.

These coefficients are used in an inverse wavelet transformation to reconstruct the data set. In this section, the universal threshold is applied, similar to [103][104] and given as:

$$T = \sigma\sqrt{2\log_e N_d} \tag{103}$$

where N_d is the size of the data set. By definition, basic thresholding methods are hard thresholding:

$$\rho_T(x) = \begin{cases} x & if|x| > T \\ 0 & if|x| < T \end{cases} \tag{104}$$

and soft thresholding:

$$\rho_T(x) = \begin{cases} x - T & if\ x \geq T \\ x + T & if\ x \leq -T \\ 0 & if\ |x| \leq T \quad . \end{cases} \tag{105}$$

Moreover, the universal thresholding method can be improved by using the translation invariant technique given in [103]. The basic idea of this method is to estimate the coefficients of all translations and take the average after a reverse translation. The final estimate is given by this average. More specifically, the coefficients $\tilde{F}^p$ of all the translated data, denoted by $X^p[n] = X[n-p]$, are calculated. By definition,

$$\tilde{F}^p = \sum_{m=0}^{N-1} \rho_T(X^p[m])g_m \quad , \tag{106}$$

where ρ_T is a hard or soft thresholding function. Then, these coefficients are shifted back and the averages are evaluated, which gives the final estimate as

$$\tilde{F}[n] = \frac{1}{N}\sum_{p=0}^{N-1} \tilde{F}^p[n+p] \quad . \tag{107}$$

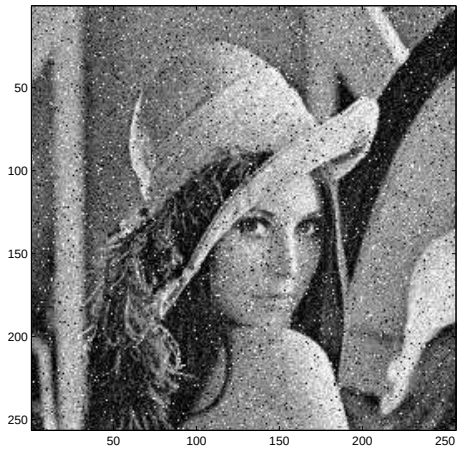
As indicated in [103], the translation invariant method produces a better denoised image than universal thresholding.

6.2.3 Experimental Results

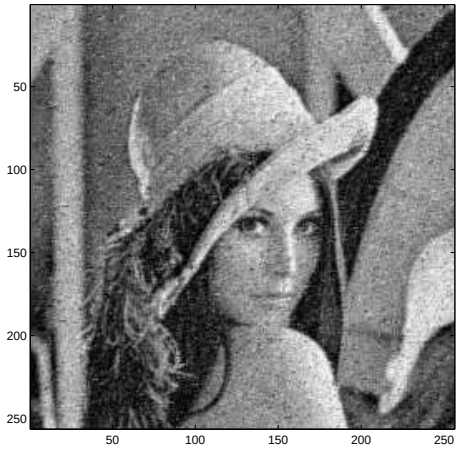
In this section, the standard 256×256 grayscale image "*Lena*" was used to assess the performance of the proposed hybrid denoising algorithm. The noisy image is generated by adding Gaussian noise with a variance of 200 and a Cauchy noise with a pdf defined by $f(x) = \theta/\pi(x^2 + \theta^2)$, where $\theta = 5$. The original and the noise corrupted images are shown in Figure 79(a) and (b), respectively. To construct the image model (95), the parameters $h_1 = 0.815493$, $h_2 = 0.489516$, $h_3 = -0.308866$ were employed. These values were estimated by using the least squares method [109].

The two-stage PF-DWT hybrid denoising process is carefully tailored to operate synergistically. In the first stage, a PF which uses Kalman filter state estimates as its proposal is utilized to suppress the heavy tailed noise. After the this stage, the remaining noise is "Gaussian-like" in character and is attenuated by wavelet thresholding. In the second stage, translation invariant soft/hard thresholding is applied to the PF result to achieve further noise removal. Two examples of images restored in this way are shown in Figure 79 and 80. As indicated in Figure 79(c), the PF removes most of the heavy-tailed noise. The remaining noise, which is observed to have a "Gaussian-like" distribution, can be removed using the wavelet thresholding method. The denoised image obtained via PF and translation invariant hard thresholding is shown in Figure 79(d). As a second example, the grayscale image "*cameraman*" is also used to test the performance of the proposed method. The original, noisy, and denoised images are show in Figure 80(a)-(d), respectively. These results show that the restored images Figure 79(d) and Figure 80(c), (d) are of remarkable visual quality. In addition, as a quantitative performance measure, the improved signal to noise ratio (ISNR) of the proposed hybrid method are also shown in Figure 79 and Figure 80.

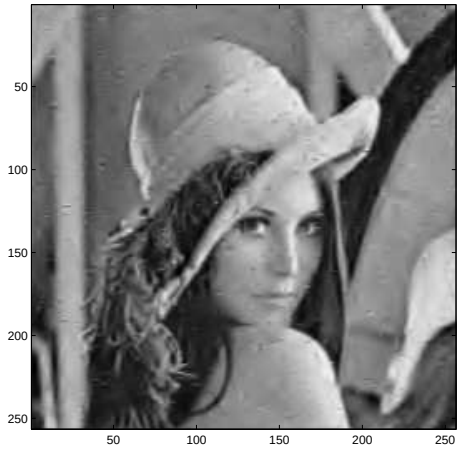
(a)The Original Image



(b) The Noisy Image



(c) Restored Image Using PF
(ISNR= 3.1991 dB)



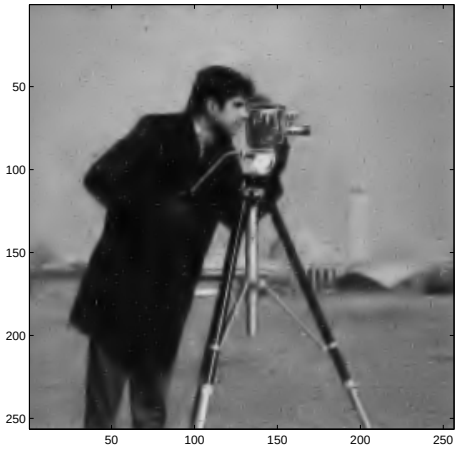
(d) PF with Translation
Invariant Hard Thresholding
(ISNR= 6.7126 dB)

