Birth Density Modeling in Multi-target Tracking Using the Gaussian Mixture PHD Filter

Rongrong CHEN

Min ZHU

This thesis is presented as part of Degree of
Master of Science in Electrical Engineering

Blekinge Institute of Technology
August 2008

Blekinge Institute of Technology

School of Engineering

Department of Applied Signal Processing

Supervisor: **Dr. Anders Johansson**

Examiner: **Dr. Anders Johansson**

ABSTRACT

A recently established method for multi-target tracking which both estimates the time-varying number of targets and their states from a sequence of observation sets in the presence of data association uncertainty, detection uncertainty, noise and false alarms is the probability hypothesis density (PHD) recursion. The approach involves modeling the respective collections of targets and measurements as random finite sets and to propagate the posterior intensity, which is a first order statistic of the random finite set of targets, in time. A closed form solution to the PHD filter recursion for multi-target tracking is provided by the Gaussian Mixture Probability Hypothesis Density filter (GM-PHD filter), whose posterior intensity function is estimated by a sum of weighted Gaussian components, including means, weights and covariances that can be propagated analytically in time.

Besides the GM-PHD filter algorithm implementation, choose the probability density function for representing target births in GM-PHD recursion and true target trajectory generation to get best tracking performance is a challenge and is the purpose of this thesis work. One reference to judge the performance of the algorithm is the target detection time, as given in this thesis.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

1.1 Problem Statement

Target tracking is an important component of a surveillance, guidance, or obstacle avoidance system. The number, movement and position of all targets have to be considered in a target tracking problem. Since there are missing targets that are not detected by the sensors and clutter that causes spurious measurements, the observations set at each time step includes both observations generated by targets and clutter detections. The objective of multi-target tracking is to estimate both the number of targets and their states from a sequence of imperfect observations.

There are several traditional multi-target tracking formulations, most of which focus on the association between measurements and targets, making up a heavy computational load. Some of them concern the propagation of association hypotheses in time, e.g. Multiple Hypotheses Tracking (MHT) [1]-[3]; others use observations weighted by their association probabilities, e.g. the Joint Probabilistic Data Association Filter (JPDAF) [4], [5], the Probabilistic Multiple Hypotheses Tracking (PMHT) [6], and the multi-target particle filter [7], [8].

One approach to solve the multi-target tracking problem which ignores explicit association between measurements and targets is using the Radom Finite Set (RFS) framework, which can be formulated in a Bayesian filtering problem by propagating the posterior intensity in time [9]-[11]. In the RFS formulation, both the collection of individual targets and the collection of individual observations are modelled as RFSs, allowing the problem of dynamically estimating multi-targets in the presence of clutters and association uncertainty to be cast in a Bayesian filtering framework [9]-[13].

A new established method for multi target tracking which both estimates the time-varying number of targets and their states from a sequence of observation sets in the presence of data association uncertainty, detection uncertainty, noise and false alarms is

the Probability Hypothesis Density (PHD) recursion. Since the method of Bayesian filtering is still computationally intensive due to the combinatorial nature of the densities, in order to alleviate the computational intractability in the multi-target Bayes filter, an approximation is developed, which is the PHD filter. Instead of propagating the multi-target posterior intensity in time, the PHD filter propagates the first-order statistical moment, or posterior intensity of the posterior multi-target state [9]. A closed form solution to the PHD filter recursion for multi-target tracking is provided by the Gaussian Mixture Probability Hypothesis Density filter (GM-PHD filter), whose posterior intensity function is estimated by a sum of weighted Gaussian components, including means, weights and covariances that can be propagated analytically in time. Particularly, the means and covariances are propagated by the Kalman filter [14].

The approach involves modeling the respective collections of targets and measurements as random finite sets and to propagate the posterior intensity, which is a first order statistic of the random finite set of targets, in time. One of the challenges in tuning this type of filters is to choose the probability density function for representing target births. This aspect of the GM-PHD filter requires further investigation and is the focus of this Masters Thesis project.

This work is focused on evaluating target discovery using the GM-PHD filter for different target birth densities and birth density models.

1.2. Scope of Thesis Work

In this thesis, the linear GM-PHD filter is implemented, and a comparison of the target detection results when using different distributions for true target trajectories and the prediction of birth targets in the GM-PHD filter recursion is made. The result of the comparison is represented in a table, and includes a statistical analysis to judge whether the modeling approach works well in order to form a reference for choosing probability

density function.

The case of nonlinear Gaussian models is not considered in our thesis work because the approaches for modeling target births are almost the same in both linear and nonlinear cases.

1.3. Division of Work

This thesis work is cooperated by Rongrong Chen and Min Zhu. Rongrong Chen is responsible for the codes of the GM-PHD filter and Min Zhu is responsible for collecting data and statistics. Constructing an evaluation table for different target births models and the conclusion of the thesis work is done by both students.

1.4. Outline of the Thesis

The structure of paper is as follows. Chapter 1 introduces some basic knowledge about target tracking and state the problem in this thesis. Chapter 2 presents a review of single-target Bayesian filtering, the Radom Finite Set (RFS) formulation of Multi-target filtering, the Probability Hypothesis Density (PHD) filter and the Kalman filter, and also discusses some related work of the thesis. Chapter 3 presents the algorithms of GM-PHD filter. Chapter 4 repents the implementation and testing of the PHD Filter algorithm. Data analysis is discussed in Chapter 5 and the results and conclusion are summarized in Chapter6. Finally, possible future research directions are given in Chapter7.

CHAPTER 2 BACKGROUND AND RELATED WORK

2.1. Background

2.1.1. Single-target Filtering

Normally in dynamic state estimation problems, the state is assumed to follow a Markov process on the state space $\mathcal{X} \subseteq \mathbb{R}^{n_x}$, and the transition density is $f_{k|k-1}(\cdot|\cdot)$, i.e. from a state x_{k-1} at time $k-1$ to state x_k at time k, the transition density is

$$f_{k|k-1}(x_k|x_{k-1}). \tag{1}$$

This Markov process is observed in the observation space $\mathcal{Z} \subseteq \mathbb{R}^{n_z}$ in part, modeling by the likelihood function $g_k(\cdot|\cdot)$, i.e. given a state x_k at time k, the probability density of receiving the observation $z_k \in Z$ is

$$g_k(z_k|x_k). \tag{2}$$

The posterior density or so called filtering density at time k is

$$p_k(x_k|z_{1:k}), \tag{3}$$

which is the probability density of the state x_k at time k, given all observations $z_{1:k} = (z_1, \cdots, z_k)$ up to time k. The posterior density $p_k(\cdot|z_{1:k})$ encapsulates all information about the state at time k.

Using the Bayes Recursion, the posterior density at time k can be calculated:

$$p_{k|k-1}(x_k|z_{1:k}) = \int f_{k|k-1}(x_k|x) p_{k-1}(x|z_{1:k-1})\, dx, \tag{4}$$

$$p_k(x|z_{1:k}) = \frac{g_k(z_k|x_k)\, p_{k|k-1}(x_k|z_{1:k-1})}{\int g_k(z_k|x)\, p_{k|k-1}(x|z_{1:k-1})\, dx}, \tag{5}$$

where $p_{k|k-1}(\cdot|z_{1:k-1})$ is the multi-target predicted density, the initial density is $p_0(\cdot)$.

2.1.2. Random Finite Sets (RFSs)

Now we consider the multi-target case. Assume that at time k, the number of targets is $M(k)$ and the target states are $x_k^i \in \mathcal{X}$, $i = 1, \cdots, M(k)$; the number of received measurements is $N(k)$ and the observations are $z_k^j \in \mathcal{Z}$, $j = 1, \cdots, N(k)$. The target states and measurements at time k can be represented as random finite sets (RFSs):

$$X_k = \left\{ x_k^i : i = 1, \cdots, M(k) \right\} \in \mathcal{F}(\mathcal{X}), \tag{6}$$

$$Z_k = \left\{ z_k^j : j = 1, \cdots, N(k) \right\} \in \mathcal{F}(\mathcal{Z}), \tag{7}$$

where $\mathcal{F}(\mathcal{X})$ and $\mathcal{F}(\mathcal{Z})$ represent the respective collections of all finite subsets of $\mathcal{X}$ and $\mathcal{Z}$, X_k and Z_k are treated as multi-target state and multi-target observation respectively. Then the multi-target tracking problem can be posed as a multi-target filtering problem, specified to Bayesian filtering problem with state space $\mathcal{F}(\mathcal{X})$ and observation space $\mathcal{F}(\mathcal{Z})$. An RFS is simply a finite-set-valued random variable which can be characterized by a discrete probability distribution and a family of joint probability densities [10], [15], [16]. The discrete distribution characterizes the cardinality of the RFS, while for a given cardinality, an approximation density characterizes the joint distribution of the elements of RFS.

The multi-target state at time k is given by

$$X_k = \left[\bigcup_{\zeta \in X_{k-1}} S_{k|k-1}(\zeta) \right] \cup \left[\bigcup_{\zeta \in X_{k-1}} B_{k|k-1}(\zeta) \right] \cup \Gamma_k, \tag{8}$$

where X_{k-1} is the multi-target state at time $k-1$, $S_{k|k-1}(\zeta)$ is the RFS of targets survived at time k that evolves from a previous state ζ, $B_{k|k-1}(\zeta)$ is the RFS of targets spawned at time k from a target with previous state ζ, and Γ_k is the RFS of spontaneous birth at time k. For simplicity, we can ignore target spawning; then the

multi-target state at time k becomes

$$X_k = \left[\bigcup_{\zeta \in X_{k-1}} S_{k|k-1}(\zeta) \right] \cup \Gamma_k . \tag{9}$$

In the same way, the multi-target measurements Z_k received by the sensor at time k is given by

$$Z_k = \left[\bigcup_{x \in X_k} \Theta_k(x) \right] \cup K_k , \tag{10}$$

where $\Theta_k(x)$ is the RFS of measurements generated by the single-target state at time k, and K_k is a RFS if false measurements, or clutters at time k.

Similar to single-target dynamic model, we have the multi-target transition density $f_{k|k-1}(\cdot|\cdot)$ and multi-target likelihood $g_k(\cdot|\cdot)$. From a state X_{k-1} at time $k-1$ to state X_k at time k, the transition density is

$$f_{k|k-1}(X_k | X_{k-1}) . \tag{11}$$

Given a state X_k at time k, the probability density of receiving the observation $Z_k \in \mathrm{Z}$ is

$$g_k(Z_k | X_k) . \tag{12}$$

The multi-target posterior density at time k is

$$p_k(X_k | Z_{1:k}), \tag{13}$$

which is the probability density of the state X_k at time k, given all observations $Z_{1:k} = (Z_1, \cdots, Z_k)$ up to time k.

Similar to the single-target case, the multi-target predicted density is

$$p_{k|k-1}(\cdot | Z_{1:k-1}) . \tag{14}$$

The optimal multi-target posterior Bayes filter propagates the multi-target posterior in

time according to the recursion

$$p_{k|k-1}\left(X_k\left|Z_{1:k}\right.\right)=\int f_{k|k-1}\left(X_k\left|X\right.\right)p_{k-1}\left(X\left|Z_{1:k-1}\right.\right)\mu_s dX\,, \tag{15}$$

$$p_k\left(X\left|Z_{1:k}\right.\right)=\frac{g_k\left(Z_k\left|X_k\right.\right)p_{k|k-1}\left(X_k\left|Z_{1:k-1}\right.\right)}{\int g_k\left(Z_k\left|X\right.\right)p_{k|k-1}\left(X\left|Z_{1:k-1}\right.\right)\mu_s dX}\,, \tag{16}$$

where μ_s is an appropriate reference measure on $\mathcal{F}\left(\mathcal{X}\right)$ [13], [17].

2.1.3. The Probability Hypothesis Density (PHD) Filter

To alleviate the computational intractability in the multi-target Bayes filter, an alternative approach is developed as PHD filter. The multi-target Bayes filter propagate the multi-target posterior density in time, while the PHD filter propagate a first-order statistical moment of the posterior multi-target state [9].

For a RFS X on $\mathcal{X}$ with a distribution P, its first moment is a non-negative function v on $\mathcal{X}$, called the Probability Hypothesis Density (PHD) [18], [19] or intensity function, with the property that for each region $S\subseteq\mathcal{X}$,

$$\int\left|X\cap S\right|P\left(dX\right)=\int_S v\left(x\right)dx\,. \tag{17}$$

That is, the integral of v over region S gives the expected number of elements of X that are in S. Hence, the total mass $\hat{N}=\int v\left(x\right)dx$ gives the expected number of elements of X. The intensity function is the first order moment of a RFS.

An important class of RFSs is Poisson RFSs, which are completely characterized by their intensity function. A RFS X is Poisson if the distribution of the cardinality of X is Poisson with the mean $\hat{N}=\int v\left(x\right)dx$, and for any finite cardinality, the elements x of X are independent and identically distributed (i.i.d.) with the probability density $v\left(\cdot\right)/\hat{N}$ [15], [16]. It is natural to model the clutters RFS [K_k in (10)] and the birth RFS [Γ_k in (9)] as Poisson RFSs for the multi-target problem.

Let v_k and $v_{k|k-1}$ denote the intensities associated with the multi-target posterior density p_k and multi-target predicted density $p_{k|k-1}$ respectively. Consider the following assumptions:

- Each target evolves and generates observations independently of one another.
- The birth RFS and the surviving RFSs are independent of each other.
- The clutters RFS is Poisson and independent of target-originated measurements.
- The predicted multi-target RFS governed by $p_{k|k-1}$ is Poisson.

The PHD recursion is given by

$$v_{k|k-1}(x) = \int p_{S,k}(\zeta) f_{k|k-1}(x|\zeta) v_{k-1}(\zeta)\, d\zeta + \gamma_k(x), \tag{18}$$

$$v_k(x) = \left[1 - p_{D,k}(x)\right] v_{k|k-1}(x) + \sum_{z \in Z_k} \frac{p_{D,k}(x) g_k(z|x) v_{k|k-1}(x)}{\kappa_k(z) + \int p_{D,k}(\xi) g_k(z|\xi) v_{k|k-1}(\xi)\, d\xi}, \tag{19}$$

where

$f_{k|k-1}(\cdot|\zeta)$ = single target transition density at time k given previous state ζ,

$p_{S,k}(\zeta)$ = probability of target existence at time k given previous state ζ,

$\gamma_k(\cdot)$ = intensity of the birth RFS Γ_k at time k,

Z_k = measurement set at time k,

$g_k(\cdot|x)$ = single target measurement likelihood at time k given current state x,

$p_{D,k}(x)$ = probability of target detection at time k given current state x,

$\kappa_k(\cdot)$ = intensity of clutters measurement at time k.

The focus in this thesis is on $\gamma_k(\cdot)$, i.e., the birth density of new targets. We can see that the PHD filter completely avoids the combinational computations arising from the unknown association of measurements with appropriate targets. The PHD recursion

requires much less computational power than the multi-target recursion operated on $\mathcal{F}(\mathcal{X})$ since the posterior intensity is a function on the single-target state space $\mathcal{X}$. However, the PHD recursion is generally intractable since implementation with numerical integration suffers from the 'curse of dimensionality' [14].

2.1.4. Kalman Filter

The Kalman filter assumes that the posterior density at every time step is Gaussian and hence exactly and completely characterized by two parameters, its mean and covariance.

The target states x_k and observation measurements z_k can be written as:

$$x_k = F_{k\text{-}1}x_{k\text{-}1} + v_{k\text{-}1} = F_{k\text{-}1}x_{k\text{-}1} + Q_{k\text{-}1}, \tag{20}$$

$$z_k = H_k x_k + w_k = H_k x_k + R_k, \tag{21}$$

where $F_{k\text{-}1}$ and H_k are known matrices defining the linear function of the dimension $n_x \times n_x$. Random sequences $v_{k\text{-}1}$ and w_k are mutually independent zero-mean white Gaussian, with covariances Q_{k-1} and R_k respectively. Note that the system and measurement matrices $F_{k\text{-}1}$ and H_k, as well as noise covariances Q_{k-1} and R_k, are allowed to be time-variant.

The Kalman filter algorithm derived from (4) and (5) can be viewed as the following recursive relationship:

$$p\left(x_{k-1}\middle|Z_{k-1}\right) = \mathcal{N}\left(x_{k-1}; \hat{x}_{k-1|k-1}, P_{k-1|k-1}\right), \tag{22}$$

$$p\left(x_k\middle|Z_{k-1}\right) = \mathcal{N}\left(x_{k-1}; \hat{x}_{k|k-1}, P_{k|k-1}\right), \tag{23}$$

$$p\left(x_k\middle|Z_k\right) = \mathcal{N}\left(x_k; \hat{x}_{k|k}, P_{k|k}\right), \tag{24}$$

where $\mathcal{N}(x; m, P)$ is a Gaussian density with argument x, mean m, and covariance P; that is

$$\mathcal{N}\left(x;m,P\right) \triangleq \left|2\pi P\right|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\left(x-m\right)^{T} P^{-1}\left(x-m\right)\right\}. \tag{25}$$

The appropriate means and covariances of Kalman filter are computed as follows:

$$\hat{x}_{k|k-1} = F_{k\text{-}1}\hat{x}_{k-1|k-1}, \tag{26}$$

$$P_{k|k-1} = Q_{k-1} + F_{k\text{-}1}P_{k-1|k-1}F_{k\text{-}1}^{T}, \tag{27}$$

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

$$P_{k|k} = P_{k|k-1} - K_{k}S_{k}K_{k\text{-}1}^{T}, \tag{29}$$

where

$$S_{k} = H_{k}P_{k|k-1}H_{k}^{T} + R_{k} \tag{30}$$

is the covariance of the innovation term $\upsilon_{k} = z_{k} - H_{k}\hat{x}_{k|k-1}$, and

$$K_{k} = P_{k|k-1}H_{k}^{T}S_{k}^{\text{-}1} \tag{31}$$

is the Kalman gain. Covariance update can be written as:

$$P_{k} = \left[I - K_{k}H_{k}\right]P_{k|k-1}, \tag{32}$$

where I is the identity matrix of dimension $n_{x} \times n_{x}$ [20].

2.2. Related Work and Comparison

The PHD filter was introduced as a multi-target detection and tracking approach in 2000 [9], [18], [21], [22]. In 2003, several sequential Monte Carlo (SMC) implementations of the PHD filter were independently proposed [23], [24], [12]. Later in 2005, a closed-form solution to the PHD recursion for linear Gaussian multi-target model was discovered by Ba-Ngu Vo and Wing-Kin Ma, which is called the Gaussian Mixture implementations [25]. Gaussian mixture implementations are much more efficient and restrictive than SMC approaches. Moreover, they obviate the need for clustering--an expensive step in the SMC implementation. Convergence results for the GM-PHD filter

were established in [26].

The GM-PHD filter has been applied to various problems. In [27], tracking in sonar images was demonstrated. Tracks were obtained using the technique proposed in [28]. The GM-PHD filter on video data was reported in [29]. [30] presents more applications of the GM-PHD filter.

According to the work of Ba-Ngu Vo and Wing-Kin Ma, this work repeats the procedure of GM-PHD filter. Based on the algorithm, different probability density functions, including Poisson, Uniform and both wide and narrow Gaussian distribution are used as both true target trajectories and target birth models in the GM-PHD recursion. This results in a reference for how well the GM-PHD filter works for different probability density functions. This investigation of the GM-PHD filter results in a set of suggestion to others researchers on how to generate true target trajectory and arrange targets births in the GM-PHD filter.

CHAPTER 3 THE PHD FILTER ALGORITHM

The design of the algorithm is based on the following propositions and mathematical formulas.

3.1. Linear Gaussian Multi-Target Model

The linear Gaussian multi-target model includes standard linear Gaussian assumptions for the transition and observation models of individual targets, as well as certain assumptions on the birth, death and detection of targets.

- Assume that each target follows a linear Gaussian dynamical model and the sensor has a linear Gaussian measurement model, i.e.

$$f_{k|k-1}\left(x|\zeta\right)=\mathcal{N}\left(x;F_{k-1}\zeta,Q_{k-1}\right), \tag{33}$$

$$g_k\left(z|x\right)=\mathcal{N}\left(z;H_k x,R_k\right), \tag{34}$$

where $\mathcal{N}\left(\cdot;m,P\right)$, m, P, F_{k-1}, Q_{k-1}, H_k and R_k is as mentioned in Section 2.1.4.

- Assume that the survival and detection probabilities are state independent, i.e.

$$p_{S,k}\left(x\right)=p_{S,k}, \tag{35}$$

$$p_{D,k}\left(x\right)=p_{D,k}. \tag{36}$$

- Assume that the intensity of the birth RFS is a Gaussian mixture of the form

$$\gamma_k\left(x\right)=\sum_{i=1}^{J_{\gamma,k}} w_{\gamma,k}^{(i)}\mathcal{N}\left(x;m_{\gamma,k}^{(i)},P_{\gamma,k}^{(i)}\right) \tag{37}$$

where $w_{\gamma,k}^{(i)}$, $m_{\gamma,k}^{(i)}$ and $P_{\gamma,k}^{(i)}$ are the weights, means and covariances of the mixture birth intensity.

3.2. The Gaussian Mixture PHD Recursion

For the linear Gaussian multi-target model, the following steps present a closed form solution to the PHD recursion. Suppose the three assumptions mentioned in Background, Section *C* hold.

Step 1: Initialization

At time $k = 0$, the initial intensity v_0 is defined as

$$v_0(x) = \sum_{i=1}^{J_0} w_0^{(i)} \mathcal{N}\left(x; m_0^{(i)}, P_0^{(i)}\right), \tag{38}$$

which is the sum of J_0 Gaussians, each Gaussian has mean state vector $m_0^{(i)}$, covariance $P_0^{(i)}$ and weight $w_0^{(i)}$.

