"PARTICLE FILTERS BASED VISUAL OBJECT TRACKING"

A DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE OF

MASTER OF TECHNOLOGY

IN

SIGNAL PROCESSING AND DIGITAL DESIGN

SUBMITTED BY

AVINASH RATRE

2K11/SPD/20(PT)

UNDER THE GUIDANCE OF

PROF. RAJIV KAPOOR

PROFESSOR AND HEAD, E&C ENGG. DEPTT., DTU

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(FORMERLY DELHI COLLEGE OF ENGINEERING)

BAWANA ROAD, DELHI – 110042

JULY, 2014

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(FORMERLY DELHI COLLEGE OF ENGINEERING)

BAWANA ROAD, DELHI – 110042

DECLARATION BY CANDIDATE AND CERTIFICATE

I, **AVINASH RATRE**, Roll No. **2K11/SPD/20 (Part Time)**, student of **M. Tech. (Signal Processing and Digital Design)**, hereby declare the dissertation titled **“Particle Filters Based Visual Object Tracking”** under the esteemed supervision of **PROF. RAJIV KAPOOR**, Professor and Head, Department of Electronics and Communication Engineering, Delhi Technological University, in partial fulfilment of the requirement for the award of the degree of Master of Technology in Signal Processing and Digital Design is an authentic record of my own work. The work has not been submitted to any other University or institution for the award of any Degree or diploma.

Place: Delhi **(AVINASH RATRE)**

Date: Roll No - **2K11/SPD/20 (PT)**

This is to certify that the above statement made by the candidate is true and correct to the best of my knowledge.

(PROF. RAJIV KAPOOR)
MTech Supervisor &
Professor and Head,
Deptt. of E&C Engg., DTU

ACKNOWLEDGEMENT

The success and final outcome of any dissertation work require a lot of guidance and assistance from many people and I am extremely fortunate to have got this all along the completion of my MTech major project.

I owe my profound gratitude to my teacher and supervisor, **Prof. Rajiv Kapoor, Professor, Department of Electronics and Communication Enngineering, DTU**, for introducing me to the present topic and for his inspiring guidance, motivation and valuable suggestions throughout the project work. His readiness for consultation at all times, his educative comments, his concern and assistance even with practical things have always been invaluable.

I respect and thank **Prof. Rajiv Kapoor, Head**, **Department of Electronics and Communication Enngineering, DTU**, for giving me an opportunity to do the major project and providing me all the necessary support which made me complete the major project on time.

I also like to pay my sincere gratitude to the **Hon'ble Vice Chancellor, DTU** for providing the necessary support during the major project.

I also acknowledge my profound sense of gratitude to all the faculty members of the department who have been instrumental for providing me the technical knowledge during MTech course and major project.

I also extend my sincere regards to all the non-teaching staff of the department and officers of the University for their timely support.

I am also thankful to my family members for their encouraging and unconditional support during this course.

Place: Delhi **(AVINASH RATRE)**

Date: Roll No - **2K11/SPD/20 (PT)**

ABSTRACT

This dissertation is concerned with the tracking of an object of interest in the video and analysis of different parameters with reference to the record of trajectories over the frames apart from the rigorous treatment of Particle filter theory.

In this dissertation, rigorous mathematical aspects and various algorithms of Particle filters have been discussed with MATLAB simulation results. Comparisons of Sequential Importance Resampling (SIR) Particle filter, Auxiliary Particle filter (APF) and Gaussian Particle filter (GPF) in terms of Mean Square Error (MSE) have also been simulated using MATLAB along with their states and estimate states for univariate non-stationary growth. Conditional linearity in the dynamic nonlinear system under the Rao-Blackwell Particle filter has also been simulated using MATLAB for tracking a maneuvering target along with its comparison with particle filter.

Under visual object tracking part, an SIR Particle filter based tracking of an object of interest in the video has been simulated using two methods such as, intensity histogram and color histogram. The shape of the object is modeled as an ellipse, along which an intensity gradient is estimated, while the interior appearance is modeled using a color histogram. MATLAB simulation shows the Belief states over the frames with handling of visual clutter along with the visual object trajectory, HSV cdf, and 3D histogram of target distribution. Other simulation shows the prediction of object position, bounding ellipse particles, posterior distribution of the moving object over the frames and SIR resampling of the tracked object.

The results show that Color histogram based particle filter tracking is robust to partial occlusion, rotation, scaling, and changes in illumination and pose. Performance of the Particle filtering tracking can also be improved by using more particles. Results also show that the tracking method is able to keep track for a fairly long time, despite the presence of clutter.

CONTENTS

TABLES

FIGURES

SYMBOLS

$x(n)$	:	state vector
$y(n)$	:	observation vector
$v(n)$	:	observation noise vector
$g(.)$	:	known function
$x^{(m)}(n),\ w^{(m)}(n)$	:	m - th particle and associated weight
χ(n)	:	discrete random measure
$E(h(X(n)))$	:	Expectation of functions of random process $X(n)$
$\pi(x(n))$	:	proposal distribution
$f(x(n)\vert y(1{:}n))$	:	a posteriori pdf
$f(x(n)\vert y(1{:}n))$	:	filtering pdf
M	:	number of particles
M_{eff}	:	effective particle size
E	:	expectation operator
μ	:	moments
N	:	Gaussian function
θ	:	vector of fixed parameters
h^2	:	smoothing parameter
$x_n(n)$	:	conditionally linear state
$v_{1,n}(n)$	:	state noise vector
$g_{1,n}(\cdot)$	:	nonlinear state transition functions

CHAPTER 1
INTRODUCTION

1.1 MOTIVATION

Visual object tracking is a classical computer vision problem with many important applications in the areas such as robotics, video surveillance and driver assistance. It is the task of estimating over time the position of objects of interest in image sequences. Humans perform accurate visual tracking with little effort, while it remains a difficult computer vision problem. It imposes major challenges, such as appearance changes, occlusions and background clutter.

Visual object tracking algorithms can roughly be divided into two categories: deterministic methods and stochastic methods. Deterministic methods typically track the object by performing an iterative search for a similarity between the template image and the current one. This methods includes the examples of Background subtraction, inter-frame difference, optical flow, skin color extraction, etc. On the other hand, stochastic methods use the state space to model the underlying dynamics of the tracking system such as Kalman filter and Particle filter. Kalman filter is a common approach for dealing with target tracking in the probabilistic framework but it cannot resolve the tracking problem when the model is nonlinear and non-Gaussian.

To overcome these problems, Particle filter, also known as Sequential Monte Carlo (SMC), is the most popular approach which recursively constructs the posterior pdf of the state space using Monte Carlo Integration. It approximates a posterior probability density of the state such as the object position by using samples or particles. The probability distribution of the state of the tracked object is approximated by a set of particles, of which each state is denoted as the hypothetical state of the tracked object, and its weight.

Many problems in adaptive filtering are nonlinear and non-Gaussian. Of the many methods proposed in the literature for solving such problems, particle filtering has become one of the most popular. Some of the most important problems in the signal processing require the sequential processing of data. The data describe a system that is mathematically represented by equations that model its evolution with time, where the evolution contains a random component. The system is defined by its state variable of which some are dynamic and some are constant. A typical assumption about the state of the system is that it is Markovian, which means that given the state at a previous time instant, the distribution of the state at the current time instant does not depend on any other older state. The state is not directly observable. Instead, we acquire measurements which are functions of the current state of the system. The measurements are degraded by noise or some other random perturbation, and they are obtained sequentially in time. The main objective of sequential signal processing is the recursive estimation of the state of the system based on the available measurements. The methods that are developed for estimation of the state are usually called filters. We may also be interested in estimates of the state in the past or would like to predict its values at future time instants.

Particle filter can be applied to any state space model where the likelihood and the prior are computable up to proportionality constants. The accuracy of the method depends on how well we generate the particles and how many particles we use to represent the random measure.

Particle filtering has been used in many different disciplines. They include surveillance guidance, and obstacle avoidance systems, robotics, communications, speech processing, seismic signal processing, system engineering, computer vision, and econometrics. Some of the applications are the following:

- Multiple target tracking using a combination of video and acoustic sensors.
- Direction of arrival tracking of multiple moving targets using passive sensor array.
- Simultaneous detection and tracking of multiple targets.
- Tracking of moving acoustic source in a moderately reverberant room.
- Joint tracking of location and speaking activity of several speakers in a meeting room with a microphone array and multiple cameras.
- Speech signal processing
- Tracking of multi-aspect targets using image sequences with background clutter.
- Mobility tracking in wireless communication networks based on received signal strength.

- Joint detection and estimation of multipath effects on GPS measurements.
- Maritime surface and underwater map-aided navigation.
- Car collision avoidance system.

1.2 CONTRIBUTIONS

Particle filters perform three basic operations: generation of new particles (sampling from the space of unobserved states), computation of particle weights (probability masses associated with the particles) and resampling (a process of removing particles with small weights and replacing them with particles with large weights). These three steps make the basis of the most commonly used type of particle filters called Sample-Importance-Resampling Particle Filters (SIR PF). Particle generation and weight computation are computationally the most intensive steps.

Rigorous mathematics involved in the theory of Particle filters always required some kind of experimental results for proper understanding. In this dissertation, MATLAB simulations have been done to understand different aspects of Particle filter algorithms. This dissertation presents the integration of color distributions into Particle filtering for visual object tracking, which has typically been used in combination with edge-based image features. MATLAB simulations have also been done to understand the mathematical aspects of tracking patterns of an object of interest in the video in the presence of clutter.

1.3 ORGANIZATION OF DISSERTATION

The reset of this dissertation is organized as follows.

Chapter 2 describes the Mathematical foundation of particle filtering, Recursions for obtaining the filtering PDF, The Basic idea with examples, Recursive computation of the random measure, The sequential importance sampling algorithm with examples, The choice of proposal distribution and resampling. MATLAB simulation results for the Particles and their weights with sampling and resampling have been discussed.

Chapter 3 describes the derivation of some Particle filtering methods such as, Sequential Importance Resampling (SIR) Particle filering, Auxiliary Particle filtering

(APF), Gaussian Particle filtering (GPF) along with their algorithms. MATLAB simulation results have also been presented to understand mathematical aspects of different parameters. Apart from the above, derivation of kernel and density assisted Auxiliary Particle filters have also been presented to understand the handling of constant parameters.

Chapter 4 describes the Rao-Blackwellized Particle filter (RBPF) with its complete derivation and algorithm. MATLAB simulation results have been presented to understand the comparison between the RBPF and PF in terms of Mean Square Error (MSE). MATLAB simulation results have also been presented for the tracking of maneuvering objects having piecewise linear dynamics.

Chapter 5 describes the Particle filter algorithms for Visual object tracking based on color and intensity histograms. MATLAB simulation have been presented to understand the Belief states over the frames with handling of visual clutter along with the visual object trajectory, HSV cdf, and 3D histogram of target distribution. Other simulation shows the prediction of object position, bounding ellipse particles, posterior distribution of the moving object over the frames and SIR resampling of the tracked object.

Chapter 6 concludes the dissertation with a summary of our contributions and enlists future directions arising from this research effort.

CHAPTER 2
THEORY OF PARTICLE FILTERING

2.1 MATHEMATICAL FOUNDATION OF PARTICLE FILTERING

In a typical setup, the observed data are modelled by

$$y(n) = g_2\big(x(n), v_2(n)\big) \qquad (2.1)$$

Where n=….,2,…..is a time index, y(n) is a vector of observations, x(n) is the state (or signal) that needs to be estimated, $v_2(n)$ is an observation noise vector, and $g_2(.)$ is a known function, which in general may change with time. We assume that all of the vectors in the above equation conform properly in their dimensions.

It is assumed that the state $x(n)$ varies according to

$$x(n) = g_1\big(x(n-1), v_1(n)\big) \qquad (2.2)$$

Where $v_1(n)$ is a state noise vector, and $g_1(.)$ is a known function (which also might vary with time). The expression (2.1) is known as observation equation and (2.2) as a state equation, and two are refereed as the state-space model. The objective is to estimate the unobserved signal $x(n)$ from the observations $y(n)$.

An alternative characterization of the state-space system can be given in terms of Probability density functions (PDFs)

$$f(x(n)|x(n-1)) \qquad (2.3)$$

$$f(y(n)|x(n) \qquad (2.4)$$

Where obviously (2.3) is derived from (2.2), and (2.4) is obtained from (2.1). The forms of the PDFs in (2.3) and (2.4) depend on the functions $g_1(\cdot)$ and $g_2(\cdot)$ as well as on the PDFs of $v_1(n)$ and $v_2(n)$.

Three main problems, namely, filtering, prediction and smoothing arise when we consider the problem of estimation. In filtering, the goal is to obtain the a posterior PDF of $x(n)$ given all of the measurements from time instant one to n, which we express by $y(1{:}n)$. This density is accordingly called filtering density and is denoted by $f\big(x(n)\big|y(1{:}n)\big), n \geq 1$. All the information about $x(n)$ is in $f\big(x(n)\big|y(1{:}n)\big)$, and for example, if one wants to find point estimate of $x(n)$, such as the minimum mean square error (MMSE) estimate or the maximum a posterior (MAP) estimate, one can obtain them from the filtering density.

In prediction, the goal is to find the predictive PDF $f\big(x(n+k)\big|y(1{:}n)\big)$, where $k > 0, n > 0$. Again, all the information about a future value of the state given the measurements $y(1{:}n)$ is in the predictive PDF and various point estimates can be obtained from it.

Finally, the problem of smoothing amounts to obtaining $f\big(x(n)\big|y(1{:}N)\big)$, where $0 \le n < N$, and where the density is called the smoothing PDF. In general, smoothing gives more accurate estimates of $x(n)$ providing delay in processing of the data. For example, we first acquire N measurements and then obtain the smoothing densities.

The objective of sequential signal processing consists of tracking the PDFs of interest by exploiting recursive relationships, that is,

- $f\big(x(n)\big|y(1{:}n)\big)$ from $f\big(x(n-1)\big|y(1{:}n-1)\big)$ for the filtering problem, where $n \ge 1$;
- $f\big(x(n+k)\big|y(1{:}n)\big)$ from $f\big(x(n+k-1)\big|y(1{:}n)\big)$ for the prediction problem, where $k > 0, n \ge 0$; and
- $f\big(x(n)\big|y(1{:}N)\big)$ from $f\big(x(n+1)\big|y(1{:}N)\big)$ for the smoothing problem, where $0 \le n < N$.

The complete information about the unknown values is in the respective densities of the unknowns. Many sequential methods provide only point estimates of these unknowns accompanied possibly with another metric that shows how variable the estimates are.

By contrast, particle filtering has a much more ambitious aim than yielding point estimates. Its objective is to track in time the approximations of all the desired densities of the unknowns in the system. It is well known that when these densities are not Gaussian, there are not many available methods that can reach this goal. Thus, the motivation for using particle filtering is in its ability to estimate sequentially the densities of unknowns of non-Gaussian and/ or nonlinear systems.

2.2 RECURSIONS FOR OBTAINING THE FILTERING PDF

Suppose that at time $(n-1)$, we know the observations $y(1{:}n-1)$ and the a posterior PDF $f\big(x(n-1)\big|y(1{:}n-1)\big)$. Once $y(n)$ becomes available, we would like to update $f\big(x(n-1)\big|y(1{:}n-1)\big)$ and modify it to $f\big(x(n)\big|y(1{:}n)\big)$. To achieve this, we formally write

$$f\big(x(n)\big|y(1{:}n)\big) \propto f\big(y(n)\big|x(n)\big)f\big(x(n)\big|y(1{:}n-1)\big) \qquad (2.5)$$

Where $\propto$ signifies proportionality. The first factor in the right of the proportionality sign is the likelihood function of the unknown state, and the second factor is the predictive density of the state.

For the predictive density, we have

$$f\big(x(n)\big|y(1{:}n-1)\big) = \int f\big(x(n)\big|x(n-1)\big)f\big(x(n-1)\big|y(1{:}n-1)\big)dx(n-1) \qquad (2.6)$$

In writing (2.6), we used the property of the state that given $x(n-1)$, $x(n)$ does not depend on $y(1{:}n-1)$. Now, the required recursive equation for the update of the filtering density is obtained readily by combining (2.5) and (2.6), that is, we formally have

$$f\big(x(n)\big|y(1{:}n)\big) \propto f\big(y(n)\big|x(n)\big) \times \int f\big(x(n)\big|x(n-1)\big)f\big(x(n-1)\big|y(1{:}n-1)\big)dx(n-1) \qquad (2.7)$$

Thus on the left of the proportionality sign, we have the filtering PDF at time instant n, and on the right under the integral, we see the filtering PDF at time instant $(n-1)$.

There are at least two problems in carrying out the above recursion, and they may make the recursive estimation of the filtering density very challenging. The first one is solving of the integral in (2.6) and obtaining the predictive density $f\big(x(n)\big|y(1{:}n-1)\big)$. The second problem is the combining of the likelihood and the predictive density in order to get the updated density.

These problems may mean that it is impossible to express the filtering PDF in a recursive form. We can state that in many problems the recursive evaluation of the densities of the state-space model cannot be done analytically, and consequently we have to resort to numerical methods. An important class of systems which allows for exact analytical recursions is the one represented by linear state-space models with Gaussian noises. These recursions are known as Kalman filtering. When analytical solutions cannot be obtained, particle filtering can be employed with elegance and with performance characterized by high accuracy.

2.3 THE BASIC IDEA

Consider the state of the system at time instant n, that is, $x(n)$. Under the particle filtering framework, the a posterior PDF of $x(n), f\big(x(n)\big|y(1{:}n)\big)$, is approximated by a discrete random measure composed of M particles and weights

$$\chi(n) = \left\{x^{(m)}(n),\ w^{(m)}(n)\right\}_{m=1}^{M} \tag{2.8}$$

where $x^{(m)}(n)$ and $w^{(m)}(n)$ represent the $m-th$ particle and weight, respectively. The particles are Monte Carlo samples of the system state, and the weights are nonnegative values that sum up to one and can be interpreted as probabilities of the particles. The previous measures allows for approximations of $f\big(x(n)\big|y(1{:}n)\big)$ by

$$f\big(x(n)\big|y(1{:}n)\big) \approx \textstyle\sum_{m=1}^{M} w^{m}(n)\delta\big(x(n) - x^{m}(n)\big) \tag{2.9}$$

Where $\delta(.)$ denotes the Dirac delta function. With these approximations, computations of expectations of functions of the random process $X(n)$ simplify to summation, that is,

$$E(h\big(X(n)\big)) = \int h\big(x(n)\big)f\big(x(n)\big|y(1{:}n)\big)dx(n) \tag{2.10}$$

$$\Downarrow$$

$$E(h\big(X(n)\big)) = \sum_{m=1}^{M} w^{m}(n)h\big(x^{m}(n)\big) \tag{2.11}$$

Where $E(\cdot)$ denotes expectations, and $h(\cdot)$ is an arbitrary function of $X(n)$.

Example 1:

Assume that independent particles, $x^{m}(n)$, can be drawn from $f\big(x(n)\big|y(1{:}n)\big)$. In that case, all the particles have the same weights, $w^{m}(n) = 1/M$, and

$$E(h\big(X(n)\big)) = \int h\big(x(n)\big)f\big(x(n)\big|y(1{:}n)\big)dx(n)$$

$$\Downarrow$$

$$\hat{E}(h\big(X(n)\big)) = \frac{1}{M}\sum_{m=1}^{M} h\big(x^{m}(n)\big) \tag{2.12}$$

Where $\hat{E}(\cdot)$ is an unbiased estimator of the conditional expectation $E(\cdot)$. Note that in this case we intrinsically approximate $f\big(x(n)\big|y(1{:}n)\big)$ by the random measure

χ(n) = $\left\{x^{(m)}(n),\ w^{(m)}(n) = 1/M\ \right\}_{m=1}^{M}$

If the variance $\sigma_h^2 < \infty$, then the variance of $\hat{E}(\cdot)$ is given by

$$\sigma^2_{\hat{E}(h(\cdot))} = \frac{\sigma_h^2}{M} \tag{2.13}$$

As $M \to \infty$, from the strong law of large numbers, we get that $\hat{E}(h\big(X(n)\big))$ converges to $E(h\big(X(n)\big))$ almost surely, that is

$$\hat{E}\left(h\big(X(n)\big)\right) \overset{a.s.}{\longrightarrow} E\left(h\big(X(n)\big)\right) \tag{2.14}$$

And from the central limit theorem, we obtain that $\hat{E}(h\big(X(n)\big))$ converges in distribution to a Gaussian distribution

$$\hat{E}\left(h\big(X(n)\big)\right) \overset{d}{\to} N(E\left(h\big(X(n)\big), \frac{\sigma_h^2}{M}\right) \tag{2.15}$$

The example shows that if we sample from $f\big(x(n)\big|y(1{:}n)\big)$ a large number of particles M ,we will be able to estimate $E(h\big(X(n)\big))$ with arbitrary accuracy. In practice, however, the problem is that we often cannot draw samples directly from the a posteriori PDF $f\big(x(n)\big|y(1{:}n)\big)$.

An attractive alternative is to use the concept of importance sampling. The idea behind it is based on the use of another function for drawing particles. This function is called importance sampling function or proposal distribution, and we denote it by $\pi(x(n))$.

When the particles are drawn from $\pi(x(n))$, the estimate of $E(h\big(X(n)\big))$ in (2.12) can be obtained either by

$$E(h\big(X(n)\big)) \approx \frac{1}{M}\sum_{m=1}^{M} w^{*m}(n)h\big(x^m(n)\big) \tag{2.16}$$

Or by

$$E(h\big(X(n)\big)) \approx \frac{1}{M}\sum_{m=1}^{M} w^{m}(n)h\big(x^m(n)\big) \tag{2.17}$$

Where

$$w^{*(m)}(n) = \frac{f\left(x^{(m)}(n)\middle|y(1:n)\right)}{\pi(x^{(m)}(n))} \tag{2.18}$$

And

$$w^{(m)}(n) = \frac{\widetilde{w}^{(m)}(n)}{\sum_{i=1}^{M} \widetilde{w}^{(i)}(n)} \tag{2.19}$$

Where $\widetilde{w}^{(m)}(n) = c\widetilde{w}^{*(m)}(n)$ with c being some unknown constant. The symbols $w^{*(m)}(n)$ and $w^{(m)}(n)$ are known as true and normalized importance weights of the particles $x^{m}(n)$, respectively. They are introduced to correct for the bias that arises due to sampling from a different function than the one that is being approximated, $f\big(x(n)\big|y(1:n)\big)$. The estimate in (2.16) is unbiased whereas the one from (2.17) is with small bias but often with a smaller mean-squared error than the one in (2.16). An advantage in using (2.17) over (2.16) is that we only need to know the ratio $f\big(x(n)\big|y(1:n)\big)/\pi(x(n))$ up to a multiplicative constant and not the exact ratio in order to compute the estimate of the expectation of $h(X(n))$.