Figure 79: *This figure shows: (a) the original image, (b) the noisy image, (c) the result obtained by only applying the particle filter, (d) the restored images obtained by the PF with the translation invariant thresholding.*

(a)The Original Image

(b) The Noisy Image

(c) PF with Translation Invariant Hard Thresholding ISNR= 5.4722 dB



(d) PF with Translation Invariant Soft Thresholding ISNR= 3.4959 dB

Figure 80: *As another example, this figure shows: (a) the original image, (b) the noisy image, (c) and (d) the restored images obtained by PF with translation invariant hard/soft thresholding.*

CHAPTER 7: CONCLUSION REMARKS AND FUTURE RESEARCH

In this dissertation, we have discussed the theories of particle filtering and its applications to various target tracking problems. We showed that the tracking estimations can be improved by designing better particle filter algorithms (*i.e.* designing better proposals), refining a target's dynamic models (*i.e.* using the multiple model method), and building better system observation models (*i.e.* utilizing multiple measurement cues). In particular, the following three particle filter frameworks have been developed:

- A particle filter based on the technique of state partitioning and parallel EKFs: The PF-SP-PEKF provides a general framework, in which two advanced particle filters were developed based this concept. The two particle filters are: (1) the multiple-sensor CPF-SP, which is used for target tracking in sensor networks, and (2) the multiple model particle filter based state partitioning (MMPF-SP) for visual target tracking. In these algorithms, the development of the proposal distribution still relies on the system linearization and the Gaussian assumption.

- A particle filter based on the Galerkin's projection method: More specifically, in this algorithm the Galerkin's method is adopted to generate the proposal distribution used in a PF framework. This algorithm has been successfully implemented for the problems of bearings-only tracking and visual target tracking, respectively. A limitation of this PF algorithm is the requirement of the computation over the entire state space when constructing the proposal distribution.

- A particle filter based on the idea of state space discretization: In this algorithm, the computation is only focused on the high probability regions in the target state space. In addition, a set of weighted cells are used to generate the proposal distribution. Based on this rationale, two sophisticated particle filters

have been developed: (1) An improved MMPF for maneuvering target tracking using both the range and the bearings measurements, and (2) An improved multiple-measurement MMPF for visual target tracking. In these algorithms, the estimation accuracy and the computational complexity can be balanced by choosing different cell size and different number of cells.

Extensive experiments and simulations have been conducted to evaluate the new designed algorithms, and promising results have been obtained. The possible future research can be carried out the following three areas:

- In multiple model particle filters, the transition matrix is predefined. However, it will be more interesting to design an algorithm that can update the transition matrix adaptively or an algorithm which has a switching structure that can choose a matrix from a finite set online. Also, a measure of model accuracy can be developed based on the variance of the particle weights.

- For visual target tracking applications, advanced pattern recognition and classification techniques can be introduced to construct a more advanced likelihood model using multiple measurement cues. For example, the support vector machine (SVM) has been proven to be a robust classifier for many applications. In future tracking algorithms, a SVM can be incorporated into the PF framework to achieve robust tracking.

- Advanced data association techniques can also be incorporated into the PF framework for multiple target tracking. Improved importance sampling techniques and new data association methods can be combined to achieve robust multiple target tracking.

In summary, there are many issues to be explored in nonlinear filtering for target tracking, especially in particle filtering. We expect the work in this dissertation will inspire more research in this area.

LIST OF REFERENCES

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

[2] A. Blake and M. Isard, *Active Contours,* Springer, New York, 1998.

[3] B. Ristic, S. Arulampalam and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*, Artech House Publishers, February 2004.

[4] A. Doucet, N. de Freitas and N. Gordon, *Sequential Monte Carlo Methods in Practice*, Springer, New York, 2001.

[5] S. Nardone, A. Lindgren and K. Gong, "Fundamental properties and performance of conventional bearings-only target motion analysis," *IEEE Transactions on Automatic Control*, vol. 29, no. 9, pp. 775 - 787, 1984.

[6] M. Orton and W. Fitzgerald, "A Bayesian approach to tracking multiple targets using sensor arrars and particle filters," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, February 2002.

[7] F. Zhao and L. Guibas, *Wireless Sensor Networks: An Information Processing Approach*, Morgan Kaufmann/Elsevier, San Francisco, 2004.

[8] I. Akyildiz, S. Weilian, Y. Sankarasubramaniam and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, issue 8, pp. 102 - 114, Aug. 2002.

[9] K. Lee and M. Reichardt, "Open standards for homeland security sensor networks," *IEEE Instrumentation and Measurement Magazine*, vol. 8, issue 5, pp. 14 - 21, December 2005.

[10] W. Chen, J. Hou and L. Sha, "Dynamic clustering for acoustic target tracking in wireless sensor networks," in *IEEE Transactions on Mobile Computing,* vol. 3, no. 3, pp. 258 - 271, August 2004.

[11] Y. Zhai, M. Yeary and J. Noyer, "Target tracking in a sensor network based on particle filtering and energy-aware design," *IEEE International Conference on Instrumentation and Measurements* (IMTC), pp. 1988 - 1992, April 2006.

[12] Y. Zhai, M. Yeary, J. Havlicek and G. Fan, "A New Centralized Sensor Fusion-Tracking Methodology Based on Particle Filtering for Power-Aware Systems," *IEEE Transactions on Instrumentation and Measurements*, submitted and under review.

[13] A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, H. Merkl, S. Pankanti, "Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking," *IEEE Signal Processing Magazine*, vol. 22, issue 2, pp. 38 - 51, March, 2005.

[14] A. Pentland, "Looking at people: sensing for ubiquitous and wearable computing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, issue 1, pp. 107 - 119, January 2000.

[15] S. Intille, J. Davis and A. Bobick, "Real-time closed-world tracking," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 697 - 703, June 1997.

[16] P. Pérez, J. Vermaak and A. Blake, "Data fusion for visual tracking with particles," *Proceedings of the IEEE*, vol. 92, issue 3, pp. 495 - 513, March 2004.

[17] Y. Zhai, M. Yeary, J. Havlicek, J. Noyer and P. Lanvin, "Visual tracking using sequential importance sampling with a state partition technique," *IEEE International Conference on Image Processing*, vol. 3, pp. 876 - 879, September 2005.

[18] Y. Zhai, M. Yeary and and J. Havlicek, "Visual target tracking using improved and computationally efficient particle filtering," *IEEE International Conference on Image Processing (ICIP)*, pp. 1757 - 1760, October 2006.