Step 2: Prediction

Suppose at time k-1, the posterior intensity v_{k-1} is a Gaussian mixture of the form

$$v_{k-1}(x) = \sum_{i=1}^{J_{k-1}} w_{k-1}^{(i)} \mathcal{N}\left(x; m_{k-1}^{(i)}, P_{k-1}^{(i)}\right). \tag{39}$$

Then, the predicted intensity $v_{k|k-1}$ at time k is also a Gaussian mixture, and is given by

$$v_{k|k-1}(x) = v_{S,k|k-1}(x) + \gamma_k(x), \tag{40}$$

where $\gamma_k(x)$ is given by (37) and

$$v_{S,k|k-1}(x) = p_{S,k} \sum_{j=1}^{J_{k-1}} w_{k-1}^{(j)} \mathcal{N}\left(x; m_{S,k|k-1}^{(j)}, P_{S,k|k-1}^{(j)}\right), \tag{41}$$

$$m_{S,k|k-1}^{(j)} = F_{k-1} m_{k-1}^{(j)}, \tag{42}$$

$$P_{S,k|k-1}^{(j)} = Q_{k-1} + F_{k-1} P_{k-1}^{(j)} F_{k-1}^T. \tag{43}$$

Step 3: Update

Suppose at time k, the predicted intensity $v_{k|k-1}$ is a Gaussian mixture of the form

$$v_{k|k-1}(x) = \sum_{i=1}^{J_{k|k-1}} w_{k|k-1}^{(i)} \mathcal{N}\left(x; m_{k|k-1}^{(i)}, P_{k|k-1}^{(i)}\right). \tag{44}$$

Then, the posterior intensity at time k is also a Gaussian mixture, and is given by

$$v_k(x) = (1 - p_{D,k}) v_{k|k-1}(x) + \sum_{z \in Z_k} v_{D,k}(x; z), \tag{45}$$

where

$$v_{D,k}(x; z) = \sum_{j=1}^{J_{k|k-1}} w_k^{(j)}(z) \mathcal{N}\left(x; m_{k|k}^{(j)}(z), P_{k|k}^{(j)}\right), \tag{46}$$

$$w_k^{(j)}(z) = \frac{p_{D,k} w_{k|k-1}^{(j)} q_k^{(j)}(z)}{\kappa_k(z) + p_{D,k} \sum_{l=1}^{J_{k|k-1}} w_{k|k-1}^{(l)} q_k^{(l)}(z)}, \tag{47}$$

$$q_k^{(j)}(z) = \mathcal{N}\left(z; \eta_{k|k-1}^{(j)}, S_{k|k-1}^{(j)}\right), \tag{48}$$

$$\eta_{k|k-1}^{(j)} = H_k m_{k|k-1}^{(j)}, \tag{49}$$

$$S_{k|k-1}^{(j)} = H_k P_{k|k-1}^{(j)} H_k^T + R_k, \tag{50}$$

$$m_{k|k}^{(j)}(z) = m_{k|k-1}^{(j)} + K_k^{(j)}\left(z - \eta_{k|k-1}^{(j)}\right), \tag{51}$$

$$P_{k|k}^{(j)} = \left[I - K_k^{(j)} H_k\right] P_{k|k-1}^{(j)}, \tag{52}$$

$$K_k^{(j)} = P_{k|k-1}^{(j)} H_k^T \left[S_{k|k-1}^{(j)}\right]^{-1}. \tag{53}$$

Step 4: Pruning

The Gaussian components with low weights are limited by pruning. The intensity function can be represented as $v_k = \left\{w_k^{(i)}, m_k^{(i)}, P_k^{(i)}\right\}_{i=1}^{J_k}$, let weights $w_k^{(1)}, \cdots, w_k^{(Np)}$ be those that are below the truncation threshold τ, and let

$$\overline{v}_k := \frac{\sum_{l=1}^{J_k} w_k^{(l)}}{\sum_{j=Np+1}^{J_k} w_k^{(j)}} \sum_{j=Np+1}^{J_k} w_k^{(i)} \mathcal{N}\left(x; m_k^{(i)}, P_k^{(i)}\right). \tag{54}$$

The new weights is defined as

$$\overline{w}_k^{(i)} = w_k^{(i)} \frac{\sum_{l=1}^{J_k} w_k^{(l)}}{\sum_{j=Np+1}^{J_k} w_k^{(j)}}, \tag{55}$$

so the new intensity $\overline{v}_k$ is given by the set

$$\overline{v}_k = \left\{ \overline{w}_k^{(i)}, m_k^{(i)}, P_k^{(i)} \right\}_{i=Np+1}^{J_k}. \tag{56}$$

Step 5: Merging

This step is to merge the Gaussian components of the distance between their means falls within a merging threshold U. The procedure for merging the components is as follows:

Set $l = 0$, and $I = \left\{ i = 1, \cdots, J_k \,\middle|\, w_k^{(i)} > \tau \right\}$.

Repeat

$$l := l + 1. \tag{57}$$

$$j := \arg \max_{i \in I} w_k^{(i)}.$$

$$L := \left\{ i \in I \,\middle|\, \left(m_k^{(i)} - m_k^{(j)} \right)^T \left(P_k^{(i)} \right)^{-1} \left(m_k^{(i)} - m_k^{(j)} \right) \le U \right\}.$$

$$\tilde{w}_k^{(l)} = \sum_{i \in L} \overline{w}_k^{(i)}.$$

$$\tilde{m}_k^{(l)} = \frac{1}{\tilde{w}_k^{(l)}} \sum_{i \in L} \overline{w}_k^{(i)} m_k^{(i)}.$$

$$\tilde{P}_k^{(l)} = \frac{1}{\tilde{w}_k^{(l)}} \sum_{i \in L} \overline{w}_k^{(i)} \left(P_k^{(i)} + \left(\tilde{m}_k^{(l)} - m_k^{(i)} \right) \left(\tilde{m}_k^{(l)} - m_k^{(i)} \right)^T \right).$$

$$I := I \setminus L.$$

Until $I = \varnothing$.

If $l > J_{\max}$, then replace $\left\{ \tilde{w}_k^{(i)}, \tilde{m}_k^{(i)}, \tilde{P}_k^{(i)} \right\}_{i=1}^{l}$ by those of the $J_{\max}$ Gaussians with largest weights, where $J_{\max}$ is maximum allowable number of Gaussian terms.

Step 6: Target State Estimation

By taking the Gaussians which have previously defined to be a target with the weights above a given threshold, the target states can be determined.

The set of estimates is $X_k = \left\{ \tilde{m}_k^{(i)} \right\}$ where the corresponding $\tilde{w}_k^{(i)} > 0.5$.

CHAPTER 4 IMPLEMENTATION

4.1. Implementation

The Matlab programming language has been used to implement the algorithm. The source code of the Gaussian Mixture PHD filter program is shown in Appendix A.

Different programs are used to perform different functions. There are several subprograms connecting to the main program: *traj.m* is to produce the true target trajectories when it's moving; *observe.m* is to produce the observation of targets in each time sample; *pruning.m* corresponds to the step of pruning in the recursion of PHD filter; *sest.m* is to extract the multi-target state; *spos.m* is the function called by other programs while using different distributions for true target trajectories and target births. Then the main program *phd.m* is to call other programs to plot out the figures for target trajectories, state observations and PHD filter multi-target estimation.

The GM-PHD filter is controlled by a number of fixed parameters. These parameters are here determined empirically to achieve good performance of the tracking algorithm. A number of rules are deducted on how the parameters affect the targets detection.

4.1.1. Initial Settings

In the examples studied here, there are three targets. Notice that while comparing the detection result for different values for the same parameter, other parameters are fixed. In our testing program, the region for tracking the target is $[0,\ 1.5]\times[0,\ 1]$. Each target has survival probability $p_{S,k} = 0.9$ and is detected with probability $p_{D,k} = 0.99$. In order to make the parameters selection more clear, we used three examples of the same true target trajectory. These are represented as subprogram *traj3.m*, see Figures 1, 2 and 3.

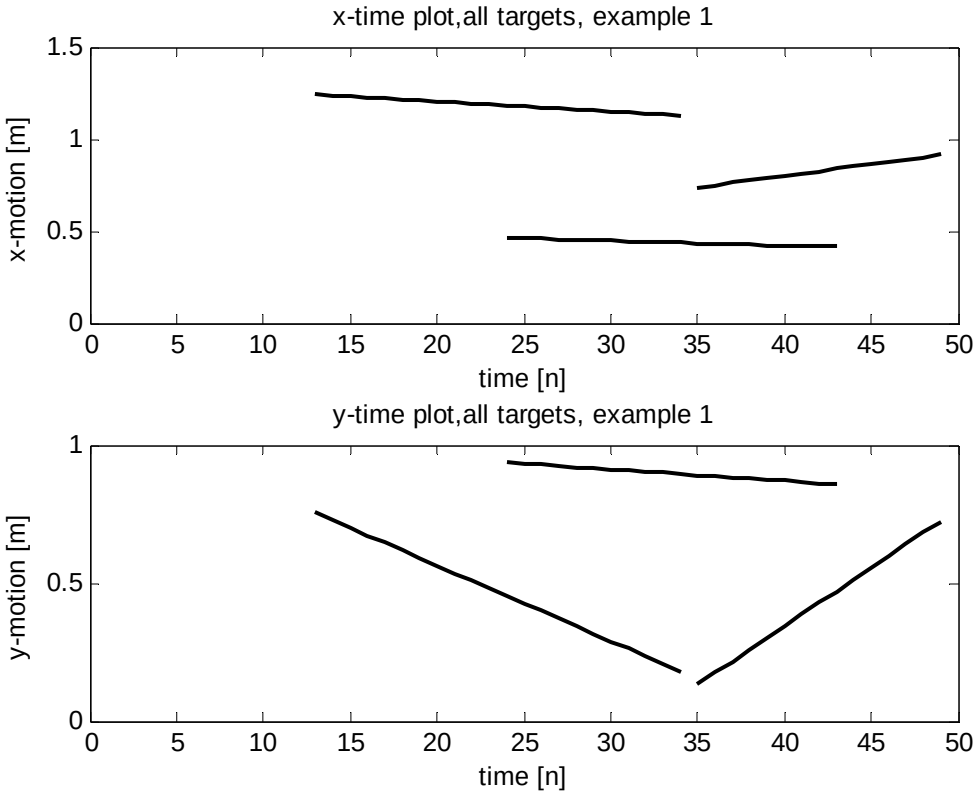

Figure 1. True trajectory of example 1 (Uniform distribution)

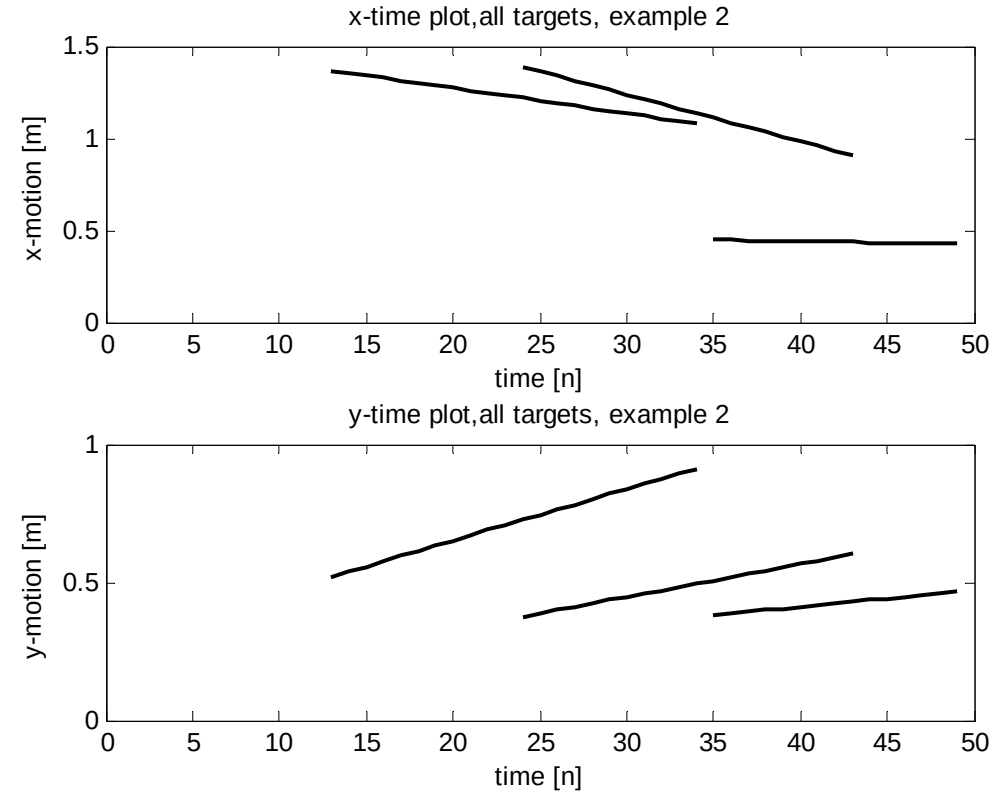

Figure 2. True trajectory of example 2 (Uniform distribution)

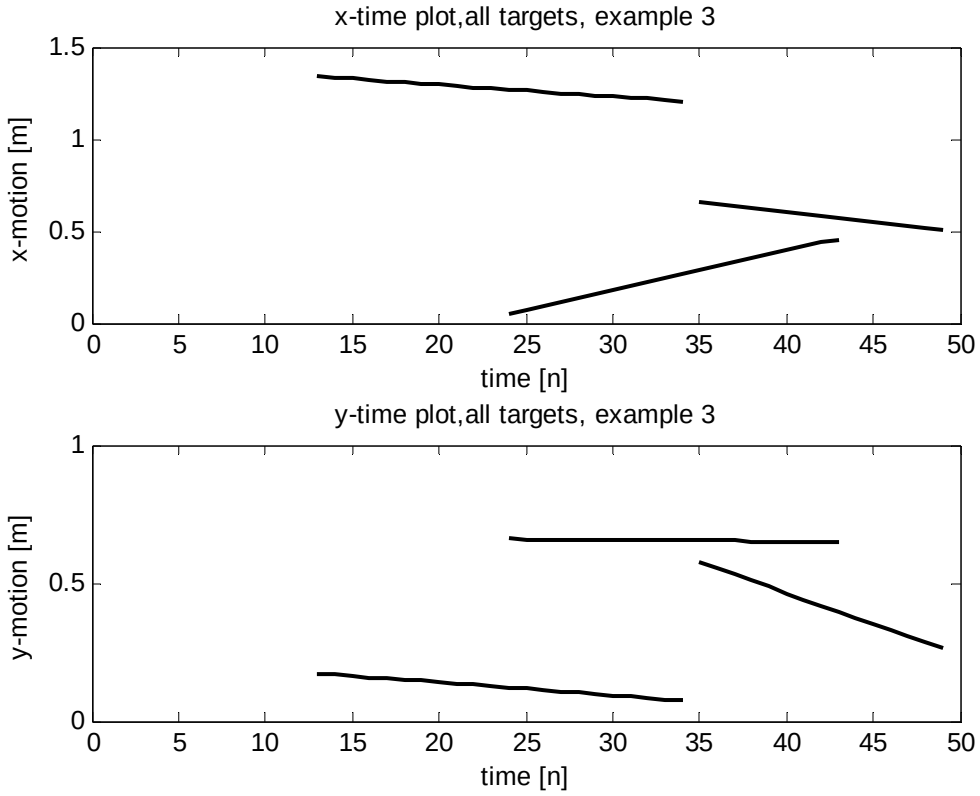

Figure 3. True trajectory of example 3 (Uniform distribution)

The distribution for the true target trajectory and the GM-PHD recursion are both uniform distributions because it makes the target births more random, matching to the true target tracking phenomenon. The selection of parameters is based on making the detection of targets correct and timely, with a minimum of clutter and short detection time.

4.1.2 Weight Threshold

In Step 6, mentioned in the Gaussian mixture PHD recursion, the threshold of the weight is $\tilde{w}_k^{(i)}$ = 0.5. During debugging of the program, it was found that the GM-PHD filter can detect the target at the first few iterations of the true trajectory, and then loose track of the target after one time sample. It may then catch the trajectory again later. The phenomenon is shown in Figure 4, where the target appears at the 24th time sample.

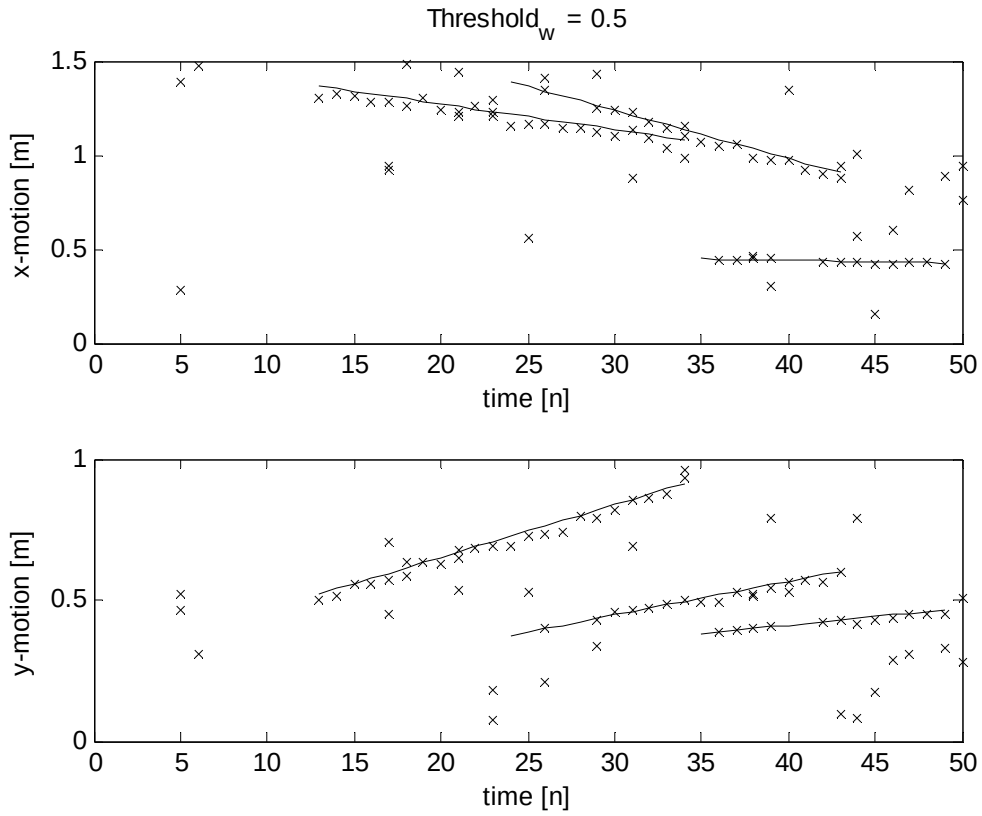

Figure 4. Detecting Performance of Threshold_weight = 0.5

True target positions (lines) and measurements (crosses), the same as follows.

Notice that while counting the detection time, only the points that are on the trajectories for both x and y motions at the same time are good estimates, and the detection time is the time from when the true trajectory appears to when estimates on the trajectory starts to appear consecutively for at least three time samples. That is, a target was considered to be detected successfully if the estimates of the true trajectory continue for at least three time samples.

In order to judge whether the first point is a correct estimate and enhance the accuracy of the conclusion, the weight threshold was reduced to 0.4 to check if there are other points following the first estimate that are removed only because its weight was below 0.5. In cases when this phenomenon still existed after reducing the threshold, the first point was considered to be noise. Though this may increase the number of spuriousities, it made the counting of detection time more precise.

4.1.3. Number of Observations per Time Sample

The number of observations per time sample M, which is defined as $N(k)$ in equation (7), cannot be less than three because there are three targets on moving. If $M < 3$, the filter cannot detect all the targets if they exist simultaneously. The program was executed for $M = 3$ and $M = 5$ twelve times respectively, and the number of spuriousities and detection times were recorded. The average values are presented in Table 1. Two outliers were discarded to ensure a more exact statistic average. The reason to use ten results is that we are not focusing on the details of the data but just to get a holistic view of how the program performs with different parameters. Ten times provides a sufficient statistic quantity. Figures 5-10 show six results for $M = 3$ and $M = 5$ for three true trajectory examples respectively.

The program was executed for $M = 8$ ten times and it was found that the number of spuriousities was too large to be useful, see Figure 11, 12 and 13.

TABLE I DATA STATISTIC OF M=3 AND M=5

No. of Observations (/ time sample)		***M=3***			***M=5***		
		Exp. 1	**Exp. 2**	**Exp. 3**	**Exp. 1**	**Exp. 2**	**Exp. 3**
No. of spuriousities/ Detection time	1	18/4	13/2	9/15	22/5	30/3	23/4
	2	12/12	~~16/18~~	10/13	~~33/15~~	19/5	~~32/18~~
	3	21/2	17/14	6/15	36/5	30/5	20/14
	4	17/12	18/14	16/13	~~33/12~~	36/3	20/3
	5	13/2	11/2	13/14	31/7	35/7	22/3
	6	~~9/14~~	9/2	20/14	32/11	25/0	22/3
	7	~~10/13~~	17/2	~~13/15~~	26/7	~~26/9~~	20/5
	8	15/5	~~12/19~~	~~12/17~~	29/1	~~24/11~~	27/8
	9	19/2	11/14	7/13	29/6	31/3	23/8
	10	7/12	12/2	16/13	19/3	27/7	24/4
	11	13/4	18/1	9/15	23/8	25/4	~~26/20~~
	12	14/12	19/15	11/15	18/1	21/7	18/1
Average of 10 times		14.9/5.7	14.5/6.8	11.7/14.2	26.5/5.4	27.9/4.4	21.9/5.3

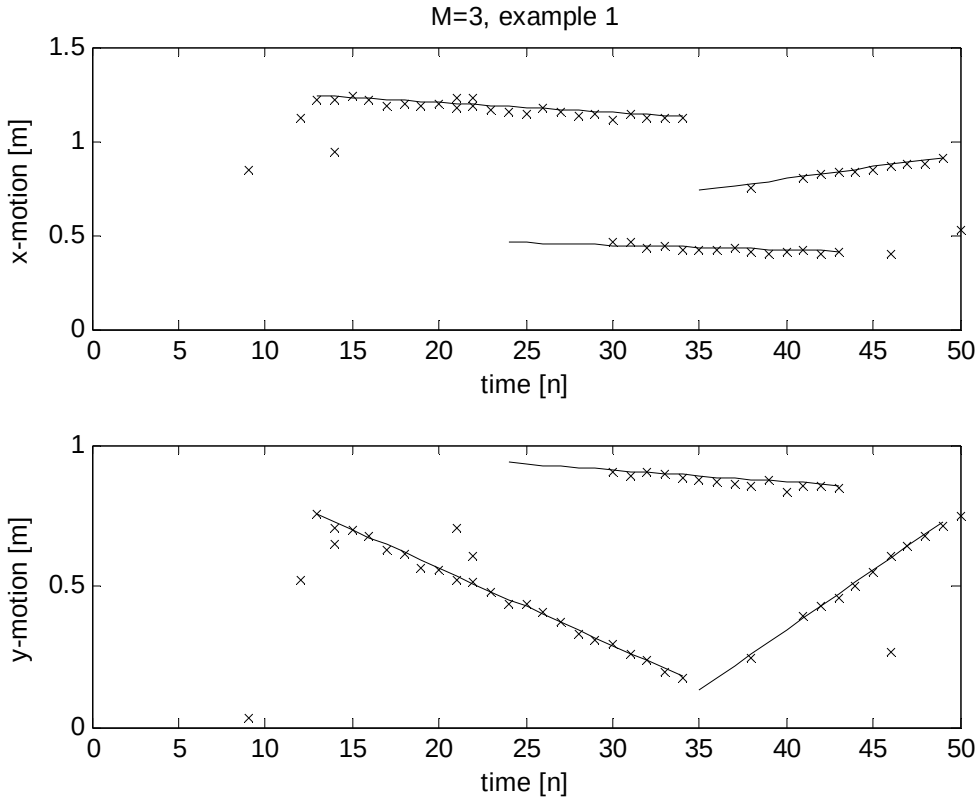

Figure 5. Detecting Performance of M = 3 (Example 1, Group 10)

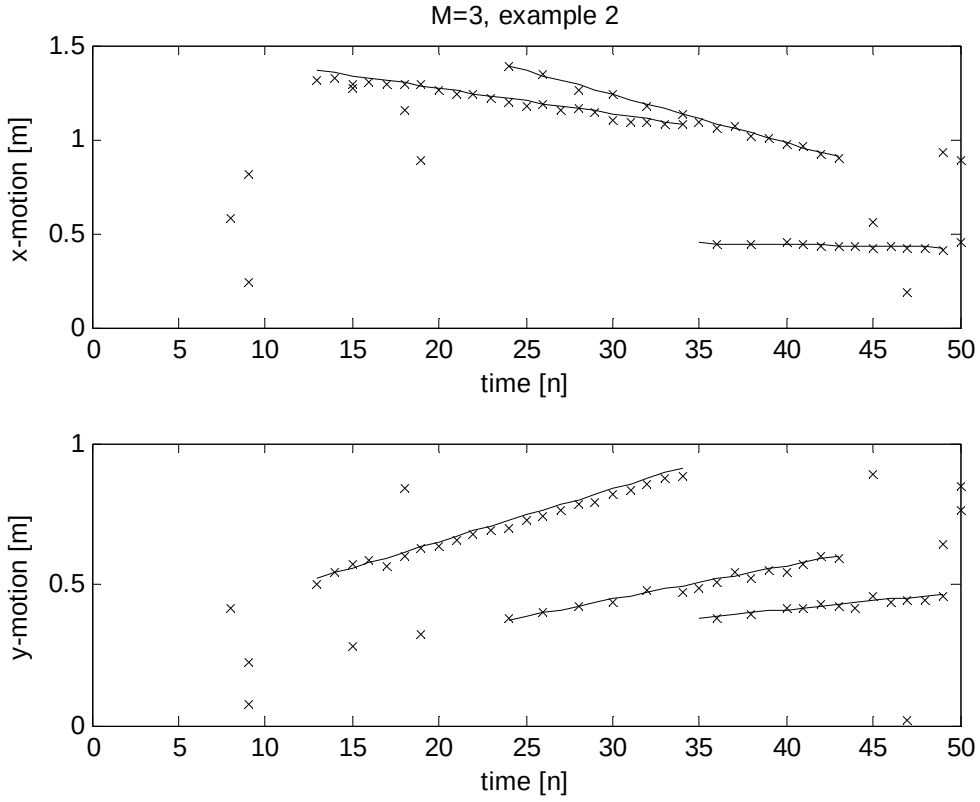

Figure 6. Detecting Performance of M = 3 (Example 2, Group 9)