How is (2.19) obtained? Suppose that the true weight cannot be found and instead we can only compute it up to a proportionality constant, that is

$$\begin{aligned} \widetilde{w}^{(m)}(n) &= c\,\frac{f\left(x^{(m)}(n)\middle|y(1:n)\right)}{\pi(x^{(m)}(n))} \\ &= c\, w^{*(m)}(n) \end{aligned} \tag{2.20}$$

Where the constant c is unknown. Since we must have

$$\begin{aligned} 1 &= \int c\,\frac{f\big(x(n)\big|y(1:n)\big)}{\pi\big(x(n)\big)}\pi\big(x(n)\big)dx(n) \\ &\simeq \frac{1}{cM}\sum_{m=1}^{M} \widetilde{w}^{(m)}(n) \end{aligned}$$

From where we can estimate c by

$$c \simeq \frac{1}{M}\sum_{m=1}^{M} \widetilde{w}^{(m)}(n) \tag{2.21}$$

Now by using (2.20), we can express (2.16) in terms of $\widetilde{w}^{(m)}(n)$. We have

$$\begin{aligned} E\big(h\big(X(n)\big)\big) &\approx \frac{1}{M}\sum_{m=1}^{M} w^{*m}(n)h\big(x^{m}(n)\big) \\ &= \frac{1}{cM}\sum_{m=1}^{M} \widetilde{w}^{(m)}(n)\, h\big(x^{m}(n)\big) \end{aligned}$$

$$\approx \sum_{m=1}^{M} \frac{\widetilde{w}^{(m)}(n)}{\sum_{i=1}^{M} \widetilde{w}^{(i)}(n)} h\big(x^m(n)\big)$$

$$= \frac{1}{cM} \sum_{m=1}^{M} w^{(m)}(n)\, h\big(x^m(n)\big)$$

Where $w^{(m)}(n)$ is the normalized weight from (2.19).

In summary, when we use a random measure to approximate a PDF, such as $f\big(x(n)\big|y(1{:}n)\big)$, we can do it by

- Drawing samples from a proposal distribution $\pi(x^{(m)}(n))$, which needs to be known only up to a multiplicative constant, that is

$$x^m(n) \sim \pi\big(x(n)\big), \quad m = 1,2,\dots,M$$

- Computing the weights of the particles $w^{(m)}(n)$ by (2.18) and 2.(19).

The resulting random measure is of the form given by (2.10). As proposal distributions we choose ones that are easy to sample from and whose shape is close to the product of $h(x(n))$ and $f\big(x(n)\big|y(1{:}n)\big)$. A rule of thumb is that we would like to generate particles from regions of the support of $x(n)$ where that product has large values.

2.4. RECURSIVE COMPUTATION OF THE RANDOM MEASURE X(N)

The random measures χ(n) and χ(n-1) approximate $f\big(x(n)\big|y(1{:}n)\big)$ and $f\big(x(n-1)\big|y(1{:}n-1)\big)$, respectively. We write (9) as

$$f\big(x(n-1)\big|y(1{:}n-1)\big)$$
$$\approx \sum_{m=1}^{M} w^m(n-1)\delta\big(x(n-1) - x^m(n-1)\big)$$

For the filtering PDF $f\big(x(n)\big|y(1{:}n)\big)$, we have

$$f\big(x(n)\big|y(1{:}n)\big)$$
$$\propto f\big(y(n)\big|x(n)\big)$$
$$\times \int f\big(x(n)\big|x(n-1)\big) f\big(x(n-1)\big|y(1{:}n-1)\big) dx(n-1)$$

$$\simeq f\big(y(n)\big|x(n)\big) \sum_{m=1}^{M} w^m(n-1) f\big(x(n)\big|x^{(m)}(n-1)\big)$$

We want to represent the new filtering density with χ(n), and to that end we need to generate particles $x(n)$ and compute their weights. If for particle generation we use the proposal distribution $\pi\left(x(n)\middle|x^{(m)}(n-1), y(1{:}n)\right)$, the newly generated particles $x^{(m)}(n)$ is appended to the particle stream $x^{(m)}(1{:}n-1)$, and we compute the value of its weight according to

$$\tilde{w}^{(m)}(n) = w^{(m)}(n-1)\frac{f\left(y(n)\middle|x^{(m)}(n)\right) f\left(x^{(m)}(n)\middle|x^{(m)}(n-1)\right)}{\pi\left(x^{(m)}(n)\middle|x^{(m)}(n-1), y(1{:}n)\right)} \tag{2.22}$$

Where $\tilde{w}^{(m)}(n)$ is a non-normalized weight of $x^{(m)}(n)$. We see that the weight of the $m-th$ stream is obtained by updating its value at time instant $(n-1)$ with the factor

$$\frac{f\left(y(n)\middle|x^{(m)}(n)\right) f\left(x^{(m)}(n)\middle|x^{(m)}(n-1)\right)}{\pi\left(x^{(m)}(n)\middle|x^{(m)}(n-1), y(1{:}n)\right)}$$

The so obtained weights are then normalized so that they sum up to one.

In summary, the PF implements two steps. One is the generation of particles for the next time instant and the other is the computation of the weights of these particles. The table 2.1 summarizes the mathematical expressions needed for carrying out the recursive update from χ(n-1) to χ(n).

Table 2.1: The sequential importance sampling algorithm

Particle generation

Basis

$$\pi\big(\, x(0{:}n) \mid y(1{:}n)\,\big) = \pi\big(\, x(n) \mid x(0{:}n-1), y(1{:}n)\,\big)\pi\big(\, x(0{:}n-1) \mid y(1{:}n-1)\,\big)$$

$$x^{(m)}(0{:}n-1) \sim \pi\big(\, x(0{:}n-1) \mid y(1{:}n-1)\,\big)$$

$$w^{(m)}(n-1) \propto \frac{f\left(x^{(m)}(0{:}n-1)\middle|y(n-1)\right)}{\pi\left(x^{(m)}(0{:}n-1)\middle|y(1{:}n-1)\right)}$$

Augmentation of the trajectory $x^{(m)}(0{:}n-1)$ to $x^{(m)}(n)$

$$x^{(m)}(n) \sim \pi\left(\, x(n) \,\middle|\, x^{(m)}(0{:}n-1), y(1{:}n)\right), m = 1, \dots, M$$

Weight update

$$w^{(m)}(n) \propto w^{(m)}(n-1)\frac{f\left(y(n)\middle|x^{(m)}(n)\right) f\left(x^{(m)}(n)\middle|x^{(m)}(n-1)\right)}{\pi\left(x^{(m)}(n)\middle|x^{(m)}(0{:}n-1), y(1{:}n)\right)},$$

$$m = 1, \dots, M$$

Example 2: Consider the state-space model given by

$$x(n) = x(n-1) + v_1(n)$$

$$y(n) = x(n) + v_2(n)$$

Where $v_1(n)$ and $v_2(n)$ are independent standard Gaussian random variables. Realize the system along with the values of the first four time instants.

Solution: In this example, we choose to use the proposal distribution $\pi\big(x(n)\big) = f\big(x(n)\big|x(n-1)\big)$. In other words, we generate the particles according to

$$x^{(m)}(n) \sim f\left(x(n) \mid x^{(m)}(n-1)\right), \qquad m = 1, \dots, M$$

That is, each particle stream has a separate proposal distribution. Since the proposal function is identical to the transition function $f\left(x(n) \mid x^{(m)}(n-1)\right)$, from (2.22) we deduce that the update of the weights is according to

$$w^{(m)}(n) \sim f\left(y(n) \mid x^{(m)}(n)\right) w^{(m)}(n-1)$$

The step by step execution of the sequential importance sampling algorithm for the three time instants proceeds as follows.

Initialization $n = 1$

$$x^{(m)}(0) \sim N(0,1), m = 1,2, \dots M$$

Note that all the weights are equal as shown in Fig. 1 of Figure 2.1.

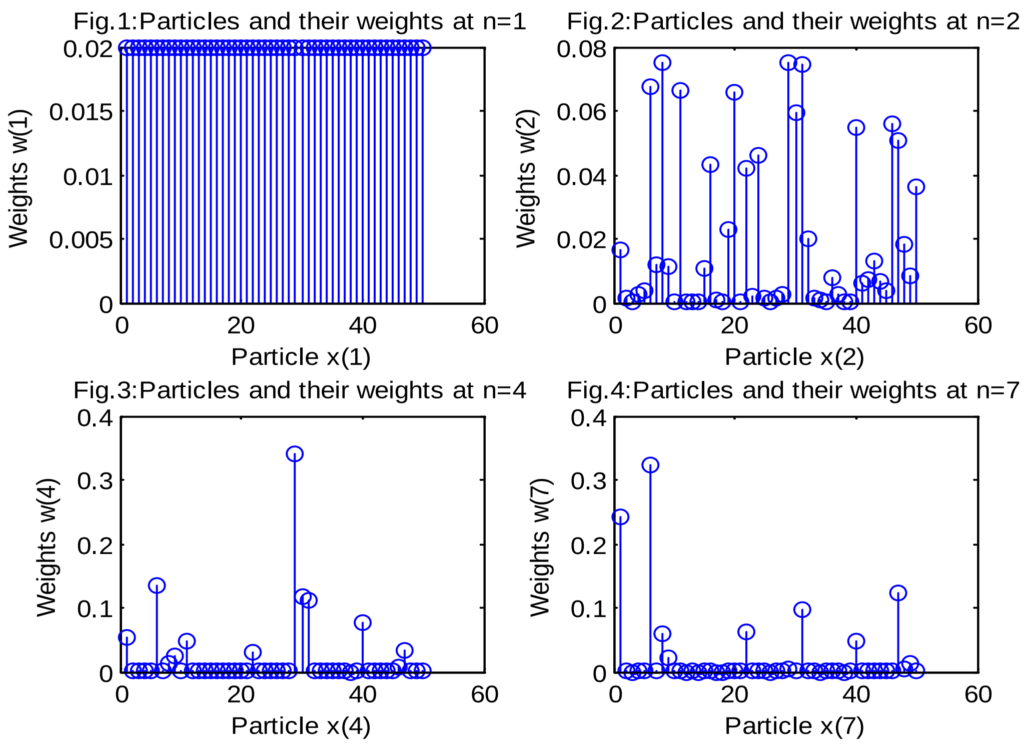


Figure 2.1: Particle and weights for Sequential Importance Sampling

Time instant $n = 2$, Generation of particles using the proposal distribution

$$x^{(m)}(2) \sim f\left(x(2) \,\middle|\, x^{(m)}(1)\right)$$

$$= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\left(x(2) - x^{(m)}(1)\right)^2}{2}\right)$$

Weight update

$$w^{(m)}(2) \propto w^{(m)}(1) \exp\left(-\frac{\left(y(2) - x^{(m)}(2)\right)^2}{2}\right)$$

The particles and their weights are shown in Fig. 2 of Figure 2.1.

Time instant $n = 4$, Generation of particles using the proposal distribution

$$x^{(m)}(4) \sim f\left(x(4) \,\middle|\, x^{(m)}(3)\right)$$

$$= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\left(x(4) - x^{(m)}(3)\right)^2}{2}\right)$$

Weight update

$$w^{(m)}(4) \propto w^{(m)}(3) \exp\left(-\frac{\left(y(4) - x^{(m)}(4)\right)^2}{2}\right)$$

The particles and their weights are shown in Fig. 3 of Figure 2.1.

Time instant $n = 7$, Generation of particles using the proposal distribution

$$x^{(m)}(7) \sim f\left(x(7) \,\middle|\, x^{(m)}(6)\right)$$

$$= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\left(x(7) - x^{(m)}(6)\right)^2}{2}\right)$$

Weight update

$$w^{(m)}(7) \propto w^{(m)}(6) \exp\left(-\frac{\left(y(7) - x^{(m)}(7)\right)^2}{2}\right)$$

The particles and their weights are shown in Fig. 4 of Figure 2.1.

The process continues in the same way as new observations become available. Note that the variance of the weights increases with time, which is an unwanted effect. Even at time instant $n = 7$, a few particle have large weights and the rest have negligible weights.

2.5 THE CHOICE OF PROPOSAL DISTRIBUTION AND RESAMPLING

In this section two important issues that affect the performance and implementation of particle filtering algorithms are discussed. One is the choice of proposal distributions and the other the concept of resampling, which turns out to be indispensable in the implementation of PFs.

2.5.1 The Choice of Proposal Distribution

The proposal distribution (importance function) plays a crucial role in the performance of particle filtering. From a practical and intuitive point of view, it is desirable to use easy-to-sample proposal distributions that produce particles with a large enough variance in order to avoid exploration of the state space in too narrow regions and thereby contributing to losing the tracks of the state, but not too large to alleviate generation of too dispersed particles. The support of of the proposal distributions has to be the same as that of the targeted distribution.

As already pointed out, the approximation of the posterior distribution with the random measure obtained by the proposal distribution will improve if the proposal becomes very similar to the posterior. In fact, the optimal choice for the importance function is the posterior distribution, that is

$$\pi\big(\, x(n) \mid x^{(m)}(0{:}\, n-1), y(1{:}\, n) \,\big) = f\big(\, x(n) \mid x^{(m)}(n-1), y(n) \,\big) \qquad (23)$$

Which corresponds to the following weight calculation

$$w^{(m)}(n) \propto w^{(m)}(n-1)\, f\left(\, y(n) \;\middle|\; x^{(m)}(n-1) \right)$$

This importance function minimizes the variance of the weights, $w^{(m)}(n)$, conditional on $x^{(m)}(0{:}\, n-1)$ and $y(1{:}\, n)$. However, the computation of the weights requires the integration of $x(n)$, that is, solving

$$f\left(y(n)\middle|x^{(m)}(n-1)\right) = \int f\big(y(n)\big|x(n)\big) f\left(x(n)\middle|x^{(m)}(n-1)\right) dx(n)$$

Thus, the implementation of the optimal importance function may be difficult for two reasons: First, direct sampling from the posterior (2.23) may not be easy, and second, the contribution of the weights may require integration.

Example 3: Consider the following decomposition of the posterior in (2.23)

$$f\big(\, x(n) \mid x^{(m)}(n-1), y(n) \,\big) \propto f\big(\, y(n) \mid x(n) \,\big) f\left(x(n)\middle|x^{(m)}(n-1)\right)$$

If the distributions on the right-hand side of the previous expressions are Gaussins, their product will also be Gaussian. Thus, the proposal is a Guassian and sampling from it can readily be performed. It is worth pointing out that if the noises in the system are additive and Gaussian and the observation is a linear function of the state, we can sample from the optimal proposal distribution even if the function in the state equation is nonlinear.

A popular choice for the importance function is the prior $\pi\big(x(n) \mid x^{(m)}(0{:}n-1), y(1{:}n) \big) = f\left(x(n) \,\middle|\, x^{(m)}(n-1) \right)$

Which yields importance weight proportional to the likelihood

$$w^{(m)}(n) \propto w^{(m)}(n-1)\, f\left(y(n) \,\middle|\, x^{(m)}(n) \right)$$

The main advantage of this choice is the ease in the computation of the weights, which amounts to obtaining the likelihood function. However, the generation of the particles is implemented without the use of observations, and therefore not all of the available information is used to explore the state space. This may lead in some practical cases to poor estimation results. Strategies to improve this performance consists of the inclusion of a prediction step like that of the auxiliary PF or the use of a hybrid importance function if part of the state is proposed from a prior and the remaining state from the optimal importance function.

Example 4: Consider a state space whose parameters can be divided in two groups $x(n) = \{x_1(n) x_1(n)\}$ and where sampling can be carried out from $f\left(x_2(n) \,\middle|\, {x_1}^{(m)}(n-1), {x_2}^{(m)}(n-1) \right)$ and $f\big(x_1(n) \mid {x_2}^{(m)}(n), {x_2}^{(m)}(n-1), {x_1}^{(m)}(n-1), y(n) \big)$. A hybrid proposal that combines the prior and the posterior importance functions is given by

$$\pi\big(x(n) \mid x^{(m)}(n-1), y(n) \big)$$
$$= f\left(x_2(n) \,\middle|\, {x_1}^{(m)}(n-1), {x_2}^{(m)}(n-1) \right) f\big(x_1(n) \mid {x_2}^{(m)}(n), {x_2}^{(m)}(n-1), {x_1}^{(m)}(n-1), y(n) \big)$$

Where ${x_2}^{(m)}(n)$ is a sample from $f\left(x_2(n) \,\middle|\, {x_1}^{(m)}(n-1), {x_2}^{(m)}(n-1) \right)$. The update of the weight can readily be obtained from the general expression given by (2.22).

2.6 RESAMPLING

In particle filtering, the discrete random measure degenerates quickly and only few particles are assigned meaningful weights. This degradation leads to a deteriorated functioning of particle filtering. Initially, all the particles have the same weights, and at each time step the particles are propagated and assigned weights. As time evolves, all the particles except for very few are assigned negligible weights. In the last step, there is only one particle with significant weight.

The mechanism of resampling as well as the use of good importance function can reduce this degeneracy. A measure of this degeneracy is the effective particle size defined by

$$M_{eff} = \frac{M}{1 + Var(w^{*(m)}(n))}$$

Where $w^{*(m)}(n) = \frac{f(x(n))}{\pi(x(n))}$ is the true particle weight. This metric can be estimated as

$$\widehat{M}_{eff} = \frac{1}{\sum_{m=1}^{M}(w^{(m)}(n))^2}$$

With $w^{(m)}(n)$ being the normalized weight corresponding to the $m-th$ particle at time instant n. If the effective particle size is below a predefined threshold, resampling is carried out. Clearly, when all the particles have the same weights, the variance of the weights is zero and the particle size is equal to the number of particles, M. The other extreme occurs when all the particles except one have negligible weights, and the particle size is equal to one.

Resampling eliminates particles with small weights and replicates particles with large weights. In general, if the random measure at time instant n is χ(n), it proceeds as follows.

- Draw M particles, $x^{(k_m)}(n)$, from the distribution given by χ(n), where the $k_m s$ are the indexes of the drawn particles.
- Let $x^{(m)}(n) = x^{(k_m)}(n)$, and assign equal weights $\frac{1}{M}$ to the particles.

At time instant n, a new set of particles is generated, and their weights are computed. Thereby the random measure χ(n) is generated and can be used for the estimation of the desired unknowns. Before the next step $(n+1)$, the effective particle size is estimated and resampling is carried out if necessary. Some of the resampled particles at n=1, n=2, n=4, and n=7 are shown in Fig. 5, 6, 7, and 8, of Figure 2.2, respectively.

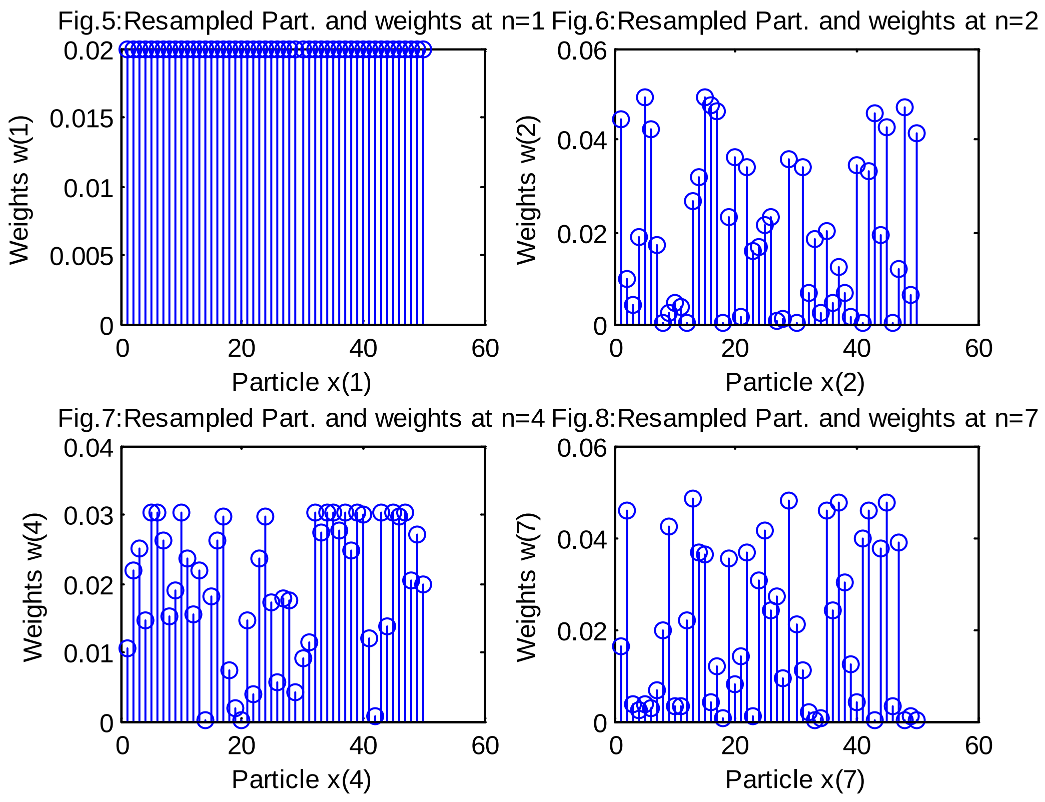


Figure 2.2: Particle and weights for Sequential Importance Resampling

CHAPTER 3
BASIC PARTICLE FILTERING METHODS

In this section, we present three different particle filtering methods. They are also known as sampling-importance-resampling (SIR), auxiliary particle filtering (APF), and Gaussian particle filtering (GPF). The common feature of each of these methods is that at time instant n, they are represented by a discrete random measure given by χ(n) = $\{x^{(m)}(n),\ w^{(m)}(n)\}_{m=1}^{M}$, where, as before, $x^{(m)}(n)$ is the $m-th$ particle of the state vector at time instant n, $w^{(m)}(n)$ is the weight of that particle, and M is the number of particles. For each of these filters, we show how this random measure is obtained from χ(n-1) by using the observation vector $y(n)$.

3.1 SIR PARTICLE FILTERING

The SIR method is the simplest of all of the particle filtering methods. It was proposed in [1] and was named bootstrap filter. Namely, the SIR method employs the prior density for drawing particles, which implies that the weights are only proportional to the likelihood of the drawn particles. We now explain in more detail.

Recall that if the particles are generated from a density function $\pi(x(n))$, and the weights of the particles in the previous time step were $w^{(m)}(n-1)$, upon the reception of the measurement $y(n)$, the weights are updated by

$$\widetilde{w}^{(m)}(n) = w^{(m)}(n-1)\,\frac{f\left(y(n)\middle|x^{(m)}(n)\right) f\left(x^{(m)}(n)\middle|x^{(m)}(n-1)\right)}{\pi\left(x^{(m)}(n)\middle|x^{(m)}(n-1), y(1{:}n)\right)}$$

Where $\widetilde{w}^{(m)}(n)$ denotes a non-normalized weight of $x^{(m)}(n)$. If the proposal distribution is equal to the prior, that is

$$\pi\left(\, x(n) \mid x^{(m)}(n-1), y(1{:}n)\,\right) = f\left(\, x(n) \,\middle|\, x^{(m)}(n-1)\,\right)$$

The computation of the weights simplifies to

$$\widetilde{w}^{(m)}(n) = w^{(m)}(n-1)\, f\left(\, y(n) \,\middle|\, x^{(m)}(n)\,\right)$$

Furthermore, if the weights from the previous time step were all equal (because of resampling), the previous update equation becomes even simpler

$$\widetilde{w}^{(m)}(n) = \ f\left(\, y(n) \,\middle|\, x^{(m)}(n)\,\right)$$

That is, the weight of the particle $x^{(m)}(n)$ is only proportional to the likelihood of that particle.