[19] Y. Zhai, M. Yeary and S. Nematifar, "Enhanced video surveillance using a multiple model particle filter," *IEEE Workshop on Signal Processing Applications for Public Security and Forensics*, April 2007, accepted.

[20] Y. Zhai, M. Yeary and N. Kehtarnavaz,"Visual tracking using multiple-model particle filter (MMPF) with state partition techniques," submitted to *IEEE Transactions on Circuits and Systems for Video Technology*, under review.

[21] A. Doucet, S. Godsill and C. Andieu,"On sequential Monte Carlo sampling methods for Bayesian filtering," *Journal of Statistics and Computing,* 10(3), pp. 197 - 208, 2000.

[22] P. Djurić, J. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. Bugallo and J. Miguez,, "Particle filtering," *IEEE Signal Processing Magazine*, vol. 20 , issue 5, pp. 19 - 38, September, 2003.

[23] M. 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, pp. 174 - 188, February 2002.

[24] N. Gordon, D. Salmond, A. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proceedings on Radar and Signal Processing*, vol. 140, issue 2, pp. 107 - 113, April 1993.

[25] R. van der Merwe, A. Doucet, N. de Freitas and E. Wan, "The unscented particle filter," *Technical Report CUED/F-INFENG/TR 380,* Department of Engineering, Cambridge University, 2000.

[26] M. Pitt and N. Shephard, "Filtering via simulation: auxiliary particle filters," *Journal of the American Statistical Association.*, vol. 94, no. 446, pp. 590 - 599, 1999.

[27] Z. Chen, T. Kirubarajan and M. Morelande, "Improved particle filtering schemes for target tracking," *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol.4, no. iv, pp. 145 - 148, March 2005.

[28] J. Liu and R. Chen, "Sequential Monte Carlo methods for dynamical systems," *Journal of the American Statistical Association*, vol. 93, issue. 443, pp. 1032 - 1044, September 1998.

[29] Y. Zhai, M. Yeary, "Implementing particle filters with Metropolis-Hastings algorithms," *IEEE Region 5 Conference*, pp. 149 - 152, April 2004.

[30] Y. Zhai and M. Yeary,"A novel nonlinear state estimation technique based sequential importance sampling and parallel filter banks," *IEEE International Conference on Control Applications*, pp. 1606 - 1611, August 2005.

[31] D. Lainiotis and P. Papaparaskeva, "A new class of efficient adaptive nonlinear filters (ANLF)," *IEEE Transactions on Signal Processing*, vol. 46, pp. 1730 - 1737, June 1998.

[32] D. Lainiotis and P. Papaparaskeva,, "Efficient algorithms of clustering adaptive nonlinear filters," *IEEE Transactions on Automatic Control*, vol. 44, pp. 1454 - 1459, July 1999.

[33] D. Lainiotis and P. Papaparaskeva,, "A partitioned adaptive approach to nonlinear channel equalization," *IEEE Transactions on Communication*, vol. 46, pp. 1325 - 1336, October 1998.

[34] V. Aidala, "Kalman filter behavior in bearings-only tracking applications," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-15, issue 1, pp. 29 - 39, January 1979.

[35] Brehard T. and J. Le Cadre, "Closed-form posterior cramer-rao bounds for bearings-only tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol.42 , issue 4, pp. 1198 - 1223, October 2006.

[36] R. Streit and M. Walsh, "Bearings-only target motion analysis with acoustic propagation models of uncertain fidelity," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, issue 4, pp. 1122 - 1137, October 2002.

[37] Y. Bar-Shalom, *Multitarget Multisensor Tracking: Advanced Applications*. Artech House, Boston, 1990.

[38] F. Martinerie, "Data fusion and target tracking using HMMs in a distributed sensor network," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, pp. 11 - 28, January 1997.

[39] X. Sheng and Y. Hu, "Energy based source localization," *IEEE Conference on Information Processing in Sensor Networks*, pp. 285 - 300, April 2003.

[40] X. Sheng and Y. Hu, "Sequential acoustic energy based source localization using particle filter in a distributed sensor network," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 972 - 975, May 2004.

[41] W. Zhang and G. Cao, "DCTC: Dynamic convoy tree-based collaboration for target tracking in sensor networks," *IEEE Transactions on Wireless Communication*, vol. 3, No. 5, pp. 1689 - 1701, September 2004.

[42] M. Coates, "Distributed particle filters for sensor networks," *IEEE Conference on Information Processing in Sensor Networks*, pp. 99 - 107, April 2004.

[43] C. Kreucher, K. Kastella and A. Hero, "Multi-target sensor menagement using Alpha-divergence measures," *IEEE Conference on Information Processing in Sensor Networks*, pp. 209 - 222, April 2003

[44] X. Sheng and Y. Hu, "Distributed particle filters for wireless sensor network target tracking," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 845 - 848, March 2005.

[45] N. Patwari, J. Ash, S. Kyperountas, A. Hero, R. Moses and M. Correal, "Locating the nodes: cooperative localization in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 22, issue 4, pp. 54 - 69, July 2005.

[46] S. Gezici, Z. Tian, G. Giannakis, H. Kobayashi, A. Molisch, H. Poor and Z. Sahinoglu, "Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks," *IEEE Signal Processing Magazine*, vol. 22, issue 4, pp. 70 - 84, July 2005.

[47] G. Sun, J. Chen, W. Guo and K. Liu, "Signal processing techniques in network-aided positioning: a survey of state-of-the-art positioning designs," *IEEE Signal Processing Magazine*, vol. 22, issue 4, pp. 12 - 23, July 2005.

[48] B. Sadler, R. Kozick and L. Tong, "Multi-modal sensor localization using a mobile access point," *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 4, pp. 753 - 756, March 2005.

[49] L. Kinsler, *Fundamentals of Acoustics*, John Wiley and Sons, New York, 1982.

[50] J. Kang, *Acoustics of Long Spaces: Theory and Design Practice*, Thomas Telford, London, 2002.

[51] R. Best and J. Norton, "A new model and efficient tracker for a target with curvilinear motion," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, issue 3, pp. 1030 - 1037, July 1997.

[52] X. Li and V. Jilkov, "Survey of maneuvering target tracking. Part I. Dynamic models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, issue 4, pp. 1333 - 1364, October 2003.

[53] X. Li and V. Jilkov, "Survey of maneuvering target tracking. Part V. Multiple-model methods," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, issue 4, pp. 1255 - 1321, October 2005.