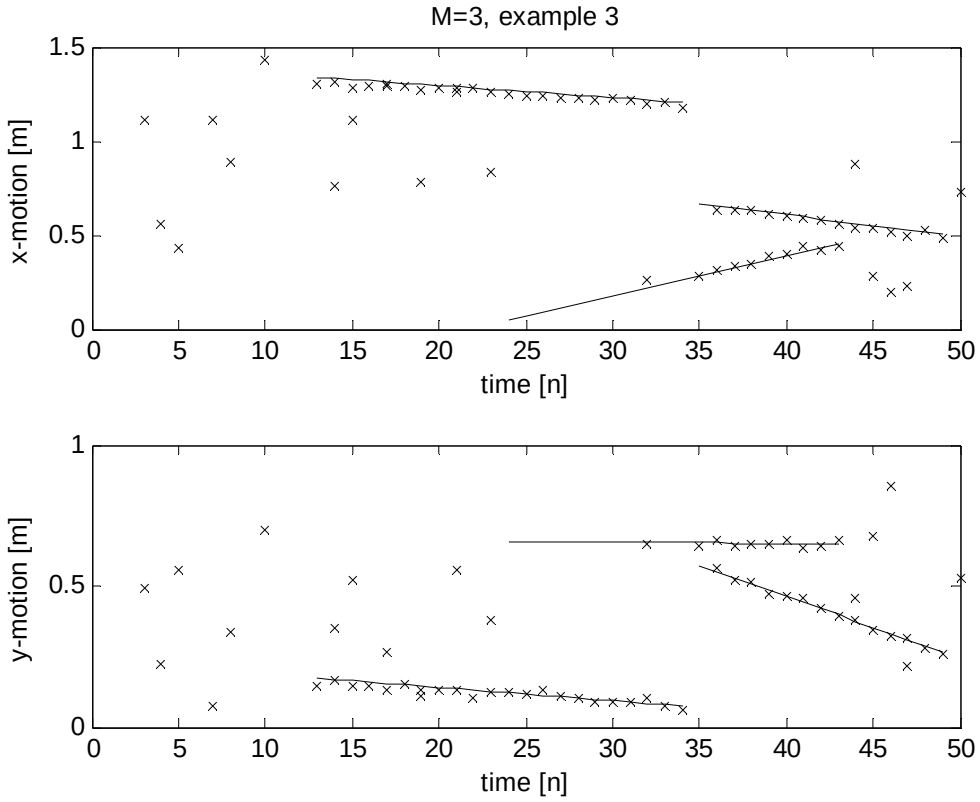

Figure 7. Detecting Performance of M = 3 (Example 3, Group 10)

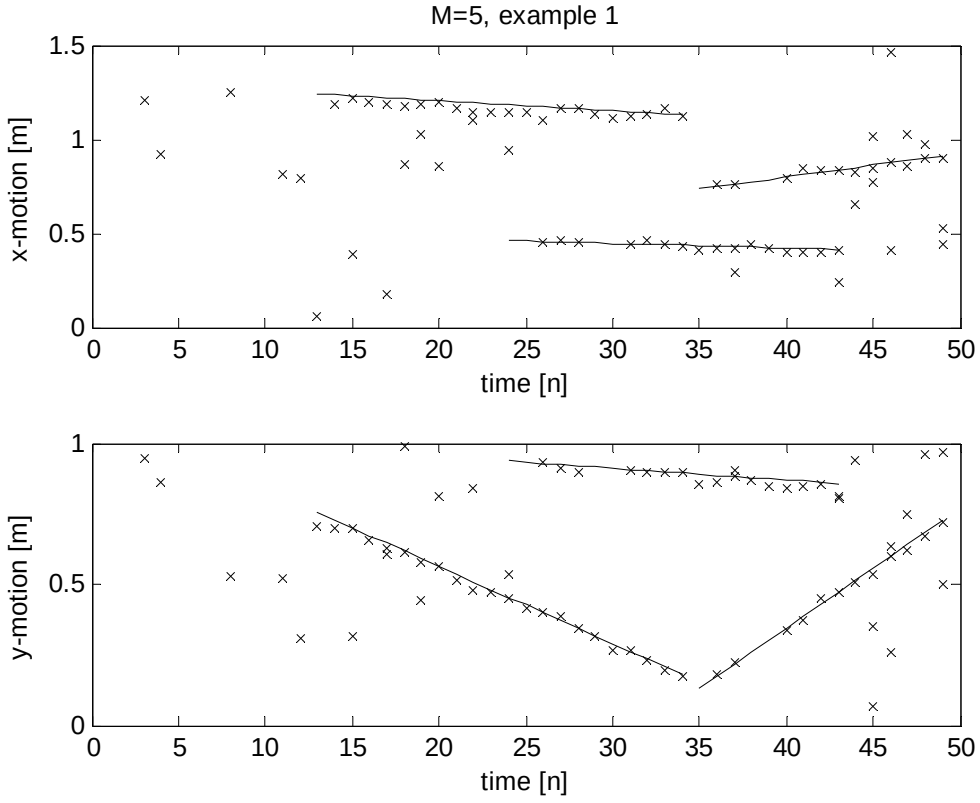

Figure 8. Detecting Performance of M = 5 (Example 1, Group 11)

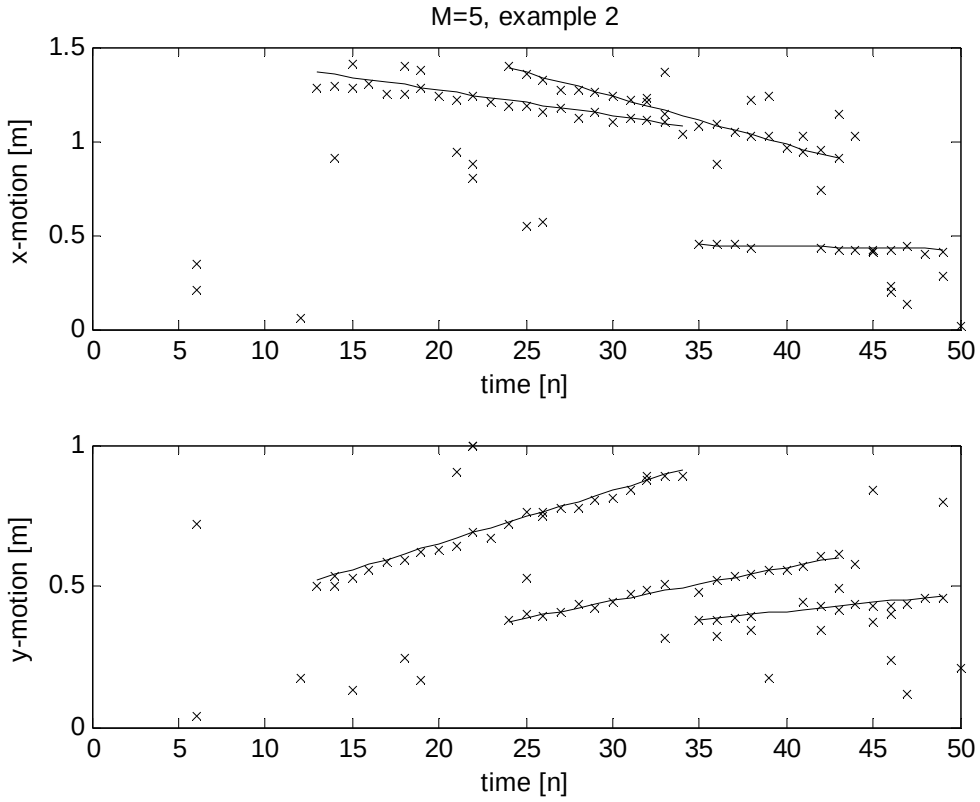

Figure 9. Detecting Performance of M = 5 (Example 2, Group 6)

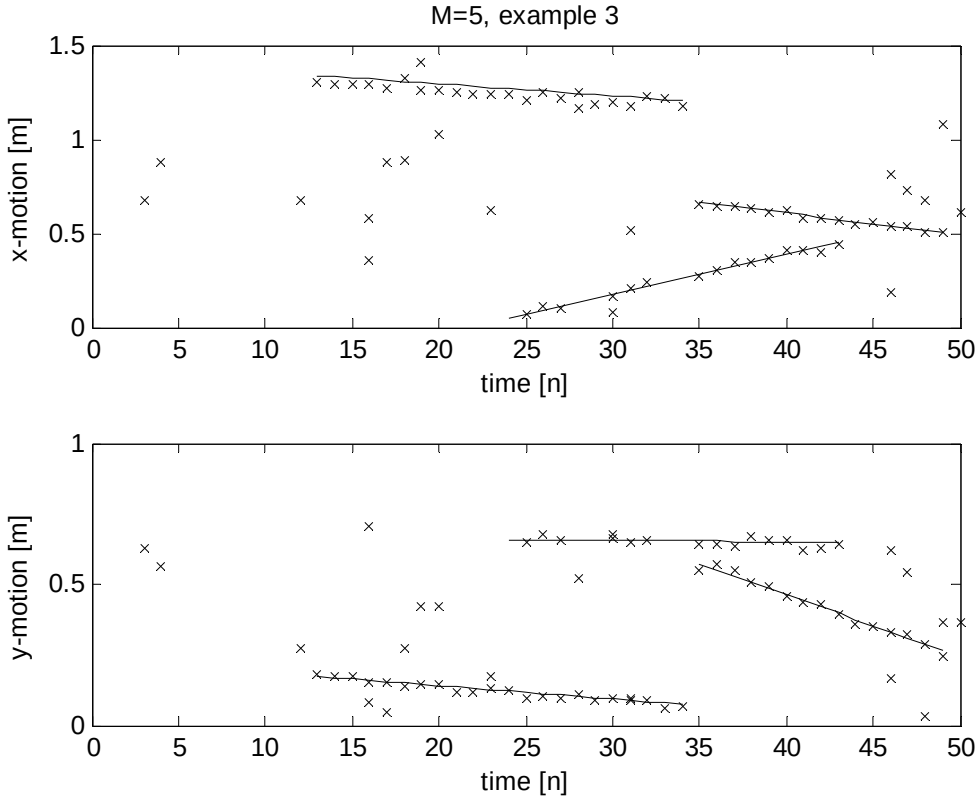

Figure 10. Detecting Performance of M = 5 (Example 3, Group 12)

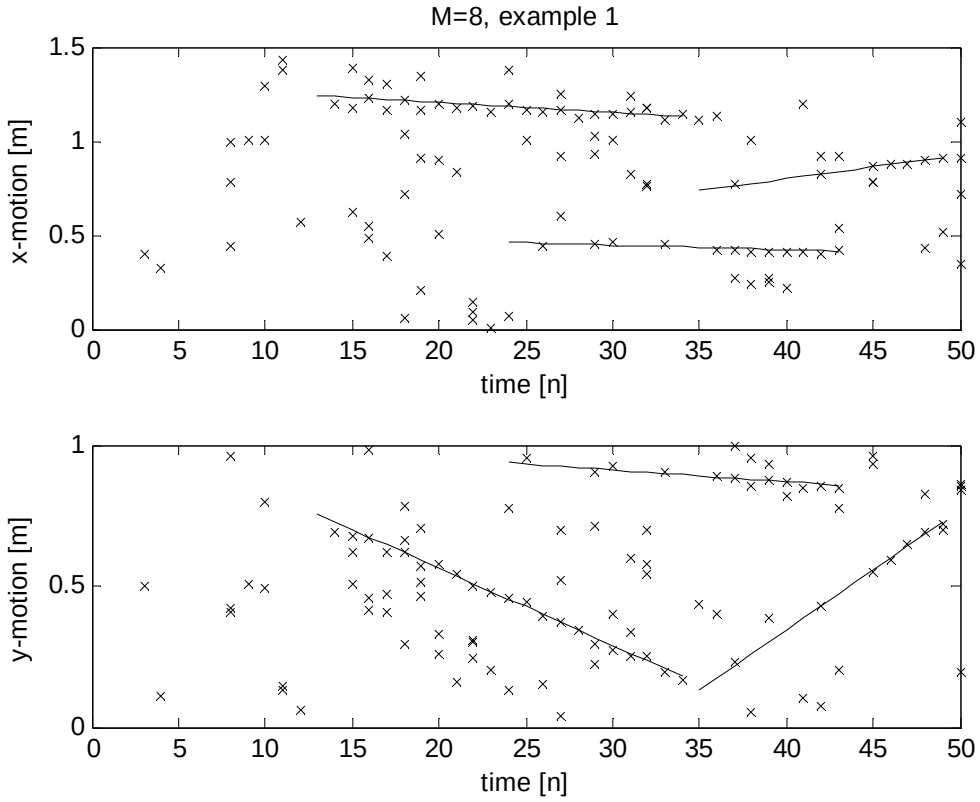

Figure 11. Detecting Performance of M = 8 (Example1)

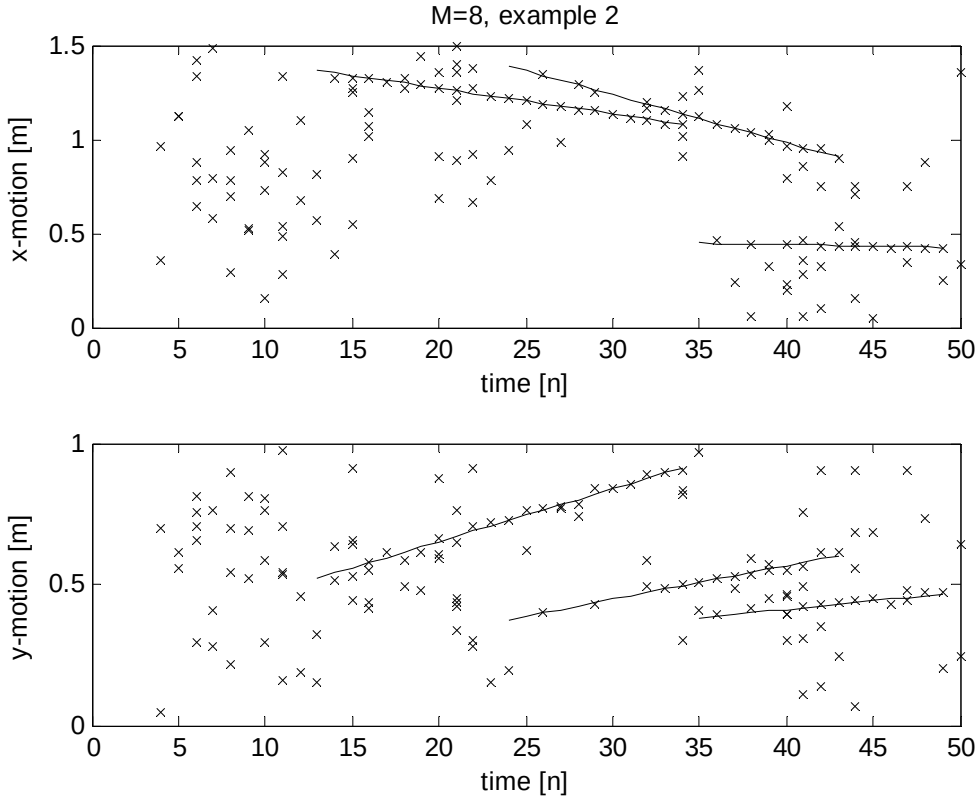

Figure 12. Detecting Performance of M = 8 (Example2)

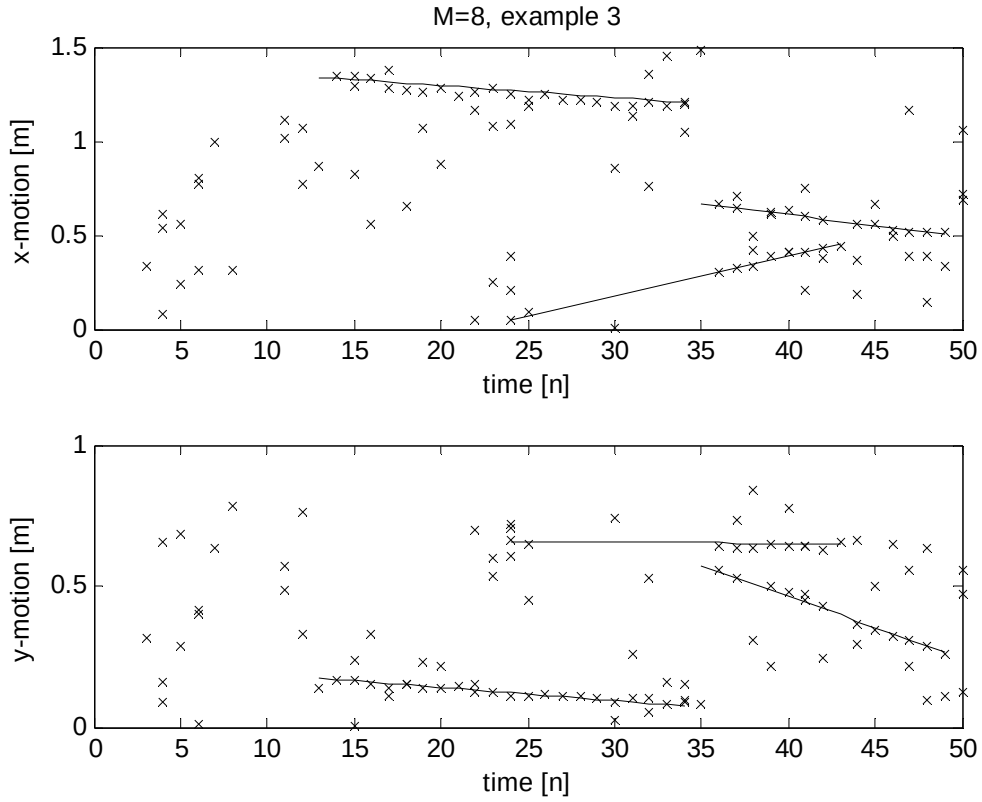

Figure 13. Detecting Performance of M = 8 (Example 3)

Analyzing data and figures, the number of observations per time sample M should be neither too large nor too small. Compare the average data for $M = 3$ and $M = 5$ for three examples, the number of spuriousities for $M = 3$ is less than for $M = 5$, but the detection time for $M = 3$ is longer than for $M = 5$. If M is too large ($M = 8$), the GM-PHD filter does not filter the observations well and there is a lot of clutter. An intermediate value $M = 5$ is a good number of observations per time sample when there are three targets, as the number of points is sufficient to catch all targets, with an acceptable number of spuriousities and a relatively low detection time.

4.1.4. The Process Noise Matrix

The process noise matrix defined in equation (20) is defined as $Q = \sigma_Q I$, where σ_Q is the process noise power and *I* stands for the identity matrix. The program is executed for $\sigma_Q \in \{0.1,\ 0.01,\ 0.001\}$. We can see that when $\sigma_Q = 0.1$, the number of spuriousities are always very large; when $\sigma_Q = 0.001$, though the spuriousities are few, the detection time is very long. Hence a value between 0.1 and 0.001 is suitable; $\sigma_Q = 0.01$ is chosen in the final program.

Figures 14-22 show nine results for $\sigma_Q \in \{0.1,\ 0.01,\ 0.001\}$ for three true trajectory examples respectively.