Next, we explain in more detail the steps of the SIR method. In the first step we draw candidate particle $x_c^{(m)}(n)$ according to

$$x_c^{(m)}(n) \sim f\left(x(n) \,\middle|\, x^{(m)}(n-1)\right), \qquad m = 1,2,\ldots,M$$

In the second step, we compute the non-normalized weights of these particles by

$$\widetilde{w}^{(m)}(n) = f\left(y(n) \,\middle|\, x_c^{(m)}(n)\right), \qquad m = 1,2,\ldots,M$$

And then normalize them so that they sum up to one, that is, we use

$$w^{(m)}(n) = \frac{\widetilde{w}^{(m)}(n)}{\sum_{i=1}^{M} \widetilde{w}^{(j)}(n)}$$

With the completion of this step, we have the random measure $\left\{x_c^{(m)}(n),\ w^{(m)}(n)\right\}_{m=1}^{M}$, which should be used for computations of desired estimates.

In the third and final step we perform resampling of the particles $x_c^{(m)}(n)$ according to the multinomial probability mass function defined by the weights $w^{(m)}(n)$. Namely, we draw indices k_m, for $m = 1,2,\ldots,M$, where $\Pr(k_m = i) = w^{(i)}(n)$. Once the indexes are drawn, we set

$$x^{(m)}(n) = x_c^{(k_m)}(n)$$

The weights of the particles $x^{(m)}(n)$ are set to $w^{(m)}(n) = 1/M$. The whole SIR procedure is summarized in Table 3.1.

If resampling was not implemented as a third step, we have

$$x^{(m)}(n) = x_c^{(m)}(n)$$

$$\widetilde{w}^{(m)}(n) = w^{(m)}(n-1) f\left(y(n) \,\middle|\, x_c^{(m)}(n)\right), \qquad m = 1,2,\ldots,M$$

And then we normalize the weights. After normalization, we would typically perform a test to decide if resampling is needed. If it was necessary, we would implement it as described; if not, we would proceed with the next time step. Recall that with resampling some particles are removed and the ones that are preserved may be replicated. We reiterate that, in general, resampling does not have to be implemented at every time step, but here we adopt to do so. If resampling is not performed at the end of the recursion, the generated particles in the first step represent the support of the random measure used for the next time instant. A stochastic volatility model showing a realization of the observations, the state and its estimate is shown in Fig.9 of Figure 3.1 . In Fig.10 of Figure 3.2, comparison between state, resampled estimate state and unsampled estimate state is shown. Particle generation and its propagation without sampling (degeneracy effect) and with resampling is also shown in Fig.11 and 12 of Figure 3.3, respectively.

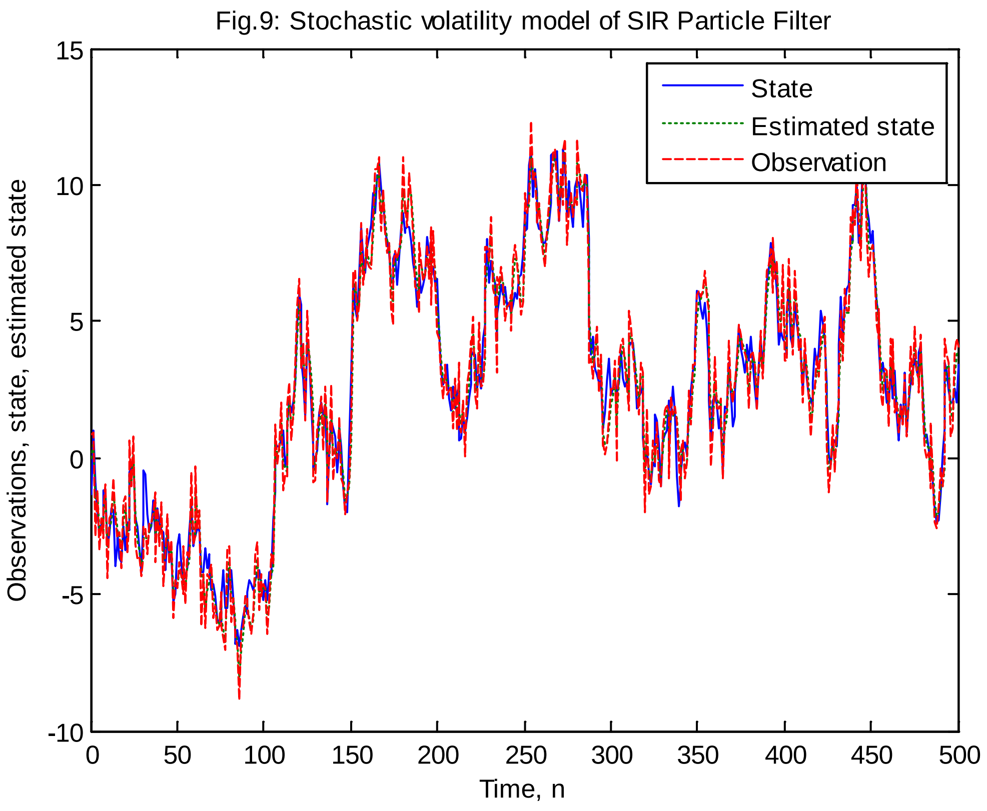


Figure 3.1 : Stochastic Volatility model of SIR Particle filter

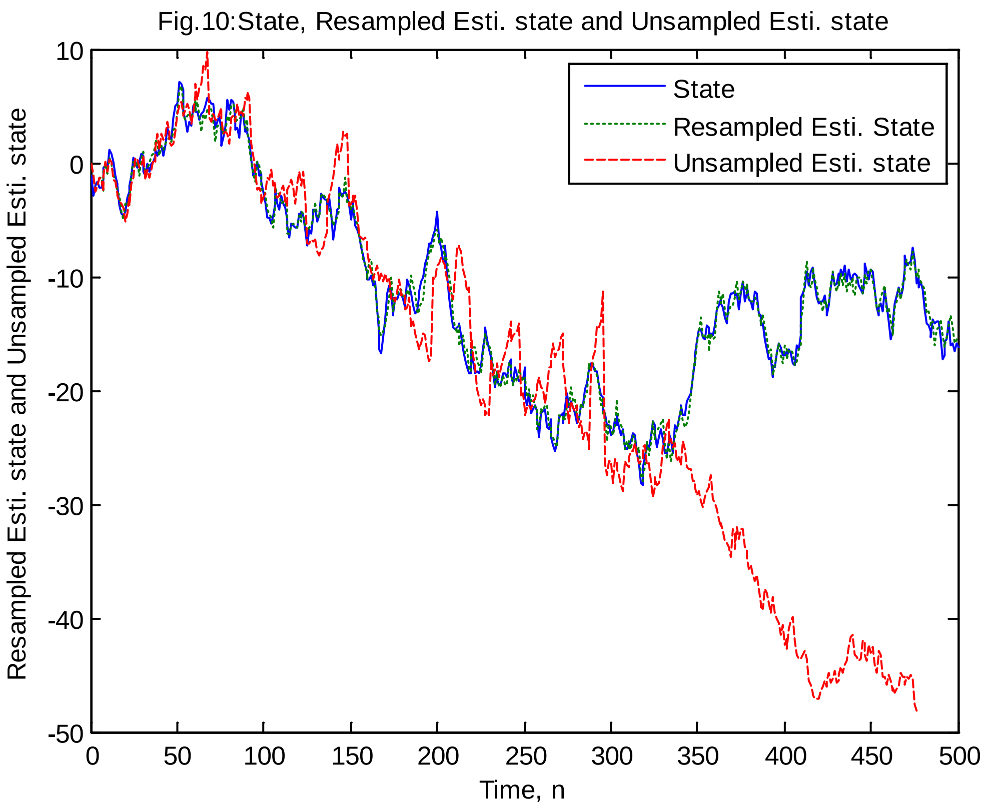


Figure 3.2 : State, Resampled Estimated state and Unsampled Estimated state

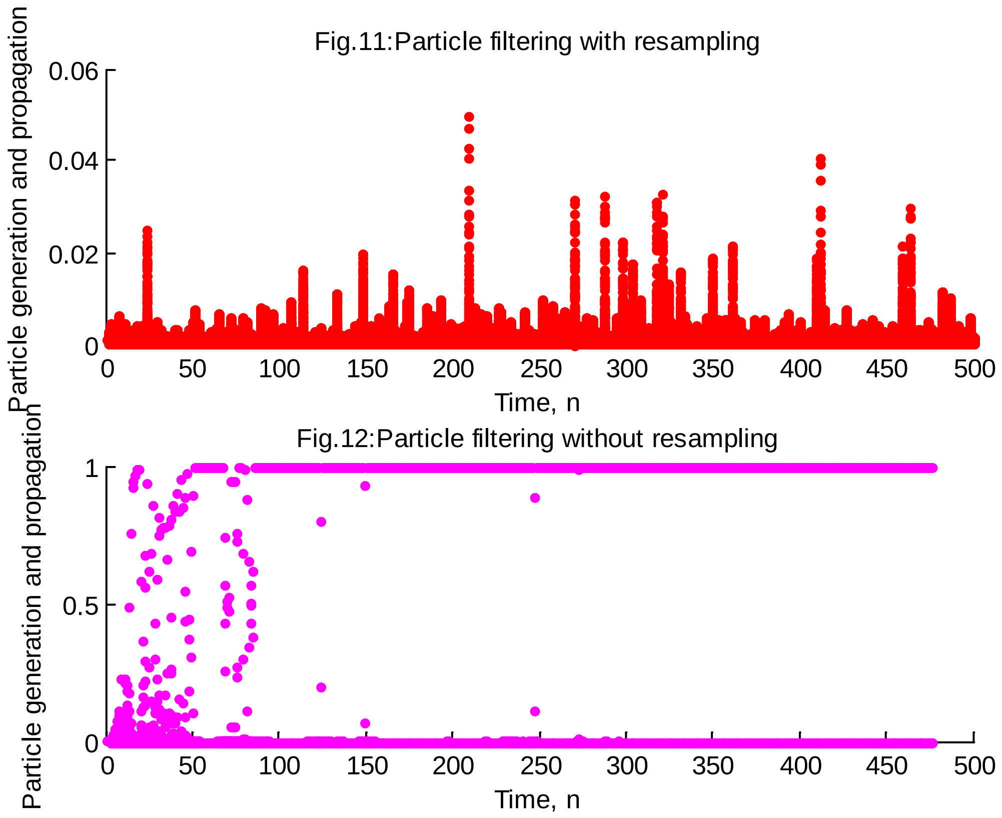


Figure 3.3 : Particle filtering with and without resampling

A clear advantage of the SIR method is that it is very simple for implementation. Its disadvantage is that in drawing the particles $x^{(m)}(n)$ for exploring the state space, we do not use the observation $y(n)$. In other words, once the particles are generated, the only thing we can do to steer the particles towards the region of the state space with large probability density is by resampling. If all of the generated particles are already far away from such regions, resampling will not help.

Table 3.1: The SIR algorithm

```
Initialization
For $m = 1,2,\dots,M$
    sample $x^{(m)}(0) \sim f(x(0))$
    $w^{(m)}(0) = \frac{1}{M}$
Recursions
For $n = 1,2,\dots,N$
  For $m = 1,2,\dots,M$
Proposal of candidate particles
 Sample $x_c^{(m)}(n) \sim f\left(x(n) \,\middle|\, x^{(m)}(n-1)\right)$
Computation of weights
      Evaluate the weights, $\tilde{w}^{(m)}(n) = f\left(y(n) \,\middle|\, x_c^{(m)}(n)\right)$
      Normalize the weights, $w^{(m)}(n) = \frac{\tilde{w}^{(m)}(n)}{\sum_{j=1}^{M} \tilde{w}^{(j)}(n)}$
  Resampling
      sample $k_m$, where $\Pr(k_m = i) = w^{(i)}(n)$
      set $x^{(m)}(n) = x_c^{(m)}(n)$ and
      $w^{(m)}(n) = \frac{1}{M}$
```

When SIR particle filter uses prior and posterior functions as proposal functions, then their performance can be compared in terms of mean square error (MSE) per sample. MSE comparison for prior and posterior proposal function is shown in Fig.13 of Figure 3.4.

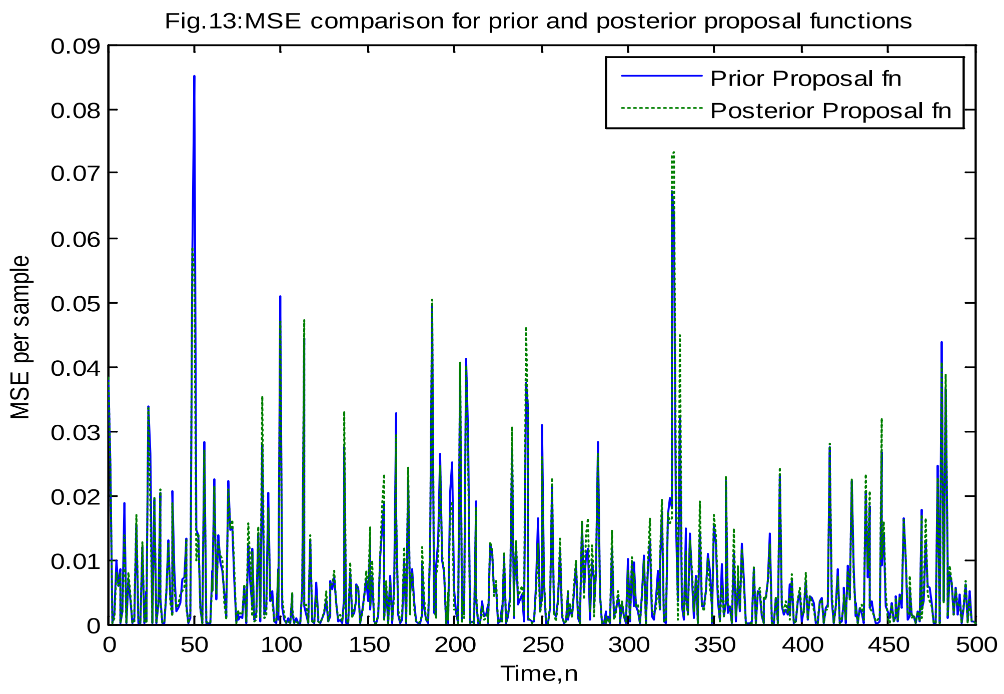


Figure 3.4 : MSE Comparison for prior and posterior proposal functions

The SIR particle filter has also been analyzed with two possible resampling schemes, one that performs the resampling at each time step, and another one where the resampling is carried out depending on the effective particle size measure. Comaparison of performance of the SIR particle filter for 1000 particles and threshold of effective particle size (0.3*1000) is shown in the Fig.14 of Figure 3.5.

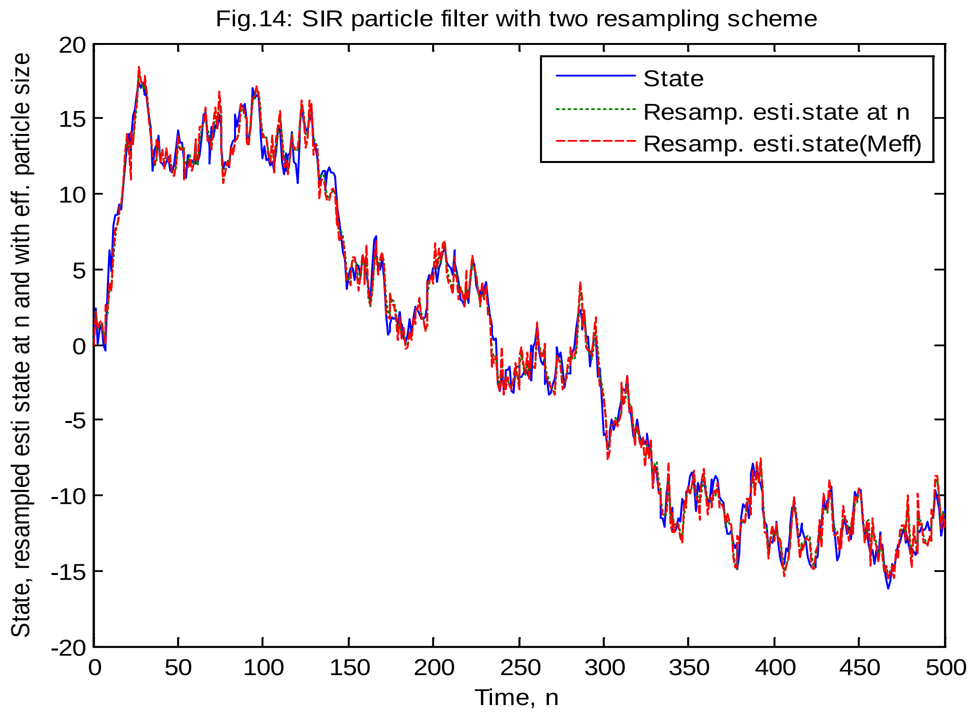


Figure 3.5 : SIR Particle filtering with two resampling scheme

Example 5: Consider a model of for univariate nonstationary growth [10]

$$x(n) = \alpha x(n-1) + \beta \frac{x(n-1)}{1 + x^2(n-1)} + \gamma\, cos\big(1.2(n-1)\big) + v_1(n)$$

$$y(n) = \frac{x^2(n)}{20} + v_2(n), n = 1, \dots, N$$

Where

$x(0) = 0.1, \alpha = 0.5, \beta = 25, \gamma = 8, N = 500, v_1(n) \sim N(0,1)\ and\ v_2(n) \sim N(0,1)$

Taking into account all the parameters, performance comparisons of SIR PF, APF, and GPF in terms of state and estimated state are shown in the Fig.15 (Figure 3.6), 16(Figure 3.7), and 17(Figure 3.8), respectively.

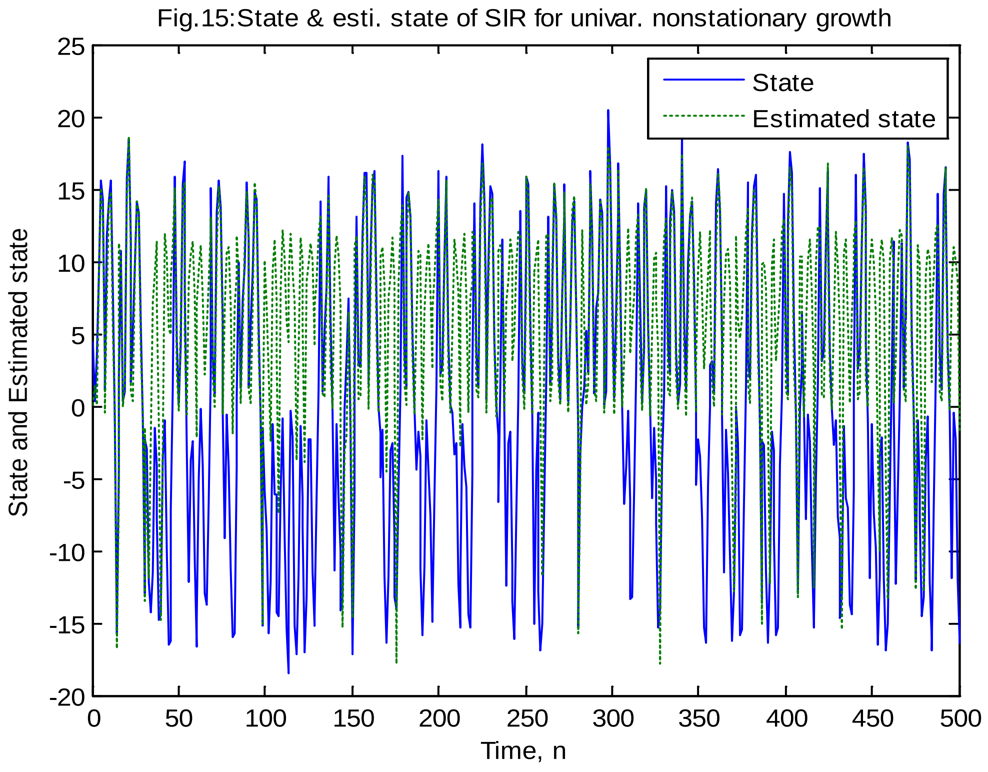


Figure 3.6 : State and estimated state of SIR Particle filter for univariate nonstationary growth

3.2 AUXILIARY PARTICLE FILTERING

The APF [10] attempts to improve the ability of the PF in exploring the state space by using the latest measurements. We know that with particle filtering, we approximate the filtering density $f\big(x(n)\big|y(1{:}n)\big)$ by the mixture density

$$f\big(y(n)\big|x(n)\big) \sum_{m=1}^{M} w^m(n-1) f\big(x(n)\big|x^{(m)}(n-1)\big) \qquad (3.1)$$

The underlying idea behind APF is to propose samples $x^{(m)}(n)$ from this density. In order to do so, we introduce an auxiliary variable, which is an index variable. We denote it by k and we index it by m, so that we write it as k_m. We draw it from the set $\{1,2,\ldots,M\}$, and it denotes the particle stream which we want to update. Thus, if we draw $k_m = 5$, we work with 5th stream, if we have $k_m = 11$, it is the 11th stream and so on.

First we describe the basic APF method. This method makes easy the problem of drawing $x(n)$ from (3.1) by using estimates of $x(n)$ for each stream of particles. If we denote the estimates by $\hat{x}^{(m)}(n)$, we modify (3.1) and create a proposal distribution given by

$$\sum_{m=1}^{M} w^{(m)}(n-1)f\big(y(n)\big|\hat{x}^{(m)}(n)\big)f\big(x(n)\big|x^{(m)}(n-1)\big) \qquad (3.2)$$

An estimate of $x^{(m)}(n)$ can be any value of $x(n)$ that is a good representative, which means that it should be a value that can easily be computed and has high likelihood.

For example, if the state equation is $x(n) = g_1\big(x(n)\big) + v_1(n)$ and the noise vector $v_1(n)$ is zero mean, an estimate of $x^{(m)}(n)$ could be $\hat{x}^{(m)}(n) = g_1\left(\hat{x}^{(m)}(n-1)\right)$.

With the estimates $\hat{x}^{(m)}(n)$ and new form of the proposal distribution (3.2), it is much easier to propose new particles $x^{(m)}(n)$. The idea has subtle point: we use (3.2) as a joint distribution of the auxiliary variable and the state. The implications is that first we draw the auxiliary variable (index) k_m from a multinomial distribution, where $\Pr(k_m = i) \propto w^{(i)}(n-1)f\big(y(n)\big|\hat{x}^{(i)}(n)\big)$.

The drawn index, say, i, identifies the distribution from which we draw $x^{(m)}(n)$, $f\big(x(n)\big|x^{(k_m=i)}(n-1)\big)$, so we proceed by drawing a particle from $f\big(x(n)\big|x^{(k_m=i)}(n-1)\big)$. Once the particles are drawn, as with SIR, the last step is the computation of the weights.

Before we derive the formula for the update of the weights, we express the proposal distribution in a form that will make the derivation easy. First, we rewrite the proposal as

$$\pi(\,x(n), k_m \mid y(1{:}n)\,) = f(\,x(n) \mid k_m, y(1{:}n)\,)f(\,k_m \mid y(1{:}n)\,)$$

The first factor on the right is

$$f(\,x(n) \mid k_m, y(1{:}n)\,) = f\big(\,x(n) \mid x^{(k_m)}(n-1)\,\big)$$

Because according to (3.2), given k_m, $y(n)$ does not affect the density of $x(n)$ and furthermore, given k_m, the density of $x(n)$ is not a function of $y(1{:}n-1)$ either, and instead it is simply the prior. This follows from the markovian nature of the state variable. Recall that k_m is an index that points to the $k_m - th$ stream and all its particles $x^{(k_m)}(0{:}n-1)$. Thus, once, k_m is known, so is $x^{(k_m)}(n-1)$, implying that all of the measurements $y(1{:}n-1)$ become irrelevant in expressing the density of $x(n)$.