[54] C. Musso, N. Oudjane and F. LeGland, "Improving regularised particle filter," in *Sequential Monte Carlo Methods in Practice* (A. Doucet, N. de Freitas and N. Gordon), New York: Springer, 2001.

[55] S. McGinnity and G. Irwin, "Multiple model bootstrap filter for maneuvering target tracking," *IEEE Transactions on Aerospace and Electronic systems*, vol. 36, no. 3, pp. 1006 - 1012, 2000.

[56] D. Angelova, T. Semerdjiev, V. Jilkov and E. Semerdjiev, "Application of Monte Carlo method for tracking maneuvering target in clutter," *Mathematics and Computers in Simulation*, vol. 1851, pp. 1 - 9, 2000.

[57] A. Farina, B. Ristic and D. Benvenuti, "Tracking a ballistic target: comparison of several nonlinear filters," *IEEE Transactions on Aerospace and Electronic systems*, vol. 38, no. 3, 2002.

[58] S. McGinnity and G. Irwin, "Multiple model bootstrap filter for maneuvering target tracking," *Sequential Monte Carlo Methods in Practice*, Springer, New York, 2001.

[59] Y. Bar-Shalom, K. Chang and H. Blom, "Tracking a maneuvering target using input estimation versus the interacting multiple model algorithm," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 25, issue 2, pp. 296 - 300, March 1989.

[60] H. Blom and Y. Bar-Shalom, Y."The interacting multiple model algorithm for systems with Markovian switching coefficients," *IEEE Transactions on Automatic Control*, vol. 33, issue 8, pp.780 - 783, August 1988.

[61] W. Blair and G. Watson, "IMM algorithm for solution to benchmark problem for tracking maneuvering targets," *Proceedings of SPIE*, pp. 476 - 488, 1994.

[62] T. Kronhamn, "Bearings-only target motion analysis based on a multi-hypothesis Kalman filter and adaptive ownship motion control," *IEE Proceedings - Radar, Sonar and Navigation*, vol. 145, issue 4, pp. 247 - 252, August 1998.

[63] J. Wang, D. Zhao, W. Gao and S. Shan, "Interacting multiple model particle filter to adaptive visual tracking," *Third International Conference on Image and Graphics*, pp. 568 - 571, December 2004.

[64] M. Arulampalam, N. Gordon, M. Orton and B. Ristic, "A variable structure multiple model particle filter for GMTI tracking," *Proceedings of the Fifth International Conference on Information Fusion*, vol. 2, pp. 927 - 934, July 2002.

[65] G. Kravaritis and B. Mulgrew, "Multitarget ground tracking with road maps and particle filters," *IEEE International Symposium on Signal Processing and Information Technology*, pp. 253 - 257, December 2005.

[66] Y. Boers and J. Driessen, "Interacting multiple model particle filter," *IEE Proceedings - Radar, Sonar and Navigation*, vol. 150, issue 5, pp. 344 - 349, October 2003.

[67] L. Hong, N. Cui, M. Bakich and J. Layne, "Multirate interacting multiple model particle filter for terrain-based ground target tracking," *IEE Proceedings - Control Theory and Applications*, vol. 153, issue 6, pp. 721 - 731, November 2006.

[68] A. Athalye, S. Hong and P. Djuric, "Distributed architecture and interconnection scheme for multiple model particle filters," *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 3, pp 920 - 923, May 2006.

[69] Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association*, Academic Press, 1998.

[70] M. Greiffenhagen, D. Comaniciu, H. Niemann and V. Ramesh, "Design, analysis, and engineering of video monitoring systems: an approach and a case study," *Proceedings of the IEEE* vol. 89, no. 10, pp. 1498 - 1517, October 2001.

[71] I. Haritaoglu, D. Harwood and L. Davis, "W4 real-time surveillance of people and their activities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.22, pp. 809 - 830, August 2000.

[72] M. Bruno and J. Moura, "Multiframe detector/tracker: optimal performance," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 37, issue 3, pp. 925 - 945, July 2001.

[73] A. Dawoud, M. Alam, A. Bal and C. Loo, "Target tracking in infrared imagery using weighted composite reference function-based decision fusion," *IEEE Transactions on Image Processing*, vol.15, no. 2, pp. 404 - 410, February 2006.

[74] N. Paragios and R. Deriche, "Geodesic active contours and level sets for the detection and tracking of moving objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, issue 3, pp. 266 - 280, March 2000.

[75] D. Comaniciu, V. Ramesh and P. Meer, "Real-time tracking of nonrigid objects using mean shift," *IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 142 - 149, June 2000.

[76] Y. Chen, Y. Rui and T. Huang, "JPDAF based HMM for real-time contour tracking," *IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 543 - 550, December 2001.

[77] R. Gonzalez and R. Woods, *Digital Image Processing*, 2nd edition, Prentice Hall, Upper Saddle River, New Jersey, January 2002.

[78] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, Prentice Hall, Upper Saddle River, New Jersey, 2003

[79] M. Isard and A. Blake, "Visual tracking by stochastic propagation of conditional density," *European Conference on Computer Vision*, pp. 343 - 356, April 1996.

[80] Y. Rui and Y. Chen, "Better proposal distributions: objects tracking using unscented particle filter," *IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 786 - 793, December 2001.

[81] M. Bruno, "Bayesian methods for multiaspect target tracking in image sequences," *IEEE Transactions on on Signal Processing*, vol.52, no.7, pp. 1848 - 1861, July 2004.

[82] N. Bouaynaya, Q. Wei and D. Schonfeld, "An Online Motion-Based Particle Filter for Head Tracking Applications," *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2, pp. 225 - 228, March 2005.

[83] D. Comaniciu, V. Ramesh and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, issue 5, pp. 564 - 577, May 2003.

[84] S. Zhou, R. Chellappa, B. Moghaddam, "Visual tracking and recognition using appearance-adaptive models in particle filters," *IEEE Transactions on Image Processing,* pp. 1491 - 1506, Nov. 2004.

[85] M. Bolić, P. Djurić and S. Hong, "Resampling algorithms for particle filters: a computational complexity perspective," *EURASIP Journal on Applied Signal Processing*, vol. 15, pp. 2267 - 2277, 2004.

[86] P. Tan and L. Rasmussen, "Multiuser detection in CDMA a comparison of relaxations, exact, and heuristic search methods," *IEEE Transactions on on Wireless Communication*, vol. 3, no. 5, pp. 1802 - 1809, September 2004.