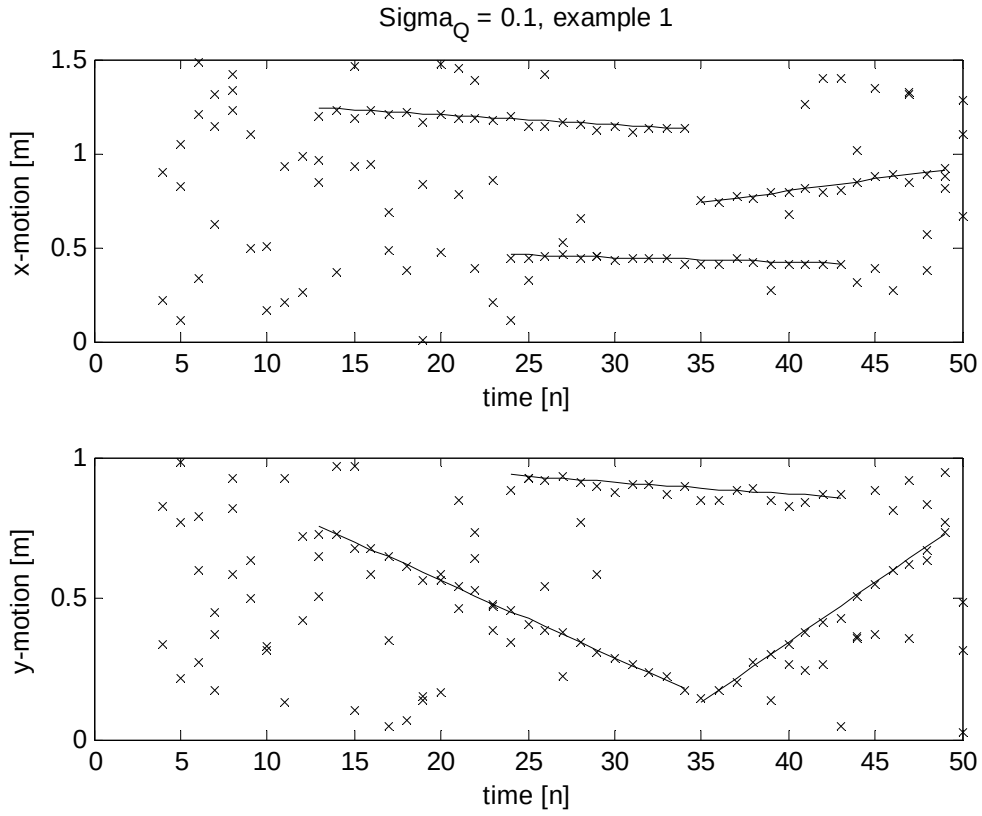

Figure 14. Detecting Performance of σ_Q = 0.1 (Example 1)

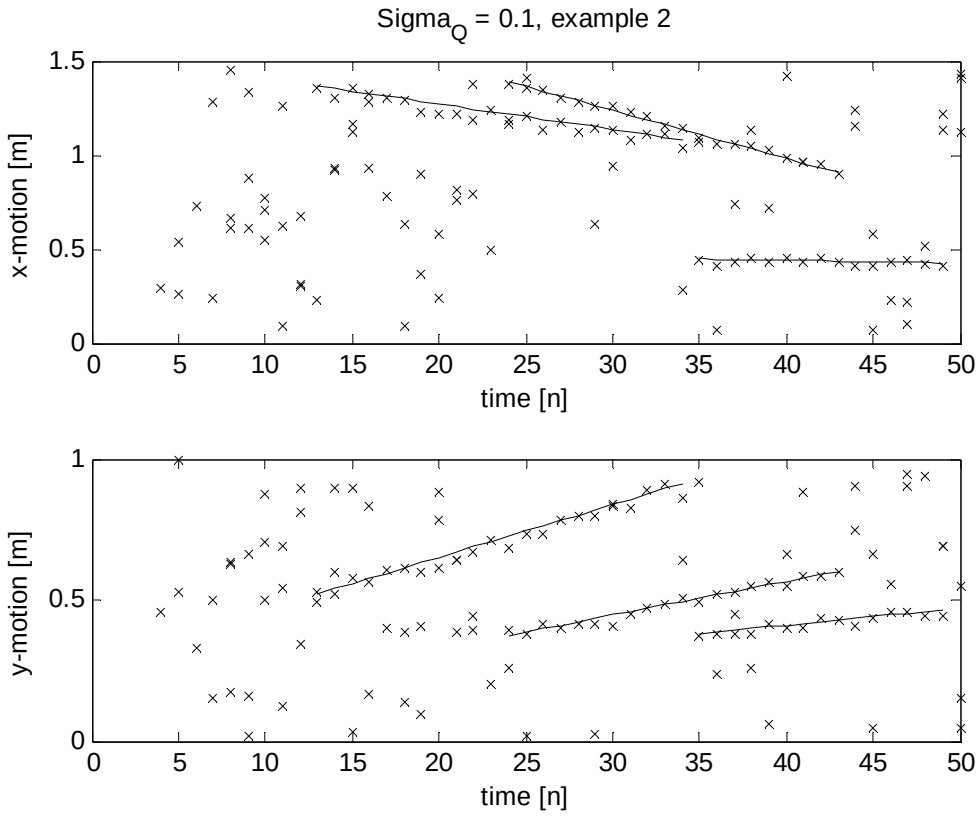

Figure 15. Detecting Performance of $\sigma_Q = 0.1$ (Example 2)

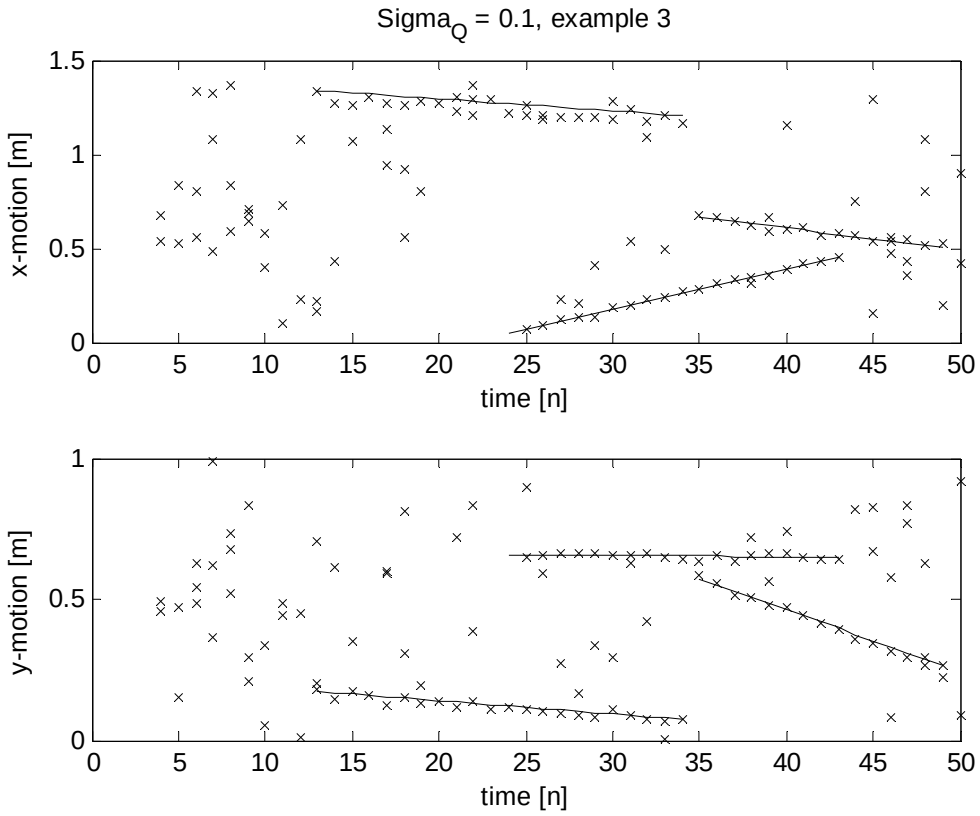

Figure 16. Detecting Performance of $\sigma_Q = 0.1$ (Example 3)

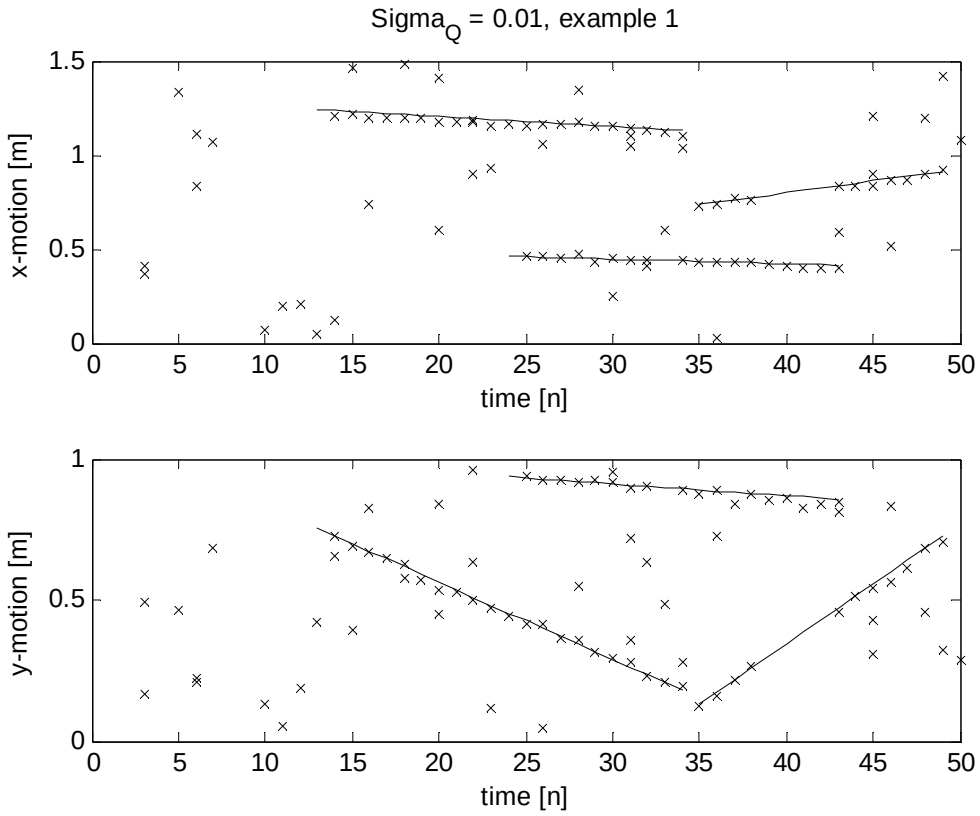

Figure 17. Detecting Performance of $\sigma_Q = 0.01$ (Example 1)

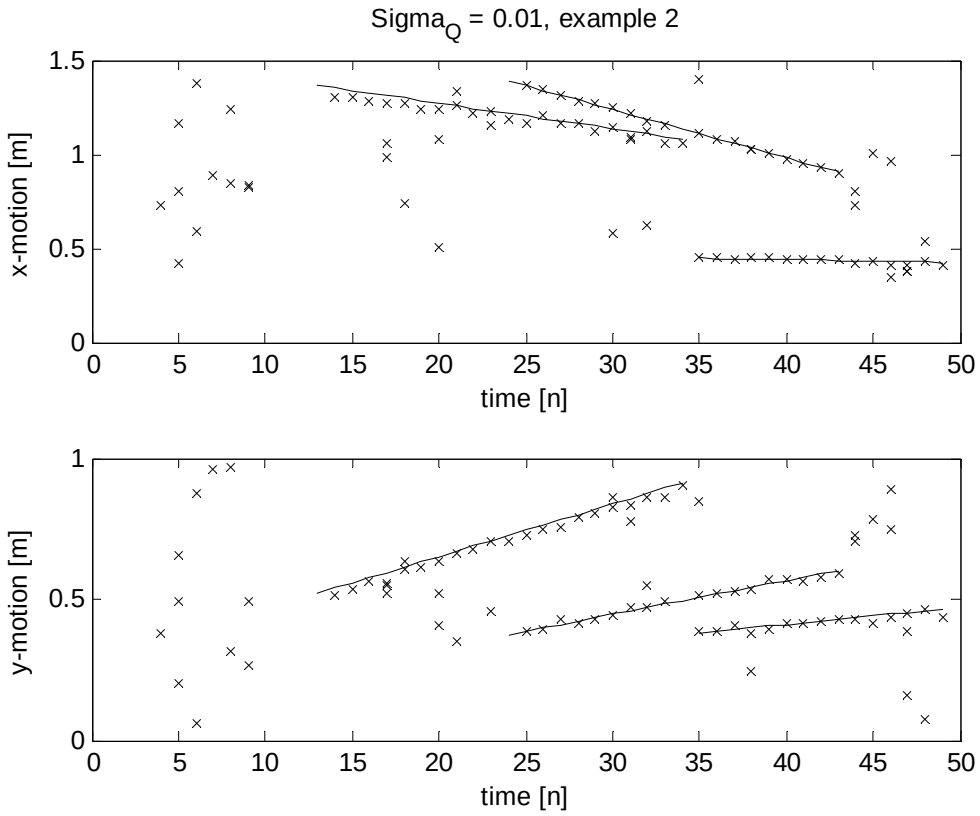

Figure 18. Detecting Performance of $\sigma_Q = 0.01$ (Example 2)

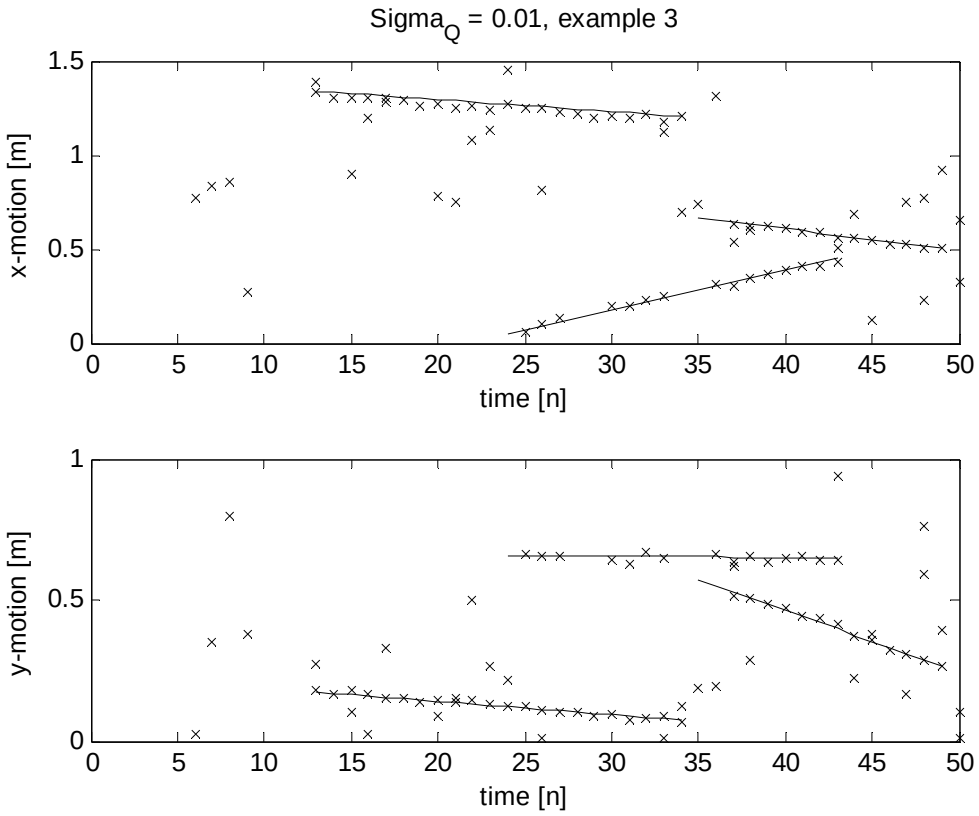

Figure 19. Detecting Performance of $\sigma_Q = 0.01$ (Example 3)

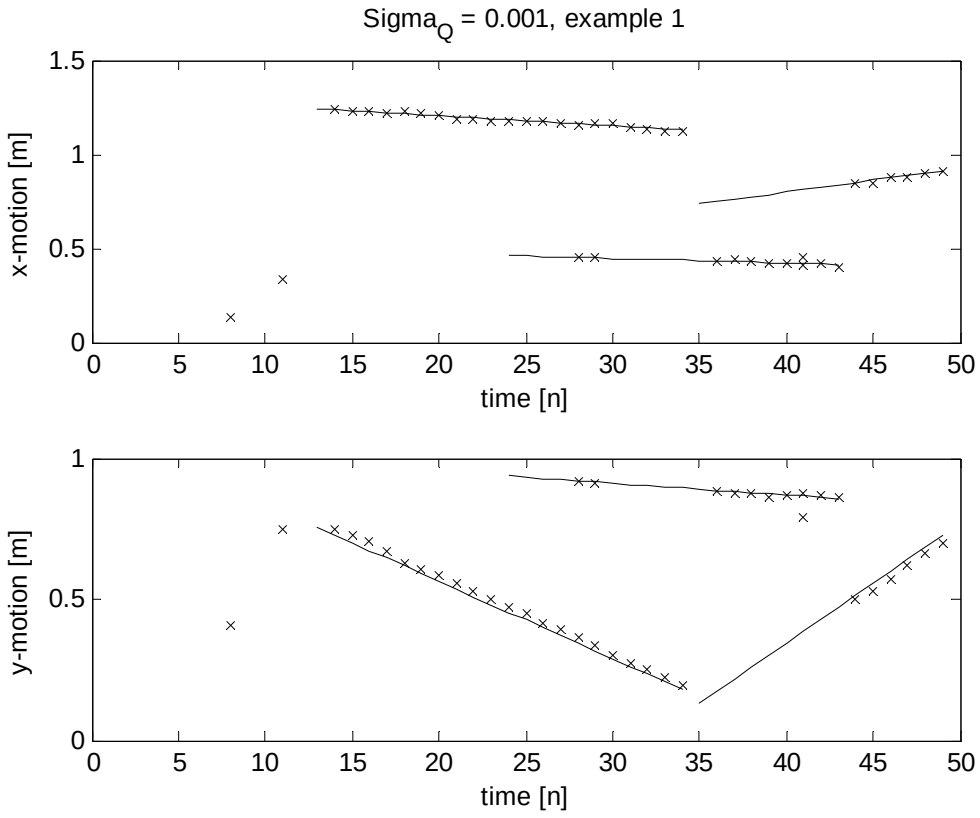

Figure 20. Detecting Performance of $\sigma_Q = 0.001$ (Example 1)

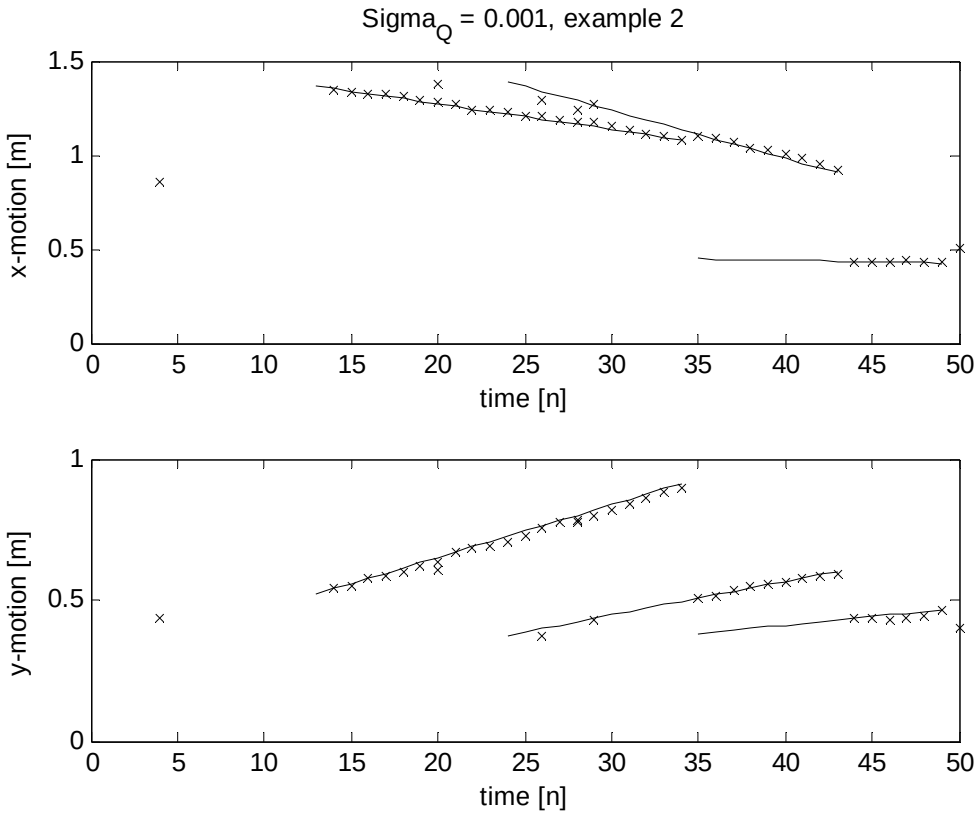

Figure 21. Detecting Performance of $\sigma_Q = 0.001$ (Example 2)

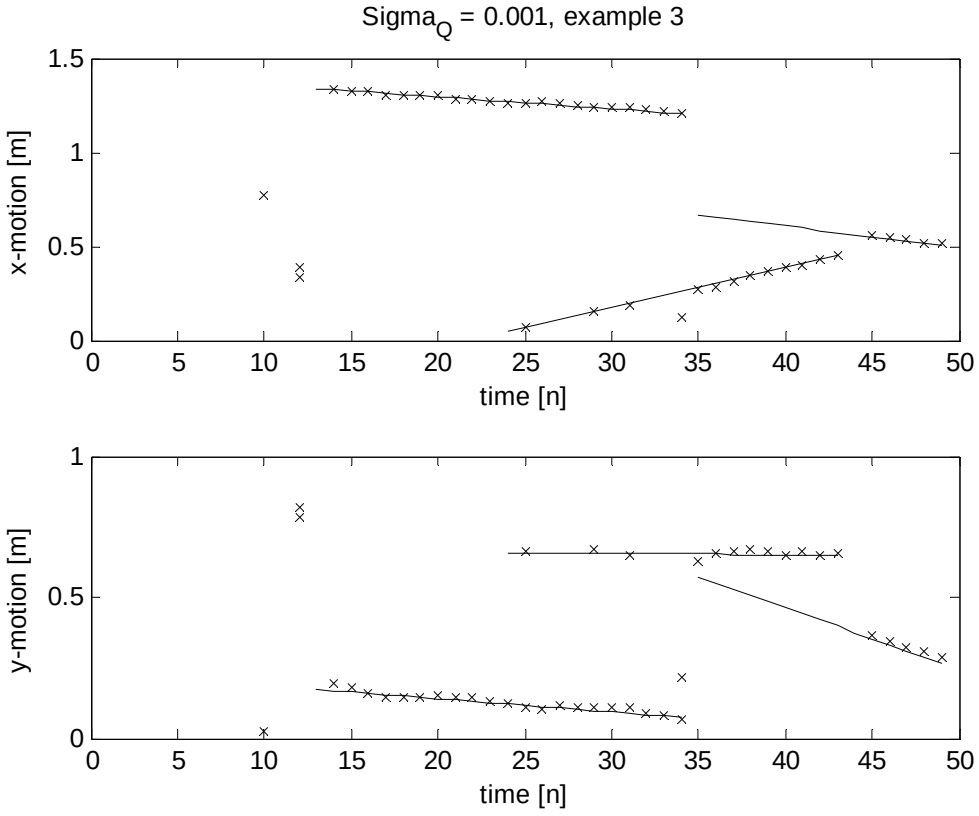

Figure 22. Detecting Performance of $\sigma_Q = 0.001$ (Example 3)

4.1.5. The Observation Noise Matrix

The observation noise matrix defined in equation (21) is defined as $R = \sigma_R I$, where σ_R is the observation noise power and *I* stands for the identity matrix. The program is executed for $\sigma_R \in \{0.1,\ 0.01,\ 0.001\}$. If σ_R is too large, the estimates may depart form the true trajectories. Compare with figures 23-31, $\sigma_R = 0.001$ is chosen is the final program.

Figures 23-31 show nine results for $\sigma_R \in \{0.1,\ 0.01,\ 0.001\}$ for three true trajectory examples respectively.