For the second factor we can write

$$f(\, k_m \mid y(1{:}n)\,) \propto w^{k_m}(n-1) f\big(y(n)\big|\hat{x}^{(k_m)}(n)\big)$$

Where the weight $w^{k_m}(n-1)$ is a function of $y(1{:}n-1)$ and, clearly, $f\big(y(n)\big|\hat{x}^{(k_m)}(n)\big)$ is a function of $y(n)$. It is obvious that $f(\, k_m \mid y(1{:}n)\,)$ represents the probability of drawing k_m.

Therefore, for the update of the weight, we have the following

$$\widetilde{w}^{(m)}(n) = w^{(k_m)}(n-1)\frac{f\left(y(n)\middle|x^{(m)}(n)\right) f\left(x^{(m)}(n)\middle|x^{(k_m)}(n-1)\right)}{\pi\big(x^{(m)}(n), k_m\big|y(1{:}n)\big)}$$

$$= w^{(k_m)}(n-1)\frac{f\big(y(n)\big|x^{(m)}(n)\big) f\big(x^{(m)}(n)\big|x^{(k_m)}(n-1)\big)}{w^{(k_m)}(n-1) f\big(y(n)|\hat{x}^{(k_m)}(n)\big) f\big(x^{(m)}(n)\big|x^{(k_m)}(n-1)\big)}$$

$$= \frac{f\big(y(n)\big|x^{(m)}(n)\big)}{f\big(y(n)|\hat{x}^{(k_m)}(n)\big)}$$

In summary, the procedure is rather straightforward. Given the particles and their weights at time instant $(n-1)$, $x^{(m)}(n-1)$ and $w^{(m)}(n-1)$, respectively, first we compute estimates $\hat{x}^{(m)}(n)$. For these estimates we evaluate the weights by

$$\widehat{w}^{(m)}(n) \propto w^{(m)}(n-1) f\big(y(n)\big|\hat{x}^{(m)}(n)\big) \qquad (3.3)$$

Next we draw the indexes of the particles streams that we will continue to append. The indexes are drawn from the multinomial distribution with parameters $\widehat{w}^{(m)}(n)$. With this, we effectively perform resampling. After this step, we draw the particles of $x(n)$ according to

$$x^{(m)}(n) \sim f\left(x(n)\middle|x^{(k_m)}(n-1)\right), \quad m = 1,2,\ldots,M$$

The last step amounts to computing the weights of these particles by

$$w^{(m)}(n) \propto \frac{f\left(y(n)\middle|x^{(m)}(n)\right)}{f\big(y(n)|\hat{x}^{(k_m)}(n)\big)} \qquad (3.4)$$

The algorithm is summarized in Table 3.2.

Table 3.2: Auxiliary Particle Filter

```
Initialization
For $m = 1,2,\dots,M$
    sample $x^{(m)}(0) \sim f(x(0))$
    $w^{(m)}(0) = \frac{1}{M}$
Recursions
For $n = 1,2,\dots,N$
  For $m = 1,2,\dots,M$
  Estimation of next particles
     Compute $\hat{x}^{(m)}(n) = E\left( x(n) \mid x^{(m)}(n-1) \right)$
      Sample the indexes $k_m$ of the streams that survive
        Sample $k_m = i$ with probability $w^{(i)}(n-1) f\left(y(n) \middle| \hat{x}^{(i)}(n)\right)$
                    Sample the new particles for time instant, $n$
                      $x^{(m)}(n) \sim f\left(x(n) \middle| x^{(k_m)}(n-1)\right)$
                    Computation of weights
                      Evaluate the weights, $\tilde{w}^{(m)}(n) = \frac{f\left(y(n) \middle| x^{(m)}(n)\right)}{f\left(y(n) \middle| \hat{x}^{(k_m)}(n)\right)}$
                      Normalize the weights, $w^{(m)}(n) = \frac{\tilde{w}^{(m)}(n)}{\sum_{i=1}^{M} \tilde{w}^{(j)}(n)}$
```

So, what do we gain by computing estimates of $x(n)$ and implementing the drawing of particles by auxiliary variables? By using the estimates of $x^{(m)}(n)$, we look ahead to how good the particle streams may be. Rather than resampling from samples obtained from the prior, we first resample by using the latest measurement and then propagate from the surviving streams. Thereby, at the end of the recursion instead of having particles propagated without the use of $y(n)$, we have particles moved in directions preferred by $y(n)$. With SIR, the data $y(n)$ affect the direction of particle propagation later they do with APF.

It is important to know that one cannot guarantee that the basic APF method will perform better than the SIR algorithm inspite of the help of the latest observations. The reason for this is that the new particles are still generated by the prior only.

Now we describe a more general version of the APF method. As with the basic APF, one preselects the streams that are propagated. Instead of a preselection based on the weights (3.3), we use weights that we denote by $w_a^{(m)}(n)$. After the k_m is selected, for propagation, we use the proposal distribution $\pi\big(\, x(n) \mid x^{(k_m)}(n-1), y(n)\,\big)$, that is,

$$x^{(m)}(n) \sim \pi\big(\, x(n) \mid x^{(k_m)}(n-1), y(n)\,\big) \qquad (3.5)$$

The new weights are then computed according to

$$\tilde{w}^{(m)}(n) = \frac{w^{(k_m)}(n-1)}{w_a^{(k_m)}(n)} \frac{f\left(y(n)\middle|x^{(m)}(n)\right) f\left(x^{(m)}(n)\middle|x^{(k_m)}(n-1)\right)}{\pi(x^{(m)}(n)|x^{(k_m)}(n-1), y(n))} \qquad (3.6)$$

Note that for the basic APF we have

$$w_a^{(k_m)}(n) \propto w^{(k_m)}(n-1) f\big(y(n)|\hat{x}^{(k_m)}(n)\big)$$

And $\quad \pi\big(x^{(m)}(n)\big|x^{(k_m)}(n-1), y(n)\big) = f\left(x^{(m)}(n)\middle|x^{(k_m)}(n-1)\right)$

As with the standard PFs, the performance of the APF depends strongly on the proposal distribution $\pi\big(x(n)\big|x^{(k_m)}(n-1), y(n)\big)$. However, its performance also depends on the choice of the weights$w_a^{(k_m)}(n)$.

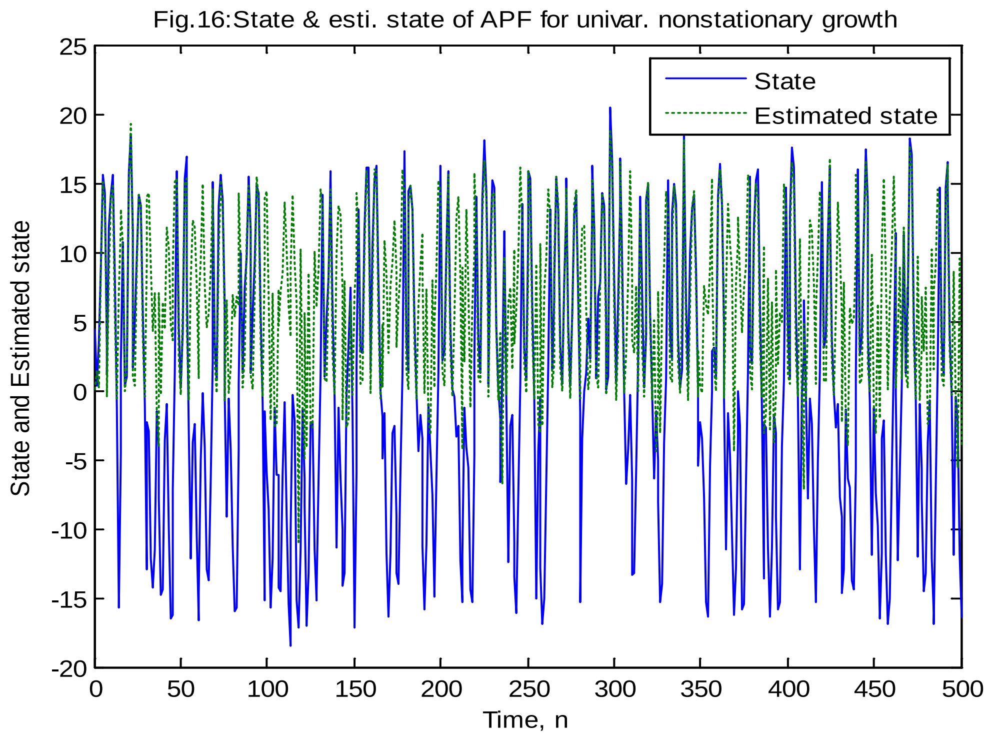


Figure 3.7 : State and estimated state of APF Particle filter for univariate nonstationary growth

3.3 GAUSSIAN PARTICLE FILTERING

With Gaussian particle filtering, we approximate the posterior and the predictive densities of the states with Gaussians [2]. The method is very simple to implement, and its advantage over other particle filtering method is that it does not require resampling. Another advantage is that the estimation of constant parameters is not a problem as is the case with other methods. In brief, the treatment of dynamic and constant states is identical. On the other hand, its disadvantage is that if the approximated densities are not Gaussians, the estimates may be inaccurate and the filter may diverge as any other method that uses Gaussian approximations. In that case, an alternative could be the use of Gaussian sum filtering where the densities in the system are approximated by mixture Gaussian [3]. The method proceeds as follows. Let the random measure at time instant $(n-1)$ be given by χ(n-1) = $\left\{x^{(m)}(n-1),\ 1/M\right\}_{m=1}^{M}$. In the first step, we draw samples $x_c^{(m)}(n)$ from the prior, that is

$$x_c^{(m)}(n) \sim\ f\left(x(n)\middle|x^{(m)}(n-1)\right)$$

For the obtained samples, we compute their weights according to

$$\widetilde{w}^{(m)}(n) = f\left(y(n)\middle|x_c^{(m)}(n)\right)$$

And then we normalize them. Now, the assumption is that the particles $x_c^{(m)}(n)$ and their weights $w^{(m)}(n)$ approximate a normal distribution whose moments are estimated by

$$\mu(n) = \sum_{m=1}^{M} w^{(m)}(n)\, x_c^{(m)}(n)$$

$$\sum(n) = \sum_{m=1}^{M} w^{(m)}(n)\, (x_c^{(m)}(n) - \mu(n))(x_c^{(m)}(n) - \mu(n))^T$$

Finally, the particles that are used for propagation at the next time instant $(n+1)$ are generated by

$$x^{(m)}(n) \sim N(\mu(n), \sum(n))$$

And they are all assigned the same weights. The method is summarized in Table 3.3.

Table 3.3: The GPF algorithm

```
Initialization
For $m = 1,2,\ldots,M$
    sample $x_c^{(m)}(0) \sim f(x(0))$
    $w^{(m)}(0) = \frac{1}{M}$
Recursions
  For $n = 1,2,\ldots,N$
  Drawing particles from the predictive density
    $x_c^{(m)}(n) \sim f\left(x(n) \,\middle|\, x^{(m)}(n-1)\right), m = 1,2,\ldots,M$
Computation of weights
    Evaluate the weights, $\tilde{w}^{(m)}(n) = f\left(y(n) \,\middle|\, x_c^{(m)}(n)\right)$
        Normalize the weights, $w^{(m)}(n) = \frac{\tilde{w}^{(m)}(n)}{\sum_{j=1}^{M} \tilde{w}^{(j)}(n)}$
            Computations of the moments of the filtering density
                $\mu(n) = \sum_{m=1}^{M} w^{(m)}(n)\, x_c^{(m)}(n)$
            $\sum(n) = \sum_{m=1}^{M} w^{(m)}(n)\, (x_c^{(m)}(n) - \mu(n))(x_c^{(m)}(n) - \mu(n))^T$
                Drawing particles for propagation at the next time instant
                $x^{(m)}(n) \sim N(\mu(n), \sum(n))\, , m = 1,2,\ldots,M$
```

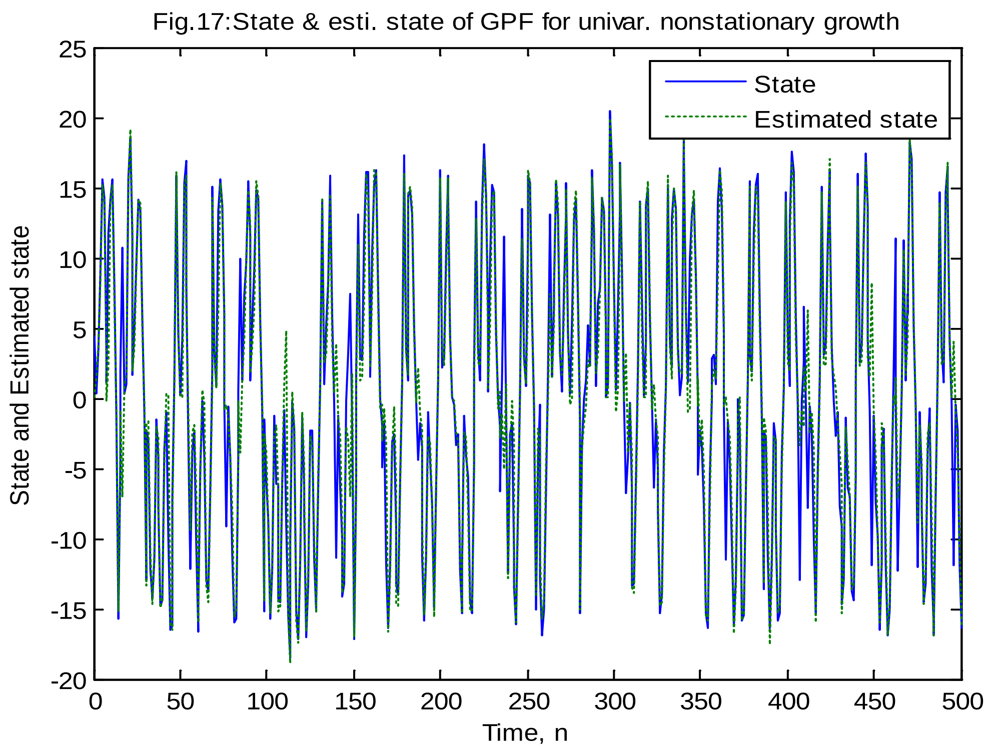


Figure 3.8 : State and estimated state of GPF Particle filter for univariate nonstationary growth

Comparison of MSEs of SIR PF, APF, and GPF with respect to total number of data samples is also shown in Fig.18 of Figure 3.9.

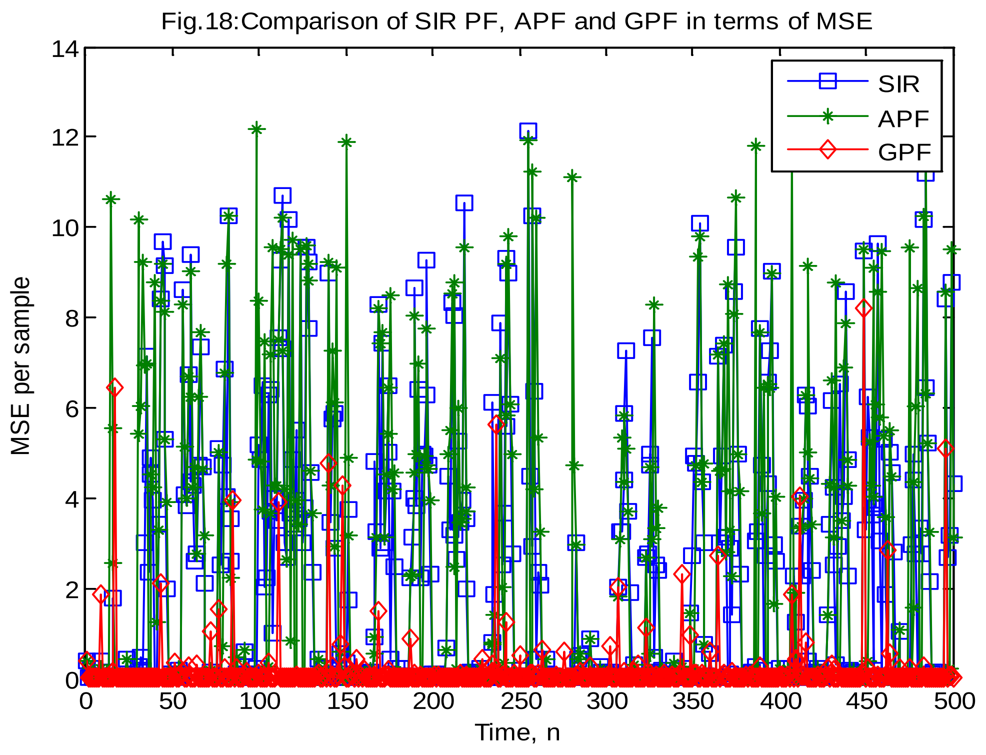


Figure 3.9 : Comparison of SIR PF, APF and GPF in terms of MSE

Comparison of particles and associated weights in respect of SIR PF, APF, and GPF at time n=1, 2, 4, and 7 are also shown in Fig.19-22 (Figure 3.10), Fig. 23-26 (Figure 3.11), and Fig. 27-30, (Figure 3.12) respectively. From this results, we can find out that, APF, rather than resampling from samples obtained from the prior, does resampling by using the latest measurement and then propagate from the surviving streams, making the particles move in directions preferred by the observation equation. In GPF results, we find out that the propagating particles are updated according to mean and variance of the approximated Normal distribution of the posterior and prior densities.

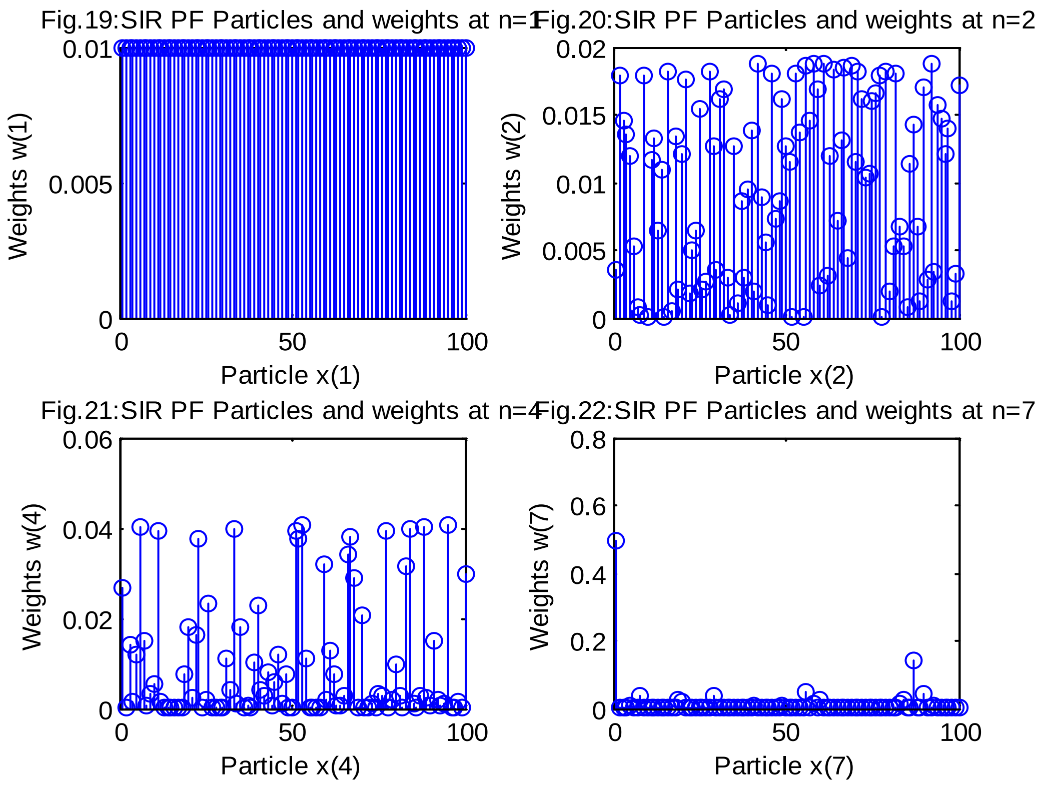


Figure 3.10 : SIR PF Particles and its weight at different time instants

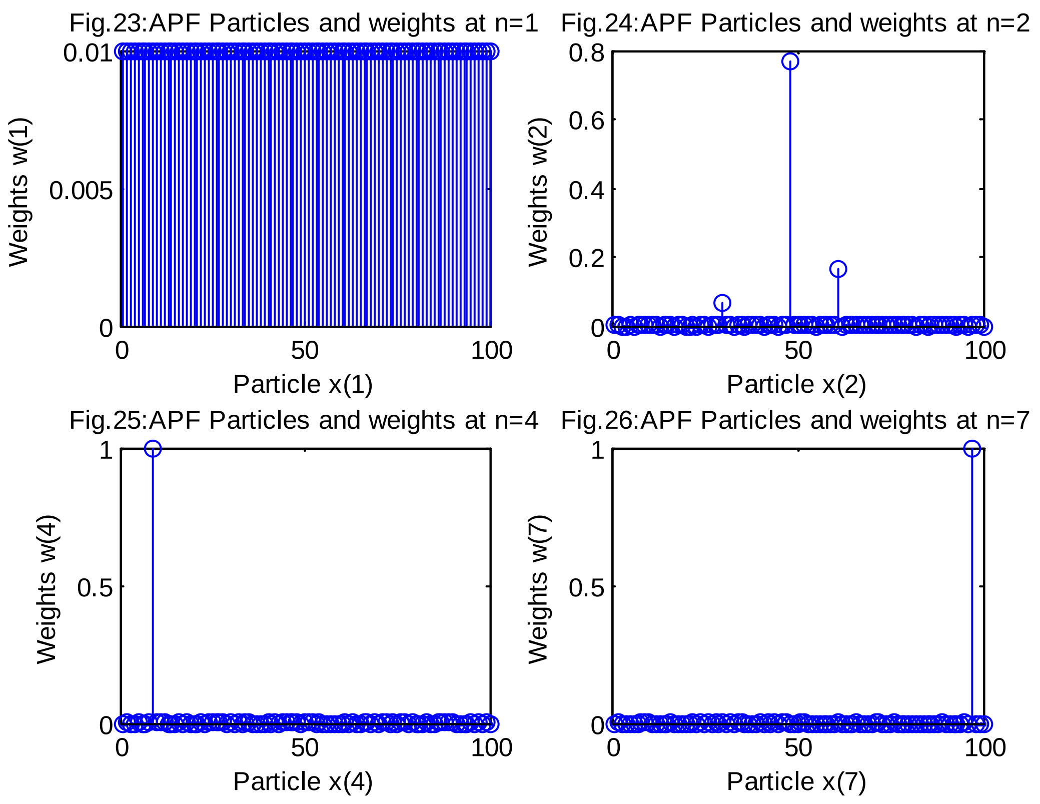


Figure 3.11 : APF Particles and its weight at different time instants

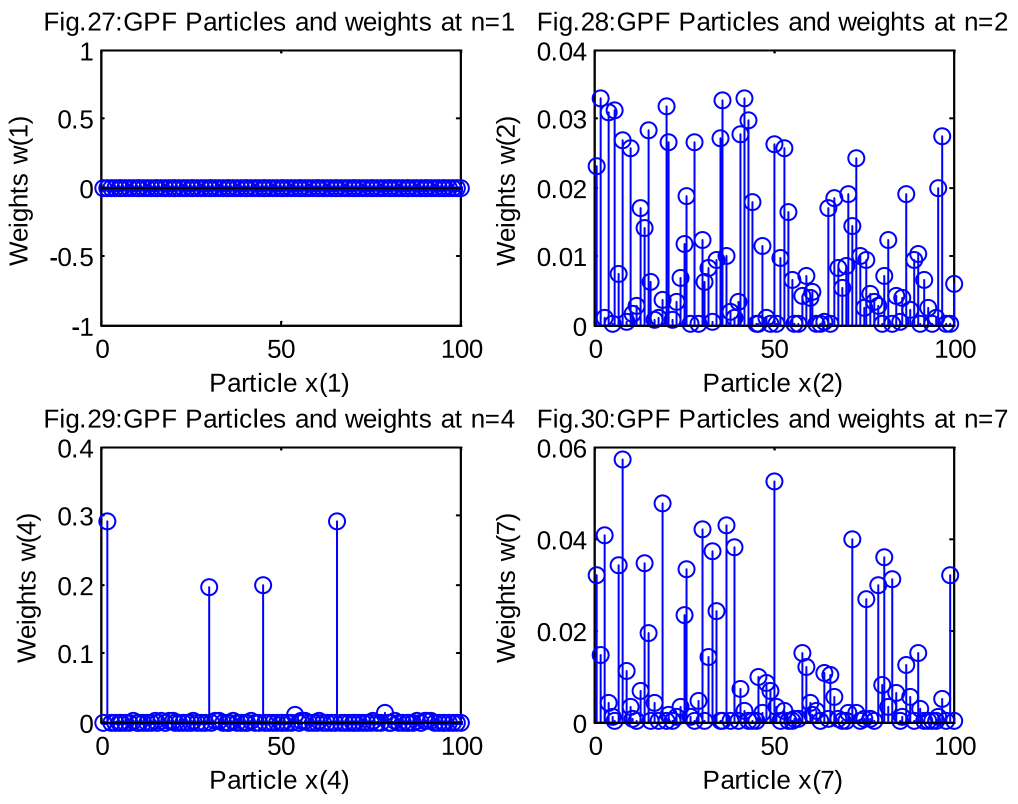


Figure 3.12 : GPF Particles and its weight at different time instants

3.4 HANDLING CONSTANT PARAMETERS

The particle filter methodology was originally devised for the estimation of dynamic signals rather than static parameters. The most efficient way to address the problem is to integrate out the unknown parameters when possible, either analytically or by Monte Carlo procedures. The former methods, however, depend on the feasibility of integration that is, on the mathematical model of the system. Generic solutions, useful for any model, are scant and limited in performance. A common feature of most of the approaches is that they introduce artificial evolution of the fixed parameters and thereby treat them in a similar way as the dynamic states of the model. Some methods insert the use of Markov chain Monte Carlo sampling to preserve the diversity of the particles.