[87] M. Zaveri, S. Merchant, U. Desai, "Interacting multiple model-based tracking of multiple point targets using an expectation maximization algorithm in an infrared image sequence," *Proceedings of SPIE*, vol. 5150, pp. 303-314, June 2003.

[88] N. Kehtarnavaz and M. Gamadia, *Real-Time Image and Video Processing: From Research to Reality*. San Rafael, CA: Morgan and Claypool Publishers, 2006.

[89] J. Vermaak, A. Doucet and P. Perez, "Maintaining Multi-Modality through Mixture Tracking," *IEEE International Conference on Computer Vision*, pp. 1110 - 1116, vol.2 October 2003.

[90] H. Wang, D. Suter and K. Schindler, "Effective appearance model and similarity measure for particle filtering and visual tracking," *9th European Conference on Computer Vision*, vol. 3953, pp. 606-618, May 2006.

[91] A. Jacquot, P. Sturm and O. Ruch, "Adaptive Tracking of Non-Rigid Objects Based on Color Histograms and Automatic Parameter Selection," *Proceedings of the IEEE Workshop on Motion and Video Computing* pp. 103 - 109, vol.2, January 2005.

[92] Y. Zhai, M. Yeary and D. Zhou, "Target tracking using a particle filter based on the projection method," *IEEE Conference Acoustics, Speech, and Signal Processing (ICASSP)*, accepted, May 2007.

[93] J. Gunther, R. Beard, J. Wilson, T. Oliphant and W. Stirling, "Fast nonlinear filtering via Galerkin's method," *American Control Conference (ACC)*, vol. 5, pp. 2815 - 2819, June 1997.

[94] R. Beard, J. Gunther, J. Lawton and W. Stirling, "Nonlinear projection filter based on Galerkin approximation," *AIAA Journal of Guidance Control and Dynamics*, vol. 22, no.2, pp. 258 - 266, March-April, 1999.

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

[96] X. R. Li and V. P. Jilkov, "A survey of maneuvering target tracking-part III: Measurement models," *SPIE Conference on Signal and Data Processing of Small Targets*, vol. 4473, pp. 423 - 446, August 2001.

[97] W. Wu and P. Cheng, "A nonlinear IMM algorithm for maneuvering target tracking," *IEEE Transactions on Aerospace and Electronics Systems*, vol. 30, no. 3, pp. 875 - 886, July 1994.

[98] S. Godsill and T. Clapp, "Improvement Strategies for Monte Carlo Particle Filters," in *Sequential Monte Carlo Methods in Practice* (A. Doucet, N. de Freitas and N. Gordon), New York: Springer, 2001.

[99] C. Berzuini, N. Best, W. Gilks and C. Larizza, "Dynamic conditional independence models and Markov chain Monte Carlo methods," *Journal of the American Statistical Association*, vol. 92, pp. 1403141, 1997.

[100] C. Robert and G. Casella, "The Metropois-Hastings Algorithm," *Monte Carlo Statistical Methods*, New York, Springer, 2001.

[101] M. Banham and A. Katsaggelos,"Digital image restoration," *IEEE Signal Processing Magazine*, pp. 24 - 41, Mar, 1997.

[102] D. Rao, E. Plotkin and M. Swamy, "A hybrid filter for restoration of color images in the mixed noise environment," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3680 - 3683, May 2002.

[103] S. Mallat, *A wavelet tour of signal processing 2nd Edition*, Academic Press, 1999.

[104] D. Donoho and I. Johnstone, "Ideal spatial adaptation via wavelet shringkage," *Biometrica*, pp. 425 - 455, 1994.

[105] M. Banham and A. Katsaggelos, "Spatially-adaptive wavelet-Based mutiscale image restoration," *IEEE Transactions on Image Processing,* pp. 619 - 634, April 1996.

[106] E. Arnaud, E. Memin and B. Cernuschi-Frias, "Conditional Filters for Image Sequence-Based TrackingApplication to Point Tracking," *IEEE Transactions on Image Processing,* pp. 63 - 79, January 2005.

[107] M. de Bruijne and M. Nielsen, "Image segmentation by shape particle filtering," *IEEE International Conference on Pattern Recognition* pp. 722 - 725, August 2004.

[108] D. Angwin and H. Kaufman, "Image restoration using a reduced order model Kalman filter," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1000 - 1003, April 1988.

[109] D. Rao, M. Swamy and E. Plotkin, "Image restoration using an hybrid approach based on DWT and SMKF," *IEEE International Conference on Image Processing*, pp. 249 - 252, October 2001.

[110] S. Chang, B. Yu and M. Vetterli, "Adaptive Wavelet Thresholding for Image Denoising and Compression," *IEEE Transactions on Image Processing*, pp. 1532 - 1546, September 2000.

Appendix CHAPTER A: THE PROOF OF THE PARTICLE WEIGHT UPDATE EQUATION

In this appendix, a detailed derivation of a particle filter is provided. Short proofs can be found in [3][4][25]. As indicated in Chapter 2, the objective of nonlinear filtering is to estimate the posterior distribution $p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})$ and the expectation $\mathbb{E}[\mathbf{f}(\mathbf{x}_{0:t})]$, where $\mathbf{f}(\cdot)$ denotes the state process in the nonlinear system given in equation (1). The expectation is given as follows:

$$\begin{aligned}\mathbb{E}[\mathbf{f}(\mathbf{x}_{0:t})] &= \int \mathbf{f}(\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})\ d\ \mathbf{x}_{0:t} \\ &= \int \mathbf{f}(\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) \frac{q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})}\ d\ \mathbf{x}_{0:t} \end{aligned} \tag{108}$$

where $q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})$ is called the proposal distribution, which is an arbitrary density function that has the support from the posterior $p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})$. According the Bayes' theorem, equation (108) can be written in the following form:

$$\int \mathbf{f}(\mathbf{x}_{0:t}) \frac{p\left(\mathbf{z}_{1:t}|p(\mathbf{x}_{0:t})\right)}{p(\mathbf{z}_{1:t})} \frac{q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})}\ d\mathbf{x}_{0:t} \quad . \tag{109}$$