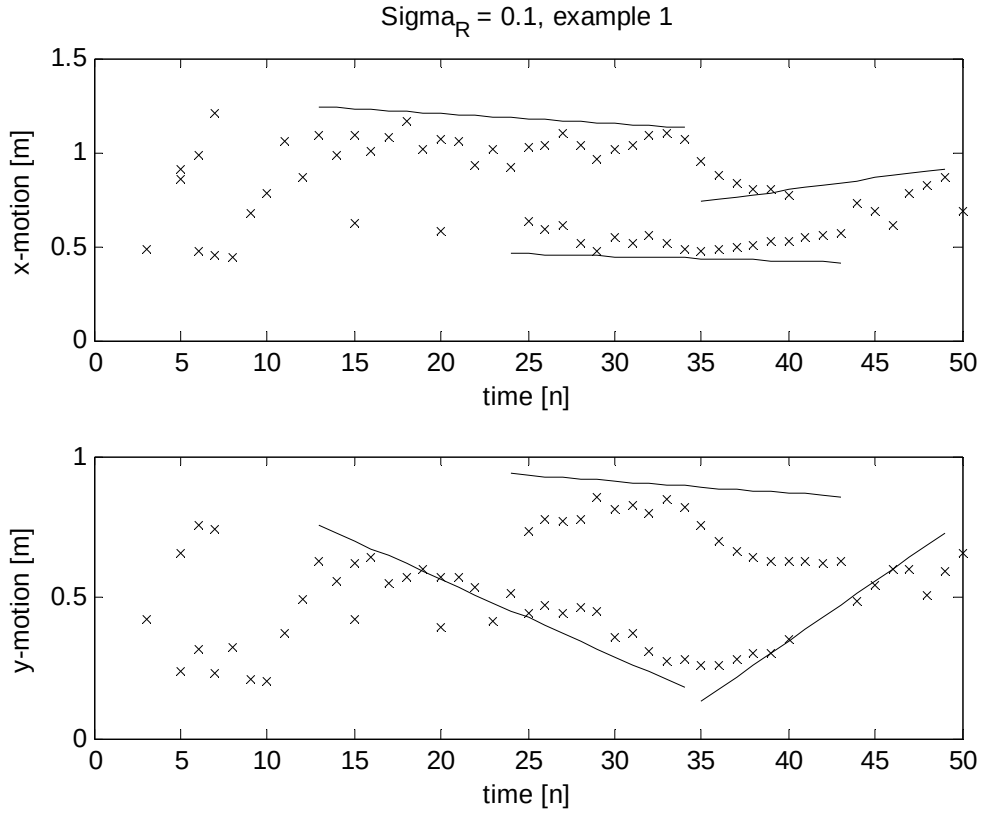

Figure 23. Detecting Performance of σ_R = 0.1 (Example 1)

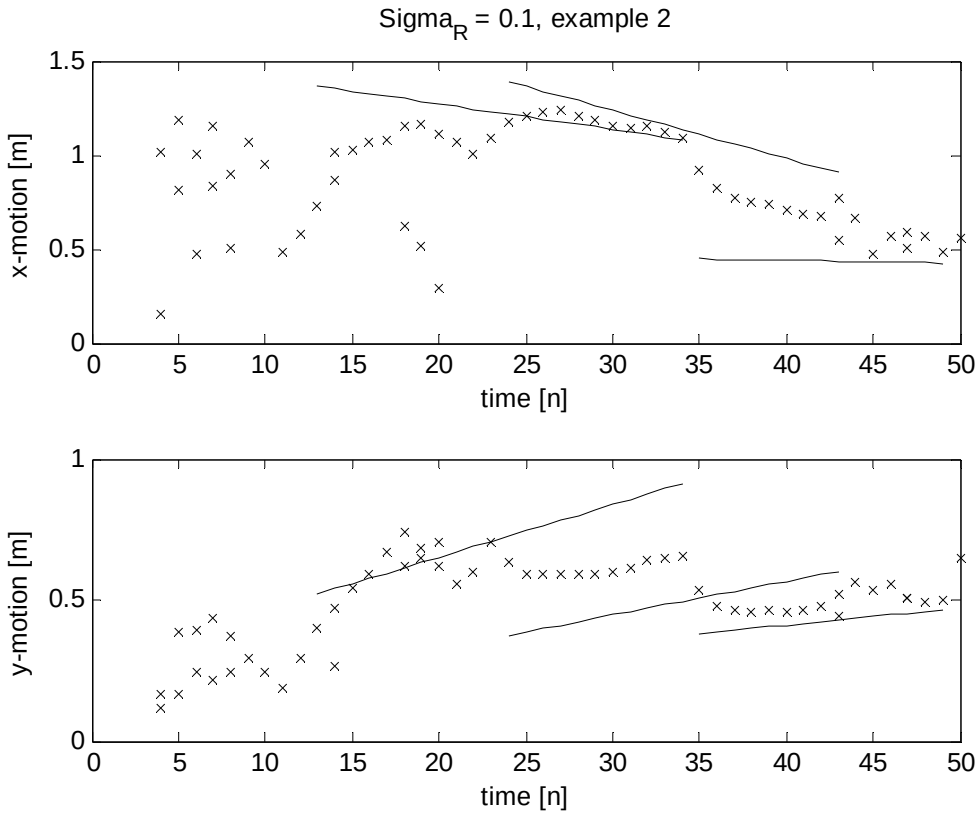

Figure 24. Detecting Performance of $\sigma_R = 0.1$ (Example 2)

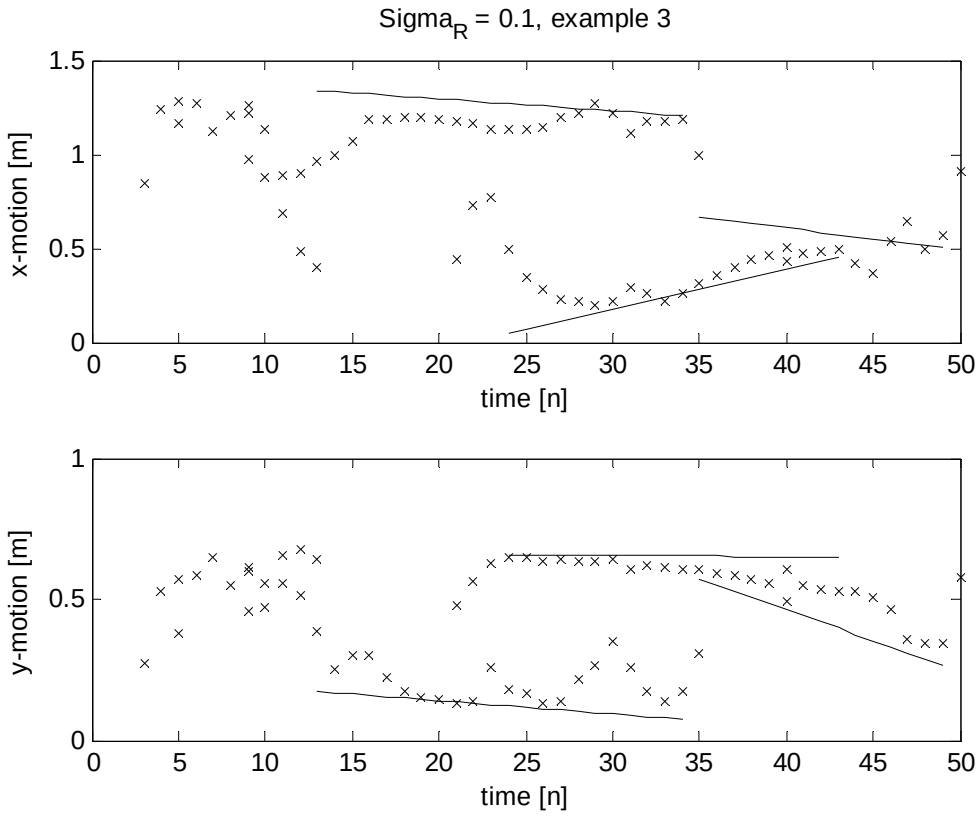

Figure 25. Detecting Performance of $\sigma_R = 0.1$ (Example 3)

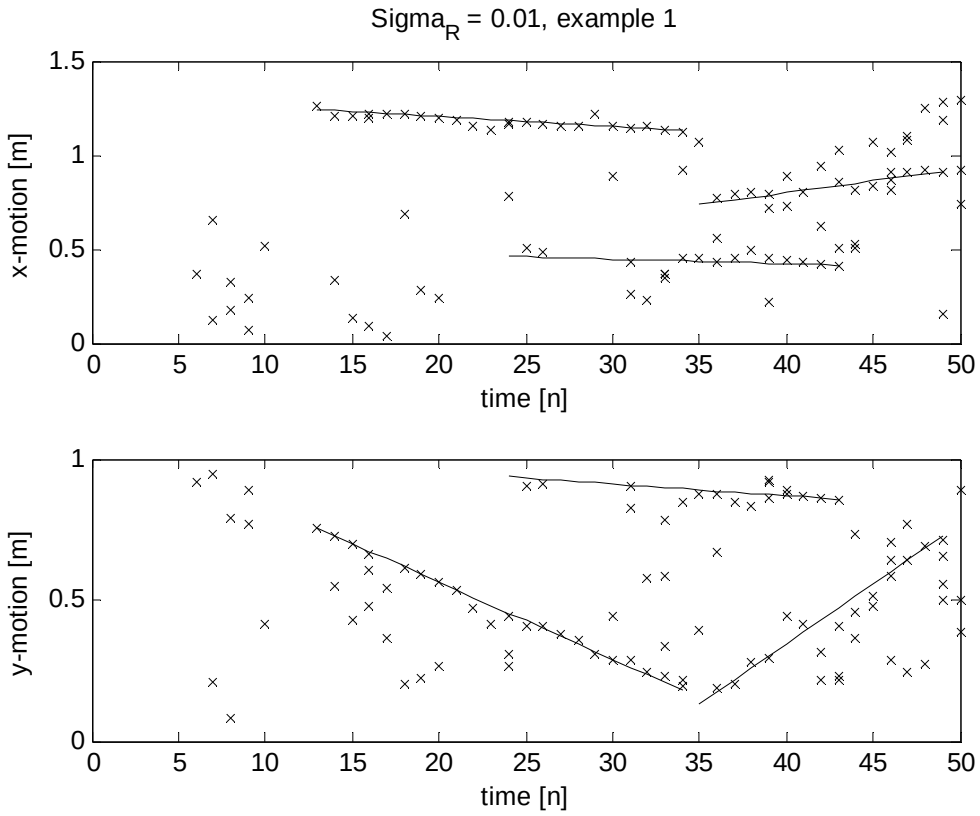

Figure 26. Detecting Performance of $\sigma_R = 0.01$ (Example 1)

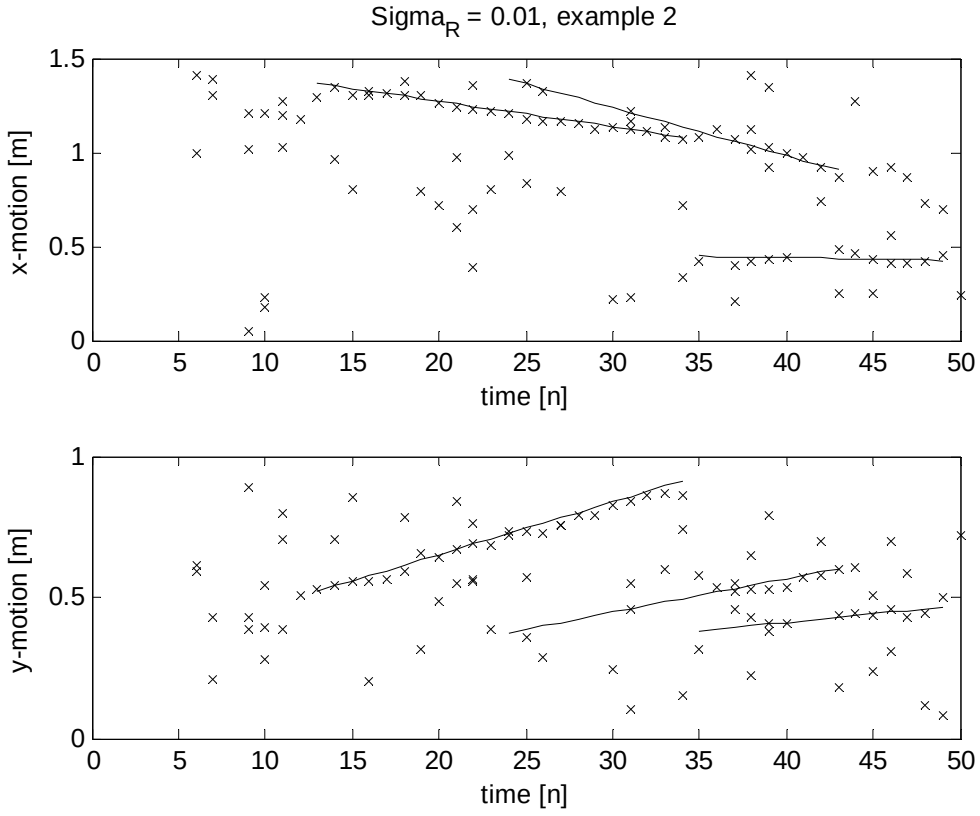

Figure 27. Detecting Performance of $\sigma_R = 0.01$ (Example 2)

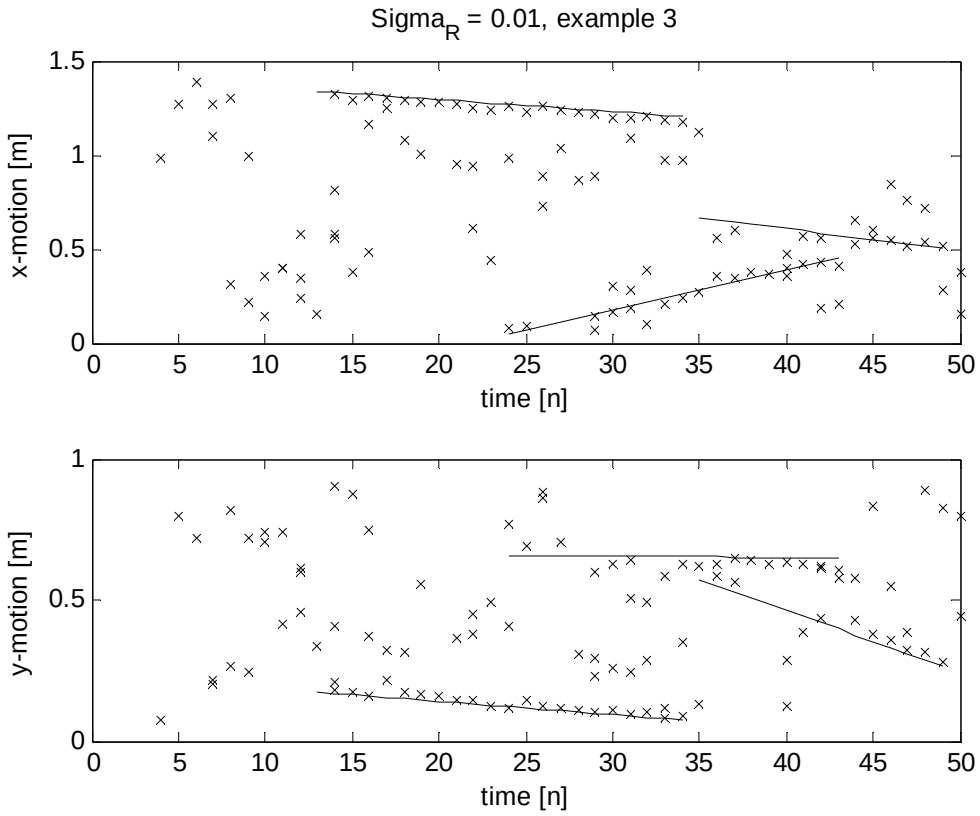

Figure 28. Detecting Performance of $\sigma_R = 0.01$ (Example 3)

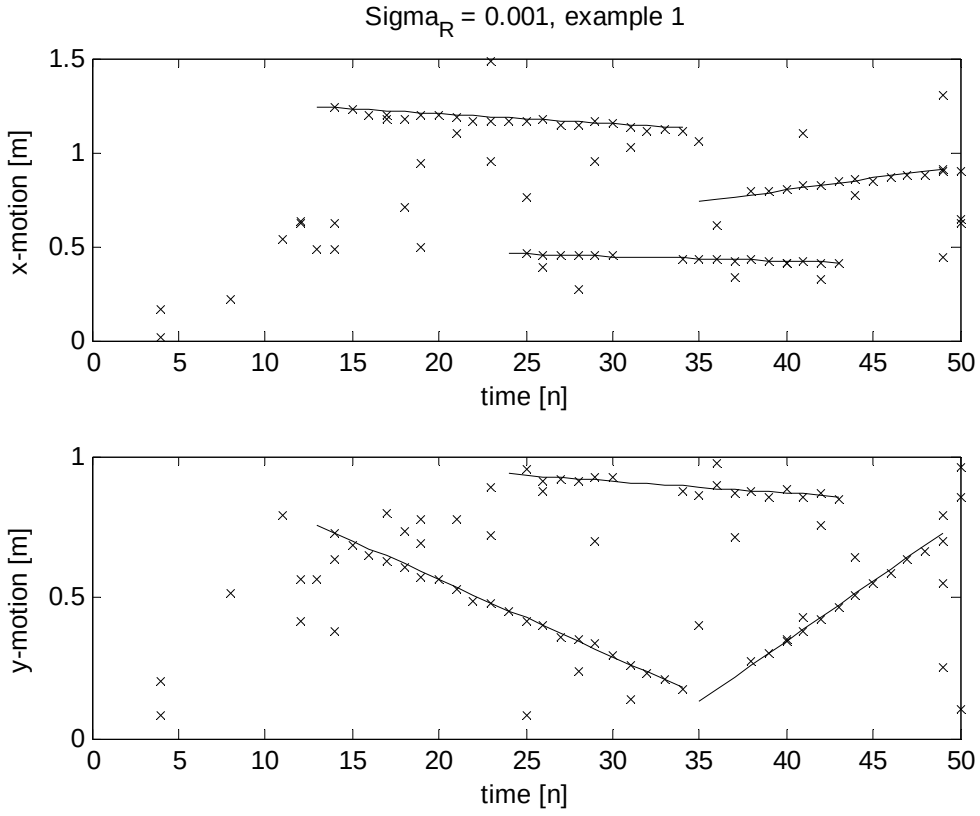

Figure 29. Detecting Performance of $\sigma_R = 0.001$ (Example 1)

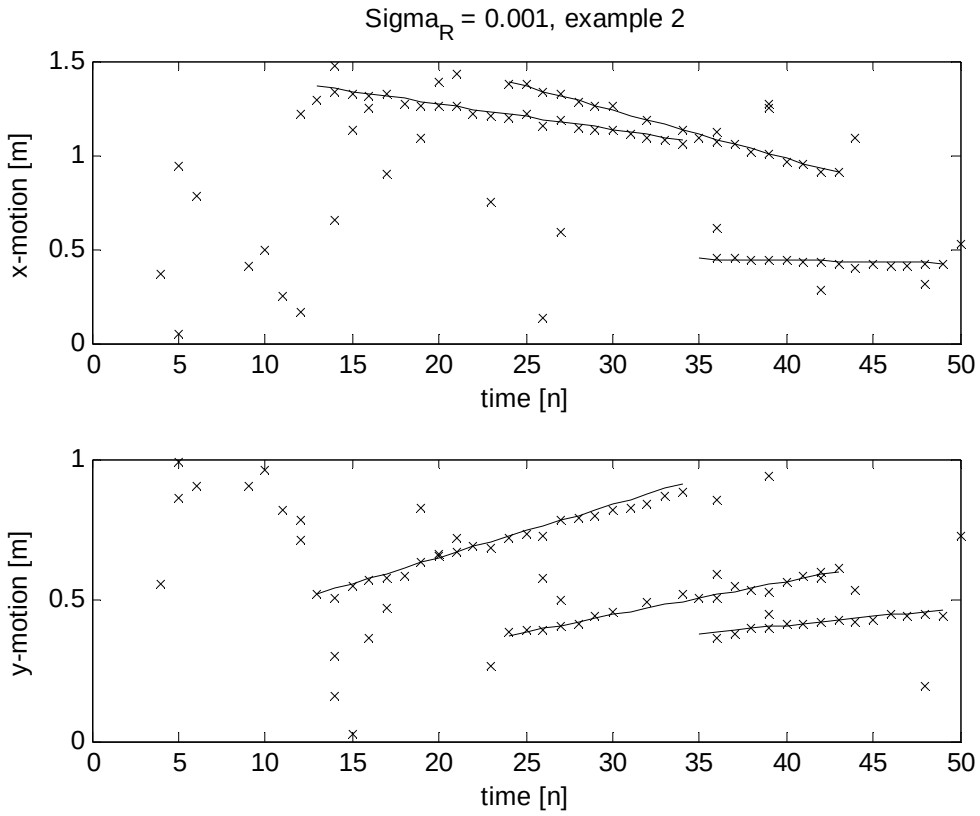

Figure 30. Detecting Performance of $\sigma_R = 0.001$ (Example 2)

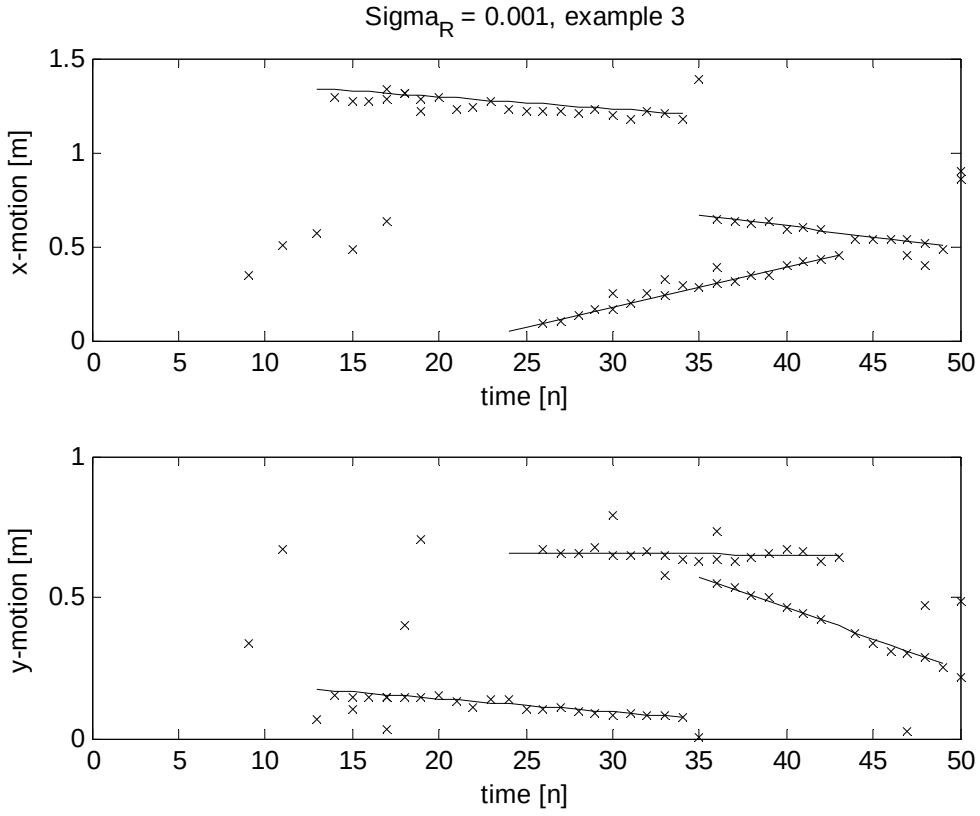

Figure 31. Detecting Performance of $\sigma_R = 0.001$ (Example 3)

4.1.6. The Initial Weight Value

The final task is to define the value of the initial weight w_0 defined in equation (38). Different from other parameters discussed before which are constants, the weight w_k is a variable. It is involved in the computation of the GM-PHD filter and always changing at different time *k*. The initial value of weight w_0 directly affects the tracking result. The program was executed for $w_0 \in \{0.1,\ 0.01,\ 0.002,\ 0.001\}$.

Consider $w_0 = 0.1,\ 0.01$ first. When $w_0 = 0.1$, the number of spuriousities are too large, see Figures 32, 33 and 34; when $w_0 = 0.01$, the number of spuriousities are still very large and the tracking may depart the trajectories, see Figures 35, 36 and 37. Hence the value of *w* still needs to be decreased. Then make a comparison between $w_0 \in \{0.002,\ 0.001\}$. Repeat the procedure above, execute the program for w_0 = 0.0002, 0.0001 twelve times respectively, and the number of spuriousities and detection times are recorded. The average values are represented in Table 2. Two outliers are discarded to ensure a more exact statistic average. Figures 38-43 shows results for w_0 = 0.0002 and 0.0001 for three true trajectory examples respectively.

As w_0 decreases, the number of clutter points reducing but the detection increase. To limit the detection time, comparing with the results shown in Table 2, the number of clutter points does not vary much between w_0 = 0.002 and w_0 = 0.001, but the detection time increases very fast as *w* is reduced form 0.002 to 0.001. Since this paper is focus on the detection time of different birth densities, w_0 = 0.002 was chosen for the final program.