A recent work [4] introduces a special class of PFs called density-assisted PFs that approximate the filtering density with a predefined parametric density by generalizing the concepts of Gaussian PFs and Gaussian sum PFs. These new filters can cope with constant parameters more naturally than the previously proposed methods. In this section, the problem of handling static parameters by PFs is reviewed under a kernel-based auxiliary PF method and the density-assisted particle filtering technique.

We can reformulate the state-space model to explicitly incorporate fixed parameters as

$$x(n) = g_1\big(x(n-1), \theta, v_1(n)\big)$$
$$y(n) = g_2\big(x(n), \theta, v_2(n)\big)$$

Where all the symbols have the same meaning as before and θ is a vector of fixed parameters. Based on the observations $y(n)$ and the assumptions, the objective is to estimate $x(n)$ and θ recursively. In the particle filtering context, this amounts to obtaining the approximation of $f\big(x(n), \theta \big| y(1{:}n)\big)$ by updating the approximation for $f\big(x(n-1), \theta \big| y(1{:}n-1)\big)$.

3.4.1 Kernel-based Auxiliary Particle Filter

The inclusion of fixed parameters in the model implies extending the random measure to the form

$$\chi(n) = \left\{x^{(m)}(n), \qquad \theta^{(m)}(n), \qquad w^{(m)}(n)\right\}_{m=1}^{M}$$

where the index n in the samples of θ indicates the approximation of the posterior at time *n* and not the time-variation of the parameter vector. The random measure approximates the density of interest $f\big(x(n), \theta \big| y(n)\big)$ which can be decomposed as

$$f\big(\, x(n), \theta \mid\; y(1{:}n) \,\big) \propto$$
$$f(\, y(n) \mid x(n), \theta)\, f(\, x(n) \mid \theta, y(1{:}n-1)\,) f(\theta \mid y(1{:}n-1))$$

From the previous expression it is clear that there is a need for approximation of the density $f(\theta \mid y(1{:}n-1))$.

$$f\big(\, \theta \mid y(1{:}n-1) \,\big) \approx \sum_{m=1}^{M} w^{(m)}(n)\, N(\, \theta \mid \bar{\theta}^{(m)}(n), h^2 \sum_{\theta}(n))$$

The above expression represents a mixture of Gaussian distributions, where the mixands $N(\theta \mid \bar{\theta}^{(m)}(n), h^2 \sum_{\theta}(n))$ are weighted by the particle weight $w^{(m)}(n)$. The parameters of the mixands are obtained using the previous time instant particles and weights, that is,

$$\bar{\theta}^{(m)}(n) = \sqrt{(1-h^2)}\theta^{(m)}(n-1) + (1-\sqrt{(1-h^2)}) \textstyle\sum_{i=1}^{M} w^{(i)}(n-1)\, \theta^{(i)}(n-1)$$

$$\sum_{\theta}(n) = \sum_{m=1}^{M} w^{(m)}(n)\, (\theta^{(m)}(n) - \bar{\theta}^{(m)}(n))(\theta^{(m)}(n) - \bar{\theta}^{(m)}(n))^T$$

Where h^2 is a smoothing parameter computed by $h^2 = 1 - (\frac{3\gamma-1}{2\gamma})^2$, and $\gamma \in (0,1)$ represents a discount factor, typically around 0.95-0.99.

We now describe the implementation of this idea in the context of auxiliary particle filtering. Assume that at time instant $(n-1)$, we have the random measure χ(n-1) = $\{x^{(m)}(n-1), \theta^{(m)}(n-1), w^{(m)}(n-1)\,\}_{m=1}^{M}$. Then, we proceed as follows.

1. Estimate the next particle by $\hat{x}^{(m)}(n) = E(x(n) \mid x^{(m)}(n-1), \theta^{(m)}(n-1))$. This step is identical to the APF.
2. Sample the indexes k_m of the streams that survive, where $k_m = i$ with probability $w_a^{(i)} \propto w^{(i)}(n-1) f(y(n) \mid \hat{x}^{(i)}(n), \bar{\theta}^{(i)}(n-1))$. Here we basically resample so that the most promising streams are kept in the random measure and the less promising are removed.
3. Draw particles of the fixed parameter vector according to

$$\theta^{(m)}(n) \sim N(\bar{\theta}^{(K_m)}(n-1), h^2 \sum_{\theta}(n-1))$$

Where the means and covariance of the mixands are obtained at the end of the cycle of computations for time instant $(n-1)$. This step takes care of the vector of constant parameters.

4. Draw particles of the state according to

$$x^{(m)}(n) \sim f(x(n) \mid x^{(K_m)}(n-1), \theta^{(m)}(n))$$

With this step we complete the drawing of new particles.

5. Update and normalize the weights, where

$$w^{(m)}(n) \propto \frac{f(y(n) \mid x^{(m)}(n), \theta^{(m)}(n))}{f(y(n) \mid \hat{x}^{(K_m)}(n), \bar{\theta}^{(K_m)}(n-1))}$$

With this step we associate weights to the particles.

6. Compute the parameters of the kernels used in constructing the mixture Gaussian according to

$$\bar{\theta}^{(m)}(n) = \sqrt{(1-h^2)}\theta^{(m)}(n) + (1-\sqrt{(1-h^2)})\sum_{i=1}^{M} w^{(i)}(n)\,\theta^{(i)}(n)$$

$$\sum_{\theta}(n) = \sum_{m=1}^{M} w^{(m)}(n)\,(\theta^{(m)}(n) - \bar{\theta}^{(m)}(n))(\theta^{(m)}(n) - \bar{\theta}^{(m)}(n))^T$$

Example 6: Consider the problem of tracking one target using two static sensors which collected bearings-only measurements. Here measurements are biased. The mathematical formulation of the problem is given by

$$x(n) = Ax(n-1) + Bv_1(n)$$
$$y(n) = g_2(x(n)) + \theta + v_2(n)$$

Where all the parameters have the same meaning as before and $\theta = [b_1 b_2]^T$ represents a vector of unknown constant biases.

In step 1, we generate $\hat{x}^{(m)}(n) = Ax^{(m)}(n)$.

The sampling is carried out by using weights that are proportional to $w^{(i)}(n-1)N(g_2\left(\hat{x}^{(i)}(n)\right) + \bar{\theta}^{(i)}(n-1), \sum_v(n-1))$, where the Gaussian is computed at $y(n)$. The remaining steps can readily be deduced from the scheme described by Table 3.3.

3.4.2 Density-assisted Particle Filter

As seen in the previous section, the GPFs approximate the predictive and filtering densities by Gaussian densities whose parameters are estimated from the particles and their weights. Similarly, the Gaussian sum PFs [3] approximate these densities with mixture of Gaussians. The approximating densities can be other than Gaussian or mixture of Gaussian, and therefore, we refer to the general class of filters of this type as density-assisted PFs (DAPFs) [4], which are a generalization of the Gaussian and Guassian sum PFs. The main adavantages of DAPFs is that they do not necessarily use resampling in the sense carried out by standard PFs and they do not share the limitations regarding the estimation of constant model parameters. Here we explain how we can use DAPFs when we have constant parameters in the model.

Let χ(n-1) = $\left\{x^{(m)}(n-1), \theta^{(m)}(n-1), w^{(m)}(n-1)\right\}_{m=1}^{M}$ be the random measure at time instant $(n-1)$, $x^{(m)}(n-1)$ and $\theta^{(m)}(n-1)$ the particles of $x(n-1)$ and θ, respectively, and $w^{(m)}(n-1)$ their associated weights. If we denote the approximating density of $f(x(n-1), \theta \mid y(1{:}n-1))$ by $\pi(\Phi(n-1))$, where $\Phi(n-1)$ are the parameters of the approximating density, the steps of the density-assisted particle filter are the following.

1. Draw particles according to
   $\{x^{(m)}(n-1), \theta^{(m)}(n-1)\} \sim \pi(\Phi(n-1))$.
2. Draw particles according to
   $x^{(m)}(n) \sim f(x(n) \mid x^{(m)}(n-1), \theta^{(m)}(n-1))$.
3. Set $\theta^{(m)}(n) = \theta^{(m)}(n-1)$.
4. Update and normalize the weights
   $w^{(m)}(n) \propto f(y(n) \mid x^{(m)}(n), \theta^{(m)}(n))$.
5. Estimate the parameters, $\Phi(n)$, of the density from
   χ(n) = $\left\{x^{(m)}(n), \theta^{(m)}(n), w^{(m)}(n)\right\}_{m=1}^{M}$.

The problem of standard PFs regarding constant parameters is avoided in the first step by drawing particles of the constant from an approximation of the posterior. It is important to note that one can combine standard PFs and DAPFs, in that we apply the standard PFs for the dynamic variables and DAPFs for the constant parameters.

Example 7: We consider the problem of tracking in a two-dimensional plane. We assume that the marginal posterior density of the bias vector θ is a Gaussian density. Suppose that at time instant $(n-1)$, we have the random measure χ(n-1) = $\{x^{(m)}(n-1), \theta^{(m)}(n-1), w^{(m)}(n-1)\}_{m=1}^{M}$. From the random measure, we can compute the parameters of the approximating Gaussian of the bias vector by

$$\mu(n-1) = \sum_{m=1}^{M} w^{(m)}(n-1)\, \theta^{(m)}(n-1)$$

$$\sum(n-1) = \sum_{m=1}^{M} w^{(m)}(n-1)\, (\theta^{(m)}(n-1) - \mu(n-1))(\theta^{(m)}(n-1) - \mu(n-1))^{T}$$

Then we draw the particles of θ for the next time step, that is, $\theta^{(m)}(n) \sim N(\mu(n-1), \sum(n-1))$. Once we have particles of the biases, we proceed by using the favourite PF. For example, if it is the APF, first we project the dynamic particles, that is $\hat{x}^{(m)}(n) = \hat{A}x^{(m)}(n)$. This is followed by resampling of the streams whose weights are given by $w^{(k_m)}(n-1)N(g_2\left(\hat{x}^{(k_m)}(n)\right) + \theta^{(k_m)}(n), \sum_v(n))$ and where the Gaussian is computed at $y(n)$. Once the indexes of the streams for propagation are known, we draw the particles of the dynamic variables, $x^{(m)}(n)$. Next , the new weights of the streams are computed by

$$w^{(m)}(n) \propto \frac{f(y(n) \mid x^{(m)}(n), \theta^{(k_m)}(n))}{f(y(n) \mid \hat{x}^{(K_m)}(n), \theta^{(K_m)}(n))}$$

CHAPTER 4
RAO-BLACKWELL PARTICLE FILTER

4.1 BRIEF INTRODUCTION

In many practical problems, the considered dynamic nonlinear system may have some states that are conditionally linear given the nonlinear states of the system. When the applied methodology is particle filtering, this conditional linearity can be expoited using the concept of Rao-Blackwellization [5, 6]. Rao-Blackwellization is a statistical procedure that is used for reducing variance of estimates obtained by Monte Carlo sampling methods, and by employing it, we can have improved filtering of the unknown states.

The main idea consists of tracking the linear states differently from the nonlinear states by treating the linear parameters as nuisance parameters and marginalizing them out of the estimation problem. This strategy allows for more accurate estimates of the unknowns because the dimension of the space that is explored with particles is reduced and therefore it is much better searched. At every time instant the particles of the nonlinear states are propagated randomly, and once they are known, the problem is linear in the rest of the states. Therefore, one can find their 'optimal' values by employing Kalman filtering and associate them with the sampled nonlinear states. Some recent applications of Rao-Blackwellized PFs include tracking of maneuvering targets in clutter [7] and joint target tracking using kinematic radar information [8]. In [9], a computational complexity analysis of Rao-Blackwellized PFs is provided.

4.2 DERIVATION

For the scenario of a nonlinear system with conditionally linear states, we write the state space model as

$$x_n(n) = g_{1,n}\big(x_n(n-1)\big) + A_{1,n}\big(x_n(n-1)\big)x_l(n-1) + v_{1,n}(n)$$
$$x_l(n) = g_{1,l}\big(x_n(n-1)\big) + A_{1,l}\big(x_n(n-1)\big)x_l(n-1) + v_{1,l}(n)$$
$$y_t = g_2\big(x_n(n)\big) + A_2\big(x_n(n)\big)x_l(n) + v_2(n)$$

Where the system state, $x(n)$, includes nonlinear and conditionally linear components, that is, $x^T(n) = [x_n^T(n)\ \ x_l^T(n)]$, with $x_n(n)$ and $x_l(n)$ being the nonlinear and conditionally linear states, respectively; $v_{1,n}(n)$ and $v_{1,l}(n)$ are state noise vectors at time instant n which are assumed to be Gaussian; $g_{1,n}(\cdot)$ and $g_{1,l}(\cdot)$ are nonlinear state transition functions; $A_{1,n}$ and $A_{1,l}$ are matrices whose entries may be functions of the nonlinear states; $g_2(\cdot)$ is a nonlinear measurement function of the nonlinear states; A_2 is another matrix whose entries may be functions of the nonlinear states; and $v_2(n)$ is observation noise vector at time n which is also assumed to be Gaussian and independent from the state noises.

Suppose that at time instant $(n-1)$, the random measure composed of M streams is given by

$$\chi(\text{n-1}) = \left\{x^{(m)}(n-1), \qquad w^{(m)}(n-1)\right\}_{m=1}^{M}$$

and that the linear state $x_l(n-1)$ in the $m-th$ stream is Gaussian distributed, that is

$$x_l(n-1) \sim N(\hat{x}_l^{(m)}(n-1), \hat{C}_{x_l}^{(m)}(n-1))$$

Where $\hat{x}_l^{(m)}(n-1)$ is the estimate of x_l in the $m-th$ stream at time instant $(n-1)$ and $\hat{C}_{x_l}^{(m)}(n-1)$ is the covariance matrix of that estimate. The scheme runs as follows.

1. Generate the nonlinear state particles according to the marginalized prior PDF

$$x^{(m)}(n) \sim f(x_n(n) \mid x^{(m)}(n-1), y(1{:}n-1))$$

Where

$$f\left(x_n(n)\middle|x_n^{(m)}(n-1), y(1{:}n-1)\right)$$
$$= \int f\left(x_n(n)\middle|x_n^{(m)}(n-1), x_l^{(m)}(n-1)\right)$$
$$\times\ f\left(x_l^{(m)}(n-1)\middle|x_n^{(m)}(n-1), y(1{:}n-1)\right)dx_l^{(m)}(n-1)$$

If the distribution inside the integral are Gaussians, $f\left(x_n(n)\middle|x_n^{(m)}(n-1), y(n-1)\right)$ is also a Gaussian distribution. It can be shown that

$$f\left(x_n(n)\middle|x_n^{(m)}(n-1), y(n-1)\right) = N(\mu_{x_n}^{(m)}(n), \sum\nolimits_{x_n}^{(m)}(n))$$

Where

$$\mu_{x_n}^{(m)}(n) = g_{1,n}\left(x_n^{(m)}(n-1)\right) + A_{1,n}^{(m)}(n-1)\,\hat{x}_l^{(m)}(n-1)$$

$$\sum_{x_n}^{(m)}(n) = A_{1,n}^{(m)}(n-1)\hat{C}_{x_l}^{(m)}(n-1){A_{1,n}^{(m)}}^T(n-1) + C_{v_{1,n}}$$

Where we have dropped in the notation that the matrix $A_{1,n}^{(m)}(n-1)$ may be a function of $x_n^{(m)}(n-1)$. The symbol $\hat{x}_l^{(m)}(n-1)$ is the estimate of the linear state at time instant $(n-1)$ from the $m-th$ stream, and $\hat{C}_{x_l}^{(m)}(n-1)$ is the covariance matrix of that estimate.

2. Update the linear states with quasi measurements. We define the quasi measurements by

$$z^{(m)}(n) = x_n^{(m)}(n) - g_{1,n}\left(x_n^{(m)}(n-1)\right)$$
$$= A_{1,n}^{(m)}(n-1)\hat{x}_l^{(m)}(n-1) + v_{1,n}(n)$$

And use them to improve the estimate $\hat{x}_l^{(m)}(n-1)$. The new estimate $\hat{\hat{x}}_l^{(m)}(n-1)$ is obtained by applying the measurement step of the Kalman filter, that is,

$$\hat{\hat{x}}_l^{(m)}(n-1) = \hat{x}_l^{(m)}(n-1) + L^{(m)}(n-1)(z^{(m)}(n) - A_{1,n}^{(m)}(n-1)\hat{x}_l^{(m)}(n-1)$$
$$L^{(m)}(n-1)$$
$$= \hat{C}_{x_l}^{(m)}(n-1)$$
$$\times (A_{1,n}^{(m)}(n-1)A_{1,n}^{(m)}(n-1)\hat{C}_{x_l}^{(m)}(n-1){A_{1,n}^{(m)}}^T(n-1) + C_{v1,n})^{-1}$$

$$\hat{\hat{C}}_{x_l}^{(m)}(n-1) = (I - L^{(m)}(n-1)A_{1,n}^{(m)}(n-1))\hat{C}_{x_l}^{(m)}(n-1)$$

3. Perform time update of the linear states according to

$$\tilde{x}_l^{(m)}(n) = g_{1,l}\left(x_n^{(m)}(n-1)\right) + A_{1,l}^{(m)}(n-1)\hat{\hat{x}}_l^{(m)}(n-1)$$
$$\tilde{C}_{x_l}^{(m)}(n) = A_{1,l}^{(m)}(n-1)\hat{\hat{C}}_{x_l}^{(m)}(n-1){A_{1,l}^{(m)}}^T(n-1) + C_{v_{1,l}}$$

4. Compute the weights by

$$w_a^{(m)}(n) \propto w^{(m)}(n-1)f\left(y(n)\middle|x_n^{(m)}(0:n), y(1:n-1)\right)$$

Where

$$f\left(y(n)\middle|x_n^{(m)}(0:n), y(1:n-1)\right)$$
$$= \int f\left(y(n)\middle|x_n^{(m)}(n), x_l^{(m)}(n)\right)$$
$$\times\ f\left(x_l^{(m)}(n)\middle|x_n^{(m)}(0:n), y(1:n-1)\right)dx_l^{(m)}(n)$$

Again, if the two densities of $x_l^{(m)}(n)$ inside the integral are Gaussian densities, the integral can be solved analytically. We obtain

$$f\left(y(n)\middle|x_n^{(m)}(0{:}n), y(1{:}n-1)\right) = N(\mu_y^{(m)}(n), \textstyle\sum_y^{(m)}(n))$$

Where

$$\mu_y^{(m)}(n) = g_2\left(x_n^{(m)}(n)\right) + A_2^{(m)}(n)\,\tilde{x}_l^{(m)}(n)$$

$$\sum_y^{(m)}(n) = A_2^{(m)}(n)\tilde{C}_{x_l}^{(m)}(n)A_2^{(m)^T}(n) + C_{v_2}$$

5. Finally we carry out the measurement update of the linear states

$$\hat{x}_l^{(m)}(n) = \tilde{x}_l^{(m)}(n) + K^{(m)}(n)(y(n) - g_2\left(x_n^{(m)}(n)\right) + A_2^{(m)}(n)\,\tilde{x}_l^{(m)}(n))$$

$$K^{(m)}(n) = \tilde{C}_{x_l}^{(m)}(n)A_2^{(m)}(n)(A_2^{(m)}(n)\tilde{C}_{x_l}^{(m)}(n)A_2^{(m)^T}(n) + C_{v_2})^{-1}$$

$$\hat{C}_{x_l}^{(m)}(n) = (I - K^{(m)}(n)A_2^{(m)}(n))\tilde{C}_{x_l}^{(m)}(n)$$

In general, the Rao-Blackwellization amounts to the use of a bank of Kalman filters, one for each particle stream. So, with M particle streams, we will have M KFs.

4.3 TRACKING OF A MANEUVERING TARGET

One application of RBPF is to track moving objects that have piecewise linear dynamics. Here we address the problem of tracking a maneuvering target in the noise. The difficulty arises from the uncertainty in the maneuvering command driving the target. The state of the target at time t is denoted as $x_t \triangleq (l_{x,t}, s_{x,t}, l_{y,t}, s_{y,t})^T$ where $l_{x,t}$ ($l_{y,t}$) and $s_{x,t}$ ($s_{y,t}$) represent the position and velocity of the target in the x (y) direction. It evolves according to following the Jump Markov Linear System (JMLS) model of parameters:

Let $r_t, t = 1, 2, \ldots..$ denote a discrete time Markkov chain with known transition probabilities. A JMLS can be modelled as

$$x_{t+1} = A(r_{t+1})x_t + B(r_{t+1})v_{t+1} + F(r_{t+1})u_{t+1}$$

$$y_t = C(r_t)x_t + D(r_t)\varepsilon_t + G(r_t)u_t$$

Where u_t denotes a known exogenous input, and v_t and ε_t denote independent white Gaussian noise sequences. A jump Markov llinear system can be viewed as a linear system whose parameters evolve $(A(r_t), B(r_t), C(r_t), D(r_t), F(r_t), G(r_t))$ evolve with time according to a finite state Markov chain r_t. Neither the continuousstate process x_t nor the finite state process r_t are observed – instead, we observe the noisy measurement process y_t.

$$A = \begin{pmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{pmatrix}, = 0.2\, I_4 \;,\; C = I_4,\; D = \sqrt{3}\,\text{diag}\,(2, 1, 2, 1)\;,\; G = 0_{4xN}$$

The switching term is $F(r_t)\, u_t$ where r_t is a three-state Markov chain corresponding to the three possible maneuvering commands: straight; left turn; right turn.