Since $p(\mathbf{z}_{1:t})$ is not a function of $\mathbf{x}_{0:t}$, equation (109) can be modified as:

$$\begin{aligned}&\int \mathbf{f}(\mathbf{x}_{0:t}) \frac{p\left(\mathbf{z}_{1:t}|p(\mathbf{x}_{0:t})\right)}{p(\mathbf{z}_{1:t})} \frac{q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})}\ d\mathbf{x}_{0:t} \\ =&\frac{1}{p(\mathbf{z}_{1:t})} \int \mathbf{f}(\mathbf{x}_{0:t}) \frac{p\left(\mathbf{z}_{1:t}|\mathbf{x}_{1:t}\right) p(\mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})} q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})\ d\mathbf{x}_{0:t} \\ =&\frac{1}{p(\mathbf{z}_{1:t})} \int \mathbf{f}(\mathbf{x}_{0:t}) \omega(\mathbf{x}_{0:t}) q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})\ d\mathbf{x}_{0:t} \end{aligned} \tag{110}$$

where $\omega(\mathbf{x}_{0:t})$ is known as the un-normalized importance weight defined as follows:

$$\omega(\mathbf{x}_{0:t}) = \frac{p\left(\mathbf{z}_{1:t}|\mathbf{x}_{1:t}\right) p(\mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})} \quad . \tag{111}$$

Next, equation (110) is rewritten in the following form:

$$
\begin{aligned}
&= \left(\int \mathbf{f}(\mathbf{x}_{0:t}) \omega(\mathbf{x}_{0:t}) q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) \; d\mathbf{x}_{0:t} \right) \Big/ \left(\int p(\mathbf{z}_{1:t}|\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t}) d\mathbf{x}_{0:t} \right) \\
&= \left(\int \mathbf{f}(\mathbf{x}_{0:t}) \omega(\mathbf{x}_{0:t}) q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) \; d\mathbf{x}_{0:t} \right) \Big/ \left(\int p(\mathbf{z}_{1:t}|\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t}) \frac{q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})} d\mathbf{x}_{0:t} \right) \\
&= \left(\int \mathbf{f}(\mathbf{x}_{0:t}) \omega(\mathbf{x}_{0:t}) q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) \; d\mathbf{x}_{0:t} \right) \Big/ \left(\int \frac{p(\mathbf{z}_{1:t}|\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})} q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) d\mathbf{x}_{0:t} \right) \\
&= \left(\int \mathbf{f}(\mathbf{x}_{0:t}) \omega(\mathbf{x}_{0:t}) q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) \; d\mathbf{x}_{0:t} \right) \Big/ \left(\int \omega(\mathbf{x}_{0:t}) q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) \; d\mathbf{x}_{0:t} \right) \qquad (112)
\end{aligned}
$$

Equation (112) can be taken as:

$$
= \frac{\mathbb{E}_{q(\cdot|\mathbf{y}_{1:t})} \left[\mathbf{f}(x_{0:t}) \omega(\mathbf{x}_{0:t}) \right]}{\mathbb{E}_{q(\cdot|\mathbf{y}_{1:t})} \left[\omega(\mathbf{x}_{0:t}) \right]}
$$

where $\mathbb{E}_{q(\cdot|\mathbf{y}_{1:t})}$ denotes that the expectation that is taken over the proposal distribution. If the samples (also known as particles) are drawn from the proposal distribution, then the expectations can be approximated by the following equation:

$$
\begin{aligned}
&= \frac{\frac{1}{N_s} \sum_{i=1}^{Ns} \mathbf{f}(\mathbf{x}_{0:t}^{(i)}) \omega(\mathbf{x}_{0:t}^{(i)})}{\frac{1}{N_s} \sum_{i=1}^{Ns} \omega(\mathbf{x}_{0:t}^{(i)})} \\
&= \sum_{i=1}^{Ns} \mathbf{f}(\mathbf{x}_{0:t}^{(i)}) \tilde{\omega}(\mathbf{x}_{0:t}^{(i)})
\end{aligned}
$$

where $\tilde{\omega}(\mathbf{x}_{0:t}^{(i)})$ is the normalized importance weight. Then, we have

$$
p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) \approx \sum_{i=1}^{Ns} \tilde{\omega}(\mathbf{x}_{0:t}^{(i)}) \delta \left(\mathbf{x}_{0:t} - \mathbf{x}_{0:t}^{(i)} \right) \quad . \qquad (113)
$$

In addition, if N_s approaches to the infinity, the above summation approximation will converge to $p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})$. The above proof is based on the concept of importance sampling (IS). Next, the framework of the sequential importance sampling (SIS) is developed. In

order to compute the posterior in a sequential framework, the proposals that satisfies the following condition will be adopted:

$$q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) = q(\mathbf{x}_{0:t-1}|\mathbf{z}_{1:t-1})q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t}) \quad . \tag{114}$$

In addition, it is assumed that the system is Markovian and the measurements are independent given the corresponding states. Then, the following equalities hold:

$$\begin{aligned} p(\mathbf{x}_{0;t}) =& p(\mathbf{x}_0) \prod_{j=1}^{t} p(\mathbf{x}_j|\mathbf{x}_{j-1}) \\ =& \left[p(\mathbf{x}_0) \prod_{j=1}^{t} p(\mathbf{x}_j|\mathbf{x}_{j-1}) \right] p(\mathbf{x}_t|\mathbf{x}_{t-1}) \end{aligned} \tag{115}$$

$$\begin{aligned} p(\mathbf{z}_{1:t}|\mathbf{x}_{0:t}) =& \prod_{j=1}^{t} p(\mathbf{z}_j|\mathbf{x}_j) \\ =& \left[\prod_{j=1}^{t-1} p(\mathbf{z}_j|\mathbf{x}_j) \right] p(\mathbf{z}_t|\mathbf{x}_t) \end{aligned} \tag{116}$$

Plug equations (114) (115) and (116) into equation (111):

$$\omega(\mathbf{x}_{0:t}) = \frac{\left[\prod_{j=1}^{t-1} p(\mathbf{z}_j|\mathbf{x}_j) \right] \cdot p(\mathbf{z}_t|\mathbf{x}_t) \cdot \left[p(\mathbf{x}_0) \prod_{j=1}^{t} p(\mathbf{x}_j|\mathbf{x}_{j-1}) \right] \cdot p(\mathbf{x}_t|\mathbf{x}_{t-1})}{q(\mathbf{x}_{0:t-1}|\mathbf{z}_{1:t-1}) q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t})}$$

$$= \frac{p(\mathbf{z}_{1:t-1}|\mathbf{x}_{0:t-1}) p(\mathbf{x}_{0:t-1})}{q(\mathbf{x}_{0:t-1}|\mathbf{z}_{1:t-1})} \cdot \frac{p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{x}_{t-1})}{q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t})}$$

$$= \omega(\mathbf{x}_{0:t-1}) \cdot \frac{p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{x}_{t-1})}{q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t})} \tag{117}$$

Equation (117) is the particle weight update equation, which is same as equation (13) in Chapter 2.