TABLE Ⅱ DATA STATISTIC OF w_0 = 0.002 AND w_0 = 0.001

Initial weight		**w_0 = 0.002**			**w_0 = 0.001**		
		Exp1	Exp2	Exp3	Exp1	Exp2	Exp3
No. of spuriousities/ Detection time	1	22/5	30/3	23/4	24/5	27/9	20/12
	2	~~33/15~~	19/5	~~32/18~~	~~29/15~~	32/10	~~25/17~~
	3	36/5	30/5	20/14	27/10	27/9	17/10
	4	~~33/12~~	36/3	20/3	27/8	~~22/16~~	15/13
	5	33/6	35/7	22/3	~~29/17~~	24/9	24/14
	6	32/11	25/0	22/3	30/13	36/11	~~21/16~~
	7	26/7	~~26/9~~	20/5	22/11	24/5	14/6
	8	29/1	~~24/11~~	27/8	24/11	15/10	18/12
	9	29/6	31/3	19/9	28/13	21/8	22/11
	10	19/3	27/7	24/4	18/3	26/8	19/11
	11	23/8	27/4	~~26/20~~	22/6	~~20/17~~	30/13
	12	18/1	21/7	18/1	30/6	20/2	14/5
Average of 10 times		26.7/5.3	24.8/4.4	21.5/5.4	25.2/8.6	25.2/8.1	19.3/10.7

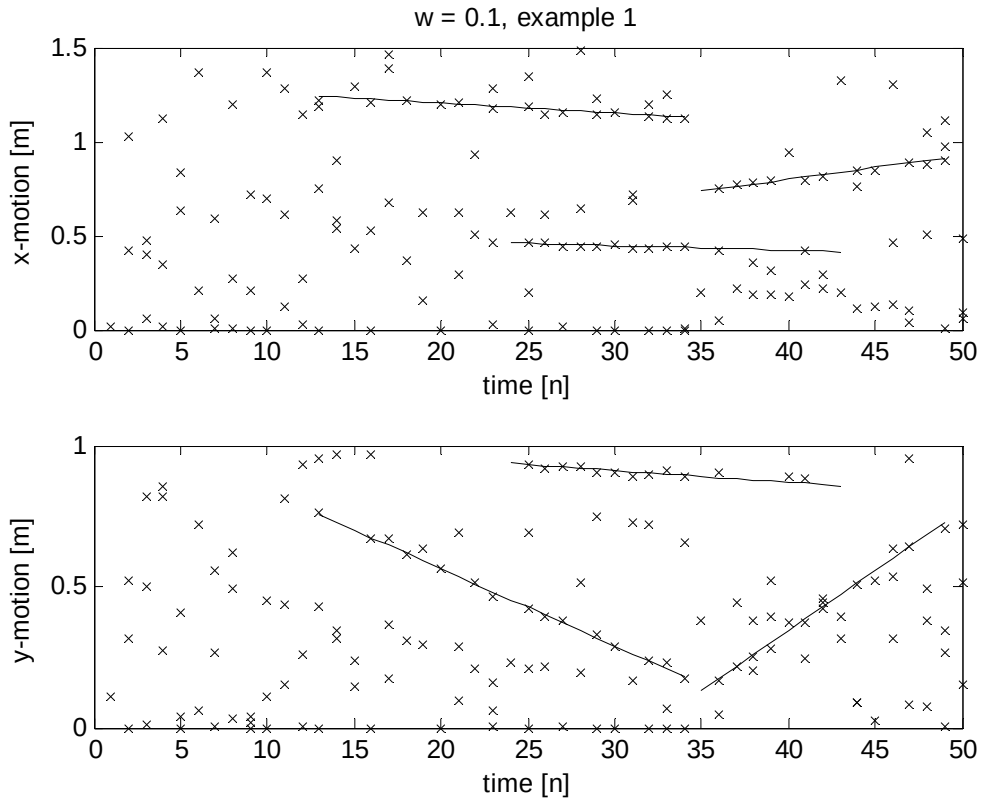

Figure 32. Detecting Performance of w_0 = 0.1 (Example 1)

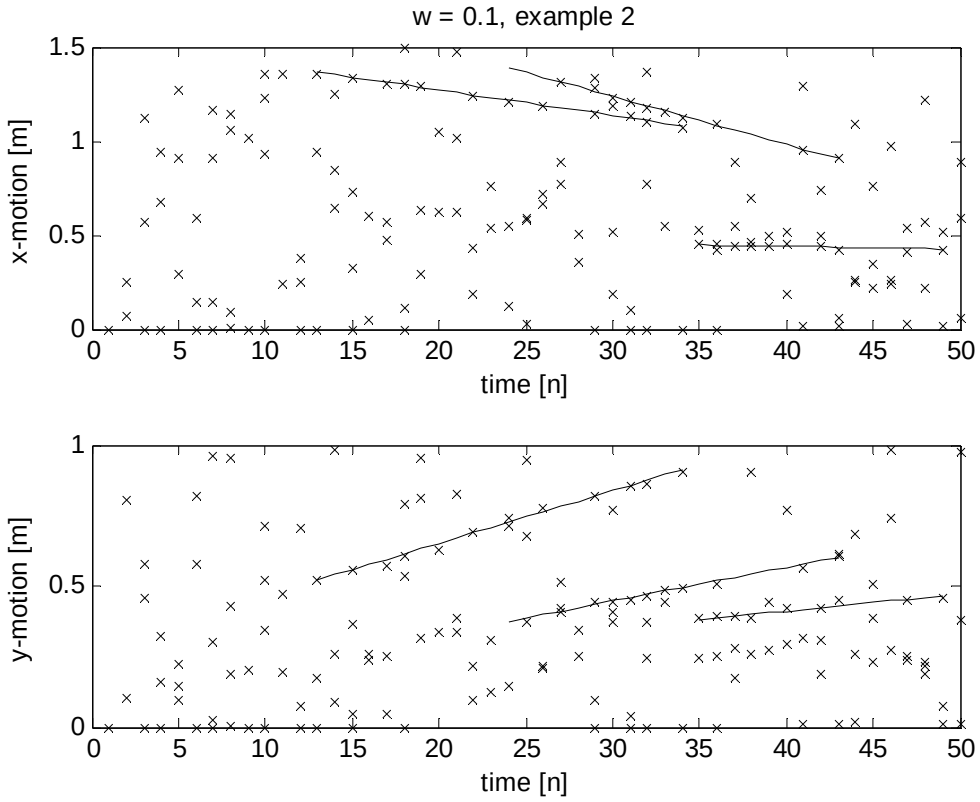

Figure 33. Detecting Performance of $w_0 = 0.1$ (Example 2)

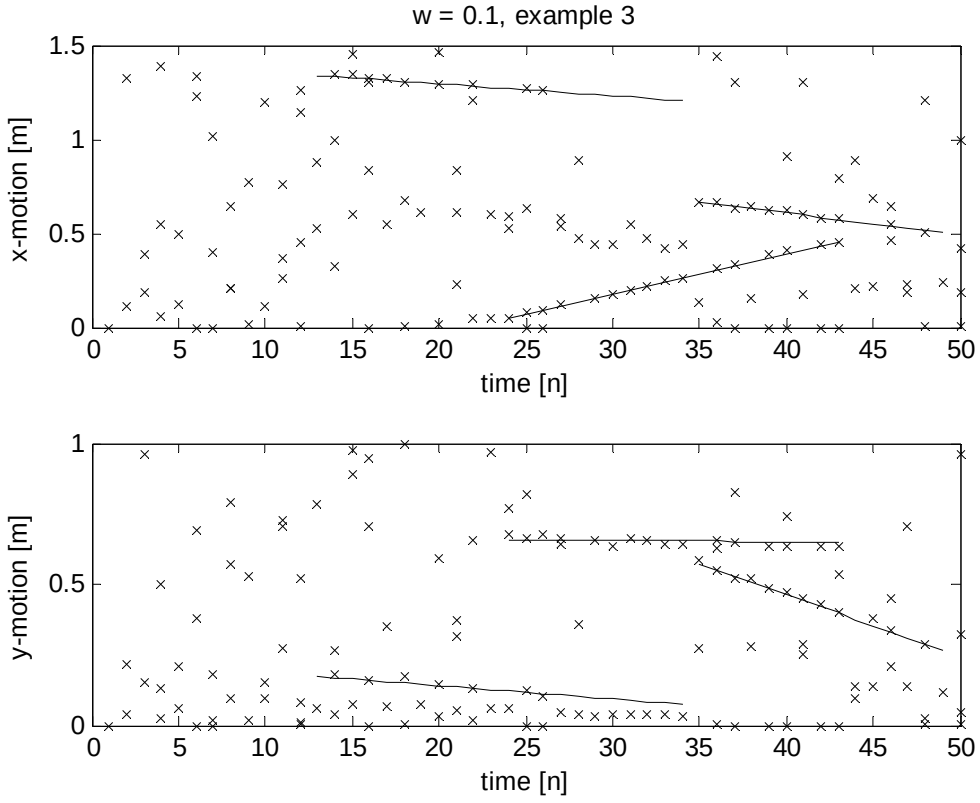

Figure 34. Detecting Performance of $w_0 = 0.1$ (Example 3)

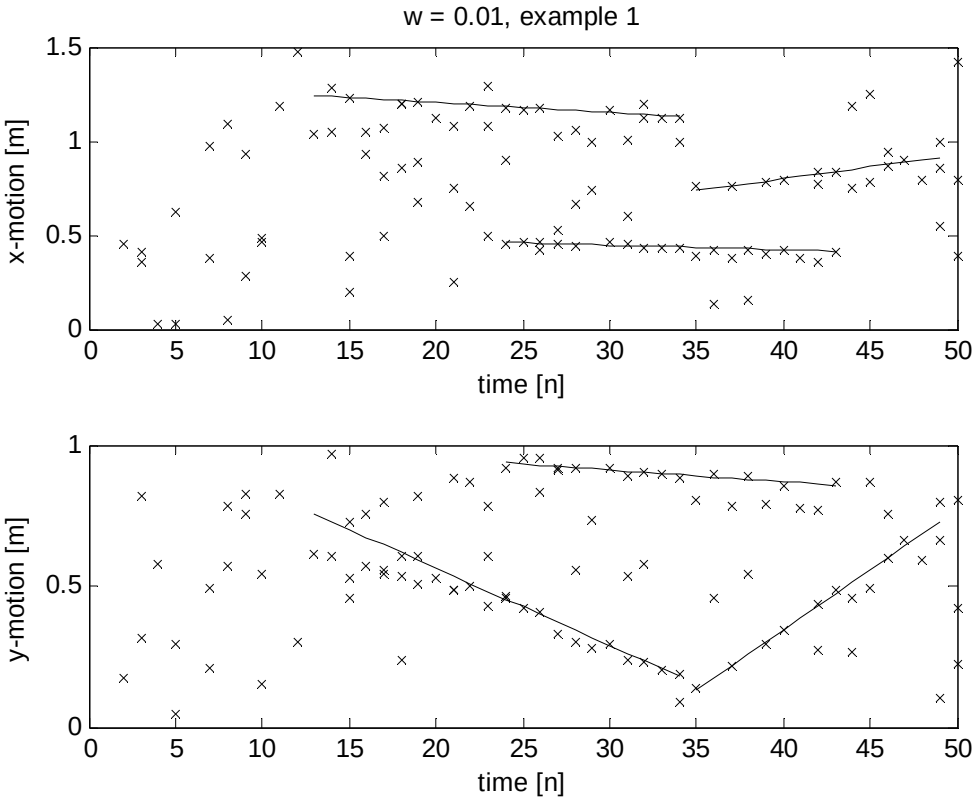

Figure 35. Detecting Performance of $w_0 = 0.01$ (Example 1)

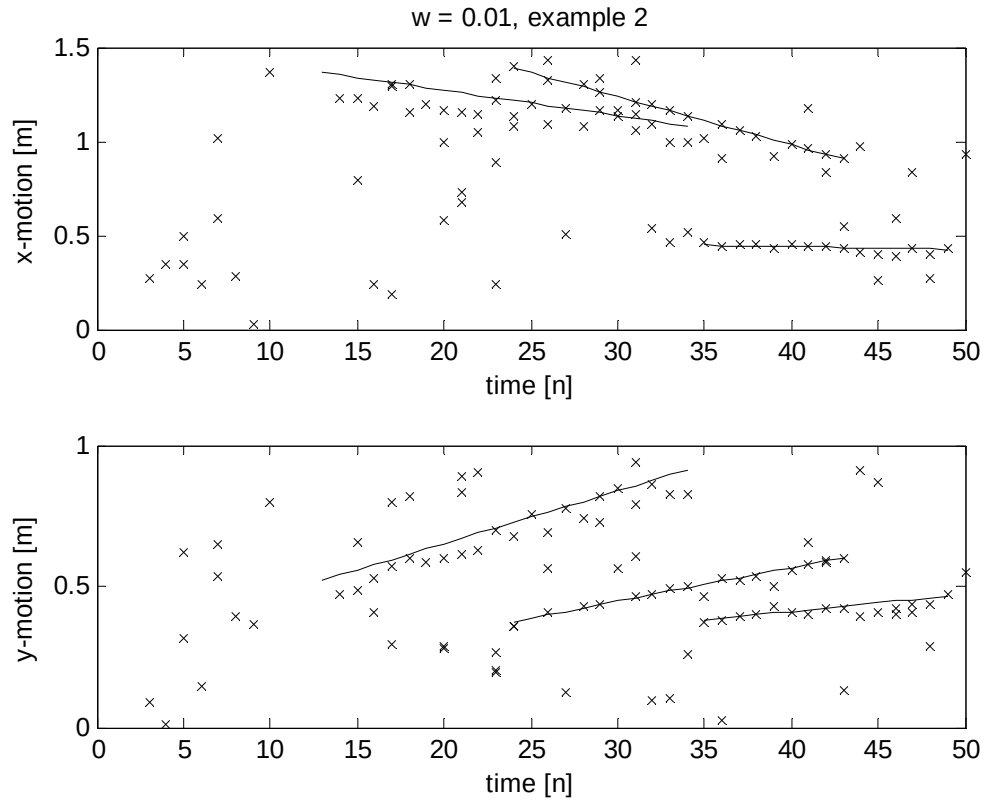

Figure 36. Detecting Performance of $w_0 = 0.01$ (Example 2)

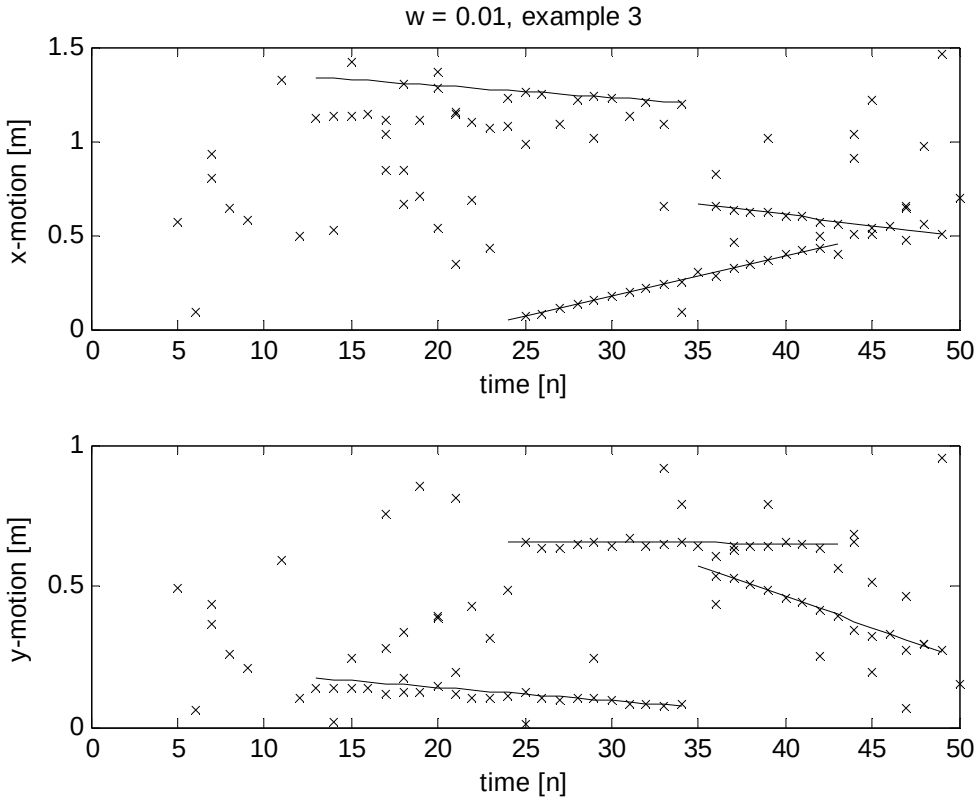

Figure 37. Detecting Performance of $w_0 = 0.01$ (Example 3)

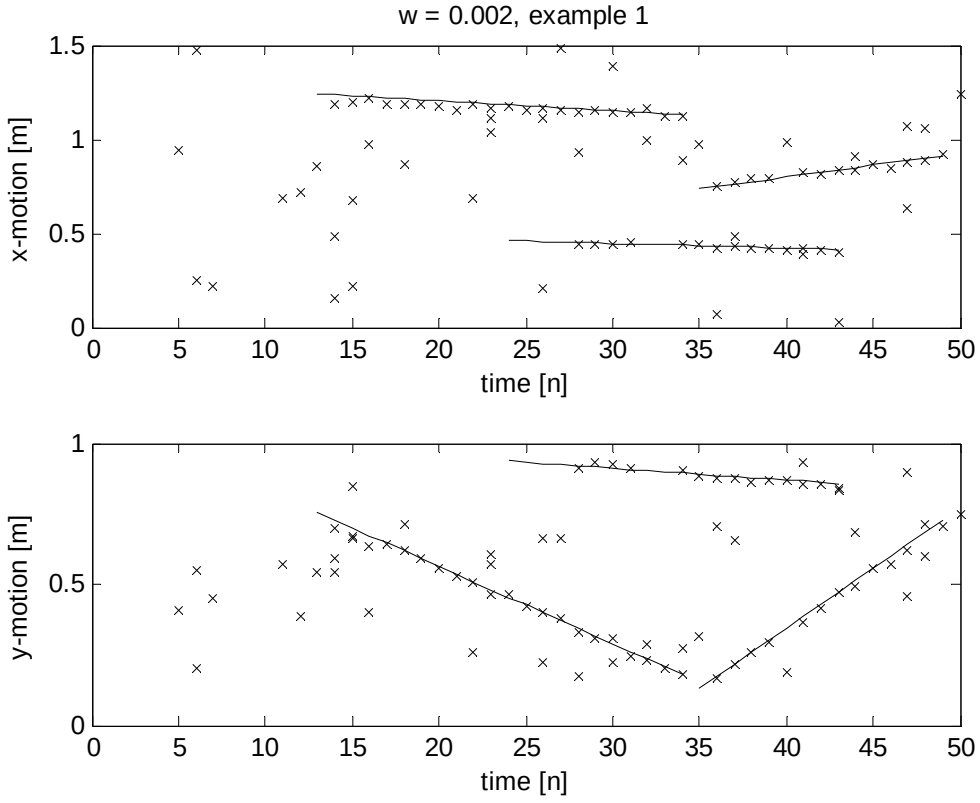

Figure 38. Detecting Performance of $w_0 = 0.002$ (Example 1, Group 5)

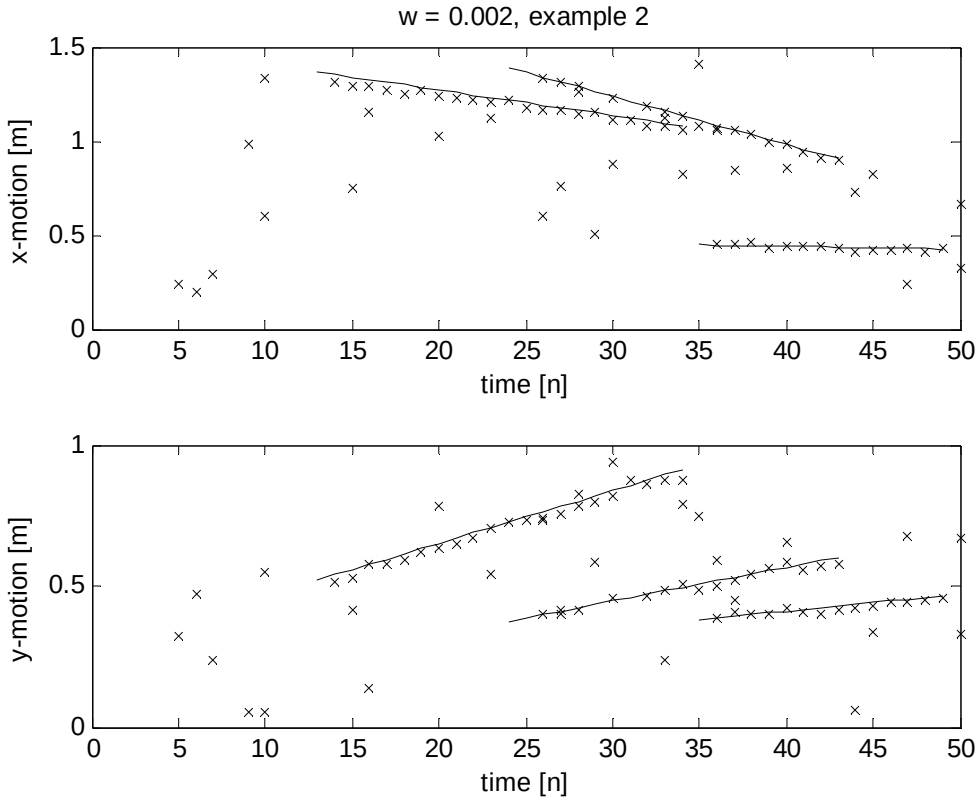

Figure 39. Detecting Performance of $w_0 = 0.002$ (Example 2, Group 11)

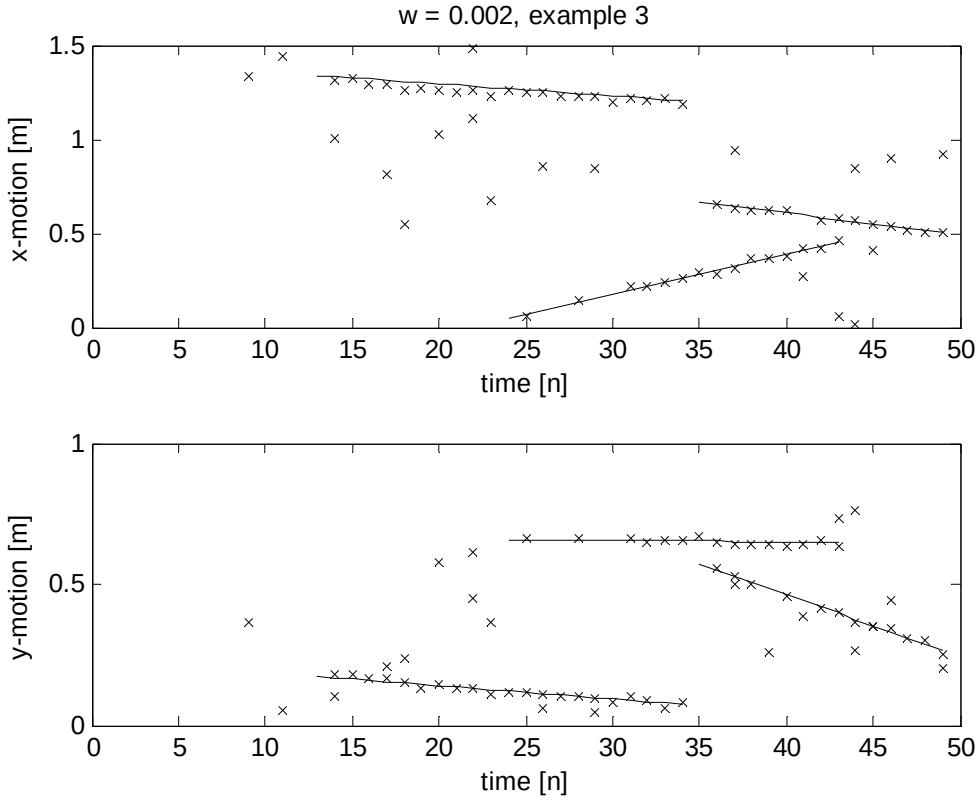

Figure 40. Detecting Performance of $w_0 = 0.002$ (Example 3, Group 9)