It has the following transition probabilities:

$p_{m,m=0.9}$ and $p_{m,n=0.05}$ for $m \neq n$. We have for any t, $F(1)u_t = (0, 0, 0, 0)^T$, $F(2)u_t = (-1.225, -0.35, 1.225, 0.35)^T$, $F(3)u_t = (1.225, 0.35, -1.225, -0.35)^T$

We sample according to the optimal distribution. We realize the signal and the MSE target position and velocity estimate computed using $N = 500$ particles.

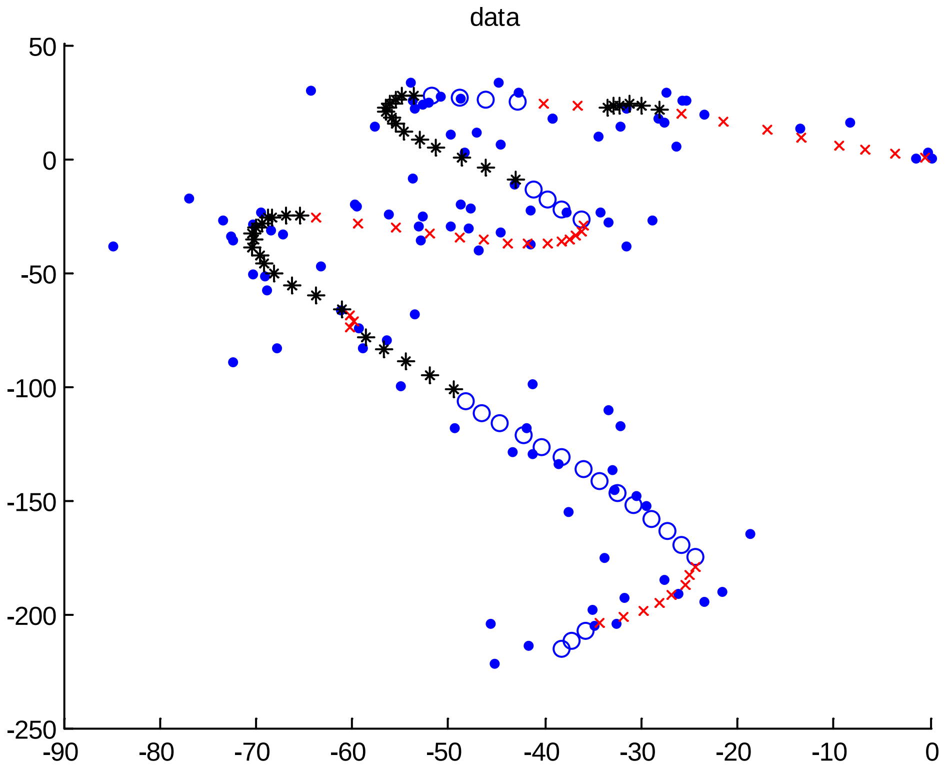


Figure 4.1 : A maneuvering target. The colored symbols represents the hidden discrete state.

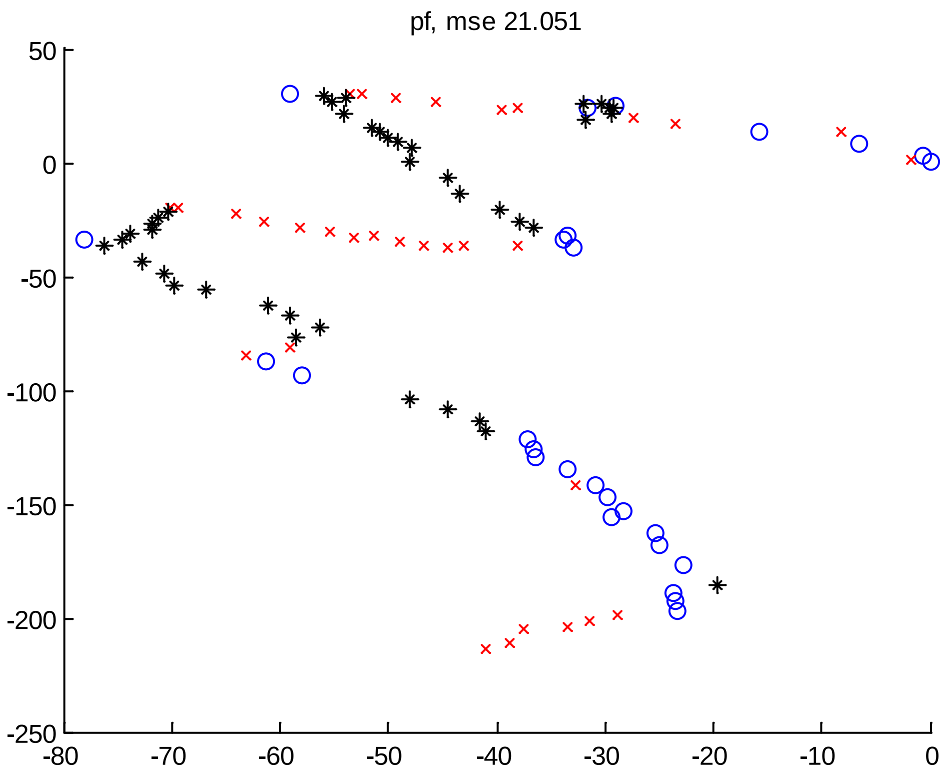


Figure 4.2 : Particle filter estimate

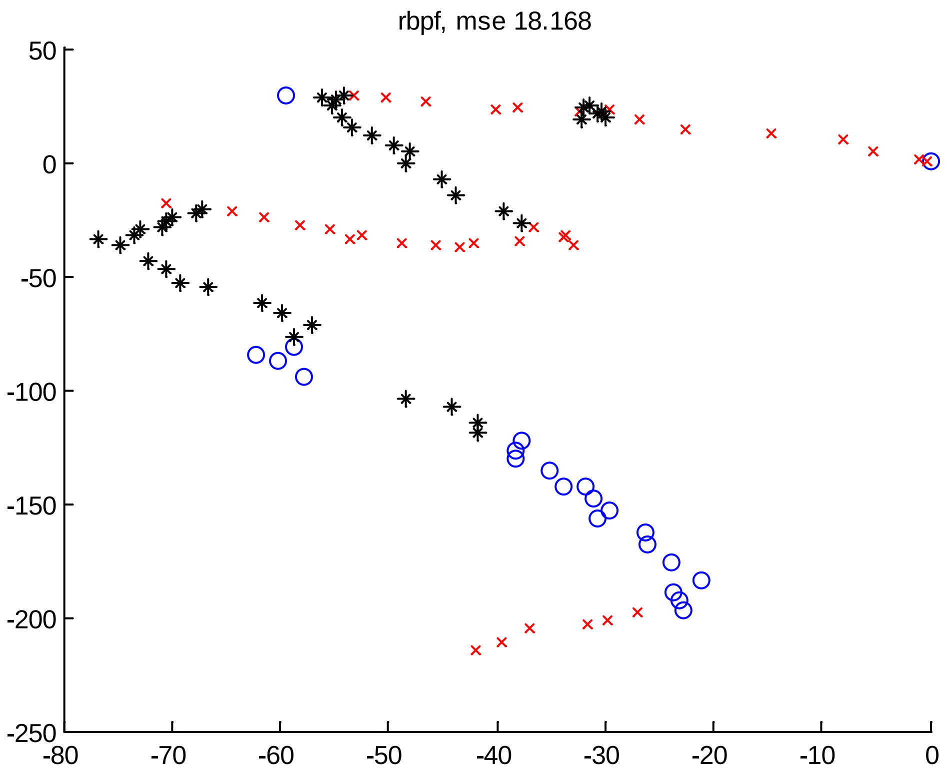


Figure 4.3 : RBPF estimate

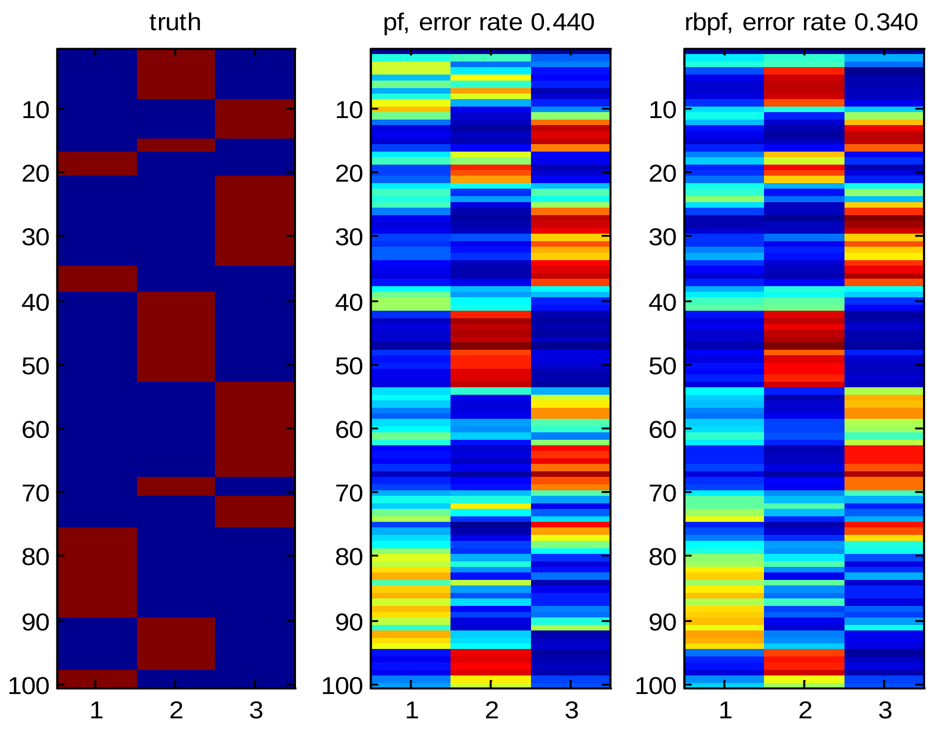


Figure 4.4 : Discrete state corresponding to Figure 4.1, 4.2 and 4.3. The system starts in state 2 (red x), then moves to state 3 (black *), returns briefly to state 2, then switches to state 1(blue circle).

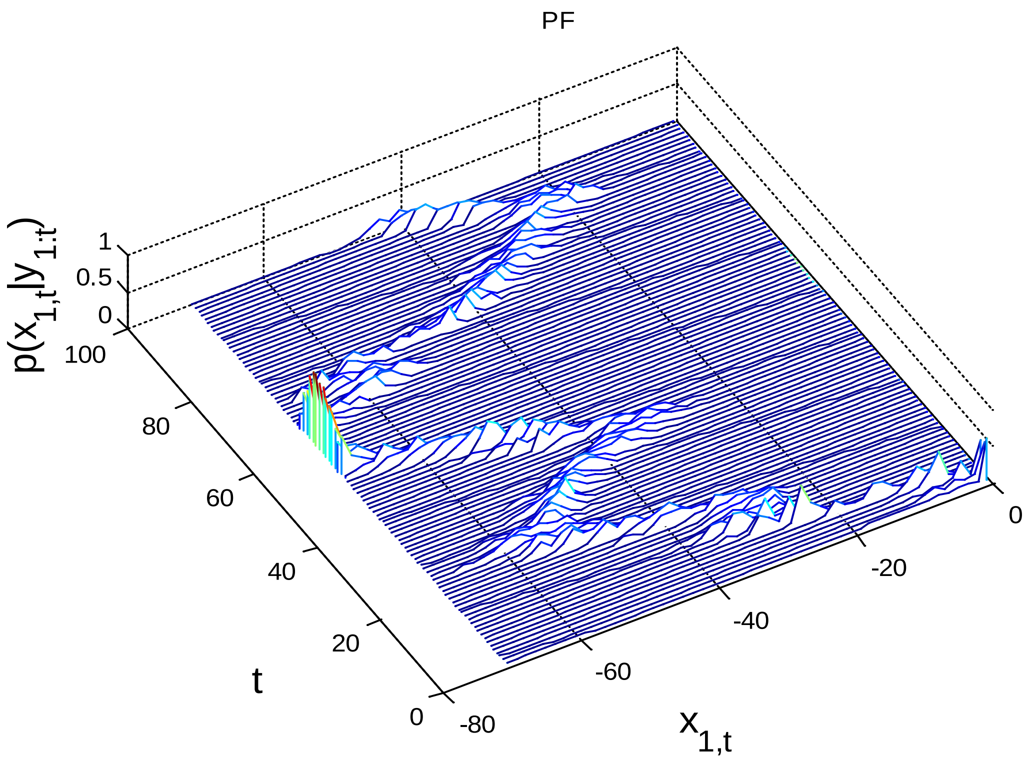


Figure 4.5 : Horizontal location of PF estimate with $X_{1,t}$

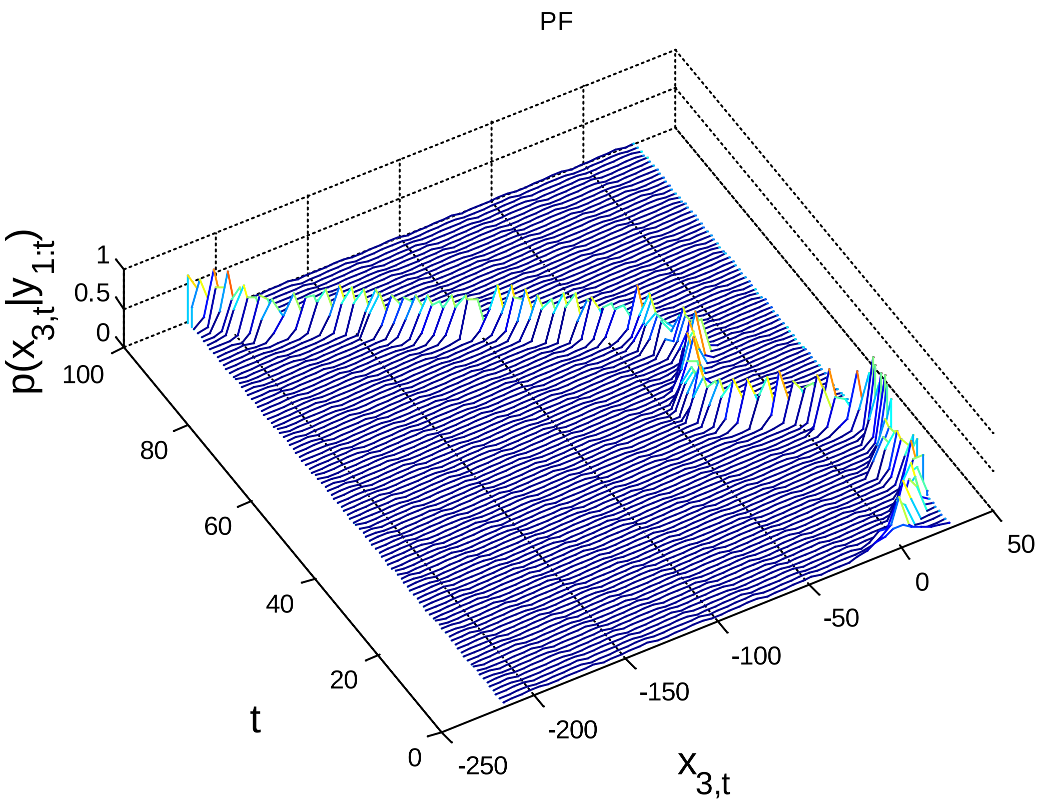


Figure 4.6 : Horizontal location of PF estimate with $X_{3,t}$

Table 4.1 : Comparison of PF and RBPF on the maneuvering target

Method	Misclassification rate	MSE	Time (seconds)
PF	0.440	21.051	7.0893
RBPF	0.340	18.168	12.6829

Figure 4.1 shows the true state of the system from a sample run, starting at (0, 0): the colored symbols denote the discrete state, and the location of the symbol denotes the (*x, y*) location. The small dots represent noisy observations.

Figure 4.2 shows the estimate of the state computed using particle filtering with 500 particles, where the proposal is to sample from the prior. The colored symbols denote the MAP estimate of the state, and the location of the symbol denotes the MMSE (minimum mean square error) estimate of the location, which is given by the posterior mean.

Figure 4.3 shows the estimate computing using RBPF with 500 particles, using the optimal proposal distribution.

Figure 4.4 visualizes the belief state of the system. we show the distribution over the discrete states. We see that the particle filter estimate of the belief state (second column) is not as accurate as the RBPF estimate (third column) in the beginning, although after the first few observations performance is similar for both methods.

Figure 4.5 and 4.6 plot the posterior over the *x* locations. For simplicity, we use the PF estimate, which is a set of weighted samples, but we could also have used the RBPF estimate, which is a set of weighted Gaussians.

Table 4.1 shows the more quantitative comparison between PF and RBPF. We see that RBPF has slightly better performance, although it is also slightly slower.

CHAPTER 5

PARTICLE FILTER FOR VISUAL OBJECT TRACKING

The particle filter was invented to numerically implement the Bayesian estimator which recursively approximates the posterior distribution using a finite set of weighted samples or particles. It has been introduced by many researchers to solve the estimation problem when the system is nonlinear and non-Gaussian. The basic idea behind the particle filter is Monte Carlo simulation, in which the posterior density is approximated by a set of particles with associated weights. As a Bayesian estimator, particle filter has two main steps: prediction and update. Prediction is done by propagating the samples based on the system model. The update step is done by measuring the weight of each samples based on the observation model. The implementation of particle filter can be described as follows.

5.1 PROPOSED METHOD

The proposed method presents tracking of the object of interest using a particle filter to naturally integrate two complementary cues: intensity gradient and color histogram. The shape of the object of interest is modeled as an ellipse, along which an intensity gradient is estimated, while the interior appearance is modeled using a color histogram. These two cues play complementary roles in tracking an object of intererst with free rotation on a cluttered background.

The object of interest is modeled as an ellipse centered at (x, y) and with size (H_x, H_y). At the first frame, we detect the location and size of the ellipse of the object. The dynamics of the (moving) object at time t are described by a state vector S_t consisting of the following eight components

$$\{x, y, X_v, Y_v, H_x, H_y, H_{vx}, H_{vy}\}$$

where (x, y) represent the center location of the object ellipse, (X_v, Y_v) represent the motion velocity, (H_x, H_y) are the lengths of the half axes, and (H_{vx}, H_{vy}) are the corresponding scale changes on the axes.

(a) Particle initialization:

Starting with a weighted set of samples at $k-1, \{X_{k-1}^{i}, \pi_{k-1}^{i}; i = 1 : N\}$ approximately distributed according to $p(x_{k-1}|y_{k-1})$ as initial distribution

$p(x_0)$, new samples are generated from a suitable proposal distribution, which may depend on the previous state and the new measurements.

(b) Prediction step:

Using the probabilistic system transition model $p(x_k|x_{k-1})$, the particles are predicted at time k. It is done by propagating each particle based on the transition or system model.

$$x_{k+1} = f_k(x_k\ ,\omega_k) = p(x_k|x_{k-1})$$

(c) Update step:

To maintain a consistent sample, the new importance weights are set to

$$\pi_k^i = \pi_{k-1}^i \frac{p\left(y_k \middle| x_k^i\right) p(x_k^i|x_{k-1}^i)}{q(x_k|x_{1:k-1},y_{1:k})}$$

It is done by measuring the likelihood of each sample based on the observation model.

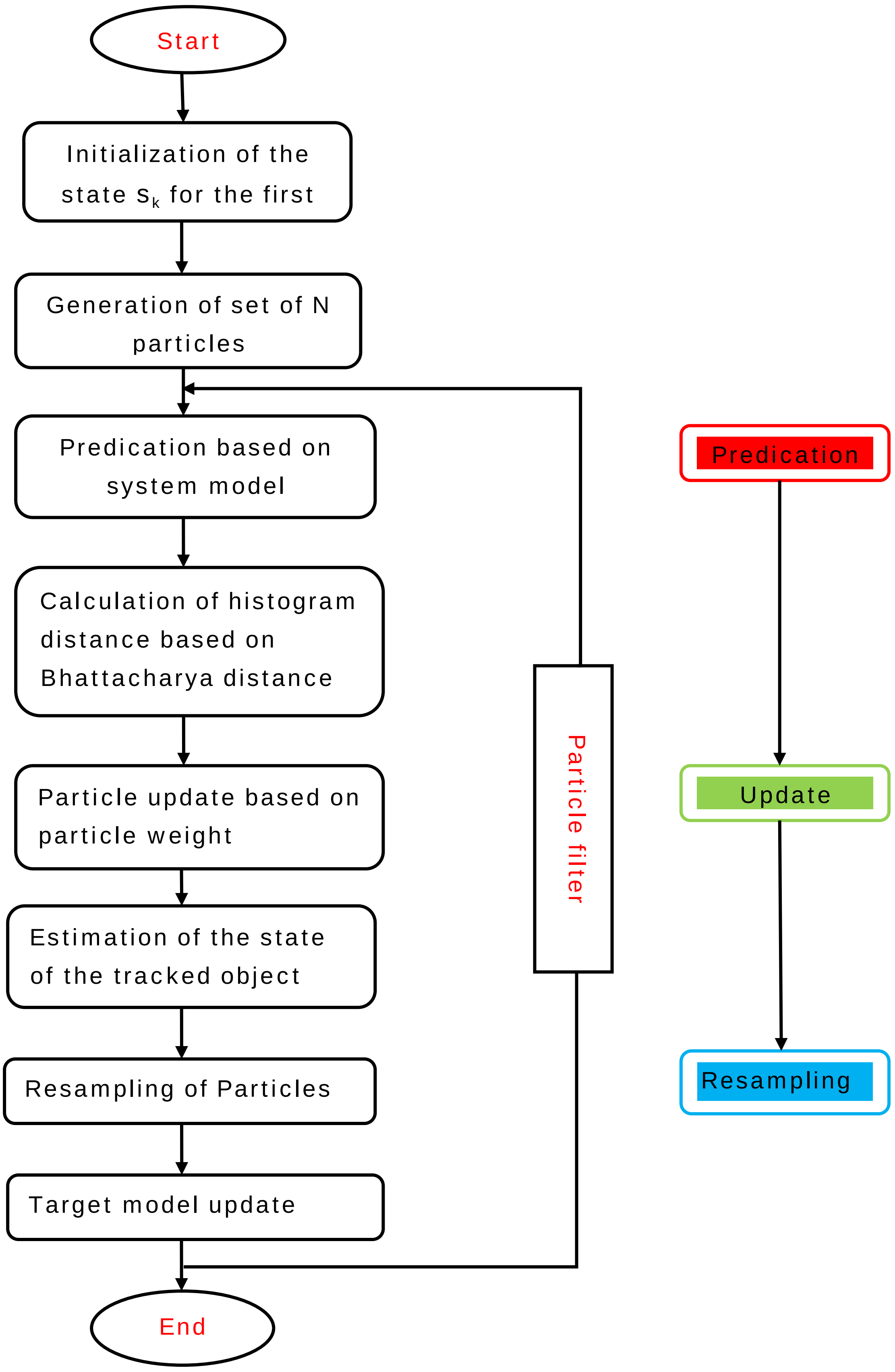
(d) Resample:

This step is performed to generate a new samples set according to their weight for the next iteration. The resample step will decrease the number of the samples with low weight and will increase the number of high weight sample. The new particles set is re-sampled using normalized weights π_k^i as probabilities. This sample set represents the posterior at time k, $p(x_k|y_{1:k})$.