Appendix CHAPTER B: A DEMO CODE FOR VISUAL TRACKING USING BOOTSTRAP FILTER

Main Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                          %%
%%  This code implements a target tracking algorith based on               %%
%%  particle filtering and cross-corelation template matching.              %%
%%                                                                          %%
%%  "Particle filter" (PF), also known as "sequential importance            %%
%%  sampling", is a family of new and powerfuly nonlinear                   %%
%%  non-Gaussian filters for state estimation. For more details,            %%
%%  see [1]-[3]                                                             %%
%%                                                                          %%
%%  References:                                                             %%
%%  [1] M. Arulampalam, and et. al., ``A tutorial on particle               %%
%%  filters for online nonlinear/non-Gaussian Bayesian tracking,"           %%
%%  IEEE Trans. Signal Process. vol. 50, pp. 174-188, Feb., 2002.           %%
%%                                                                          %%
%%  [2] A. Blake and M. Isard, Active Contours, NY: Springer, 1998.         %%
%%                                                                          %%
%%  [3] Y. Zhai, and et. al., ``Visual tracking using sequential            %%
%%  importance sampling with a state partition technique,'' IEEE            %%
%%  ICIP, vol. 3, no. 3, pp. 876-879, Sep. 2005.                            %%
%%                                                                          %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%  ---------------------------------------------------------------
%%  Created in: 6-10-2006    by Y.Zhai
%%  Modifications:
%%  (1) The MMPF using edge-detection were deleted. Date: 2-2-2007
%%  (2) Variable names for particles were changed.  Date: 2-2-2007
%%  ---------------------------------------------------------------

clear all
close all
clc

%loading the video data (Matlab format)
load NoisyFrameData.mat

% --------  Setup initial conditions ------------
var_w=5; var_v=8; var_o=1;

ctr_1=[69 63]; % target centers in the first two frames
ctr_2=[68 61];
v_ini=ctr_2-ctr_1; % initial velocity vector
```

```
% get the video size
[RowMax ColMax VideoLength] = size(NoisyFrameData)

% number of particles (samples)
Ns=200;

% get the initial gray-scale template
G_temp_hw=3;
G_temp_hh=3;
[G_template, Temp_bdries] = Func_GetWindow( ...
    ctr_1, G_temp_hw, G_temp_hh, NoisyFrameData(:,:,1) );

% the aid window size
AW_hw=20; AW_hh=20;

% Setup particle buffers
p_row  = zeros(VideoLength,Ns);    % These are the particles for the estimates
p_col  = zeros(VideoLength,Ns);    % of locations and velocities. Note that
p_vrow = zeros(VideoLength,Ns);    % there's no need to store them for all t.
p_vcol = zeros(VideoLength,Ns);    % The only reason for doing this here is for
                                   % the optional tracking performance analysis
                                   % (not in this code)
% Initializing the particles:
p_row(1,:)  = ctr_1(1)*ones(1,Ns);
p_col(1,:)  = ctr_1(2)*ones(1,Ns);
p_vrow(1,:) = v_ini(1)*ones(1,Ns);
p_vcol(1,:) = v_ini(2)*ones(1,Ns);

est_r(1)=ctr_1(1);  % temporary variables
est_c(1)=ctr_1(2);

estimated_centroid=zeros(VideoLength,2); % final estimations
estimated_centroid(1,:)=ctr_1;
% ------------- end of initial condition ------------------------

% ------------- the main loop -------------------------
for FrNo=2:VideoLength
    % FrNo is the frame number ( or time index)

    % Show the current frame No.
    fprintf('frame: %g \n', FrNo)

    %% target dynamic model (constant velocity model):
    %%
    %%  [ r(t)   ]   [ 1 0 1 0 ][ r(t-1)   ]   [     0.5         0]
    %%  [ c(t)   ]=  [ 0 1 0 1 ][ c(t-1)   ] + [       0       0.5] [w_r(t-1)]
    %%  [ v_r(t) ]   [ 0 0 1 0 ][ v_r(t-1) ]   [       1         0] [w_c(t-1)]
    %%  [ v_c(t) ]   [ 0 0 0 1 ][ v_c(t-1) ]   [       0         1]
```

```
%%
%%  where r and c are row and column indices, v_r and v_r denote
%%  the row and column velocities, respectively.

% Particle propagation ( prediction based the dynamic model )
particle_row = p_row( FrNo-1,:) + p_vrow( FrNo-1,: ) + var_w^0.5*randn(1,Ns);
particle_col = p_col( FrNo-1,:) + p_vcol( FrNo-1,: ) + var_w^0.5*randn(1,Ns);


% Take the mean of the predicted centers as the aid-window center
AW_Ctr = round( [ mean( particle_row ) ...
                  mean( particle_col ) ] );


% Target detection using cross correlation to find the a location
% that the target has the highest probability to present.
%
% obser_ctr: observed target center
obser_ctr = Func_TempMatching( NoisyFrameData(:,:,FrNo), ...
                                     G_template, ...
                                     AW_Ctr, AW_hw, AW_hh);

% We use 2D symmetric Gaussian to approximate the likelihood model.
% Then calculate the likelihood of each particle
lik_row = ( 2*pi*var_o )^(-0.5) * exp( ...
            -( obser_ctr(1) - particle_row ).^2 / (2*var_o));

lik_col = ( 2*pi*var_o )^(-0.5) * exp( ...
            -( obser_ctr(2) - particle_col ).^2 / (2*var_o));

% Calculate overall likelihood
lik = lik_row.*lik_col;

% Normalize the likelihood.
% The particle weight is the likelihood ( The condensation method )
lik=lik/sum(lik);

% Particles needs to resample, see ref [1] - [3] for theories behind this.
% Using the systematic resampling method
outIndex = systematicR(1:Ns,lik');

p_row(FrNo,:)   = particle_row(outIndex);
p_col(FrNo,:)  = particle_col(outIndex);

% The estimated locations are the mean of the resampled particle.
est_r(FrNo)=mean(p_row(FrNo,:));
est_c(FrNo)=mean(p_col(FrNo,:));