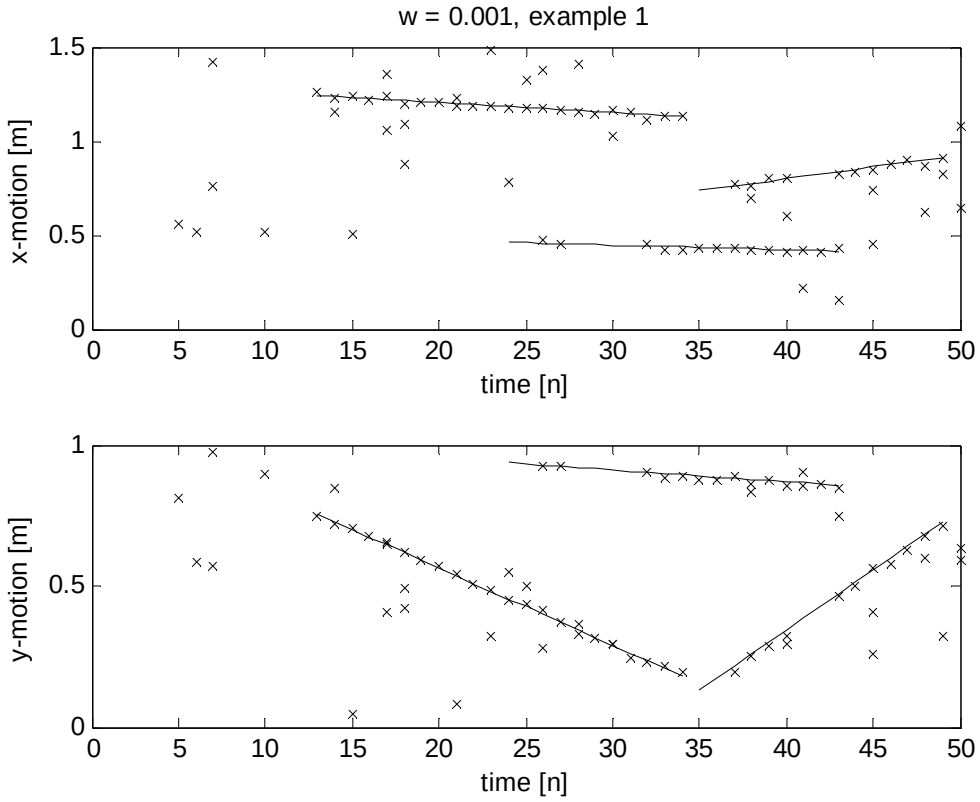

Figure 41. Detecting Performance of $w_0 = 0.001$ (Example 1, Group 3)

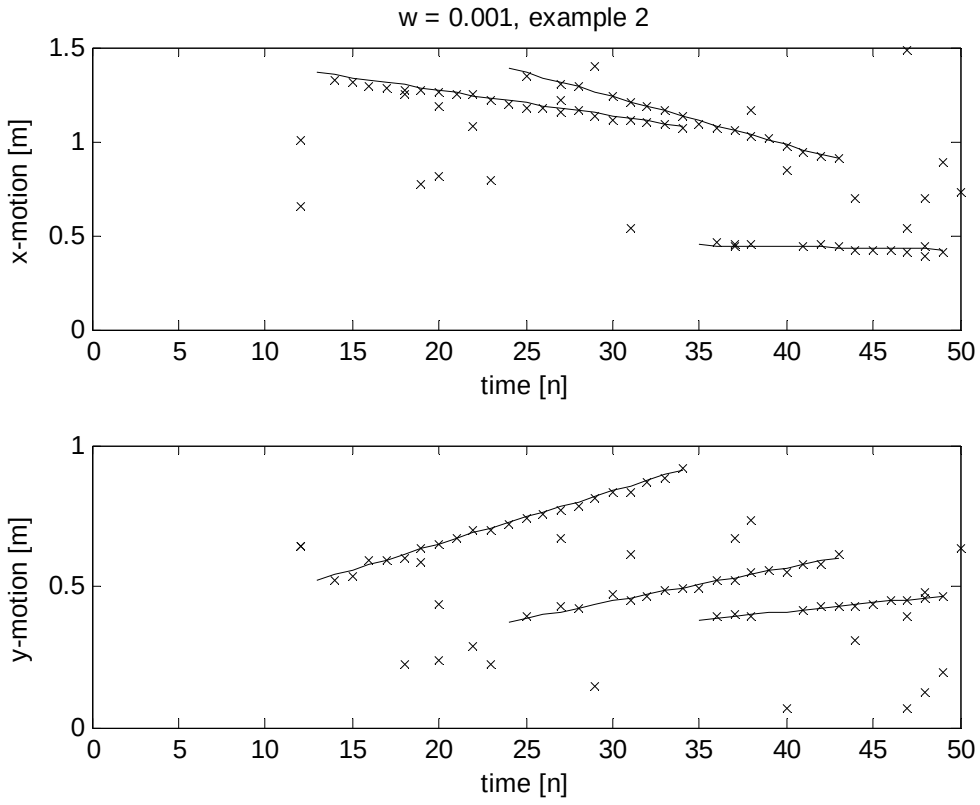

Figure 42. Detecting Performance of $w_0 = 0.001$ (Example 2, Group 9)

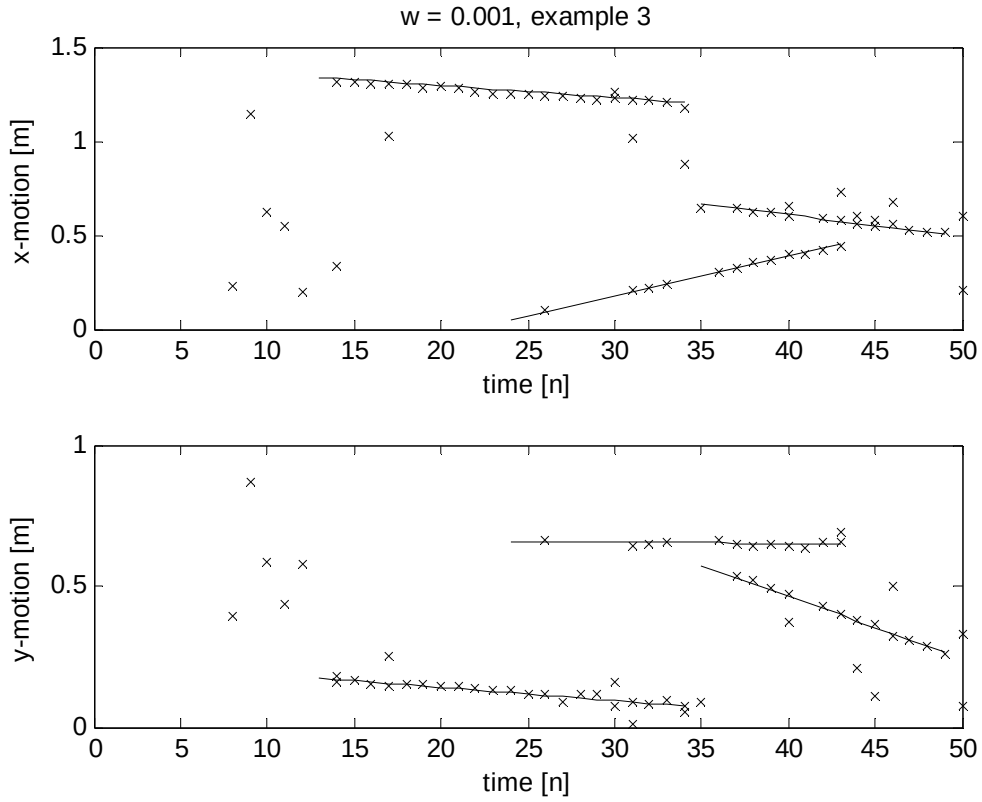

Figure 43. Detecting Performance of $w_0 = 0.001$ (Example 3, Group 3)

Generally, the parameters which affect the detecting performance of the GM-PHD filter are defined as follows:

Number of observations per time sample: $M = 5$;

Process noise matrix: $Q = 0.01 \cdot I$;

Observation noise matrix: $R = 0.001 \cdot I$;

Initial weight: $w_0 = 0.002$.

4.2. Testing

After defining an acceptable setting of the parameters, it goes to test how well the whole program works. In this paper four distributions are discussed: Poisson distribution, narrow Gaussian distribution, wide Gaussian distribution and uniform distribution. In both true target trajectories and target births in PHD filter, the probability density function of each distribution will be used. For example, the Poisson distribution is used to generate the true target trajectory, and then all four distributions are used in the PHD recursion to see how well the GM-PHD filter performs; the other three distributions will be used in true target trajectory respectively too. Hence there are sixteen distribution combinations.

In the case of wide Gaussian distribution, the standard deviation σ is defined as 0.2, while in narrow Gaussian distribution, σ is 0.01. The size of uniform distribution lies in the region $[0,\ 1.5]\times[0,\ 1]$. The centre of Poisson distribution is located at $(x, y) = (0.5,\ 0.4)$. Figure 44 shows the performance where the probability density functions for true target trajectory and GM-PHD filter are both narrow Gaussian.

When the true target trajectories are generated by random, some situations can be ignored considering. An example is shown in Figure 45, where two targets are too close to each other for several time samples; this would not be possible in reality.

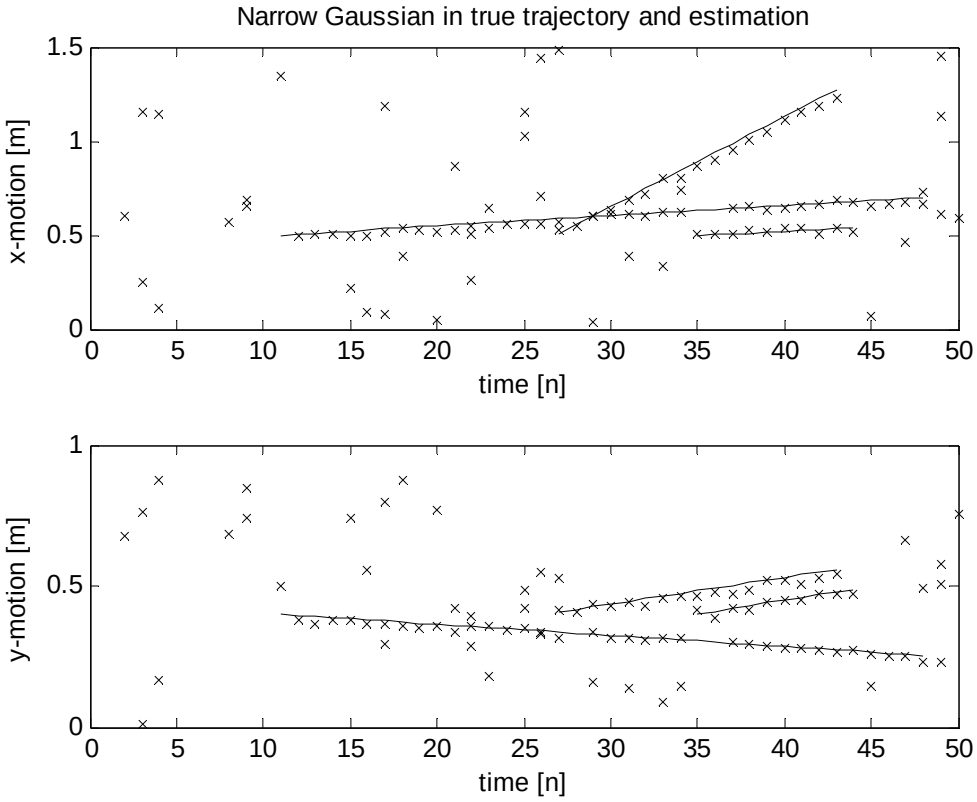

Figure 44. Tracking Performance for narrow Gaussian in true trajectory and estimation.

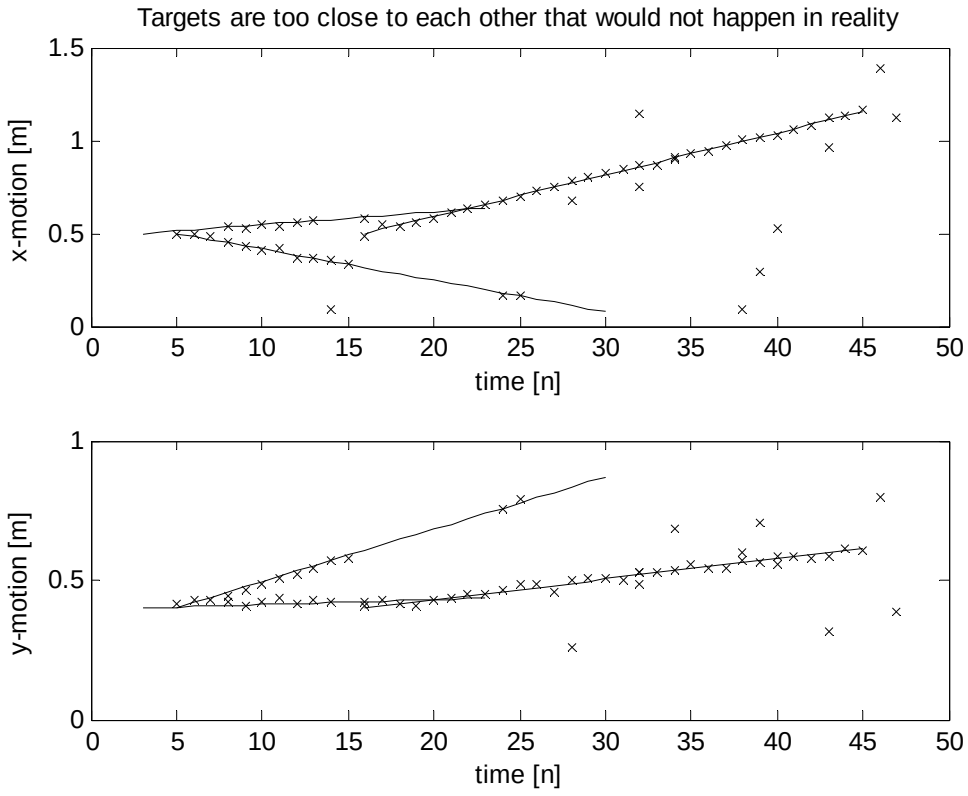

Figure 45. Targets are too close to each other that would not be possible in reality

CHAPTER 5 DATA ANALYSIS

In the data analysis, the focus is on the detection time of the GM-PHD filter to catch the trajectory of a moving target, the shorter the detection time, the better the combination of distributions performs.

5.1. Data Statistics

The program was executed for each combination 200 times and the aggregate detection time was recorded, that is how many time samples it took to detect all the targets successfully. The average detection time is calculated for each combination, and presented in a table. The abbreviations *WG*, *NG*, *P* and *U* stand for Wide Gaussian, Narrow Gaussian, Poisson and Uniform distribution respectively.

A minimum of three estimation points must appear on the true trajectory consecutively for a target to be said to be detected successfully. As shown in Figure 46, the detection time for the true trajectory born on the 22nd time sample and dead on the 49th time sample is 7. If there are too few or no consecutive estimates on the true trajectory, the target is not regarded to be detected, see Figure 47.

The program was executed until getting 200 successful detection figures were obtained. Average detection time for all 16 combinations are shown in Table Ⅲ.

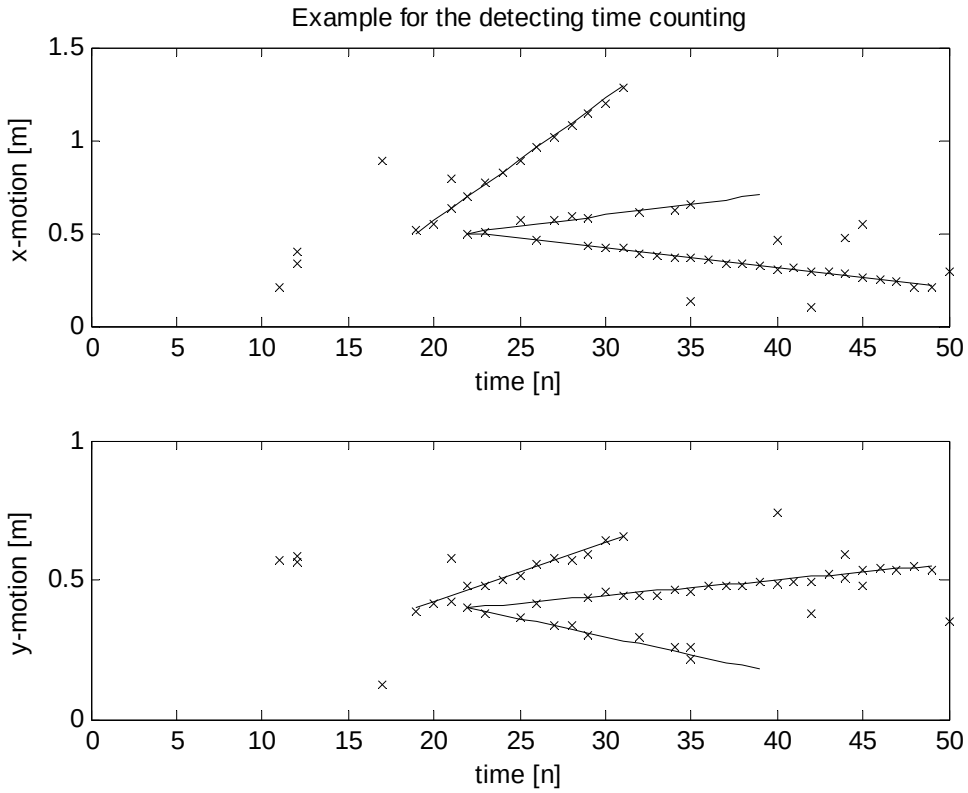

Figure 46.Example for counting the precise detection time

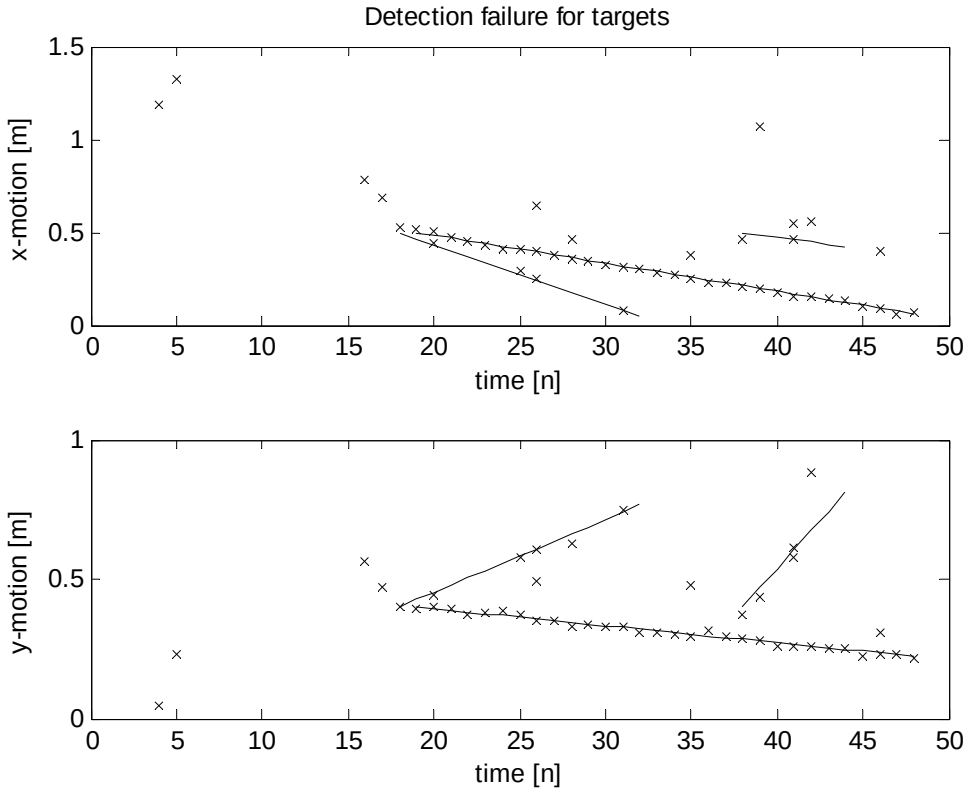

Figure 47. Example for detection failure

TABLE Ⅲ AVERAGE DETECTION TIME FOR 16 DISTRIBUTION COMBINATIONS

Average detection time in 200 times		**Distribution in True Target Trajectory**			
		P	***NG***	***WG***	***U***
Distribution in GM-PHD filter	***P***	2.23	1.985	2.555	3.835
	NG	2.1	2.495	2.38	3.875
	WG	2.24	2.09	2.235	4.015
	U	2.03	2.32	2.615	3.45

A reference average detection time of 1 or 3 time samples is defined, and the percentage that the detection time is less than the reference for each combination is calculated. The result from this calculation is shown in Table Ⅳ and Ⅴ, respectively.

TABLE Ⅳ

PERCENTAGE OF DETECTION TIME < 1 TIME SAMPLE FOR 16 COMBINATIONS

detection time < 1 time sample in 200 times		**Distribution in True Target Trajectory**			
		P	***NG***	***WG***	***U***
Distribution in GM-PHD filter	***P***	0.53	0.515	0.415	0.165
	NG	0.515	0.455	0.44	0.275
	WG	0.55	0.52	0.465	0.23
	U	0.485	0.48	0.415	0.245

TABLE Ⅴ

PERCENTAGE OF DETECTION TIME < 3 TIME SAMPLE FOR 16 COMBINATIONS

detection time < 3 time sample in 200 times		**Distribution in True Target Trajectory**			
		P	***NG***	***WG***	***U***
Distribution in GM-PHD filter	***P***	0.775	0.81	0.75	0.585
	NG	0.8	0.78	0.77	0.6
	WG	0.815	0.8	0.82	0.585
	U	0.835	0.78	0.765	0.66

5.2. Data Analysis

According to Tables Ⅲ, Ⅳ and Ⅴ, we can conclude that a combination of distribution with small average detection time and large percentage of detection time within the reference while executing the program 200 times performs best. Four superior combinations are *NG-P*, *U-P*, *P-NG* and *WG-N*, whose average detection time is within 2.1 time samples, the average detection time less than 1 time sample is above a percentage of 48% while less than 3 time sample is above a percentage of 80%.

Another group of combinations not as good as the four above but still work quite well is *P-P*, *WG-P*, *U-NG* and *WG-WG*, whose average detection time is within 2.3 time samples, the average detection time less than 1 time sample is above a percentage of 45% while less than 3 time sample is above a percentage of 77%.

The third grade for the combinations includes *NG-NG*, *P-WG*, *NG-WG* and *U-WG*, whose average detection time is within 2.7 time samples and the average detection time less than 1 time sample is between a percentage of 41.5% and 45.5% while less than 3 seconds is between a percentage of 75 % and 77%. A special case can be considered to be good is *U-U*, with the shortest detection time (3.45 comparing to 3.835, 3.785, 4.015 time samples) when using uniform distribution in the true trajectory, since the *U-U* combination was used to set the parameters.

The remaining combinations are *P-U, NG-U and WG-U*, whose average detection time is around 4 time samples while the percentage of average detection time less than 1 time sample is lower than 30 % and less than 3 time samples are around 60 %. We can say these combinations perform bad results.

Generally, we give four grades to the 16 distribution combinations, which are very good (*VG*), quite good (*QG*), good (*G*) and bad (*B*). The analysis above is summarized in Table Ⅵ.

Table Ⅵ gives out a reference to choose the distribution for true target trajectory

generation and GM-PHD filter recursion in the research of GM-PHD filter multi-target tracker. It is expected that Poisson in true trajectory will have a good performance with Poisson in GM-PHD filter while narrow Gaussian will have a good performance with narrow Gaussian in GM-PHD filter, etc., which may construct a diagonal in the table. There may be also existed some regulars along rows or columns of the table for the range of distribution is expanding that may affect the performance of the GM-PHD filter.