(e) Then, the expectations can be approximated as

$$Ep(x_k|y_{1:k}) = \sum_{i=1}^{N} \pi_k^i x_k^i$$

5.2 FLOW CHART OF PROCEDURE



Start
Initialization of the state s_k for the first
Generation of set of N particles
Predication based on system model
Calculation of histogram distance based on Bhattacharya distance
Particle update based on particle weight
Estimation of the state of the tracked object
Resampling of Particles
Target model update
End
Particle filter
Predication
Update
Resampling

11.

Figure 5.1 : Flowchart of procedure

5.3 INTENSITY GRADIENT HISTOGRAM BASED ON THE OBJECT SHAPE

If we obtain the contour of the object of interest, then computing the gradient along the object's contour provides a good measure for distinguishing the object from the background. Assuming that the ellipse in the model matches the object contour, then, for a particular state sample specified by s, the normalized sum of the gradient magnitude around the ellipse boundary is computed as:

$$\psi_g(s) = \frac{1}{N_s} \sum_{i=1}^{N\,(H_x,H_y)} g(x_i, y_i)$$

Where $g(x_i, y_i)$ is the intensity gradient of pixel (x_i, y_i) located at the boundary of ellipse specified by s and N_s is the number of pixels on the perimeter of the ellipse.

Since an ellipse does not accurately describe the contour of the object, to make the above gradient estimate more useful in case of the inaccurate modeling, the gradient at pixel (x_i, y_i) is established as the maximum gradient by a local search along the normal direction.

$$g(x_i, y_i) = max_{(x_n,y_n)\varepsilon L_n}\{g(x_n,y_n)\}$$

Where L_n represents the normal line, (x_n, y_n) is the coordinate of the points that are located on the normal line. (x_n, y_n) must satisfy the following criterion:

$$\sqrt{(x_n - x_i)^2 + (y_n - y_i)^2} < Search\,range$$

$$y_n = \frac{(y_i - C_y) * H_x^2}{(y_i - C_y) * H_y^2} * (x_n - x_i) + y_i$$

The first formula specifies that the distance in the normal direction between points (x_n, y_n) and (x_i, y_i) must be within a certain search range. This will help to restrict our search around the object contour and avoid hitting other distracting points on the background (or within the object) which have big intensity gradients. The second one is the normal line equation at point (x_i, y_i). (C_x, C_y) denotes the ellipse center.

A simple operator is used to compute the gradient in x direction and y direction for pixel (x_n, y_n).

$$g_x(x_n, y_n) = I(x_n - 2, y_n) + 2 * I(x_n - 1, y_n) - 2 * I(x_n + 1, y_n) - I(x_n + 2, y_n)$$

$$g_y(x_n, y_n) = I(x_n, y_n - 2) + 2 * I(x_n, y_n - 1) - 2 * I(x_n, y_n + 1) - I(x_n, y_n + 2)$$

And finally the gradient at point (x_n, y_n) is computed as

$$g(x_n, y_n) = \sqrt{g_x^2(x_n, y_n) + g_y^2(x_n, y_n)}$$

5.4 COLOR HISTOGRAM MODEL

Color distributions are used as target models as they achieve robustness against non-rigidity, rotation and partial occlusion. Suppose that the distributions are discretized into m - bins. The histograms are produced with the function of $h_{(x_i)}$, that assigns the color at location (x_i) to the corresponding bin. In the present method, the histograms are typically calculated in the RGB space using 8x8x8 bins. To make the algorithm less sensitive to lighting conditions, the HSV color space could be used instead of less sensitivity to V (e.g. 8x8x4 bins)

We determine the color distribution inside an upright elliptic region with half axes H_x and H_y. To increase the reliability of the color distribution when boundary pixels belong to the background or get occluded, smaller weights are assigned to the pixels that are further away from the region center by employing a weighting function

$$k(r) = \begin{Bmatrix} 1 - r^2, r < 1 \\ 0, otherwise \end{Bmatrix}$$

Where r is the distance from the region center. Thus, we increase the reliability of the color distribution when these boundary pixels belong to the background or get occluded.

The color distribution $p_y = \left\{p_y^{(u)}\right\}_{u=1..m}$ at location y is calculated as

$$p_y^{(u)} = f \sum_{i=1}^{I} k \left(\frac{||y - x_i||}{a}\right) \delta(h(x_i) - u)$$

Where I is the number of pixels in the region, δ is the Kronecker delta function, the parameter $a = \sqrt{H_x^2 + H_y^2}$ is used to adapt the size of the region, and the normalization factor

$$f = \frac{1}{\sum_{i=1}^{l} k\left(\frac{||y - x_i||}{a}\right)}$$

Ensures that $\sum_{u=1}^{m} p_y^{(u)} = 1$.

In tracking approach, the estimated state is updated at each time step by incorporating the new observations. Therefore, we need a similarity measure, which is based on color distributions. A popular measure between two distributions $p(u)$ and $q(u)$ is the Bhattacharyya coefficient.

$$\rho(p, q) = \int \sqrt{p(u) q(u) du}$$

Considering discrete densities such as color histograms $p = \{p^{(u)}\}_{u=1..m}$ and $q = \{q^{(u)}\}_{u=1..m}$, the coefficient is defined as

$$\rho(p,q) = \sum_{u=1}^{m} \sqrt{p^{(u)}q^{(u)}}$$

The larger ρ is, the more similar the distributions are. For two identical normalized histograms, we obtain $\rho = 1$, indicating a perfect match. As distance between two distributions we define the measure

$$d = \sqrt{1 - \rho[p,q]}$$

Which is called the Bhattacharyya distance.

To weight the sample set, the Bhattacharyya coefficient has to be computed between the target histogram and the histogram of the hypotheses. Each hypothetical region is specified by its state vector $s^{(n)}$. Both the target histogram q and the candidate histogram $p_{s^{(n)}}$ are calculated where the target is centered at the origin of the elliptic region.

As we want to favor samples whose color distributions are similar to the target model, small Bhattacharyya distances correspond to large weights:

$$\pi^{(n)} = \frac{1}{\sqrt{2\pi\sigma}}\, e^{-\frac{d^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi\sigma}}\, e^{-\frac{(1-\rho[p_{s^{(n)}},q])}{2\sigma^2}}$$

That are specified by a Gaussian with variance σ. During filtering, samples with high weight may be chosen several times, leading to identical copies, while others with relatively low weights may not be chosen at all. The samples are located around the maximum of the Bhattacharyya coefficient, which represents the best match to the target model.

5.5 RESULTS OF VISUAL OBJECT TRACKING

5.5.1 Visual object tracking of Helicopter with color histogram

This section is concerned with tracking a remote-controlled helicopter in a video sequence. The method uses a simple linear motion model for the centroid of the object, and a color histogram for the likelihood model, using Bhattacharya distance to compare histograms. The proposal distribution is obtained by sampling from the likelihood.

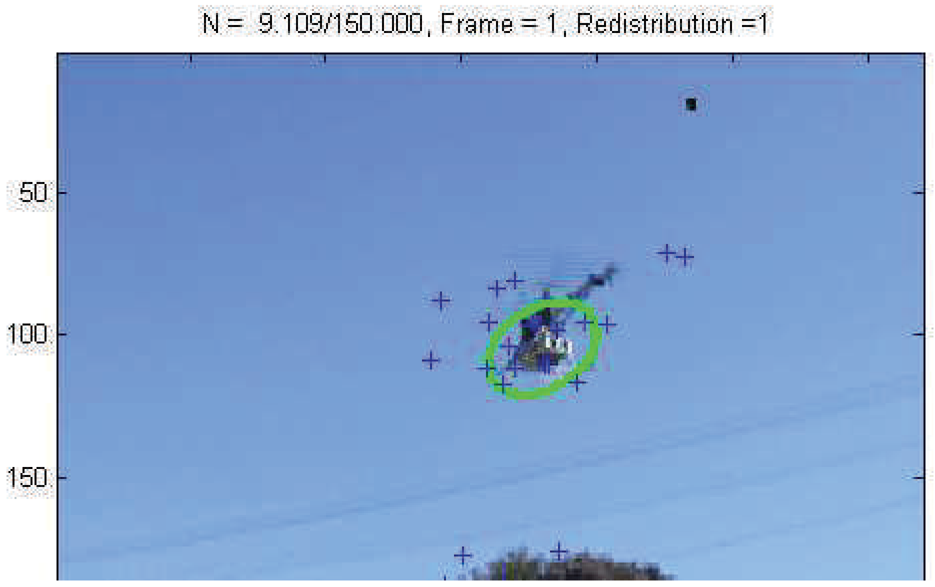


Figure 5.2 : Belief state at frame 1

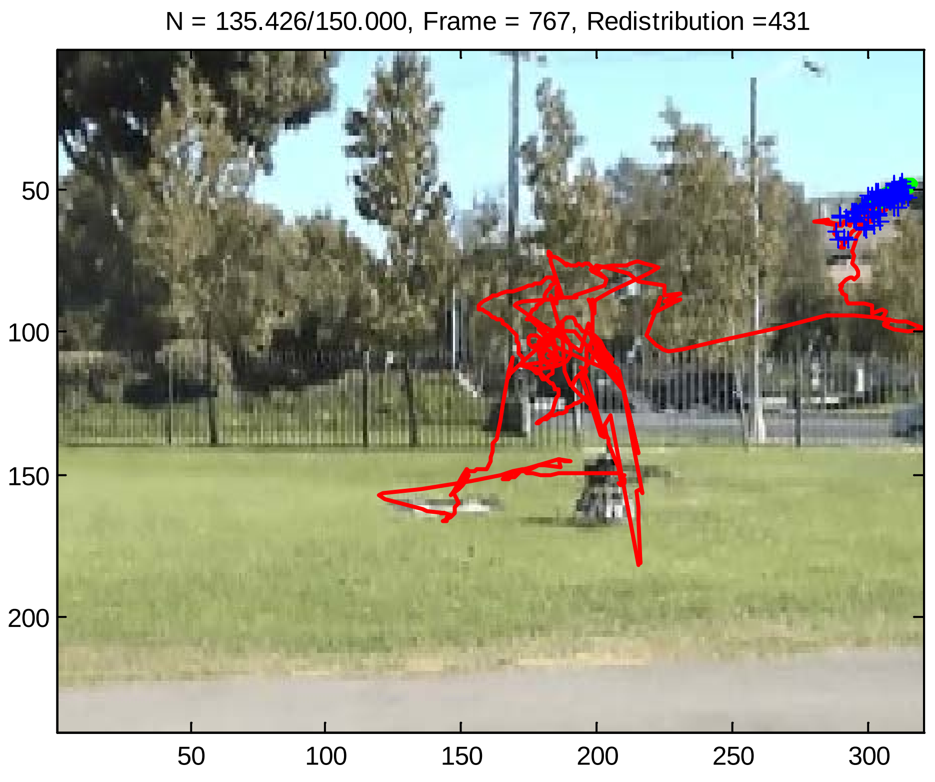


Figure 5.3 : Trajectory of the object over the frames with red line.