% Calculate the final estimated velocities
```

```
    p_vrow(FrNo,:) = est_r(FrNo)-est_r(FrNo-1);
    p_vcol(FrNo,:) = est_c(FrNo)-est_c(FrNo-1);

    % The final estimaed center (integer row and column index)
    % Keep the record of all tracking history for analysis
    est_centroid(FrNo,:)=[round(est_r(FrNo))  round(est_c(FrNo)) ];

    % need update the grayscale template sequentially
    [G_template, Temp_bdries] = Func_GetWindow( ...
                                [est_centroid( FrNo,1 ) ...
                                 est_centroid( FrNo,2 )], ...
                                 G_temp_hw, G_temp_hh, ...
                                 NoisyFrameData(:,:,FrNo) );

    %% plotting the tracking result ON-LINE (optional)
    OUTPUT(:,:,FrNo-1)=NoisyFrameData(:,:,FrNo);
    %% plot the center
    OUTPUT( est_centroid( FrNo,1 ), ...
            est_centroid( FrNo,2 )-2 : ...
            est_centroid( FrNo,2 )+2, ...
            FrNo-1)=0;

    OUTPUT( est_centroid( FrNo,1 )-2 : ...
            est_centroid( FrNo,1 )+2, ...
            est_centroid( FrNo,2 ), ...
            FrNo-1)=0;

    colormap(gray(256));
    image(OUTPUT(:,:,FrNo-1));
    pause(0.1)

end
% ------------- end of the main loop ------------------------

%%  %%% Making an avi file ( Optional )
%%
%%  aviobj = avifile('TrackingResult.avi', 'fps',15,'quality', 100);
%%
%%  for i=1:VideoLength-1  %% from the second frame
%%      colormap(gray(256));
%%      image(OUTPUT(:,:,i));
%%      frame = getframe(gca);
%%      aviobj = addframe(aviobj,frame);
%%  end
%%  aviobj = close(aviobj);
%%
%%  close all
```

Func_GetWindow

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  This function returns a windowed image
%%  which will be used either as the target template
%%  or the aid-window for further processing.
%%
%%  In other words, this function cuts off a
%%  piece of image from the input image.
%%  This small piece of image will be used
%%  either as the target template (grayscale template)
%%  or the aid-window for further processing.
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%  Function Usage:
%%
%%  [Windowed_Image, AW_bdries]=Func_GetWindow(center, AW_hw, AW_hh, InputImage)
%%
%%  Function inputs :
%%  center ( 1-by-2 vector ) -- the window center
%%  AW_hw ( scalar ) -- half of the window width  (AW: stands for aid window)
%%  AW_hh ( scalar ) -- half of the window height
%%  InputImage ( actual frame size) -- the input video frame
%%
%%  Function returns:
%%  AW_bdries ( 1-by-4 vector ) -- the corners of the aid-window or the template
%%  Windowed_Image ( AW_hh*2+1--by--AW_hw*2+1 ) -- the aid-window or the template

%% ------------------------------------------------------  %%
%%  Created in: 6-2-2006    by Y.Zhai
%%  Modifications:
%%  (1) adding the returned variable "AW_bdries"  Date: 6-6-2006
%% ------------------------------------------------------  %%
function [Windowed_Image, AW_bdries]=Func_GetWindow(center, AW_hw, AW_hh, InputImage)

[RowMax, ColMax]=size(InputImage);

% find the aid-window corners
AW_RowMin = max(1, center(1)-AW_hh);
AW_RowMax = min(RowMax, center(1)+AW_hh);
AW_ColMin = max(1, center(2)-AW_hw );
AW_ColMax = min(ColMax, center(2)+AW_hw);

% the aid-window corners to be returned
AW_bdries=[AW_RowMin, AW_RowMax, AW_ColMin, AW_ColMax];

% the aid-window to be returned
Windowed_Image=InputImage( AW_RowMin : AW_RowMax, AW_ColMin : AW_ColMax);
```

Func_TempMatching

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  This function implements a template matching using the
%%  cross correlation method, for more information see [1] [2].
%%
%%  More specifically, this function filters the input
%%  image with the target template (actually, it only
%%  searchs inside a piece of the input image: the
%%  aid-window), and returns the location of a pixel,
%%  which has the largest respone.
%%
%%  The target is assumed to have the highest probability
%%  to appear at this location.
%%
%% References:
%% [1]  R. Gonzalez and R. Woods, ``Chapter 12 Object Recognition,''
%% \emph{Digital Image Processing}, 2nd Edition, Prentice Hall, NJ, 2001.
%%
%% [2]  D. Forsyth and J. Ponce, ``Chapter 7 Linear Filters: filter as
%% templates,'' \emph{Computer Vision: A Modern Approach}, Prentice
%% Hall, NJ, 2003
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%  Function Usage:
%%  TargetCtr_Temp_Matching = Func_TempMatching( ...
%%       InputImage, TemplateIM, WindowCenter, AW_hw, AW_hh)
%%
%%  function inputs :
%%  InputImage (2-D array) -- the input image
%%  TemplateIM (2-D array) -- the target template image
%%  WindowCener( 1-by-2 vertor) -- the aid-window
%%  AW_hw ( scalar ) -- half of the window width
%%  AW_hh ( scalar ) -- half of the window height
%%
%%  function returns:
%%  TargetCtr_Temp_Matching ( 1-by-2 vertor)
%%        -- Target Center (i.e. the target location)

%% ------------------------------------------------
%%  Created in: 6-2-2006  by Y.Zhai
%%  Modifications:
%% (1) replaced self-written cross-corelation function
%%     with filter2    Date: 6-5-2006
%% ------------------------------------------------

function   TargetCtr_Temp_Matching = Func_TempMatching( ...
            InputImage, TemplateIM, WindowCenter, AW_hw, AW_hh)
```

```
% Get the aid-window
[Windowed_Image, AW_bdries] = Func_GetWindow(WindowCenter, ...
                              AW_hw, AW_hh, InputImage);
AW_RowMin=AW_bdries(1);
AW_RowMax=AW_bdries(2);
AW_ColMin=AW_bdries(3);
AW_ColMax=AW_bdries(4);

% 2D filtering
out = filter2(TemplateIM,Windowed_Image);

% seek the peak
out_peak = max(max(out));

% find the location
[TarI, TarJ] = find(out == out_peak);
TarI=AW_RowMin+TarI-1;
TarJ=AW_ColMin+TarJ-1;

% return the location
TargetCtr_Temp_Matching=[TarI TarJ];
```