TABLE Ⅵ GRADES OF DETECTION FOR 16 DISTRIBUTION COMBINATIONS

Grades of Distribution Combinations		**Distribution used in True Target Trajectory**			
		P	***NG***	***WG***	***U***
Distribution used in GM-PHD filter	***P***	*QG*	*VG*	*G*	*B*
	NG	*VG*	*G*	*G*	*B*
	WG	*QG*	*VG*	*QG*	*B*
	U	*VG*	*QG*	*G*	*G*

Analyzing Table Ⅲ and Ⅵ, some interesting regulars can be summarized:

a) Two diagonals can be derived, as shown in Table Ⅵ. The program always performs well by the two diagonals and below them.

b) *NG-P* performs better than *P-P*, and *WG-NG* performs better than *NG-NG*. One possible reason is that when the birth density in true trajectory is Poisson, the PHD filter has only one chance to detect the target at the first moment if it uses Poisson distribution but narrow Gaussian distribution has more chances to detect it. While the detection range becomes too wide (e.g. expand to wide Gaussian), the detection time will increase. A similar case is the comparison of *NG-NG* and *WG-NG*, in which *WG-NG* gets a shorter detection time than *NG-NG*.

c) Two special cases are *P-NG* and *U-P*, whose detection time is thought to be longer,

because the range of uniform distribution may be too wide to detect a Poisson true target birth and the range of Poisson may be too narrow to detect a narrow Gaussian true target birth at the first moment. One possible explanation for the short detection time of *P-NG* is that the mean of narrow Gaussian distribution lies at (0.5, 0.4) with a small deviation σ = 0.01, which means the true target may born in a small range around (0.5, 0.4); the centre of Poisson distribution is the same as narrow Gaussian, the probability to detect a new birth target at the first moment may be quite large at a fixed point (0.5, 0.4) for the true target has a large probability to born just at the center. For the other special case *U-P*, the short detection time may be caused by the parameters are set based on *U-U* combination, which may help to improve the performance of uniform distribution in PHD recursion. The explanation of these phenomena still needs further study.

d) Without considering the column of Uniform distribution in the true target trajectory, and assume the detection time of *P-NG* and *U-P* to be longer, a trend of detection time by the columns can be concluded as “long-short-long” from Poisson to narrow Gaussian to wide Gaussian to uniform distribution, which has a “valley” between to “peaks”. This is consistent with the expected outcome of the experiment.

CHAPTER 6 RESULTS AND CONCLUSION

In multi-target tracking research, find out a closed form solutions to the PHD recursion is very important, which alleviates the computational intractability. In this paper, an algorithm called Gaussian Mixture PHD filter which has the ability to estimate the number of targets and track the trajectories of targets has been implemented. Discussions about how the parameters affect the tracking based on the most random case have been made. Several regulars of the performance of different distribution combinations have been deduced and possible explanations to these regulars are also given. A reference for choosing the probability density function in both true target trajectory simulations and GM-PHD filter recursion has been provided, distributions by and below the two diagonals shown in Table Ⅵ always have a better performance than others.

CHAPTER 7 FUTURE WORK

There are several possible future research directions. The number of clutters after GM-PHD filter still needs to be reduced. How the parameters affect the result of target tracking and the statistic of relay time in GM-PHD recursion with different probability density function needs deeper and more accurate studying. Performance of *P-NG* and *U-P* needs to be discussed with more reasonable explanations. How to apply the GM-PHD filter in actual target tracking problems is also an interesting research area.

APPENDIX A SOURCE CODE OF THE FINAL PROGRAM

A.1. Main Program: *phd.m*

```
close all;clear;

K = 50; % Time samples
N = 3;    % Total number of targets
M = 5;    % Number of observations per time sample
space = [0 1.5 ; 0 1]; % Xmin Xmax Ymin Ymax



p = traj(K,N,space);
z = observe(p,M,space);

h = figure(1);
clf(h);
hold on
for n=1:size(p,3)
   plot(p(:,1,n),p(:,2,n),'r');
end
for n=1:size(z,3)
   plot(z(:,1,n),z(:,2,n),'x');
end
axis([space(1,1) space(1,2) space(2,1) space(2,2)])
hold off
xlabel('x-motion [m]')
ylabel('y-motion [m]')
title('xy-plot, all targets and observations')



figure(2)
subplot(2,1,1)
title('separate x and y motion vs time')
plot(squeeze(p(:,1,:)),'k','LineWidth',2)
hold on;
plot(squeeze(z(:,1,:)),'.b')
axis([0 50 space(1,1) space(1,2)])
hold off;
```

```
ylabel('x-motion [m]')
xlabel('time [n]')
title('x-time plot,all targets and observations')
subplot(2,1,2)
plot(squeeze(p(:,2,:)),'k','LineWidth',2)
hold on;
plot(squeeze(z(:,2,:)),'.b')
axis([0 50 space(2,1) space(2,2)])
hold off;
ylabel('y-motion [m]')
xlabel('time [n]')
title('y-time plot,all targets and observations')


% PHD filter, z is observations, space is possible observation space

[K,L,M] = size(z); % K= #iterations, L= #space dims, M= #estimates

J    = 0;    % Current number of targets
Jg = 5; % Number of normals in birth intensity function
Jm = 20; % Max number of concurrent targets

ps = 0.9; % Probability of survival between generations
pd = 0.99; % Probability of detection
kk = (M/4)/prod(space(:,2)-space(:,1));

% Process and observation models
F = eye(L);          % State transition matrix for kinematic model
Q = 0.01*eye(L);    % Process noise matrix
H = eye(L);          % Observation matrix
R = 0.001*eye(L); % Observation noise matrix

% Definition of normal
N.w = 1.0e-4;            % Weight
N.m = zeros(L,1); % Mean
N.P = eye(L);        % Covariance

Ng(1:Jg) = N;        % New births
```

```
Nt(1:Jm) = N;         % Normal set
Ns(1:Jm,1:M) = N; % Spawned set

T = zeros(L,Jm,K); % Targets in each iteration
Jk = zeros(1,K);      % Nr of target in each iteration

% Loop over time
S    = zeros(L,L,Jm);
Kt = zeros(L,L,Jm);
nu = zeros(L,Jm);
q    = zeros(1,Jm);
detS = zeros(1,Jm);
invS = zeros(L,L,Jm);

for k=1:K
   % Pop some random kids (step.1 prediction for birth targets)
   for j=1:Jg
      Ng(j).m = spos(1,space);
   end

   % Propagate known targets (step2. prediction for existing targets)
   for j=1:J
      Nt(j).w = ps*Nt(j).w;            % Decay
      Nt(j).m = F*Nt(j).m;              % Update mean
      Nt(j).P = Q + F*Nt(j).P*F'; % Update covariance
   end

   % Add new kids to set
   Nt(J+1:J+Jg) = Ng;
   J = J+Jg;

   % Observe (step 3. construction of the PHD update components)
   for j=1:J
      S(:,:,j)          = H*Nt(j).P*H'+R;
      detS(j)         = 1/sqrt(det(S(:,:,j)));
      invS(:,:,j)     = pinv(S(:,:,j));
      Kt(:,:,j)      = Nt(j).P*H' * pinv(S(:,:,j));
      Nt(j).P      = (eye(L) - Kt(:,:,j)*H)*Nt(j).P;
```

```
      nu(:,j)       = H*Nt(j).m;
    end

    % update
    for m=1:M
      wqs = 0.0;
      for j=1:J
        Ns(j,m).m = Nt(j).m + Kt(:,:,j)*(z(k,:,m)' - nu(:,j));
        q(j) = detS(j)*...
            exp(-0.5*(z(k,:,m)'-nu(:,j))'*invS(:,:,j)*(z(k,:,m)'-nu(:,j)));
        wqs = wqs + Nt(j).w*q(j);
      end
      for j=1:J
        Ns(j,m).w = pd*Nt(j).w*q(j)/(kk+pd*wqs);
        Ns(j,m).P = Nt(j).P;
      end
    end
    % Prune
    [Nt,J] = pruning(Ns,J,Nt);
    [T(:,:,k),Jk(k)] = sest(Nt,J,Jm);
end

figure(3)
subplot(2,1,1)
title('separate x and y motion vs time')
plot(squeeze(p(:,1,:)),'k')
hold on;
plot(squeeze(T(1,:,:))','xk')
axis([0 50 space(1,1) space(1,2)])
hold off;
ylabel('x-motion [m]')
xlabel('time [n]')
subplot(2,1,2)
plot(squeeze(p(:,2,:)),'k')
hold on;
plot(squeeze(T(2,:,:))','xk')
axis([0 50 space(2,1) space(2,2)])
hold off;
```

```
ylabel('y-motion [m]')
xlabel('time [n]')
[r1,c1]= find(isnan(squeeze(T(1,:,:))')==0);
numel(r1);
```

A.2. Function for Generating True Trajectory: *traj.m*

```
function p=traj(K,N,space)
% Create trajectories K = nr of time samples, N total number of targets,
% space is possible observation space
for n=1:N
   first = ceil(0.75*rand(1)*K);
   last    = first + floor((0.4+0.6*rand(1))*(K-first));
   pf = spos(1,space);
   pl = [space(1,1)+rand(1)*(space(1,2)-space(1,1)) ; ...
      space(2,1)+rand(1)*(space(2,2)-space(2,1))];
   x = [nan*zeros(1,first) ...
         linspace(pf(1),pl(1),last-first) ...
         nan*zeros(1,K-last)];
   y = [nan*zeros(1,first) ...
         linspace(pf(2),pl(2),last-first) ...
         nan*zeros(1,K-last)];
   p(:,:,n) = [x;y]';
end
```

A.3. Function for Generating a Fixed True Trajectory: *traj3.m*

```
function p=traj3(K,N,space)
% Create trajectories K = nr of time samples, N total number of targets,
% space is possible observation space
mat=[0.3 0.6 0.9];
for n=1:N
   first = ceil(0.75*mat(n)*K);
   last    = first + floor((0.4+0.6*mat(n))*(K-first));
```

```
%     a(n) = rand(1)
%     b(n) = rand(1)
%     c(n) = rand(1)
%     d(n) = rand(1)

%%%%%%%%%% Example 1 %%%%%%%%%%
    a = [0.8281 0.3081 0.4904];
    b = [0.7549 0.9366 0.1324];
    c = [0.7518 0.2756 0.6088];
    d = [0.1800 0.8571 0.7237];

%%%%%%%%%% Example 2 %%%%%%%%%%
%   a = [0.9115 0.9264 0.2990];
%   b = [0.5206 0.3754 0.3804];
%   c = [0.7208 0.6042 0.2830];
%   d = [0.9129 0.6024 0.4653];

%%%%%%%%%% Example 3 %%%%%%%%%%
%   a = [0.8933 0.0353 0.4413];
%   b = [0.1732 0.6595 0.5739];
%   c = [0.8031 0.3043 0.3387];
%   d = [0.0750 0.6491 0.2648];

  pf = [space(1,1)+a(n)*(space(1,2)-space(1,1)) ; ...
     space(2,1)+b(n)*(space(2,2)-space(2,1))];
%     pf = spos2(n,space);
  pl = [space(1,1)+c(n)*(space(1,2)-space(1,1)) ; ...
     space(2,1)+d(n)*(space(2,2)-space(2,1))];
  x = [nan*zeros(1,first) ...
         linspace(pf(1),pl(1),last-first) ...
         nan*zeros(1,K-last)];
  y = [nan*zeros(1,first) ...
         linspace(pf(2),pl(2),last-first) ...
         nan*zeros(1,K-last)];
  p(:,:,n) = [x;y]';
end
```

A.4. Function for Creating Observations: observe.m

```
function z=observe(p,M,space)
% Create observations p = actual positions, M number of observations per
% time sample, space is possible observation space

sigma = 0.01; % Observation noise

L = size(p,2);

for k=1:size(p,1)
   m = 1;
   % Generate true source position measurements
   for n=1:size(p,3)
      if ~isnan(p(k,1,n))
         zt = sigma * randn(1,2) + p(k,:,n);
         zt = max(zt,space(:,1)');
         zt = min(zt,space(:,2)');
         z(k,:,m) = zt;
         m = m+1;
      end
   end

   % Generate clutter
   for m=m:M
      z(k,:,m) = space(:,1)+rand(L,1).*(space(:,2)-space(:,1));
   end
end
```

A.5. Function for Selecting Distributions: spos.m

```
function m=spos(dist,space)

x = 0.5;
y = 0.4;

sigma1 = 0.01;
sigma2 = 0.2;

if dist == 1
      m = [x y]'; % Poisson
elseif dist == 2
      m = [x y]' + sigma1 * randn(2,1); % Normal narrow
elseif dist == 3
    m = [x y]' + sigma2 * randn(2,1); % Normal wide
else
    m = [space(1,1)+rand(1)*(space(1,2)-space(1,1)) ; ... % Uniform
          space(2,1)+rand(1)*(space(2,2)-space(2,1))];
end

if m(1) < space(1,1)
      m(1) = space(1,1);
end

if m(2) < space(2,1)
      m(2) = space(2,1);
end

if m(1) > space(1,2)
      m(1) = space(1,2);
end

if m(2) > space(2,2)
      m(2) = space(2,2);
end
```

A.6. Function for Pruning the Gaussian Components: *pruning.m*

```
function [Nt,J] = pruning(Ns,J,Nt)

[d,M]=size(Ns);
for m=1:M
    for j=1:J
        wt(j,m) = Ns(j,m).w;
    end
end

wt = wt(:);
Ns = Ns(:); % Indices in wt corresponds to indices in Ns
sum_w = sum(wt(:));
idx = find(wt>1e-5); % Finding weights which are below truncation threshold
Ns = Ns(idx);
Jk = length(idx);
sum_above_thr = sum(wt(idx));

for i=1:Jk
    Ns(i).w = Ns(i).w* sum_w/sum_above_thr;
%       Nsw(i)=Ns(i).w;
end

% Merging
M_threshold = 4;    % Merging threshold
l = 0;
[U1,II] = sort(wt(idx),'descend'); % Sorting N_Ns_w in descending order, II is the index of the
                                   % elements in N_Ns_w
JJ = length(II);

while JJ > 0
    l = l+1;
    for d = 1:JJ
        U_value(d) = (Ns(II(d)).m-Ns(II(1)).m)'*inv(Ns(II(d)).P)*(Ns(II(d)).m-Ns(II(1)).m);
% Because II is the index after sorted in descending order, II(1) is always the index of the
% maximum value of Ns_w each time
    end
```

```
    [U2,Inx2] = sort(U_value(1:JJ)); % Sorting U_value(1:JJ) in ascending order
    ind2 = find(U2 <= M_threshold);
    lth = length(ind2); % The number of elements that U_value <= merging threshold

    for in2 = 1:lth
        LL(in2) = II(Inx2(in2));
        % LL is the index of Ns that the aggregate of U_value < merging threshold
    end
    L1 = LL(1:lth); % L1 is a vector to save LL each time

    sum_NSw = 0;
    sum_NSwx = zeros(2,1);

    for e = 1:lth
        sum_NSw = sum_NSw + Ns(L1(e)).w;
        sum_NSwx = sum_NSwx + Ns(L1(e)).w*Ns(L1(e)).m;
    end

    Nt(l).w = sum_NSw;
    Nt(l).m = sum_NSwx/Nt(l).w;

    sum_NSP = zeros(2,2);

    for g = 1:lth
        sum_NSP = sum_NSP +
                    Ns(L1(g)).w*(Ns(L1(g)).P+(Nt(l).m-Ns(L1(g)).m)*(Nt(l).m-Ns(L1(g)).m)');
    end

    Nt(l).P = sum_NSP/Nt(l).w;
    II = setdiff(II, L1); % Remove sets LL from II, II is a new vector.
    JJ = length(II);

end

J=l;
Nt = Nt(1:J);
```

A.7. Function for Extracting the Multi-target State: *sest.m*

```
function [T,J]=sest(Nt,J,Jm)

T = zeros(length(Nt(1).m),Jm)+nan;
w=[];
for li = 1:J
    w(li) = Nt(li).w;
end
num1 = find(w>0.4);
for q1 = 1:length(num1)
    T(:,q1) = Nt(num1(q1)).m;
end
J=length(num1);
```

APPENDIX B VALUE OF CONSTANTS USED IN FINAL PROGRAM

B.1. Constants in main program *phd.m*:

Time samples: $K = 50$;

Total number of targets: $N = 3$;

Number of observations per time sample: $M = 5$;

Tracking area: $[0,\ 1.5]\times[0,\ 1]$;

Probability of survival between generations : $ps = 0.9$;

Probability of detection: $pd = 0.99$;

State transition matrix for kinematic model: $F = I$;

Process noise matrix: $Q = 0.01I$;

Observation matrix: $H = I$;

Observation noise matrix : $R = 0.001I$;

Weight: $N.w = 0.0001$.

B.2. Constant in *observe.m*

Observation noise: $\sigma = 0.01$.

B.3. Constants in *traj3.m*

Vector for fixing the first and last position of the true trajectory: *mat* = [0.3 0.6 0.9];

Constants for generating the fixed true trajectory using uniform distribution, where *a*, *b*, *c* and *d* are obtained by the Matlab command rand(1) respectively:

Example 1:

a = [0.8281 0.3081 0.4904];
b = [0.7549 0.9366 0.1324];
c = [0.7518 0.2756 0.6088];

d = [0.1800 0.8571 0.7237];

Example 2:

a = [0.9115 0.9264 0.2990];
b = [0.5206 0.3754 0.3804];
c = [0.7208 0.6042 0.2830];
d = [0.9129 0.6024 0.4653];

Example 3:

a = [0.8933 0.0353 0.4413];
b = [0.1732 0.6595 0.5739];
c = [0.8031 0.3043 0.3387];
d = [0.0750 0.6491 0.2648];

B.4. Constants in *spos.m*

Centre of Poisson Distribution: $(x, y) = (0.5, 0.4)$;

Standard deviation for Narrow Gaussian Distribution: $\sigma_1 = 0.01$;

Standard deviation for Wide Gaussian Distribution: $\sigma_2 = 0.2$;

B.5. Constants in *pruning.m*

Truncation threshold: $\tau = 10^{-5;}$

Merging threshold: $U = 4$;

B.6. Constants in *sest.m*

Threshold of weights for selecting Gaussian components: $T = 0.4$.

REFERENCE LIST

[1] S. Blackman, *Multiple Target Tracking with Radar Applications*. Artech House, Norwood, 1986.

[2] D. Reid, "An algorithm for tracking multiple targets," *IEEE Trans. AC*, vol. AC-24, no.6, pp. 843-854, 1979.

[3] S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE A & E Systems Magazine*, vol. 19, no. 1, part 2, pp. 5-18, 2004.

[4] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*. Academic Press, San Diego, 1988.

[5] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Sonar tracking of multiple targets using joint probabilistic data association," *IEEE J. Oceanic Eng.*, vol. OE-8, no.July, pp. 173-184, 1983.

[6] R. L. Streit and T. E. Luginbuhl, "Maximum likelihood method for probabilistic multi-hypothesis tracking," in *Proc. SPIE*, vol. 2235, pp. 5-7, 1994.

[7] C. Hue, J. P. Lecadre, and P. Perez, "Sequential Monte Carlo methods for multiple target tracking and data fusion," *IEEE Trans. Trans. SP*, vol. 50, no. 2, pp. 309-325, 2002.

[8] --------, "Tracking multiple objects with particle filtering," *IEEE Trans. AES*, vol. 38, no. 3, pp. 791-812, 2002.

[9] R. Mahler, "Multi-target Bayes filtering via first-order multi-target moments," *IEEE Trans. AES*, vol. 39, no. 4, pp. 1152-1178, 2003.

[10] I. Goodman, R. Mahler, and H. Nguyen, *Mathematics of Data Fusion*. Kluwer Academic Publishers, 1997.

[11] --------, "An introduction to multisource-multitarget statistics and applications," *Lockheed Martin Technical Monograph*, 2000.

[12] B. Vo, S. Singh, and A. Doucet, "Sequential Monte Carlo implementation of the PHD filter for multi-target tracking," in *Proc. Int'l Conf. on Information Fusion*, Cairns, Australia, pp. 792-799, 2003.

[13] --------, "Sequential Monte Carlo Methods for multi-target filtering with random finite sets," *IEEE Trans. Aerospace and Electronic Systems*, vol. 41, no. 4, pp. 1224-1245, 2005, also http://www.ee.unimelb.edu.au/staff/bv/publications.html

[14] Ba-Ngu Vo, Wing-Kin Ma, "The Gaussian Mixture Probability Hypothesis Density filter," *IEEE Trans. Signal Processing*, vol. 54, no. 11, pp. 4091-4104, Nov, 2006.

[15] D. Daley and D. Vere-Jones, *An introduction to the theory of point processes*. Springer-Verlag, 1998.

[16] D. Stoyan, D. Kendall, and J. Mecke, *Stochastic Geometry and its applications*. John Wiley & Sons, 1995.

[17] B. Vo and S. Singh, "Technical aspects of the Probability Hypothesis Density recursion." *Tech. Rep. Tr05-006* EEE Dept. The University of Melbourne, Australia, 2005.

[18] R. Mahler, "A theoretical foundation for the Stein-Winter Probability Hypothesis Density (PHD) multi-target tracking approach," in *Proc. 2002 MSS Nat'l Symp. On Sensor and Data Fusion*, vol. 1, (Unclassified) Sandia National Laboratories, San Antonio TX, 2000.

[19] M. C. Stein and C. L. Winter, "An adaptive theory of probabilistic evidence accrual," *Los Alamos*

National Laboratories Report, LA-UR-93-3336, 1993.

[20] M. Sanjeev, Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp, “A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking,” *IEEE Trans. Signal Processing*, vol. 50, no. 2, pp. 174-188, Feb, 2002.

[21] R. Mahler, "A theoretical foundation for the Stein-Winter Probability Hypothesis Density (PHD) multi-target tracking approach," *Proc. MSS Nat'l Symp. on Sensor and Data Fusion*, vol. I (Unclassified), San Antonio TX, June 2000.

[22] R. Mahler, "Multi-target moments and their application to multi-target tracking," *Proc. Workshop on Estimation, Tracking and Fusion: A tribute to Yaakov Bar-Shalom*, Monterey, pp. 134-166, 2001.

[23] T. Zajic, R. Ravichandran, R. Mahler, R.Mehra, and M. Noviskey, "Joint tracking and identification with robustness against unmodeled targets," in *Signal Processing, Sensor Fusion and Target Recognition XII, SPIE Proc.*, vol. 5096, pp. 279–290, 2003.

[24] H. Sidenbladh, "Multi-target particle filtering for the Probability Hypothesis Density," in *Proc. Int'l Conf. on Information Fusion*, Cairns, Australia, pp. 800–806, 2003.

[25] Ba-Ngu Vo, Wing-Kin Ma, "A closed-form solution for the probability hypothesis density filter," *Proc. Information Fusion, 2005 8th International Conference*, vol. 2, pp. 25-28, July 2005.

[26] D. E. Clark, and B.-N. Vo, "Convergence analysis of the Gaussian Mixture Probability Hypothesis Density filter," *IEEE Trans. Signal Processing*, vol. 55, no. 4, pp. 1204-1212, 2007.

[27] D. E. Clark, B.-N. Vo, and J. Bell, "GM-PHD filter multi-target tracking in sonar images," *Proc. SPIE'06*, Florida, USA, 2006.

[28] D. E. Clark, K. Panta, and B.-N. Vo, "The Gaussian mixture PHD filter Multiple Target Tracker," *Proc. 9th Annual Conf. Information Fusion*, Florence, Italy, 2006.

[29] N. T. Pham, W. Huang, S. H. Ong, *Tracking Multiple Objects using Probability Hypothesis Density Filter and Color Measurements Multimedia and Expo,* 2007 IEEE International Conference, pp. 1511 – 1514, July 2007.

[30] R. Mahler, "A survey of PHD filter and CPHD filter implementations," *Signal Processing, Sensor Fusion, and Target Recognition XV, SPIE Defense & Security Symposium*, April 2007.