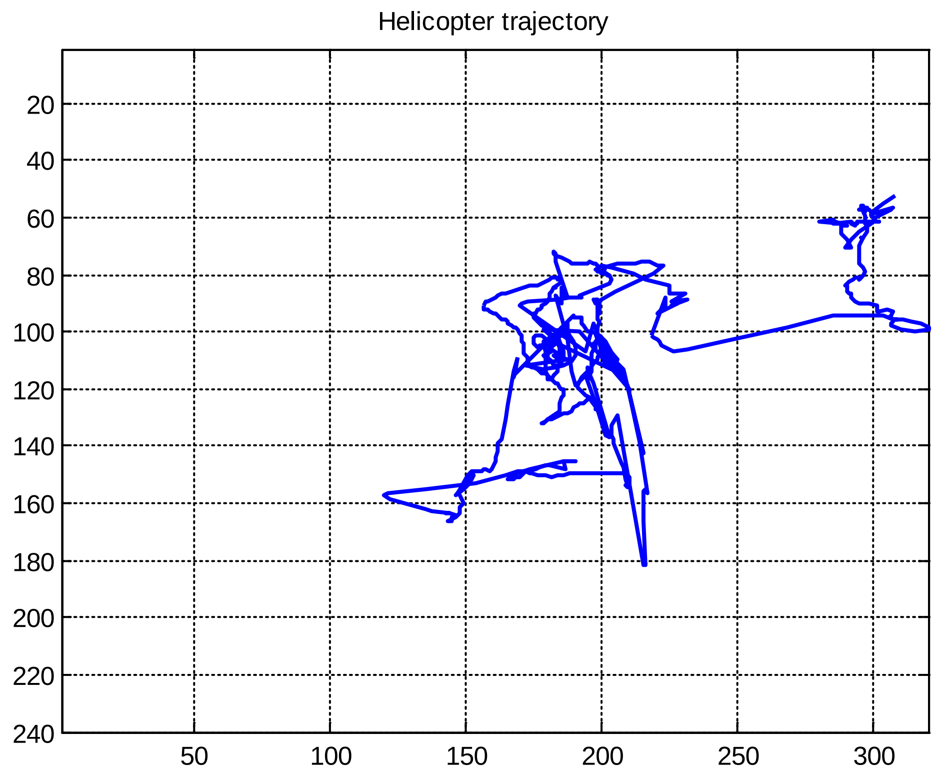


Figure 5.4 : Trajectory of the object over the frames with blue line

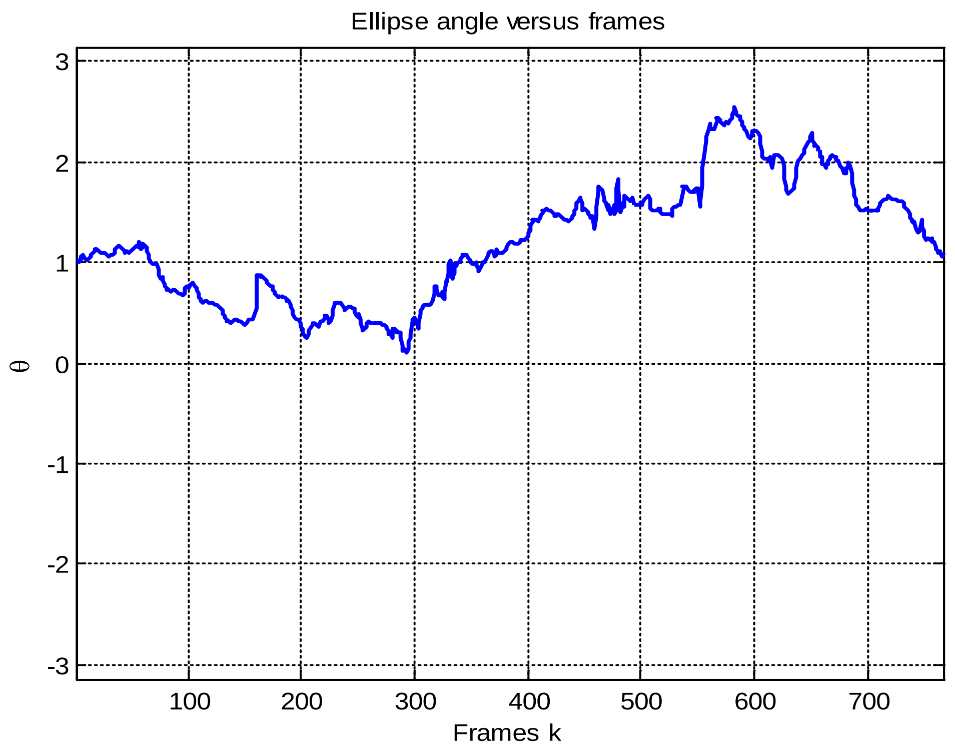


Figure 5.5 : Plot of ellipse angle over the frames

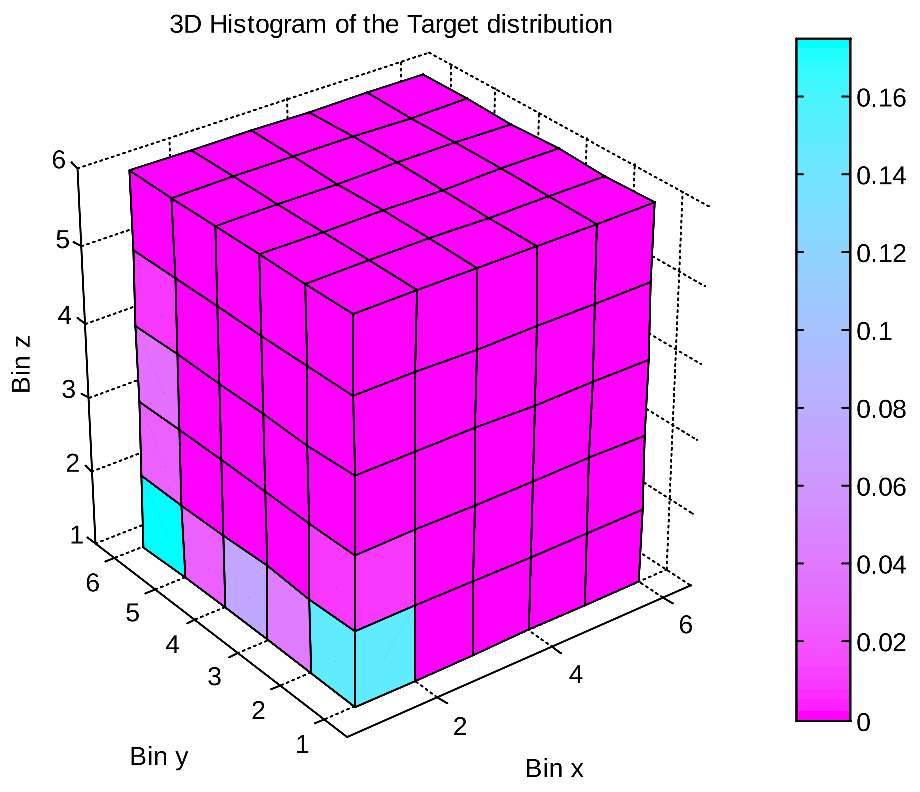


Figure 5.6 : 3D Histogram of the target distribution

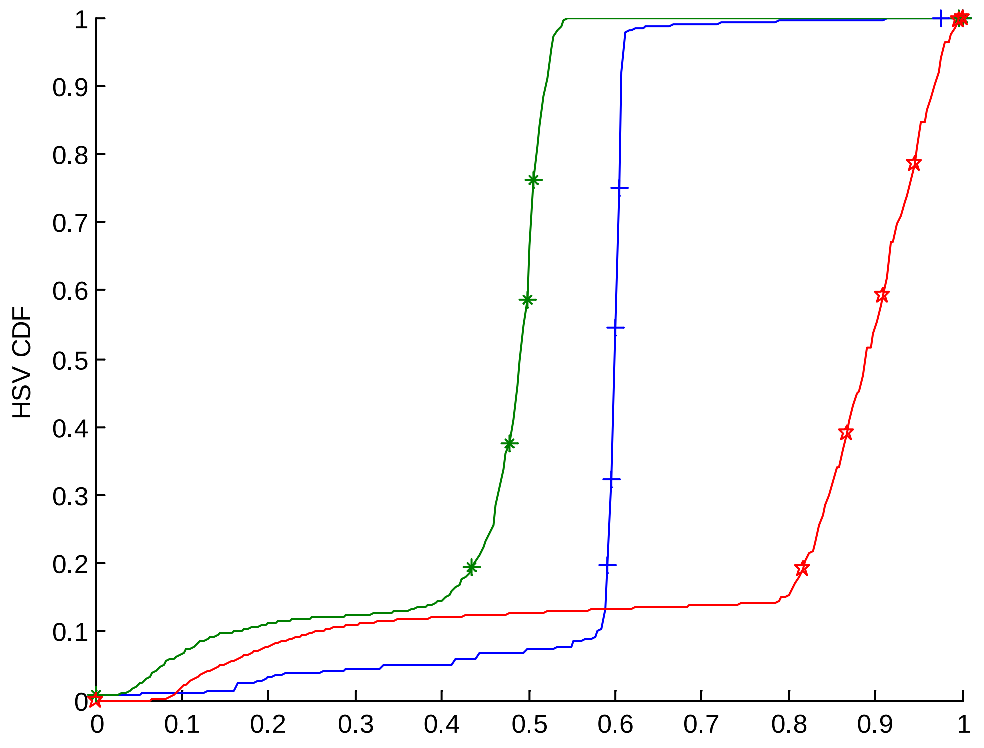


Figure 5.7 : HSV cumulative Density Function of the object

5.5.2 Visual object tracking of Moving car with color histogram

Object bounding ellipse

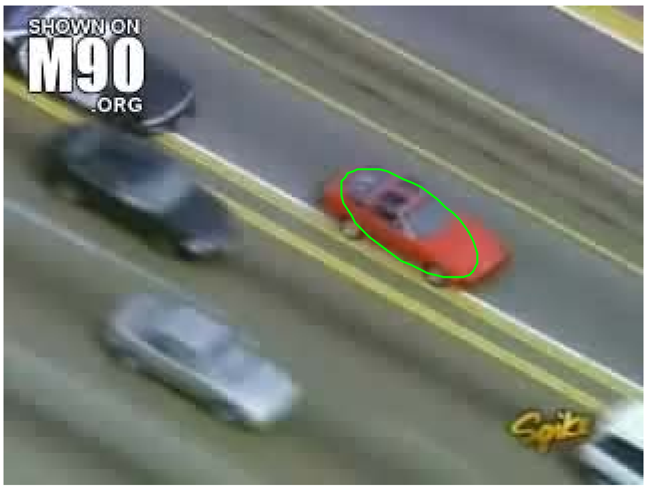


Figure 5.8 : Object Bounding ellipse for the moving red car

Prediction of object position

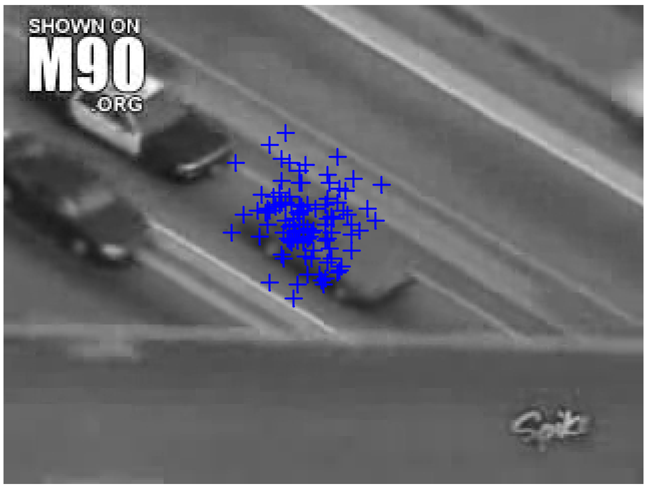


Figure 5.9 : Prediction of object position

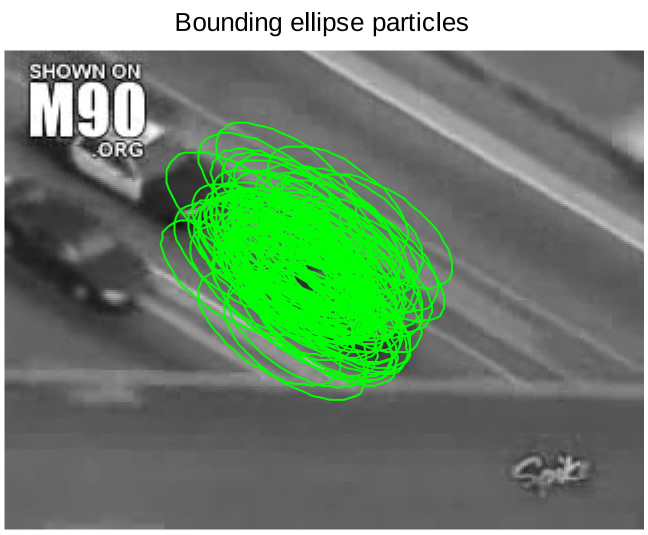


Figure 5.10 : Bounding ellipse particles for the object

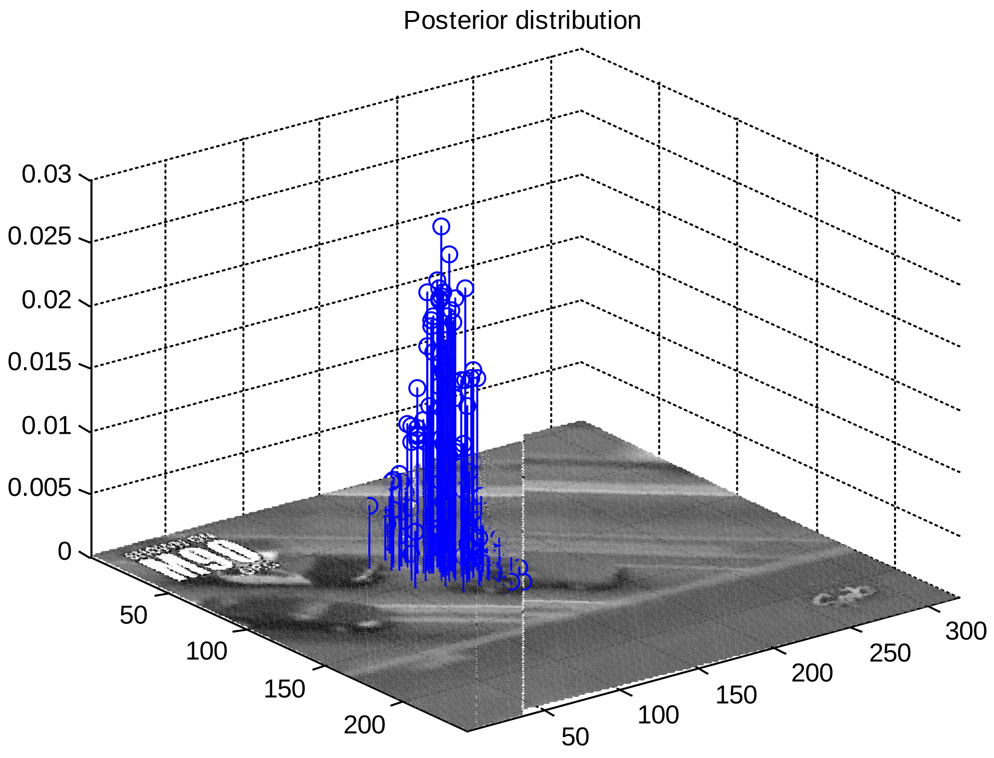


Figure 5.11 : Posterior distribution of the object over the frames

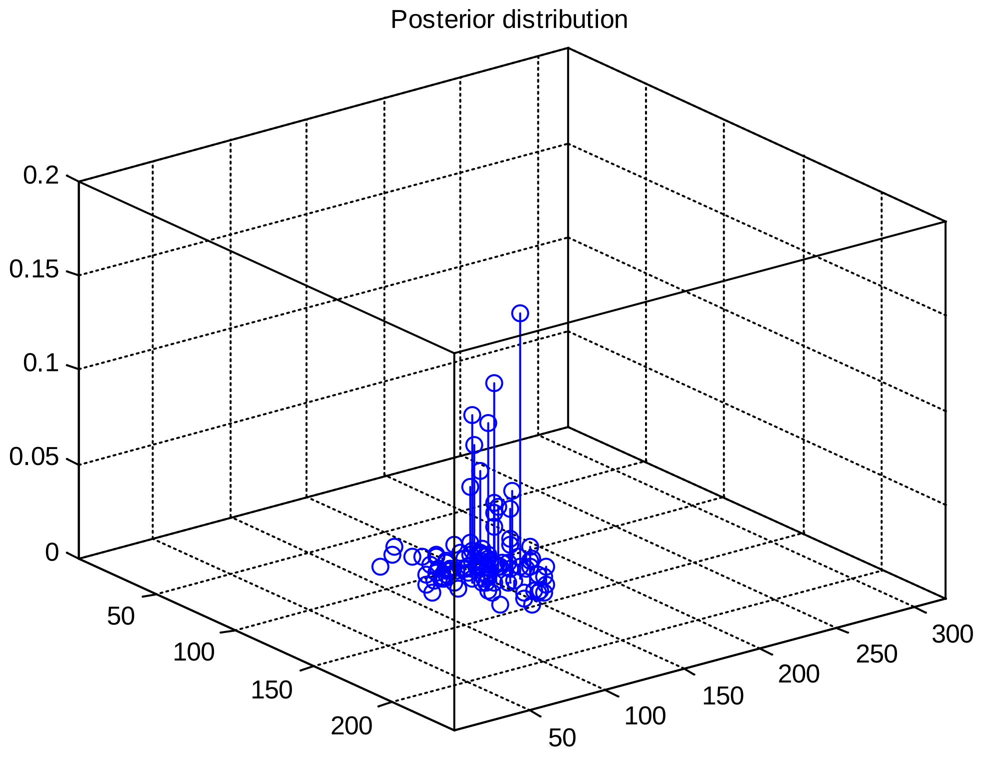


Figure 5.12 : 3D Posterior distribution of the object over the frames

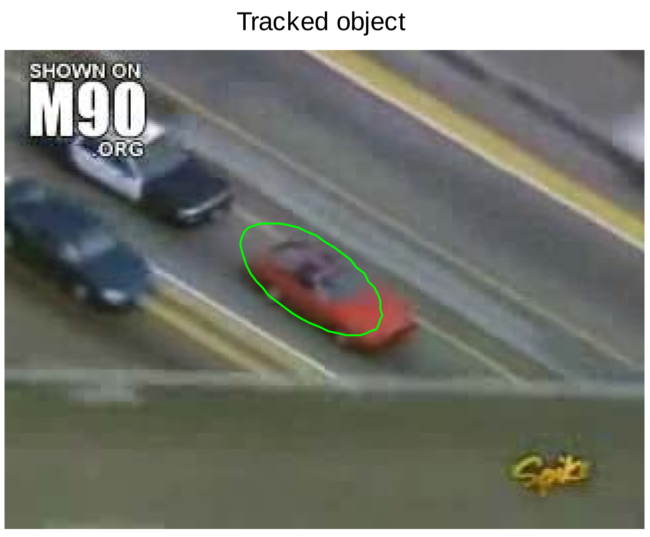


Figure 5.13 : Tracked object at the last frame

SIR resampling

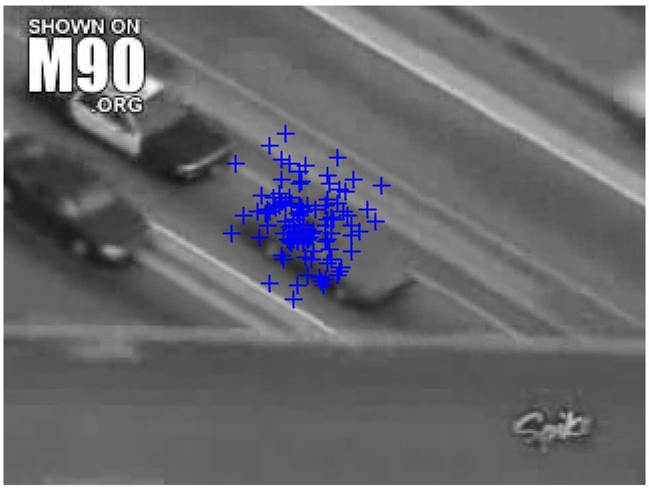


Figure 5.14 : SIR resampling of the object over the frames

5.5.3 Visual object tracking of Moving car with Intensity histogram

Object bounding ellipse

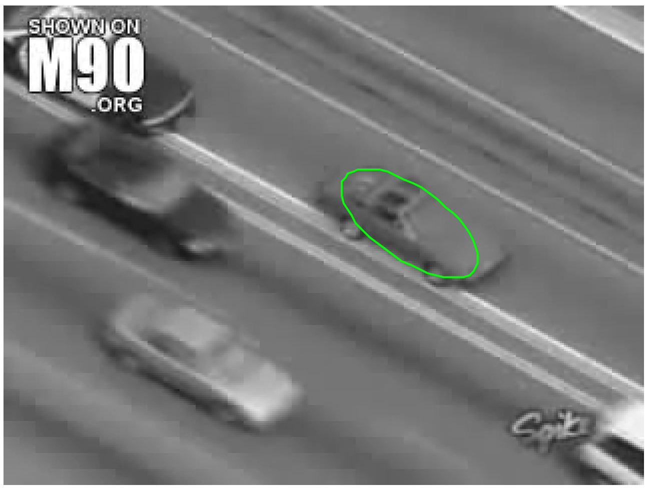


Figure 5.15 : Object Bounding ellipse for the moving car with intensity histogram

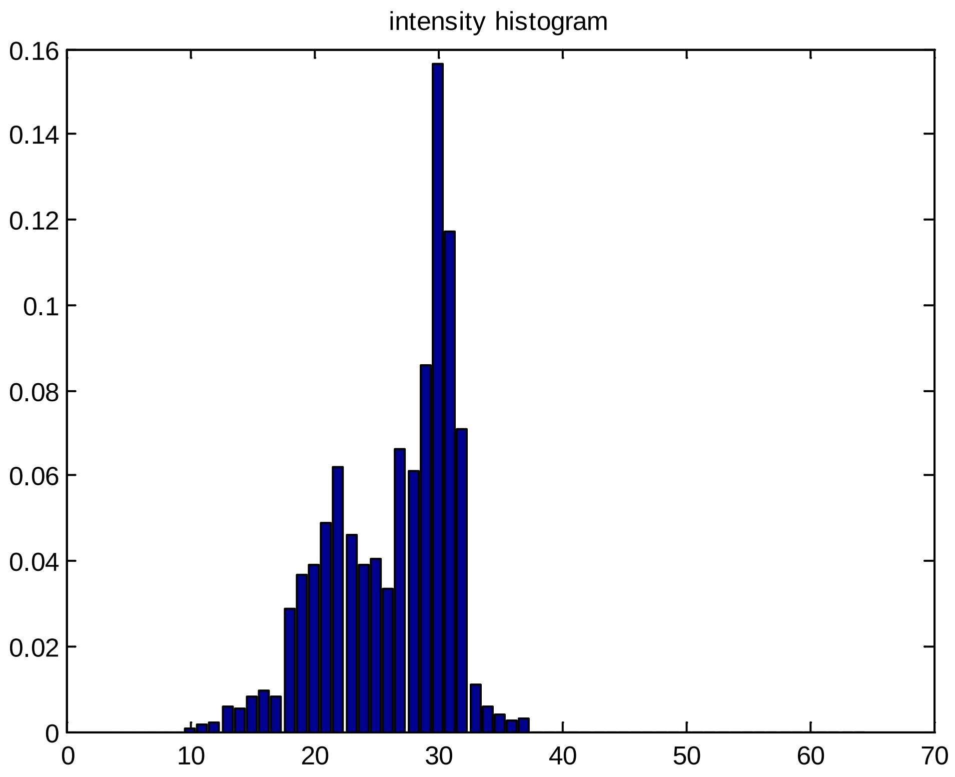


Figure 5.16 : Intensity histogram of the moving car

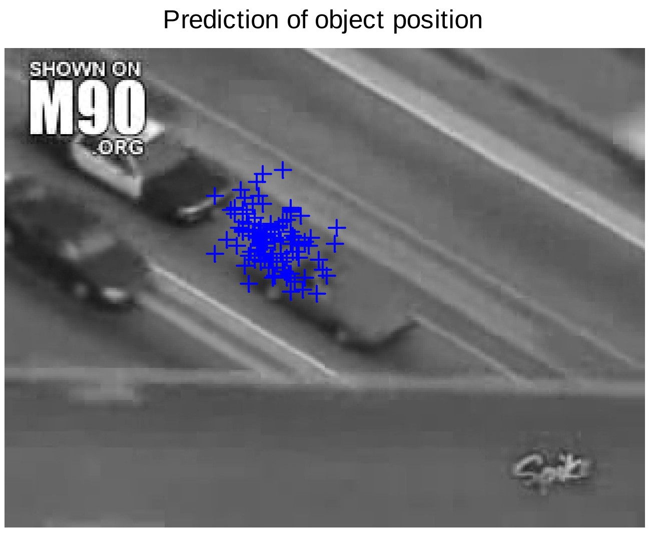


Figure 5.17 : Prediction of object position with intensity histogram

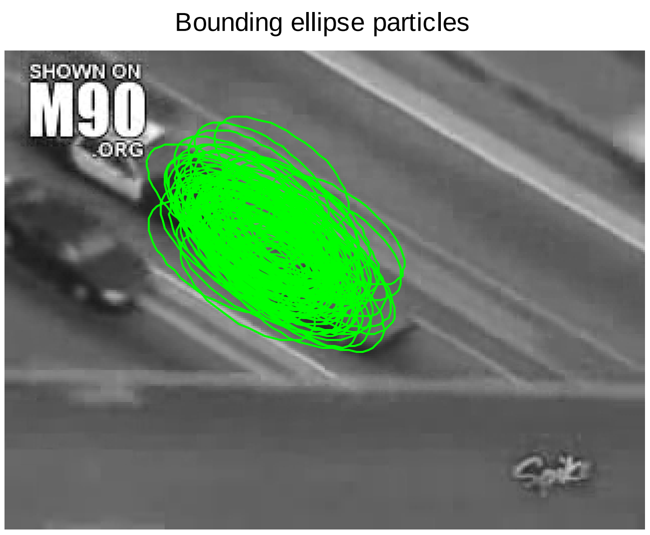


Figure 5.18 : Bounding ellipse particles with intensity histogram

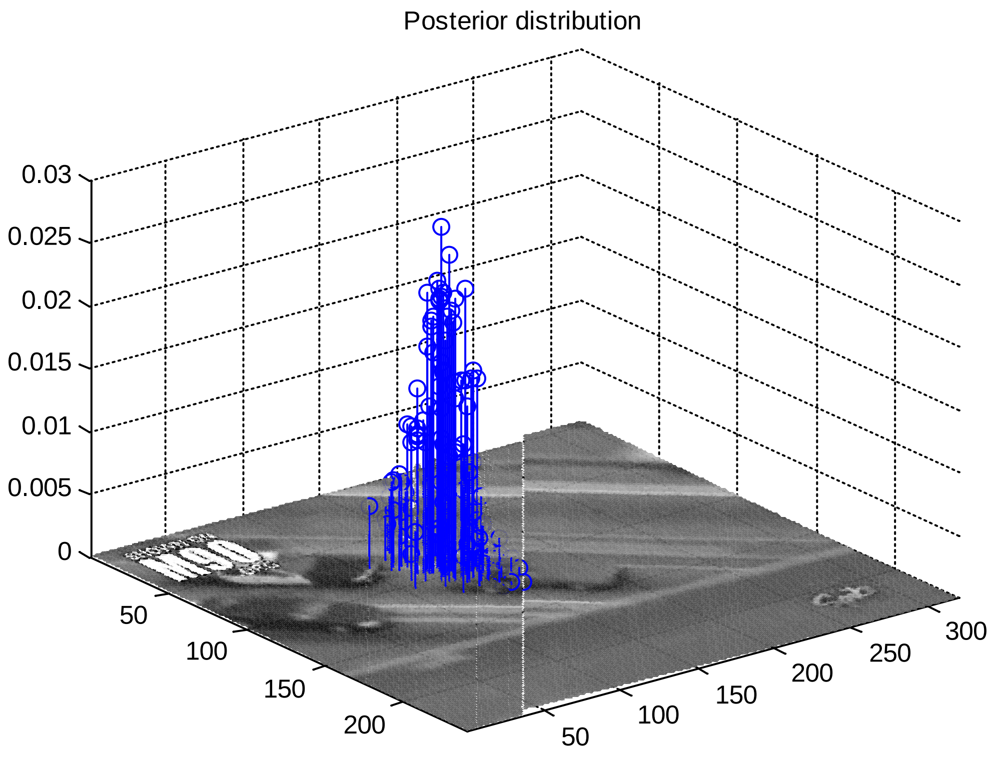


Figure 5.19 : Posterior distribution of moving car with intensity histogram

Tracked object

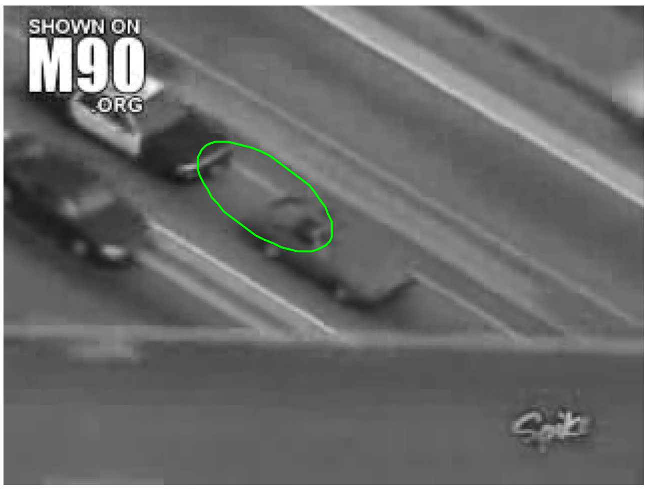


Figure 5.20 : Tracked object with intensity histogram

SIR resampling

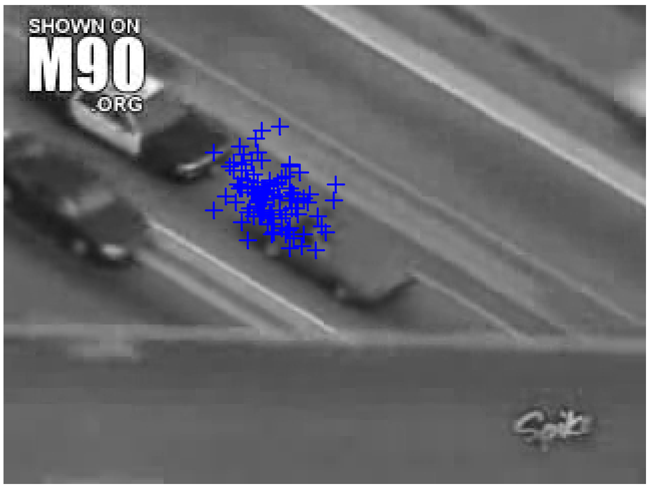


Figure 5.21 : SIR resampling of the moving car with intensity histogram

CONCLUSION

In this dissertation, rigorous mathematical foundations of Particle filters and its types have been simulated using MATLAB and the same is sufficient to have the clear and analytical insights to this highly mathematical topic.

Having built on the foundation, we showed in MATLAB simulations that Rao-Blackwellized Particle filter in conjuction with Jump markov linear system outperformed the traditional Particle filters for the tracking of Maneuvering target in the presence of noise.

Further, we showed in MATLAB simulations that color and intensity histogram based tracking handled the fast moving objects efficiently and successfully in the two different videos under different appearance changes.

Both methods are able to keep track for a fairly long time, despite the presence of clutter. The simplest way to improve the performance is to use more particles. Amongst the results, posterior distribution of a moving object gives clear understanding about the changes in number of particles and corresponding weights over the frames.

FUTURE SCOPE

This dissertation dealt with the tracking of single object in the video whereas tracking a varying number of non-rigid objects has two major difficulties. First, the observation models and target distributions can be highly non-linear and non-Gaussian. Second, the presence of a large, varying number of objects creates complex interactions with overlap and ambiguities. To surmount these difficulties, we can go for a Particle filter based system that is capable of learning, detecting and tracking the objects of interest. Various classifiers can be merged with the particle filters to make the system work more efficiently.

REFERENCES

[1] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, “Novel approach to nonlinear/non-Gaussian Bayesian state estimation”, *IEEE Proc.-F,* 140, 107, 1993.

[2] J. Kotecha and P. M. Djuric, “Gaussian particle filtering”, *IEEE Trans. On Signal Processing,* 51, 2592, 2003.

[3] J. Kotecha and P. M. Djuric, “Gaussian sum particle filtering”, *IEEE Trans. On Signal Processing,* 51, 2602, 2003.

[4] P. M. Djuric, M. F. Bugallo, and J. Miguez, “Density assisted particle filters for state and parameter estimation”, *Proc. Of the IEEE international conference on Acoustic, Speech, and Signal Processing,* Montreal, Canada, 2004.

[5] R. Chen and J. S. Liu, “Mixture Kalman filters”, *Journal of the Royal Statistical Society,* 62, 493, 2000.

[6] A. Doucet, S. J. Godsill, and C. Andrieu, “On sequential Monte Carlo sampling methods for Bayesian filtering”, *Statistics and Computing,* 10, 197, 2000.

[7] M. R. Morelande and S. Challa, “Manoeuvering target tracking in clutter using particle filters”, *IEEE Trans. On Aerospace and Electronic Systems,* 41, 252, 2005.

[8] D. Angelova and L. Mihaylova, “Joint target tracking and classification with particle filtering and mixture Kalman filtering using kinematic radar information”, *Digital Signal Processing,* 16, 180, 2006.

[9] R. Karlsson and F. Gustafsson, “Complexity analysis of the marginalized particle filter”, *IEEE Trans. on Signal Processing,* 53, 4408, 2005.

[10] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “ A tutorial on particle filters for online nonlinear/nonGaussian Bayesian tracking”, *IEEE Trans. on Signal Processing,* 50, 174, 2002.

[11] Hanna Goszczynska, “Object Tracking”, *Intechweb.com*, Edition 2011.

[12] Katja Nummiaro, esther Koller-Meier, Luc Van Gool, “Ad adaptive color-based particle filter”, *Image and Vision Computing, Elsevier*, 21, pp. 99-110, 2003.

[13] Arnaud Doucet, Neil J Gordon, Vikram Krishnamurthy, “Particle filters for state estimation of Jump markov linear systems”, *IEEE Transaction on Signal Processing,* Vol 49, No 3, pp. 613-624, March 2001.

[14] Emilio Maggio and Andrea Cavallaro, “Video Tracking”, John Wiley and Sons, Edition 2011.

[15] Alper Yilmaz, Omar Javed, Mubarak Shah, “Objject tracking : A survey”,, ACM Computing Surveys, Vol 38, No 4, Article 15, December 2006.

[16] O. Cappe, S. J. Godsill, and E. Moulines, “An overview of existing methods and recent advances in sequential Monte carlo”, *Proc. of the IEEE,* 95, 899, 2007